长安大学

高性能计算平台 AI 模块
使用手册

V1.0

目　录

第一章 概述与架构

人工智能平台提供了丰富的算法和建模框架，不仅提供了用户图形化、拖拽式的建模方式，支持用户“零编码建模”；也提供了开放式接口，支持高级用户自定义算法和模型。用户创建的模型可以用来创建自己的 AI 场景解决方案，也可以共享给其他用户使用。人工智能软件提供了数据集、模型共享功能，用户可以把自己的数据集和模型共享出去，帮助团队简单、快速地构建模型。人工智能平台支持主流深度学习框架，包括 Tensorflow、Pytorch、Mxnet、PaddlePaddle、MindSpore 等，还提供了常用的 VSCode、Jupyter 等 Web IDE，灵活扩展用户 AI 实践过程。

系统的整体功能架构如下图所示：

系统功能架构图

整个系统主要包括 Web 应用门户、数据集管理、模型管理、方案管理、图像标注、镜像管理等模块。

第二章 主要功能

2.1. 数据集

人工智能平台提供了丰富的数据集管理功能，满足了不同类型数据在不同训练场景下的管理需求，支持数值、图像、视频、音频、文本以及其他类型数据的管理，也支持进行用户数据的在线标注。

数据集中包含“我的数据集”“共享数据集”和“数据标注”功能，在“我的数据集”和“共享数据集”下可以创建“数值数据”“图像数据”“视频数据”“音频数据”“文本数据”和“其他”数据集。

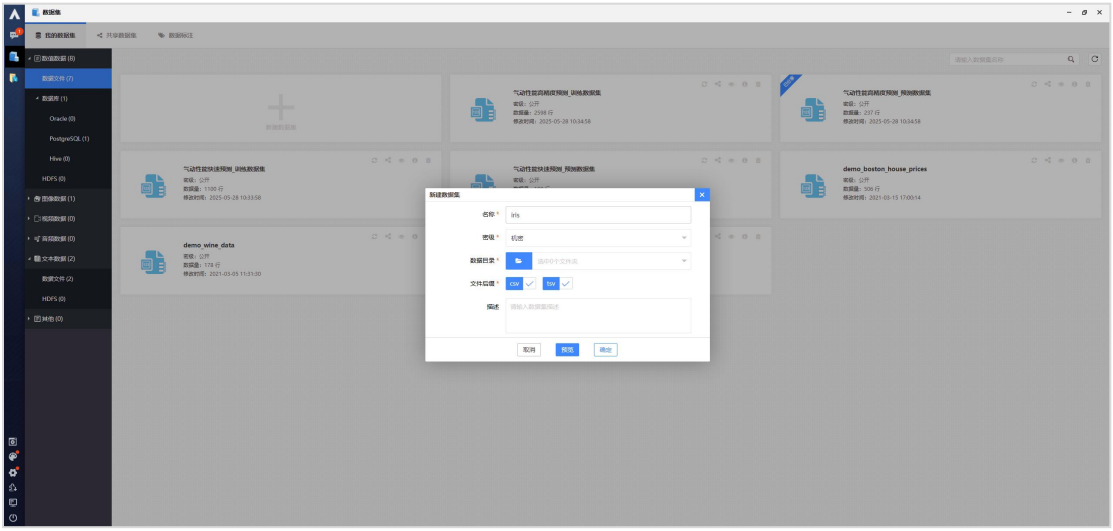
数据集同时支持数据集的共享功能，让不同用户之间可以相互共享数据，并支持设置共享的权限。

方案设计中支持以可视化方式引用数据集，可以使用已有的数据集进行数据处理和模型训练。

2.1.1. 数值数据

2.1.1.1. “数值数据-数据文件”数据集

新建“数值数据-数据文件”数据集，操作步骤如下：依次点击“我的数据集”-“数值数据”-“数据文件”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“数值数据-数据文件”数据集

图中每个参数的具体含义如下：

名称：数据集名称，输入任意符合命名规则的名称即可。

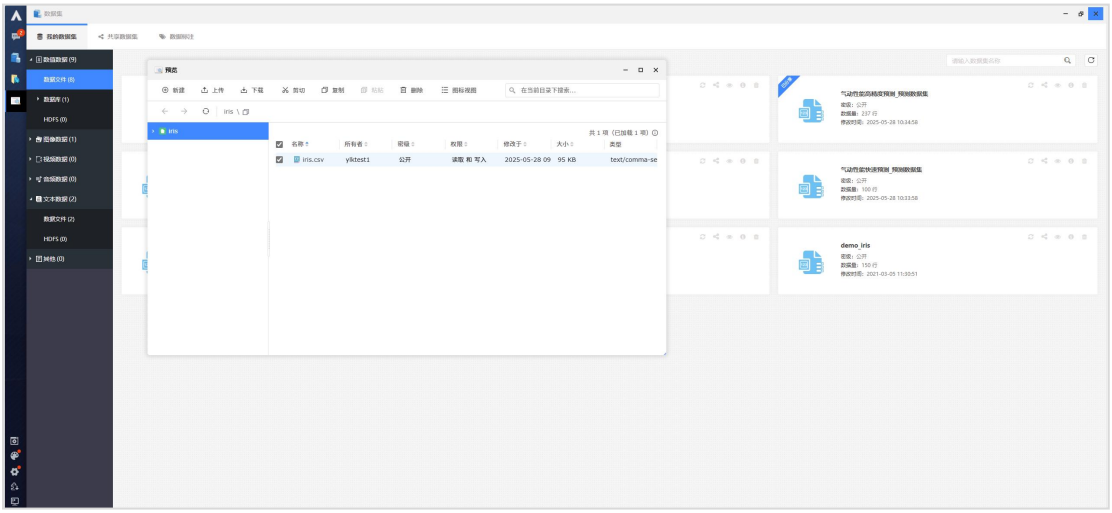
密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

数据目录：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择数据目录”窗口中选择数据目录即可。当数据目录中包含多个数据文件时，需要保证每个数据文件的表头相同，否则会导致新建数据集失败。

文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 2 种，分别是 csv 和 tsv，可单选和全选，至少选择其中一项。

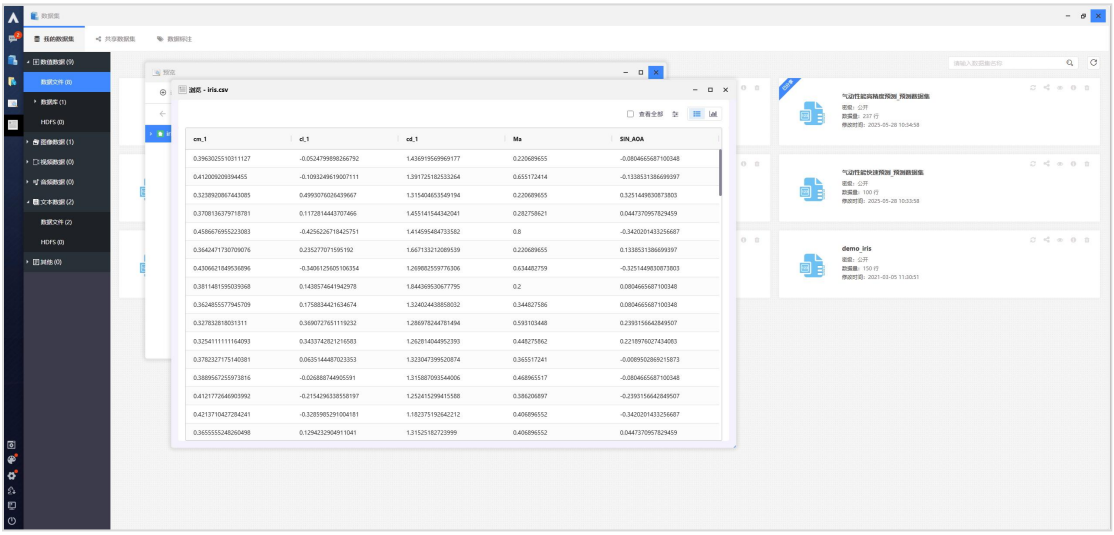
描述：输入该数据集的描述信息，可选。

配置完成后，点击“预览”按钮，即可预览数据集的数据文件，如下图所示：



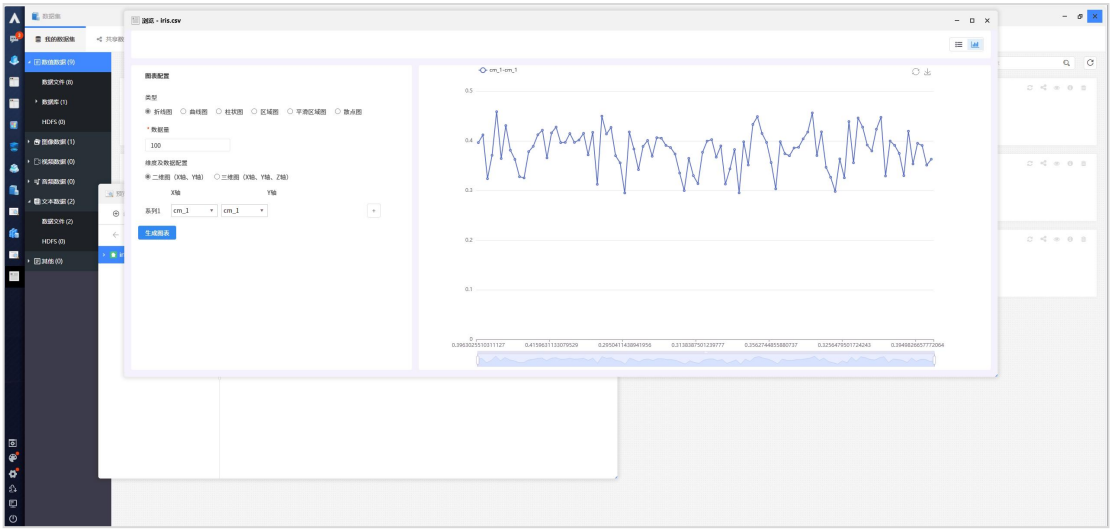
预览“数值数据-数据文件”数据集

双击文件，可以预览数据文件的内容，如下图所示：



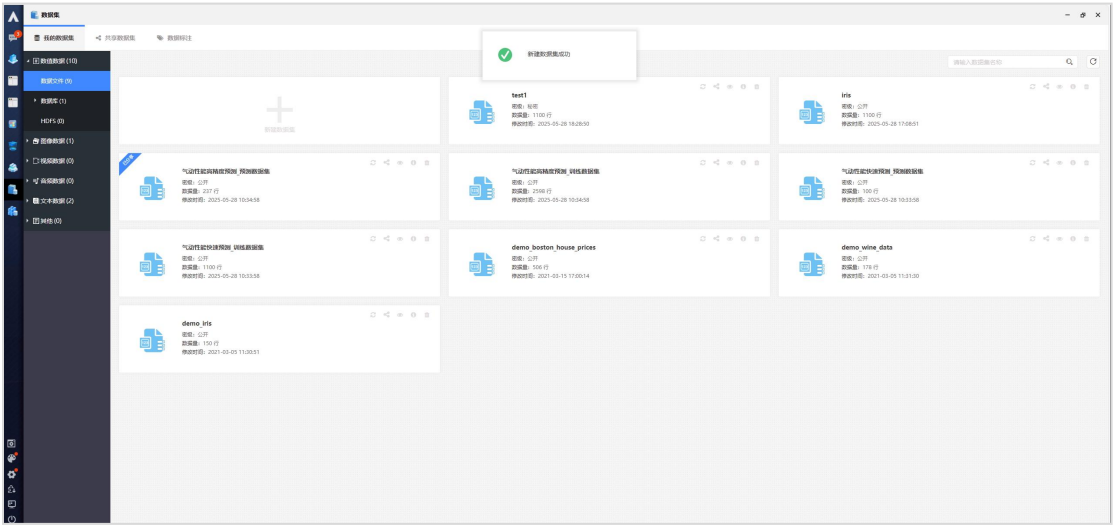
以列表视图形式，预览“数值数据-数据文件”数据集

点击浏览窗口右上角的“图表视图”按钮，将列表视图界面切换至图表视图界面，如下图所示：



以图标视图形式，预览“数值数据-数据文件”数据集

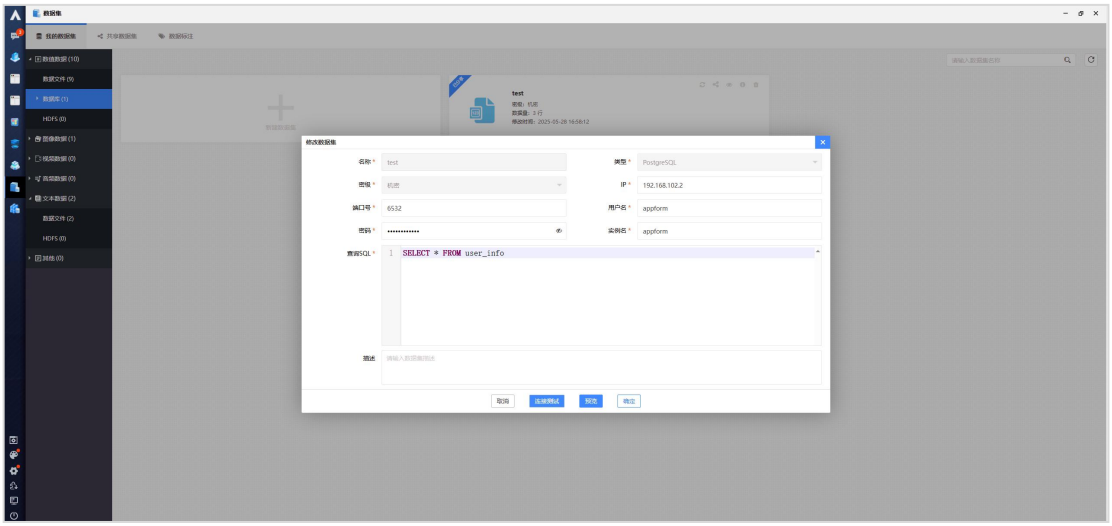
点击“确定”按钮，新建数据集，如下图所示：



新建“数值数据-数据文件”数据集成功

2.1.1.2. “数值数据-数据库”数据集

以新建“PostgreSQL 数据库”数据集为例，操作步骤如下：依次点击“我的数据集”-“数值数据”-“数据库”-“PostgreSQL”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“数值数据-PostgreSQL 数据库”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

类型：选择数据库的类型，默认回填当前数据分类的类型，如：PostgreSQL。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

IP：输入数据库所在机器的 IP。

端口号：输入数据库使用的端口号。

用户名：输入数据库的用户名。

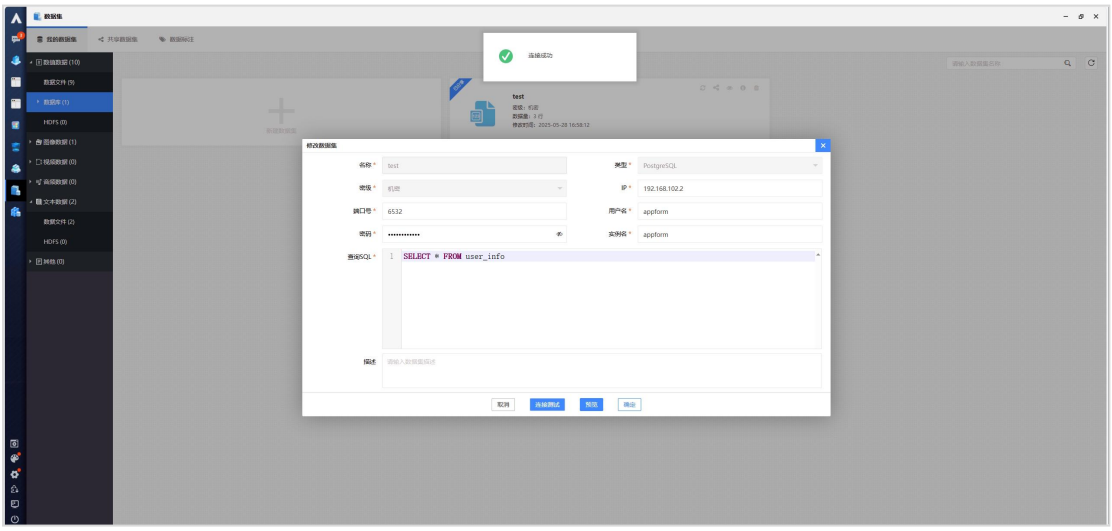
密码：输入数据库用户对应的密码。

实例名：输入数据库名。

查询 SQL：输入 SQL 查询语句，例如：select * from data_set（注意：不要在查询语句后面加分号）。

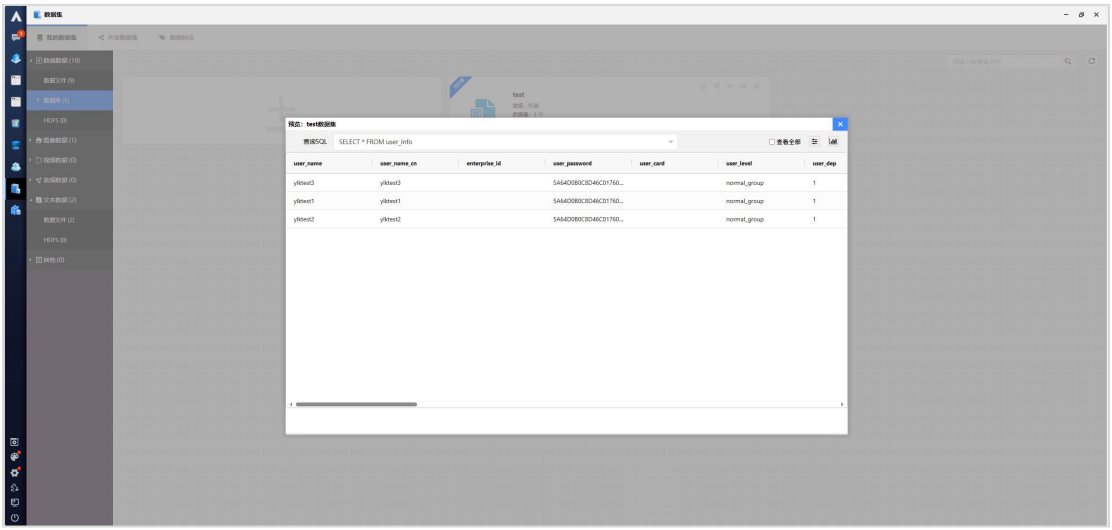
描述：输入该数据集的描述信息，可选。

配置完对应信息后，点击“连接测试”按钮，即可测试数据库的连通性，会有成功和失败提示。连接测试效果如下图所示：



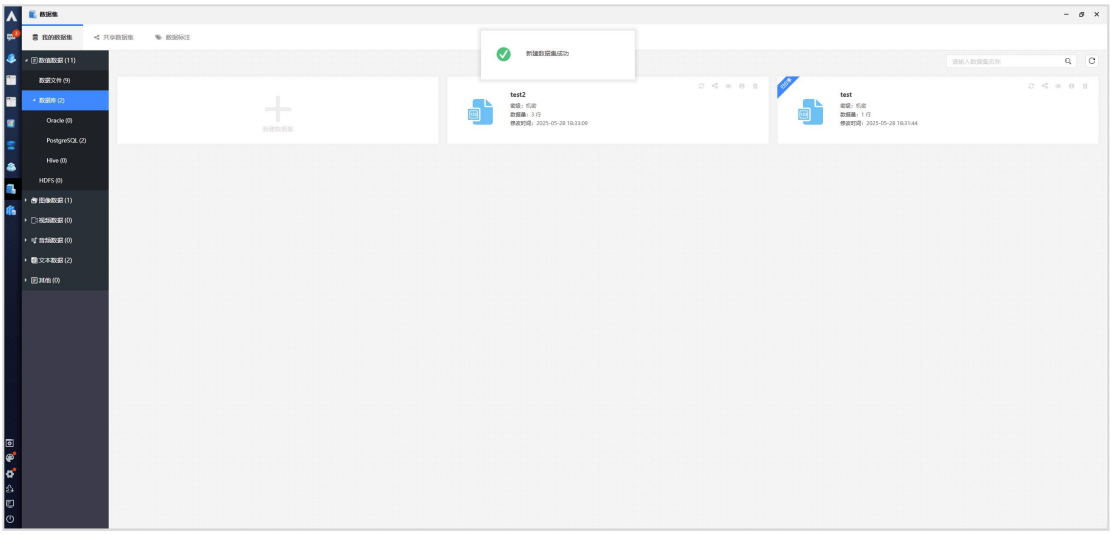
数据库连接测试

点击“预览”按钮，即可查看“查询 SQL”的结果。如下图所示：



预览“数值数据-PostgreSQL 数据库”数据集

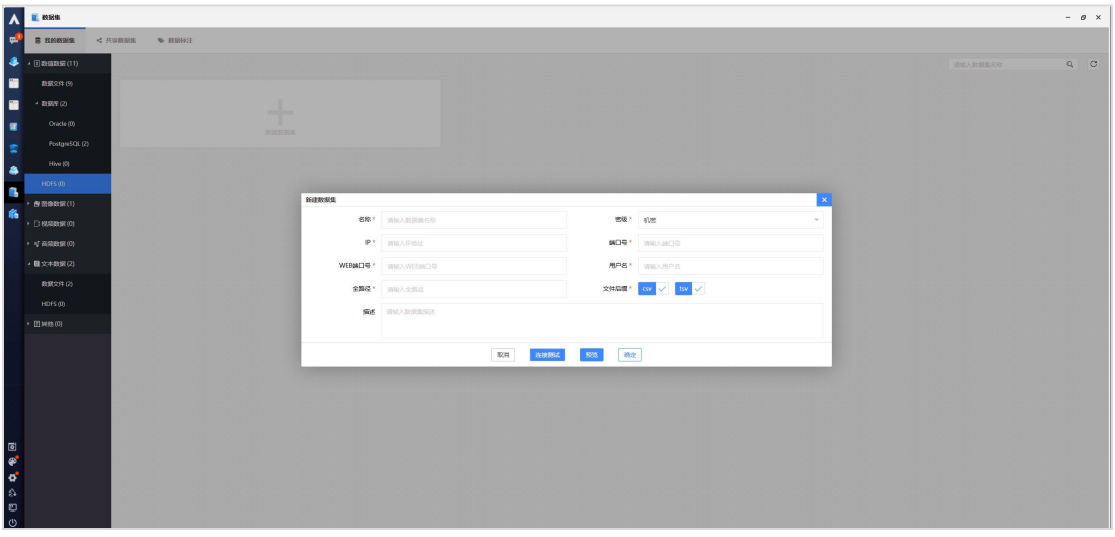
点击“确定”按钮，新建数据集。如下图所示：



新建“数值数据-PostgreSQL 数据库”数据集成功

2.1.1.3. “数值数据-HDFS”数据集

新建“数值数据-HDFS”数据集，操作步骤如下：依次点击“我的数据集”-“数值数据”-“HDFS”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“数值数据-HDFS”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安

全、保密。但是目前不支持在打开密级功能的时候，使用 HDFS 功能。

IP：输入 HDFS 服务节点的 IP。

端口号：HDFS 集群的 RPC 通信端口，即$HADOOP_HOME/etc/hadoop/core-site.xml HDFS 配置文件中的“fs.defaultFS”配置参数的值。该端口一般为：8020 或 9000。

WEB 端口号：HDFS NameNode 的 Http UI 端口，用于方案设计中应用 HDFS 数据集时，调用该端口。需要根据 hadoop 版本，配置 WEB 端口，如：hadoop2 的默认 WEB 端口为 50070，hadoop3 的默认 WEB 端口为 9870。

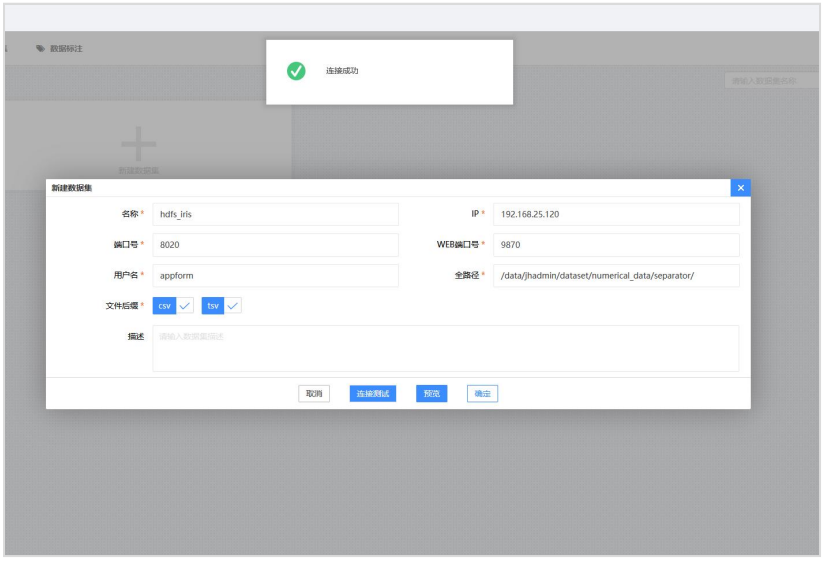
用户名：输入 HDFS 服务的用户名。

全路径：输入数据目录的绝对路径，系统默认自动拼接“管理门户-系统管理-系统配置-人工智能-人工智能基础配置”配置文件的“HDFS 根路径挂载点”的值。

文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 2 种，分别是 csv 和 tsv，可单选和全选，至少选择其中一项。

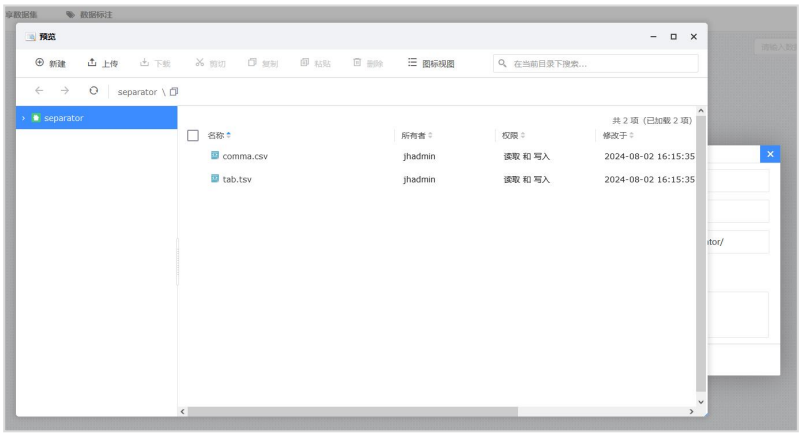
描述：输入该数据集的描述信息，可选。

配置完对应信息后，点击“连接测试”按钮，即可测试 HDFS 服务的连通性，会有成功和失败提示。连接测试效果如下图所示：



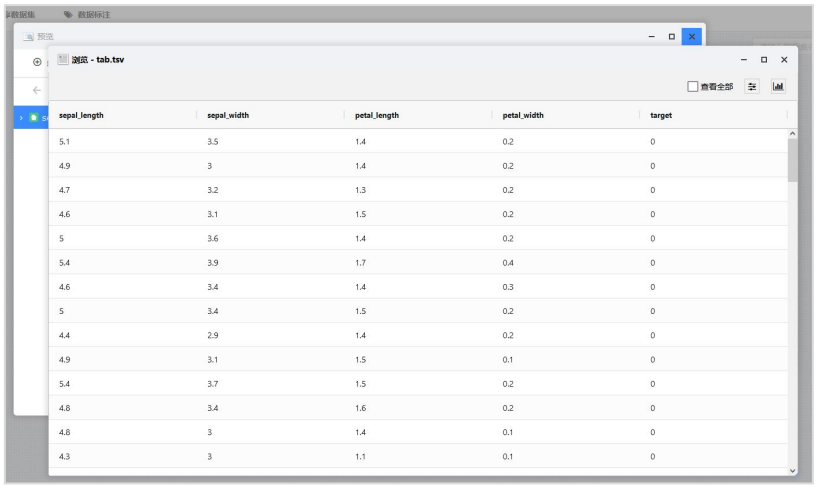
连接测试

点击“预览”按钮，即可预览数据集的数据文件。如下图所示：



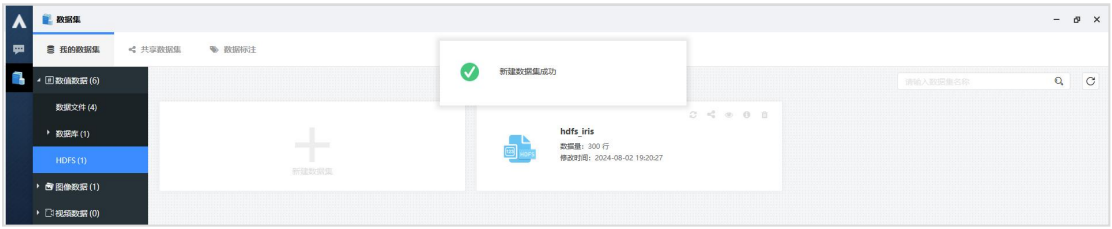
预览“数值数据-HDFS”数据集

双击文件，可以预览数据文件的内容，如下图所示：



预览“数值数据-HDFS”数据集

点击“确定”按钮，新建数据集。如下图所示：

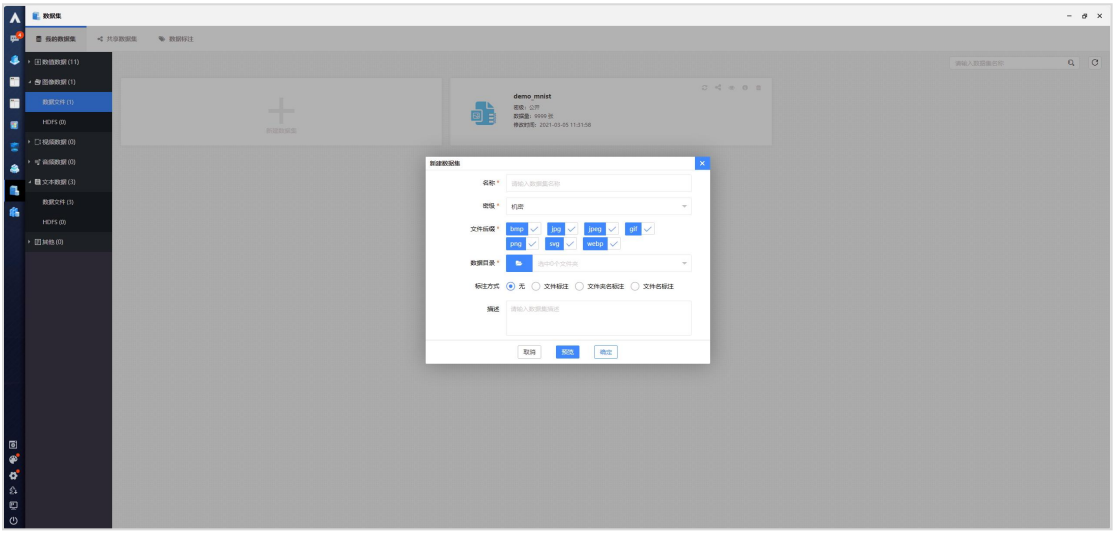


新建“数值数据-HDFS”数据集成功

2.1.2. 图像数据

2.1.2.1. “图像数据-数据文件”数据集

新建“图像数据-数据文件”数据集，操作步骤如下：依次点击“我的数据集”-“图像数据”-“数据文件”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“图像数据-数据文件”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 bmp、jpg、jpeg、gif、png、svg 和 webp，可单选和全选，至少选择其中一项。

数据目录：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择数据目录”窗口中，选择图像文件夹即可。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目

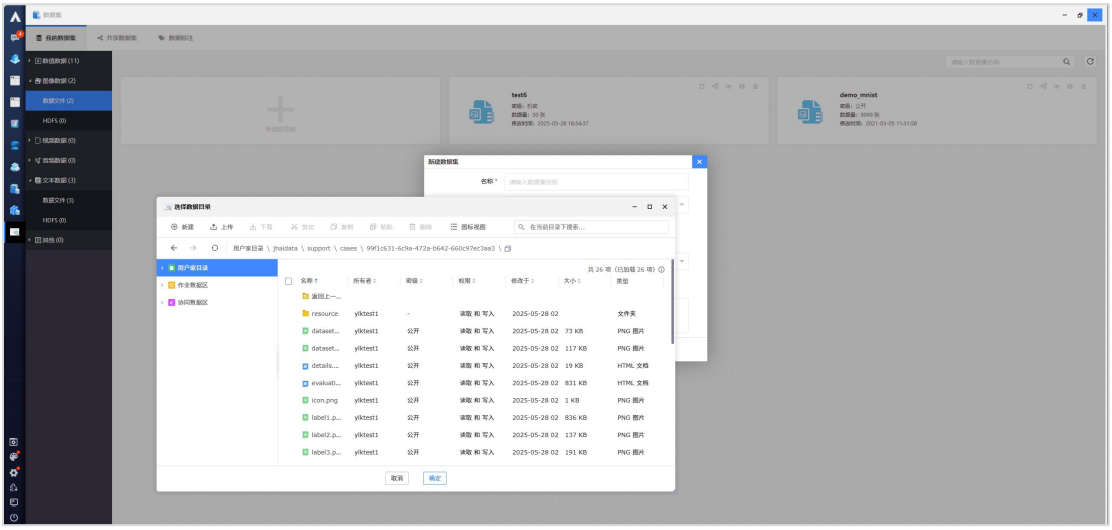
录项对应的标签文件，目前仅支持“json”格式的标签文件；

3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；

4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

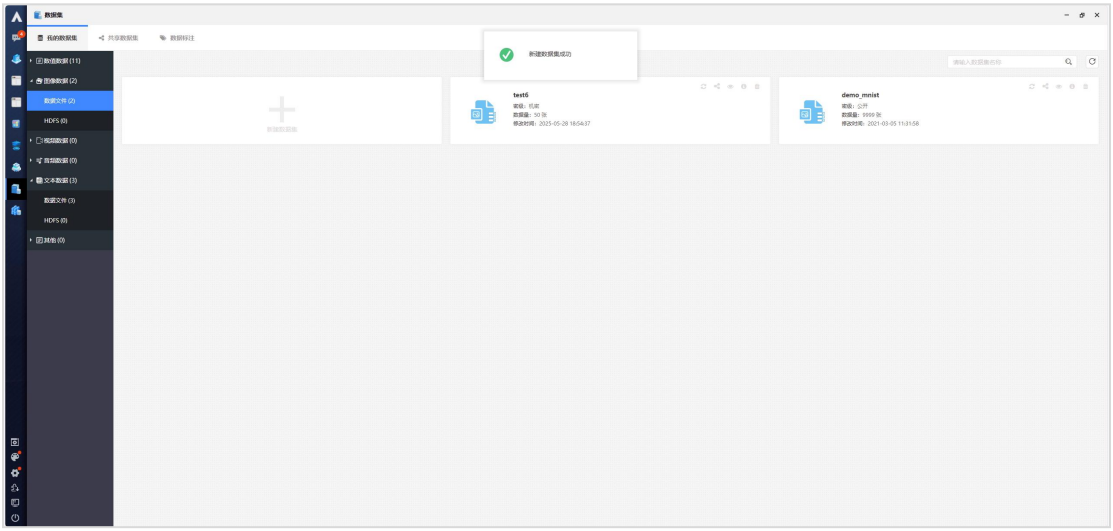
描述：输入该数据集的描述信息，可选。

配置完对应的信息后，点击“预览”按钮，即可预览数据集的数据文件。预览效果如下图所示：



预览“图像数据-数据文件”数据集

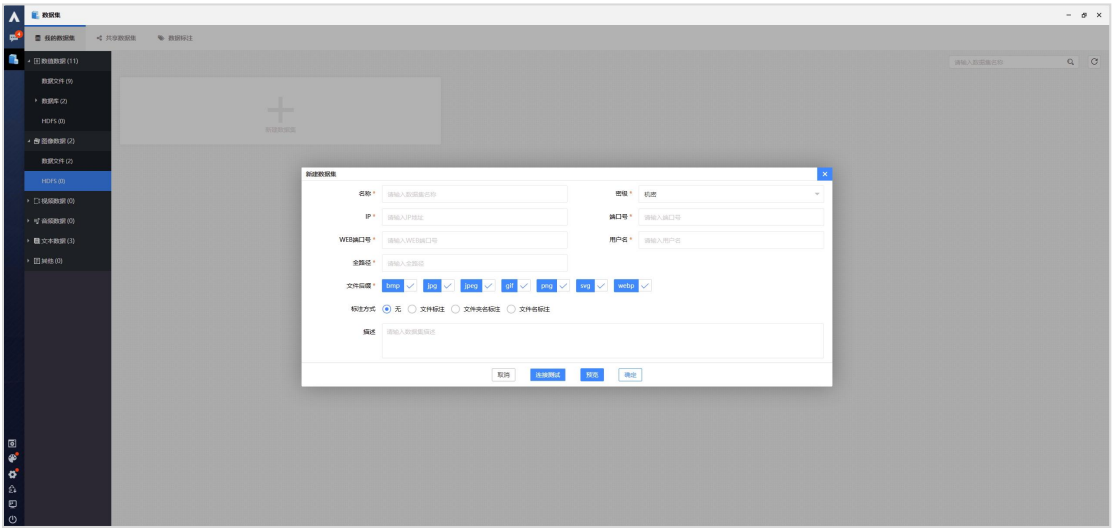
点击“确定”按钮，新建数据集。如下图所示：



新建“图像数据-数据文件”数据集成功

2.1.2.2. “图像数据-HDFS”数据集

新建“图像数据-HDFS”数据集，操作步骤如下：依次点击“我的数据集”-“图像数据”-“HDFS”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“图像数据-HDFS”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。但是目前不支持在打开密级功能的时候，使用 HDFS 功能。

IP：输入 HDFS 服务节点的 IP。

端口号：HDFS 集群的 RPC 通信端口，即$HADOOP_HOME/etc/hadoop/core-site.xml HDFS 配置文件中的“fs.defaultFS”配置参数的值。该端口一般为：8020 或 9000。

WEB 端口号：HDFS NameNode 的 Http UI 端口，用于方案设计中应用 HDFS 数据集时，调用该端口。需要根据 hadoop 版本，配置 WEB 端口，如：hadoop2 的默认 WEB 端口为 50070，hadoop3 的默认 WEB 端口为 9870。

用户名：输入 HDFS 服务的用户名。

全路径：输入数据目录的绝对路径，系统默认自动拼接“管理门户-系统管理-系统配置-人工智能-人工智能基础配置”配置文件的“HDFS 根路径挂载点”的值。

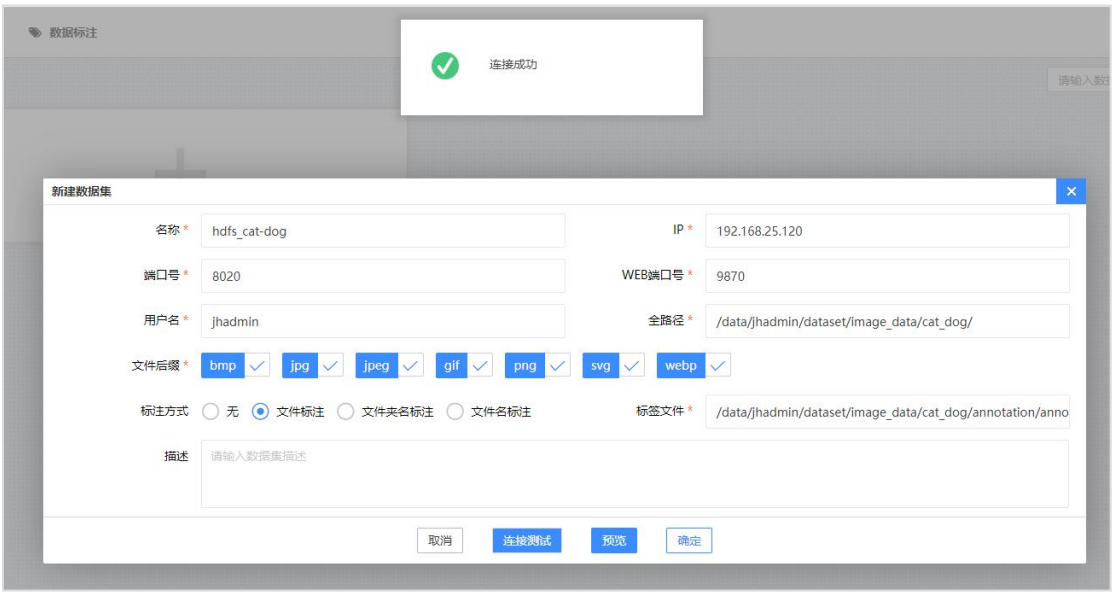
文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 bmp、jpg、jpeg、gif、png、svg 和 webp，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；
3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；
4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

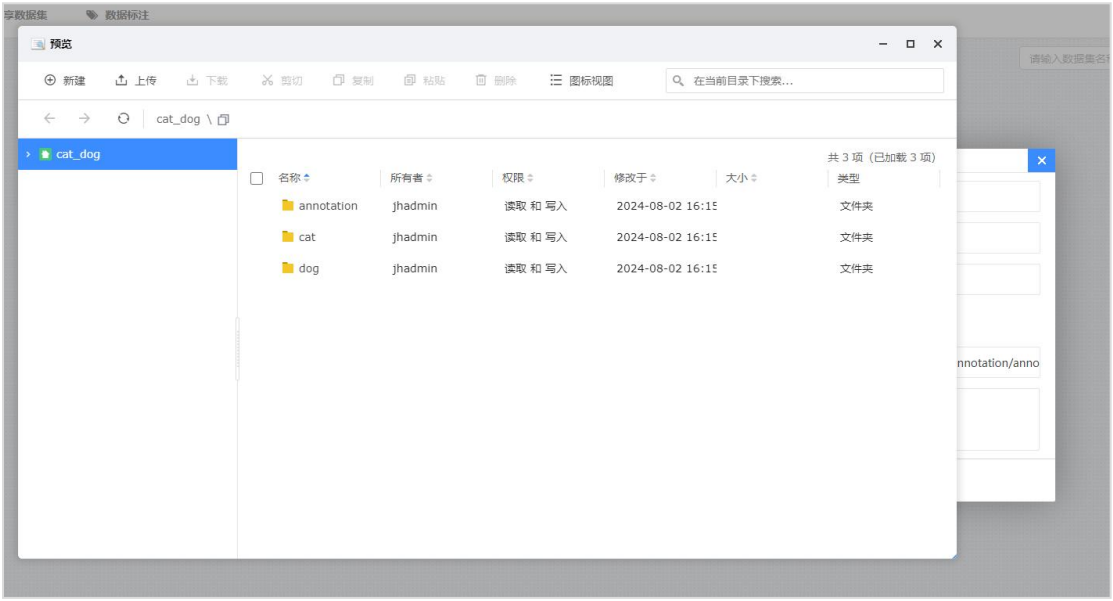
描述：输入该数据集的描述信息，可选。

配置完对应信息后，点击“连接测试”按钮，即可测试 HDFS 服务的连通性，会有成功和失败提示。连接测试效果如下图所示：



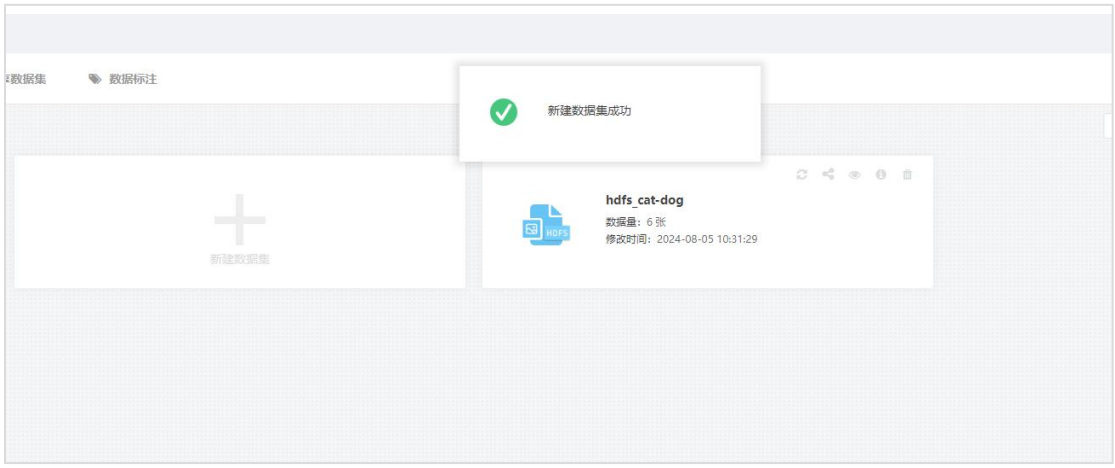
连接测试

点击“预览”按钮，即可查看数据集的数据文件数据。如下图所示：



预览“图像数据-HDFS”数据集

点击“确定”按钮，新建数据集。如下图所示：

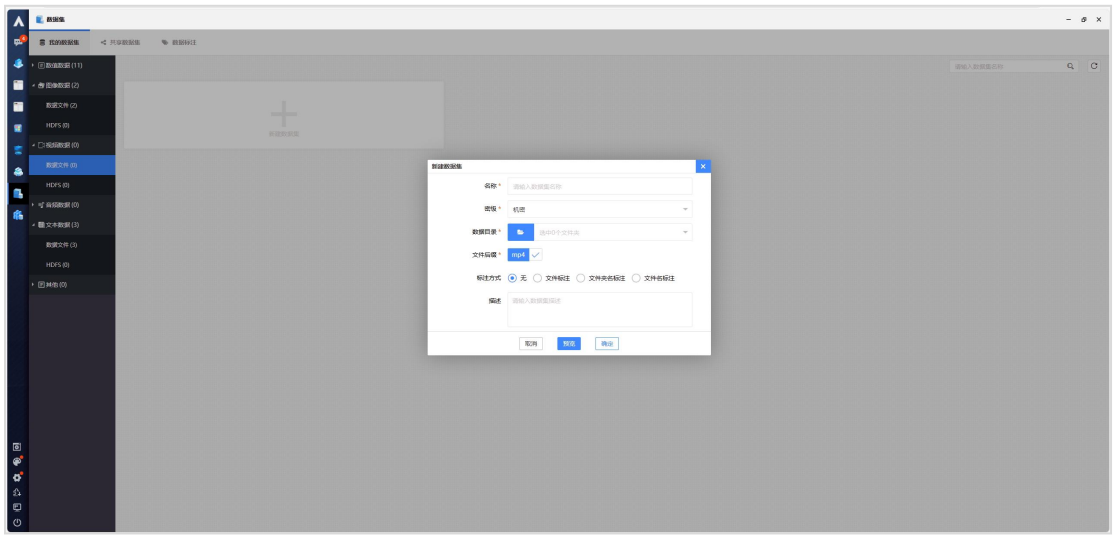


新建“图像数据-HDFS”数据集成功

2.1.3. 视频数据

2.1.3.1. “视频数据-数据文件”数据集

新建“视频数据-数据文件”数据集，操作步骤如下：依次点击“我的数据集”-“视频数据”-“数据文件”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“视频数据-数据文件”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

数据目录：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择数据目录”窗口中，选择视频文件夹即可。

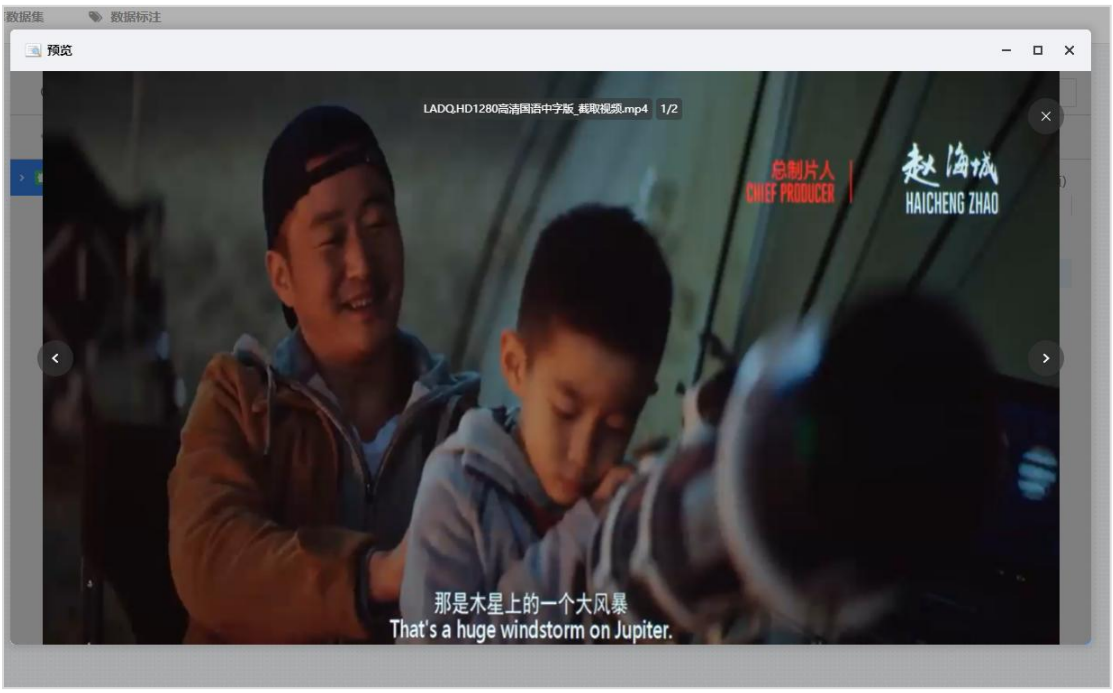
文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型仅有 mp4，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；
3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；
4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

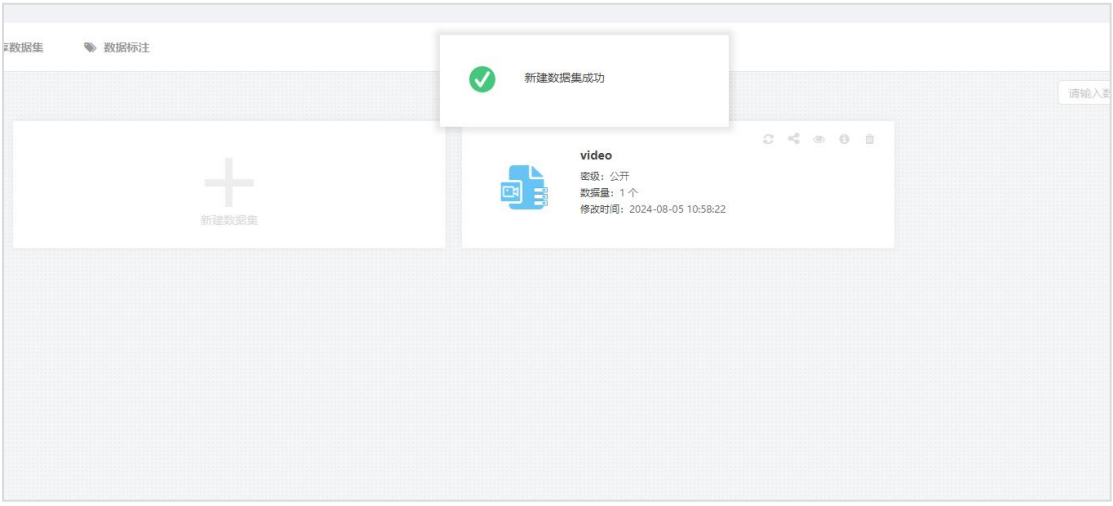
描述：输入该数据集的描述信息，可选。

配置完对应的信息后，点击“预览”按钮，即可查看数据集的数据文件数据。预览效果如下图所示：



预览“视频数据-数据文件”数据集

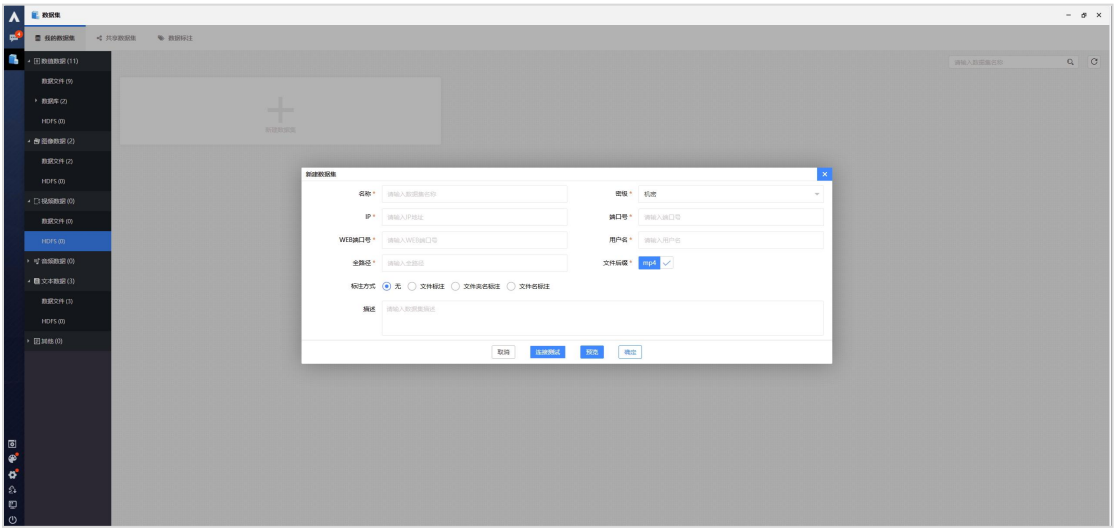
点击“确定”按钮，新建数据集。如下图所示：



新建“视频数据-数据文件”数据集成功

2.1.3.2. “视频数据-HDFS”数据集

新建“视频数据-HDFS”数据集，操作步骤如下：依次点击“我的数据集”-“视频数据”-“HDFS”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“视频数据-HDFS”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。但是目前不支持在打开密级功能的时候，使用 HDFS 功能。

IP：输入 HDFS 服务节点的 IP。

端口号：HDFS 集群的 RPC 通信端口，即$HADOOP_HOME/etc/hadoop/core-site.xml HDFS 配置文件中的“fs.defaultFS”配置参数的值。该端口一般为：8020 或 9000。

WEB 端口号：HDFS NameNode 的 Http UI 端口，用于方案设计中应用 HDFS 数据集时，调用该端口。需要根据 hadoop 版本，配置 WEB 端口，如：hadoop2 的默认 WEB 端口为 50070，hadoop3 的默认 WEB 端口为 9870。

用户名：输入 HDFS 服务的用户名。

全路径：输入数据目录的绝对路径，系统默认自动拼接“管理门户-系统管理-系统配置-人工智能-人工智能基础配置”配置文件的“HDFS 根路径挂载点”的值。

文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型仅有 mp4，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1） “无”：默认值，表示没有使用任何标注方式。用法：可用来当作

测试集；

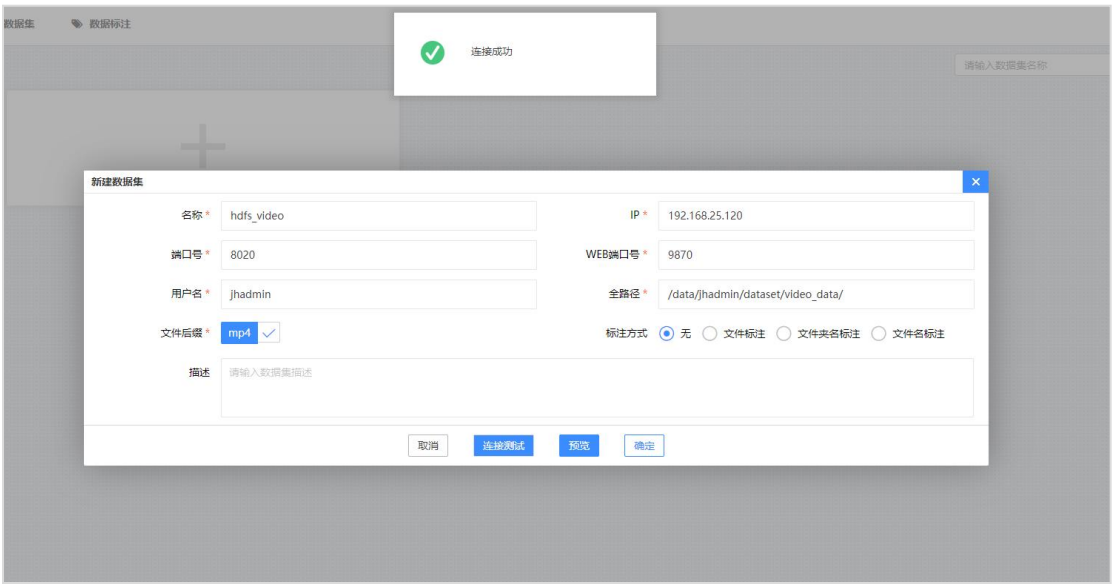
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；

3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；

4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

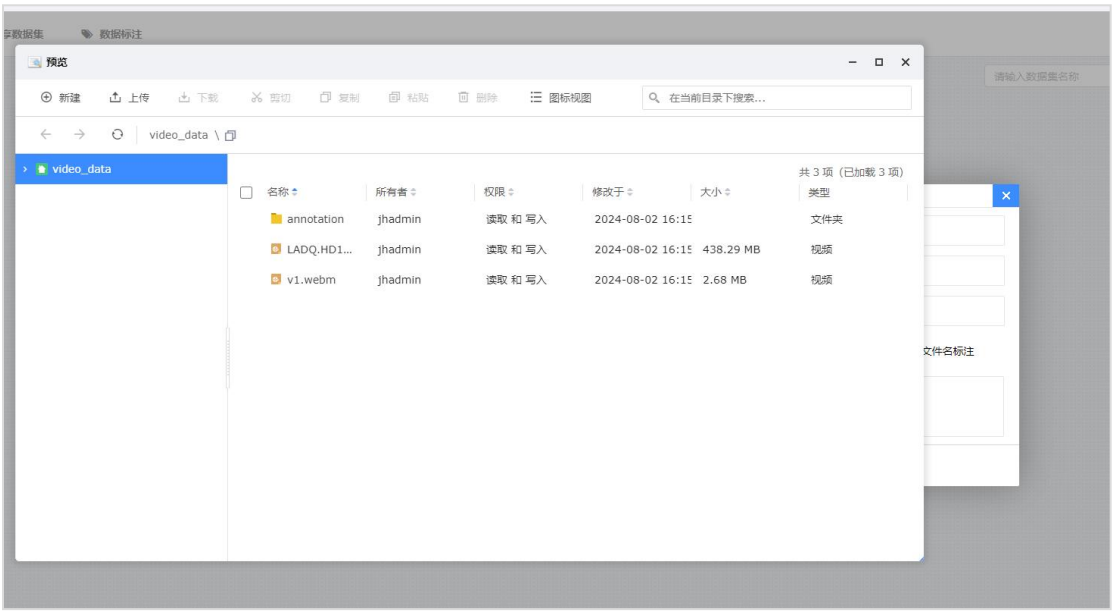
描述：输入该数据集的描述信息，可选。

配置完对应信息后，点击“连接测试”按钮，即可测试 HDFS 服务的连通性，会有成功和失败提示。连接测试效果如下图所示：



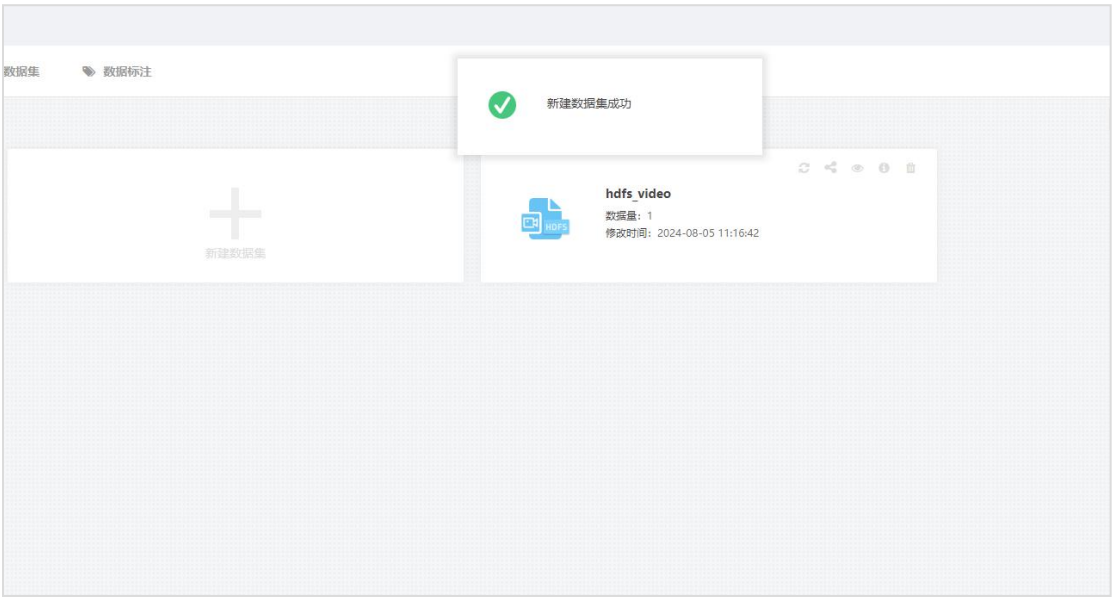
连接测试

点击“预览”按钮，即可查看数据集的数据文件数据。如下图所示：



预览“视频数据-HDFS”数据集

点击“确定”按钮，新建数据集。如下图所示：

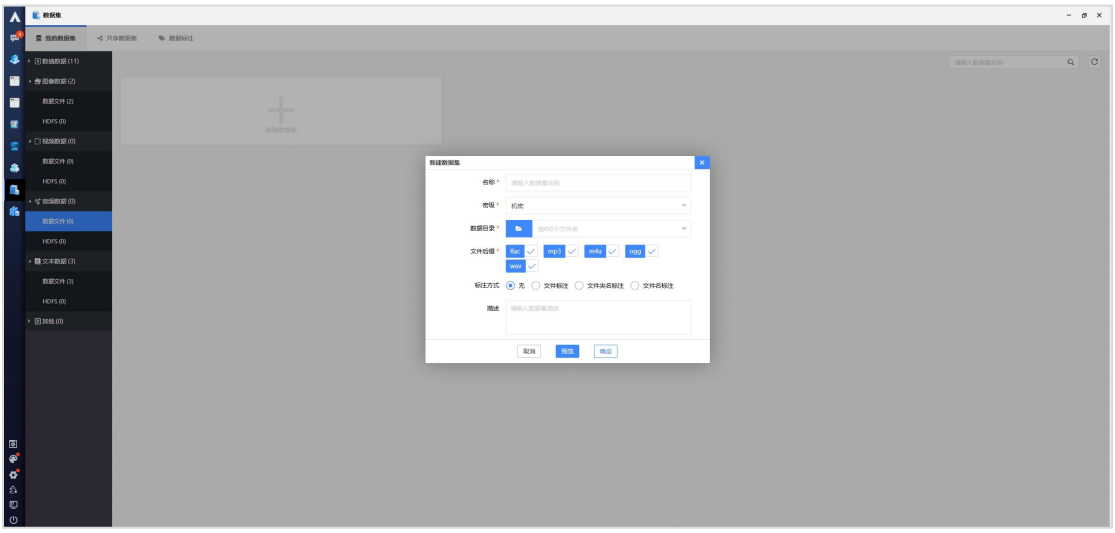


新建“视频数据-HDFS”数据集成功

2.1.4. 音频数据

2.1.4.1. “音频数据-数据文件”数据集

新建“音频数据-数据文件”数据集，操作步骤如下：依次点击“我的数据集”-“音频数据”-“数据文件”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“音频数据-数据文件”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

数据目录：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择数据目录”窗口中，选择音频文件夹即可。

文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 flac、mp3、m4a、ogg 和 wav，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目

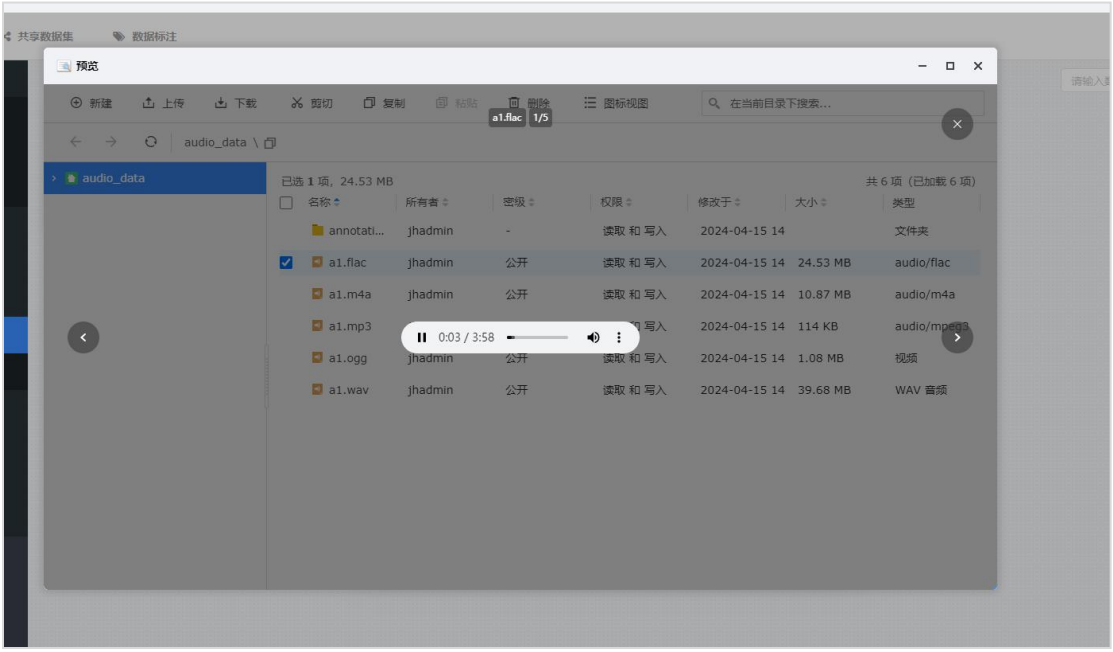
录项对应的标签文件，目前仅支持“json”格式的标签文件；

3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；

4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

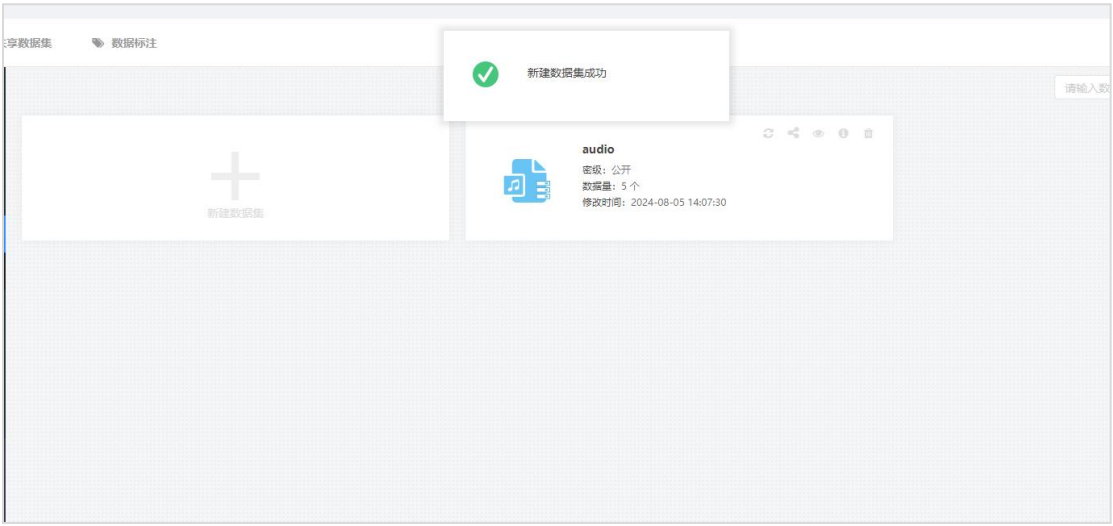
描述：输入该数据集的描述信息，可选。

配置完对应的信息后，点击“预览”按钮，即可查看数据集的数据文件数据。预览效果如下图所示：



预览“音频数据-数据文件”数据集

点击“确定”按钮，新建数据集。如下图所示：

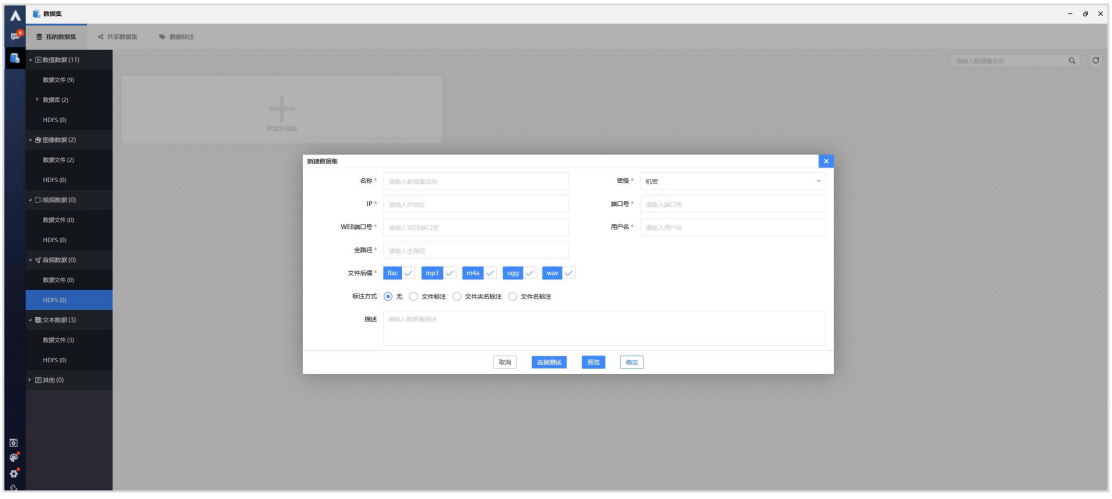


新建“音频数据-数据文件”数据集成功

2.1.4.2. “音频数据-HDFS”数据集

注意：应用 HDFS 功能时，需要管理员关闭密级功能，才能正常使用。

新建“音频数据-HDFS”数据集，操作步骤如下：依次点击“我的数据集”-“音频数据”-“HDFS”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“音频数据-HDFS”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。但是目前不支持在打开密级功能的时候，使用 HDFS 功能。

IP：输入 HDFS 服务节点的 IP。

端口号：HDFS 集群的 RPC 通信端口，即$HADOOP_HOME/etc/hadoop/core-site.xml HDFS 配置文件中的“fs.defaultFS”配置参数的值。该端口一般为：8020 或 9000。

WEB 端口号：HDFS NameNode 的 Http UI 端口，用于方案设计中应用 HDFS 数据集时，调用该端口。需要根据 hadoop 版本，配置 WEB 端口，如：hadoop2 的默认 WEB 端口为 50070，hadoop3 的默认 WEB 端口为 9870。

用户名：输入 HDFS 服务的用户名。

全路径：输入数据目录的绝对路径，系统默认自动拼接“管理门户-系统管理-系统配置-人工智能-人工智能基础配置”配置文件的“HDFS 根路径挂载点”的值。

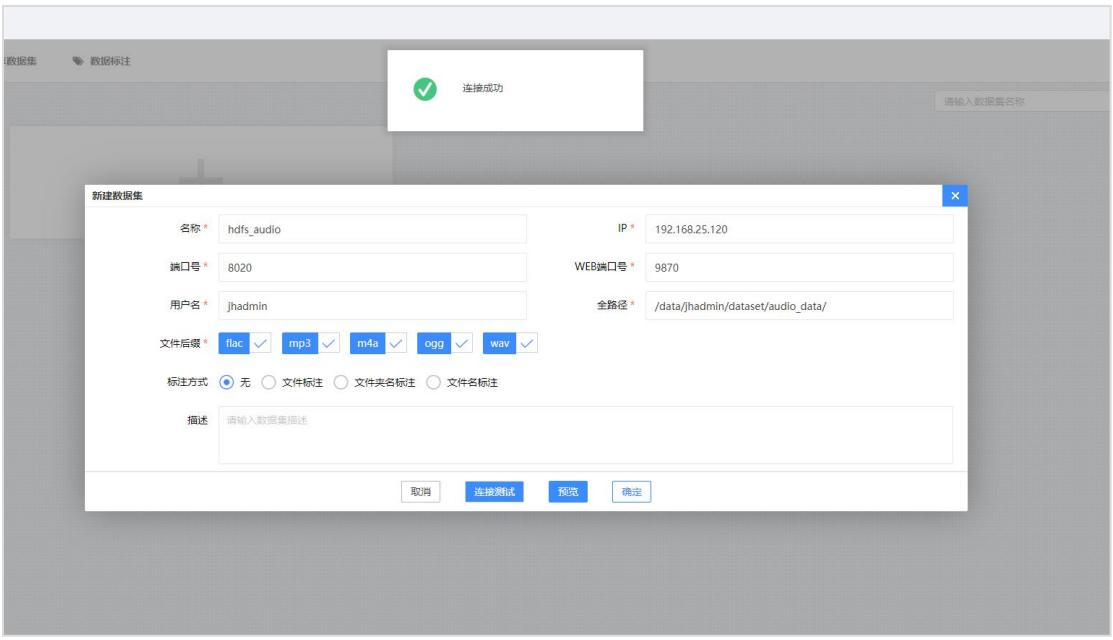
文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 flac、mp3、m4a、ogg 和 wav，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；
3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；
4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

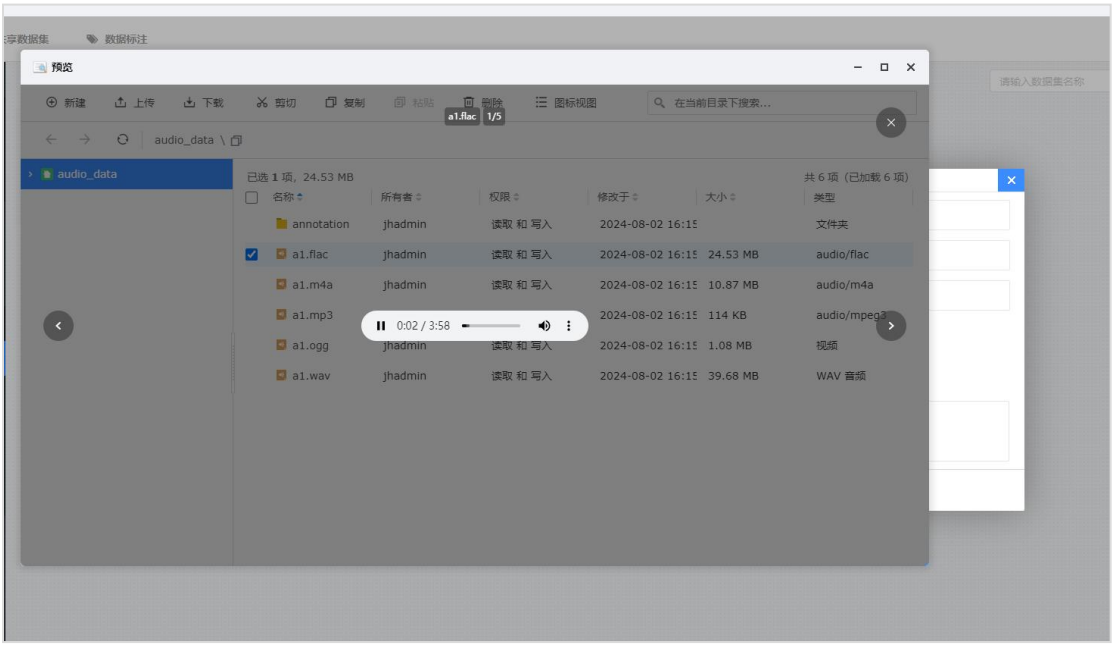
描述：输入该数据集的描述信息，可选。

配置完对应信息后，点击“连接测试”按钮，即可测试 HDFS 服务的连通性，会有成功和失败提示。连接测试效果如下图所示：



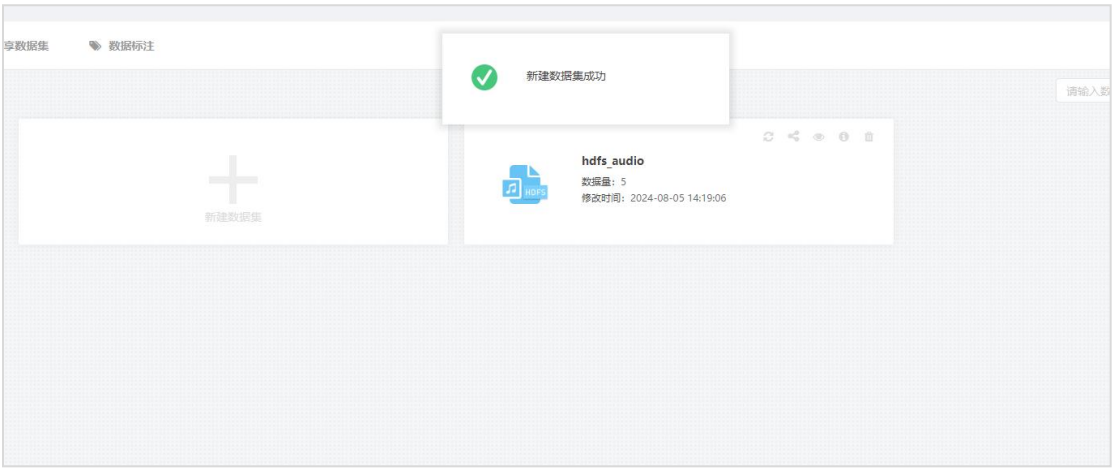
连接测试

点击“预览”按钮，即可查看数据集的数据文件数据。预览效果如下图所示：



预览“音频数据-HDFS”数据集

点击“确定”按钮，新建数据集。如下图所示：

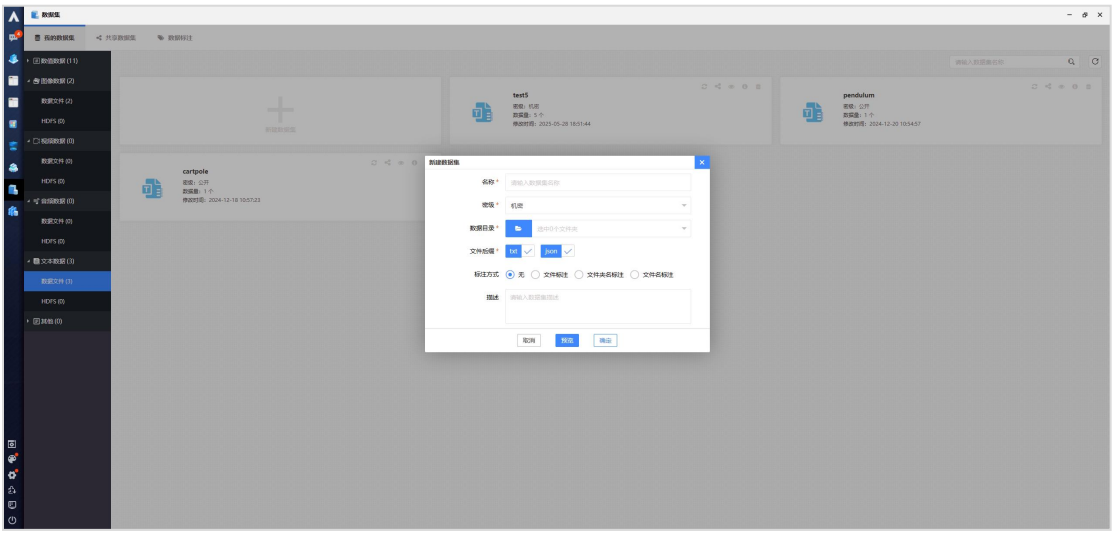


新建“音频数据-HDFS”数据集成功

2.1.5. 文本数据

2.1.5.1. “文本数据-数据文件”数据集

新建“文本数据-数据文件”数据集，操作步骤如下：依次点击“我的数据集”-“文本数据”-“数据文件”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“文本数据-数据文件”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

数据目录：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择数据目录”窗口中，选择文本文件夹即可。

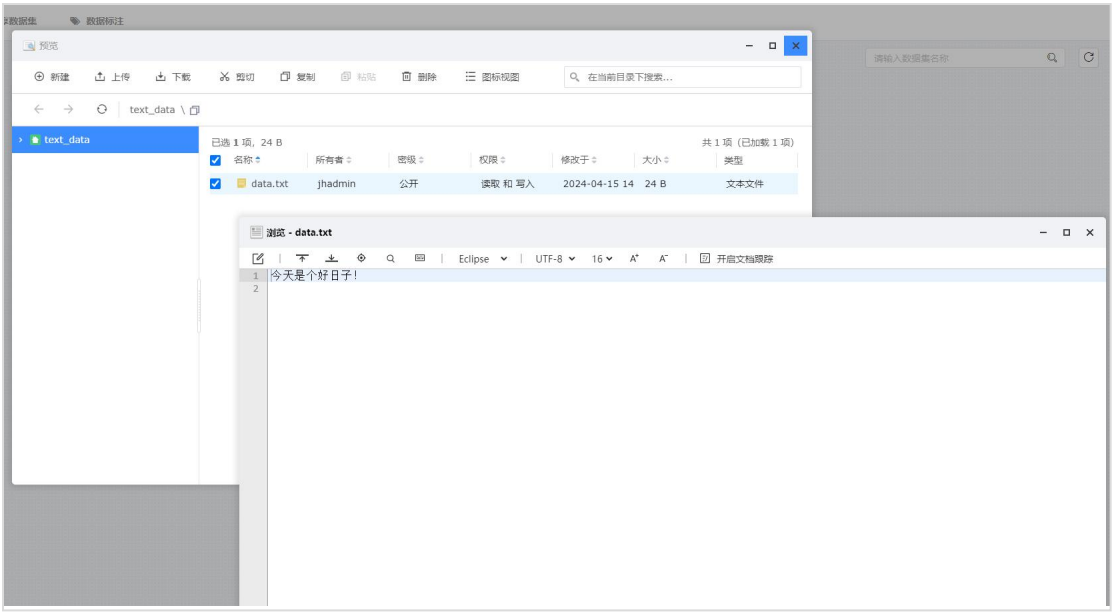
文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 txt 和 json，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；
3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；
4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

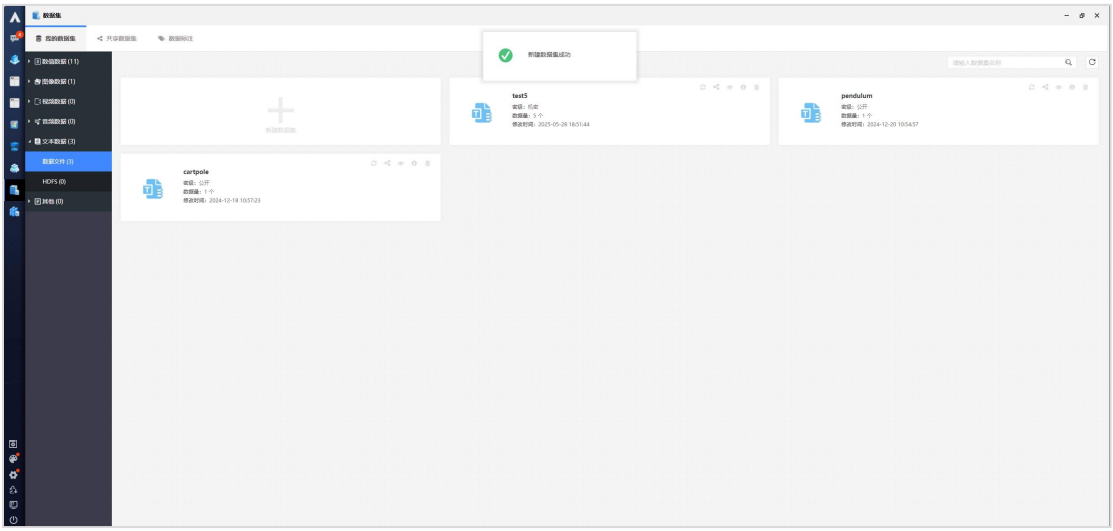
描述：输入该数据集的描述信息，可选。

配置完对应的信息后，点击“预览”按钮，即可查看数据集的数据文件数据。预览效果如下图所示：



预览“文本数据-数据文件”数据集

点击“确定”按钮，新建数据集。如下图所示：



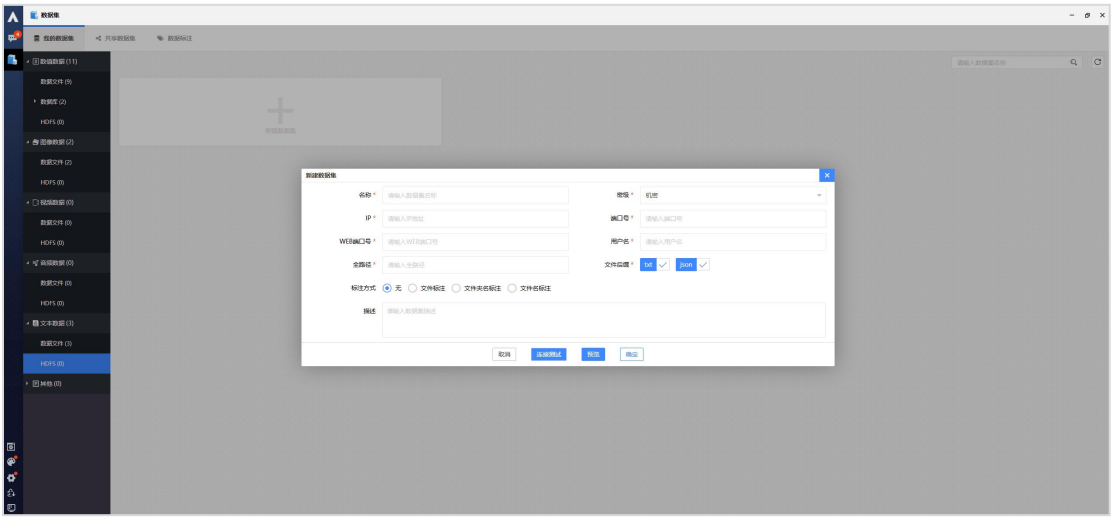
新建“文本数据-数据文件”数据集成功

2.1.5.2. “文本数据-HDFS”数据集

注意：应用 HDFS 功能时，需要管理员关闭密级功能，才能正常使用。

新建“文本数据-HDFS”数据集，操作步骤如下：依次点击“我的数据集”-“文本数据”-“HDFS”分类按钮，然后在右侧的工作区，点击“新建数据集”

卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“文本数据-HDFS”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。但是目前不支持在打开密级功能的时候，使用 HDFS 功能。

IP：输入 HDFS 服务节点的 IP。

端口号：HDFS 集群的 RPC 通信端口，即$HADOOP_HOME/etc/hadoop/core-site.xml HDFS 配置文件中的“fs.defaultFS”配置参数的值。该端口一般为：8020 或 9000。

WEB 端口号：HDFS NameNode 的 Http UI 端口，用于方案设计中应用 HDFS 数据集时，调用该端口。需要根据 hadoop 版本，配置 WEB 端口，如：hadoop2 的默认 WEB 端口为 50070，hadoop3 的默认 WEB 端口为 9870。

用户名：输入 HDFS 服务的用户名。

全路径：输入数据目录的绝对路径，系统默认自动拼接“管理门户-系统管理-系统配置-人工智能-人工智能基础配置”配置文件的“HDFS 根路径挂载点”的值。

文件后缀：用于筛选出需要使用的数据文件，支持的文件后缀类型有 txt 和 json，可单选和全选，至少选择其中一项。

标注方式：有 4 种标注方式，如下所示：

1) “无”：默认值，表示没有使用任何标注方式。用法：可用来当作测试集；

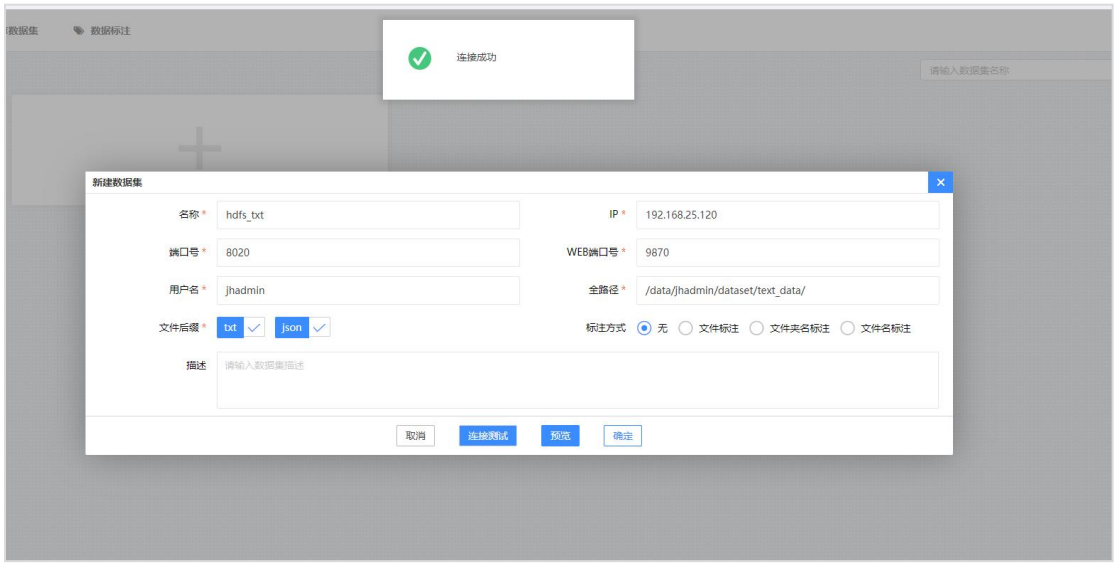
2) “文件标注”：勾选文件标注项，展示标签文件项，选择与数据目录项对应的标签文件，目前仅支持“json”格式的标签文件；

3) “文件夹名标注”：每个文件夹就是一个类别的集合，并且通过文件夹名来标注类别；

4) “文件名标注”：图像的类别通过图像文件名称标识，例如：1-pic.png，即该图像中的内容是 1，此时需要使用“标注信息-文件名”类别的标注方式；

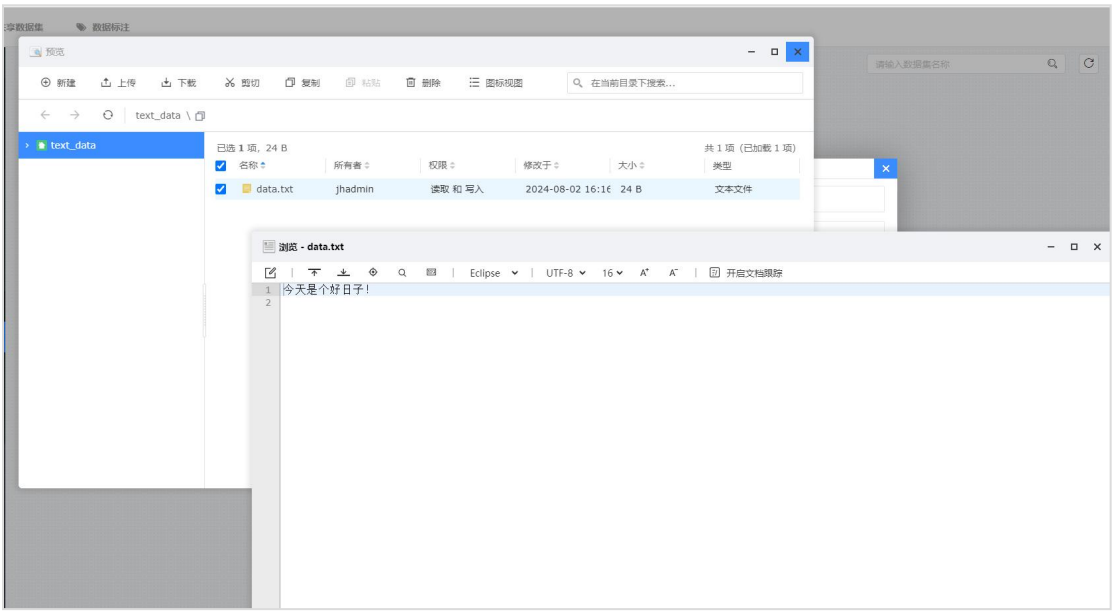
描述：输入该数据集的描述信息，可选。

配置完对应信息后，点击“连接测试”按钮，即可测试 HDFS 服务的连通性，会有成功和失败提示。连接测试效果如下图所示：



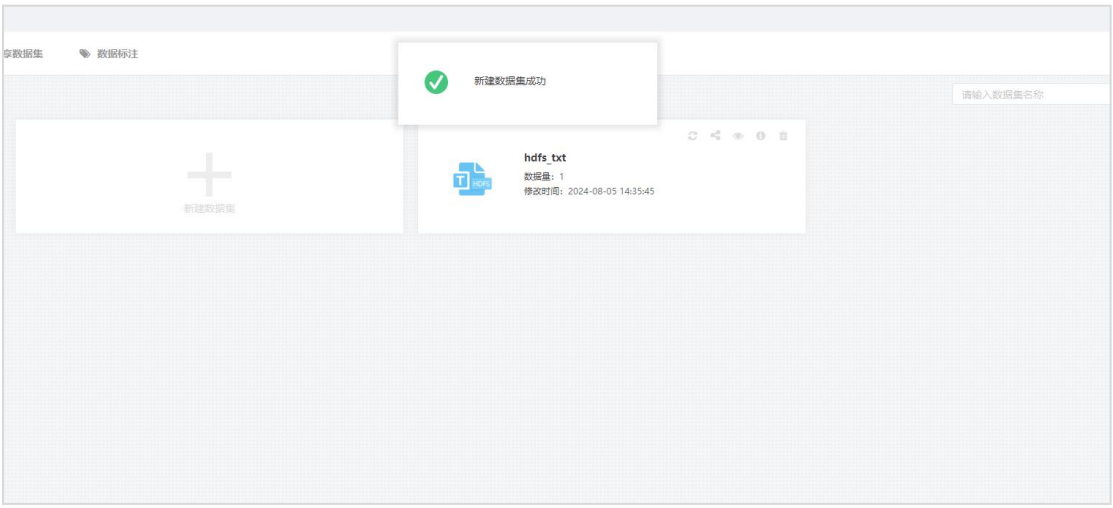
连接测试

点击“预览”按钮，即可查看数据集的数据文件数据。预览效果如下图所示：



预览“文本数据-HDFS”数据集

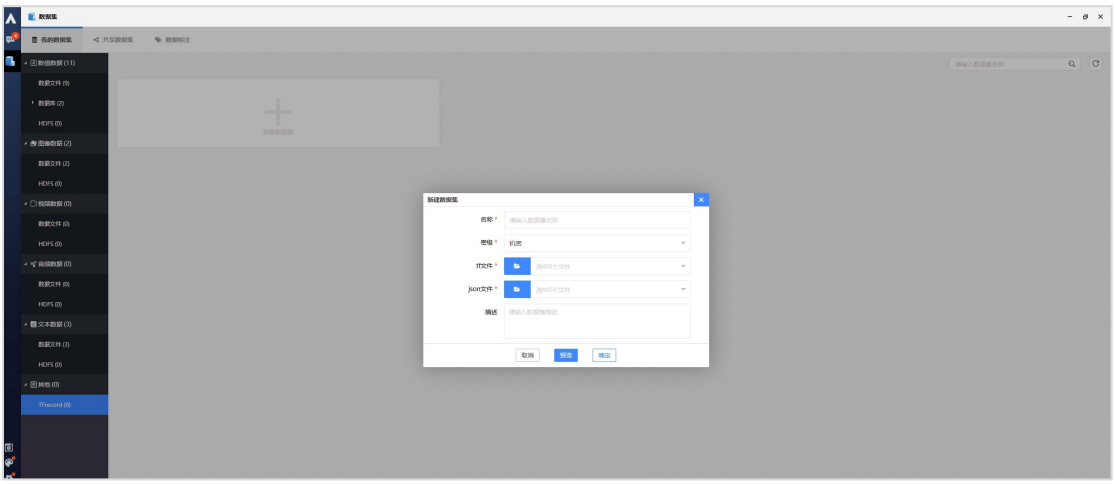
点击“确定”按钮，新建数据集。如下图所示：



新建“文本数据-HDFS”数据集成功

2.1.6. 其他

新建“其他-TFrecord”数据集，操作步骤如下：依次点击“我的数据集”-“其他”-“TFrecord”分类按钮，然后在右侧的工作区，点击“新建数据集”卡片按钮，此时会弹出“新建数据集”窗口，如下图所示：



新建“其他-TFrecord”数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

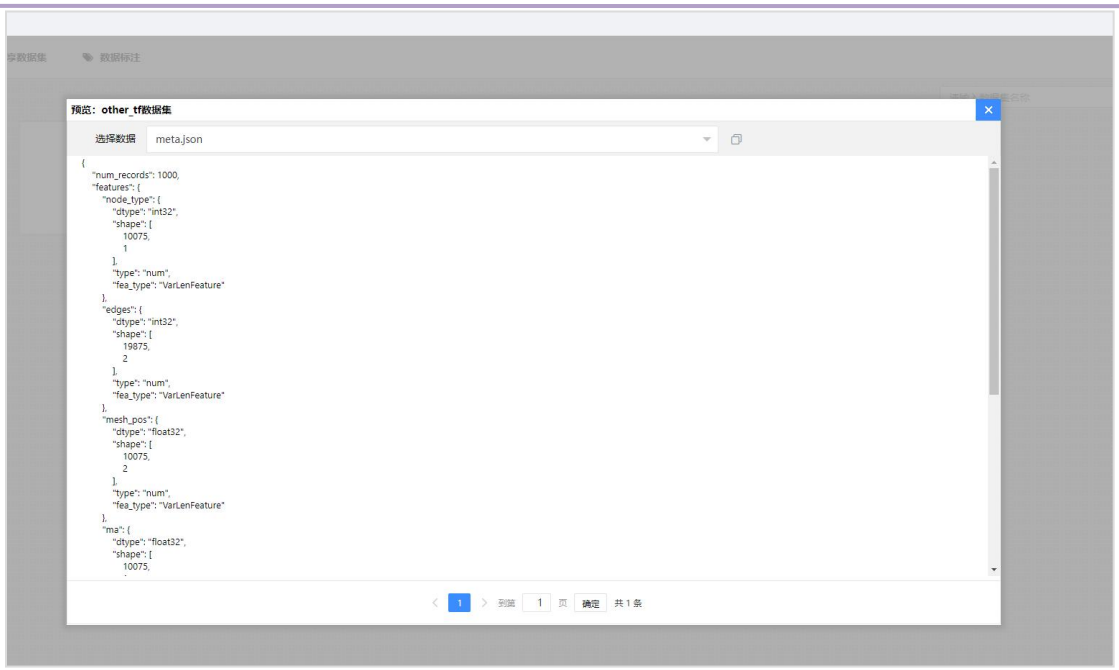
密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

tf 文件：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择 tf 文件”窗口中，选择 tfrecord 文件即可。只允许选择一个 tfrecord 文件。

json 文件：点击蓝色的“文件夹”图标按钮，然后在弹出的“选择 json 文件”窗口中，选择 json 文件即可。只允许选择一个 json 文件。

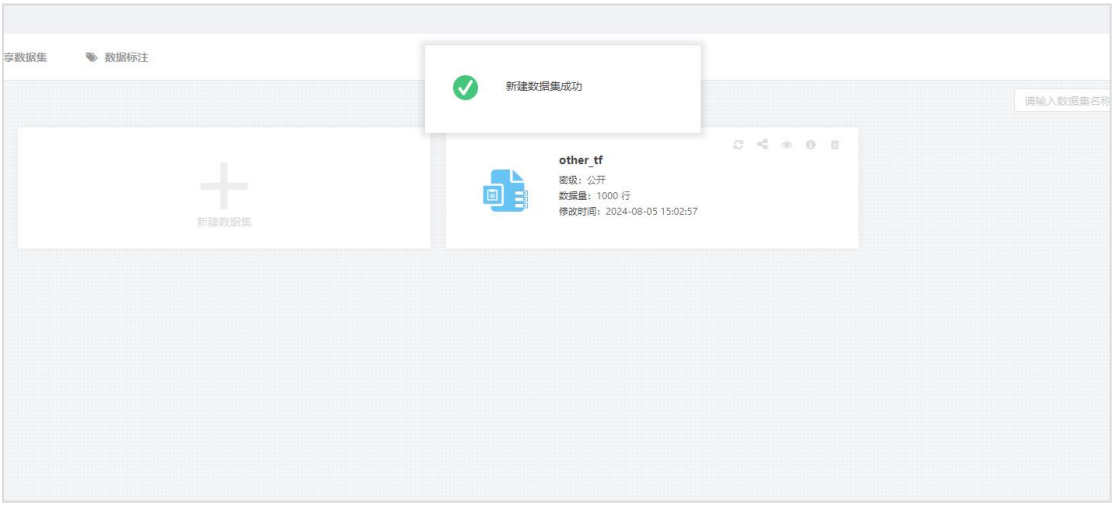
描述：输入该数据集的描述信息，可选。

配置完对应的信息后，点击“预览”按钮，即可查看数据集的数据文件数据。如下图所示：



预览“其他-TFrecord”数据集

点击“确定”按钮，新建数据集。如下图所示：



新建“其他-TFrecord”数据集成功

2.1.7. 共享数据集

共享数据集的产生有 2 种方式：

- 在我的数据集中，将我的数据集共享至共享数据集；
- 在共享数据集中，应用共享组中的数据文件，新建共享数据集；

各种角色对共享数据集的操作权限：

共享者：对共享数据集有修改、预览、使用和删除的权限；

共享组用户：对共享数据集有预览、使用的权限；

管理员：对共享数据集仅有删除权限；

其他用户：看不到共享数据集；

用途：共享组成员可以在方案设计和 AI 作业中使用共享数据集。

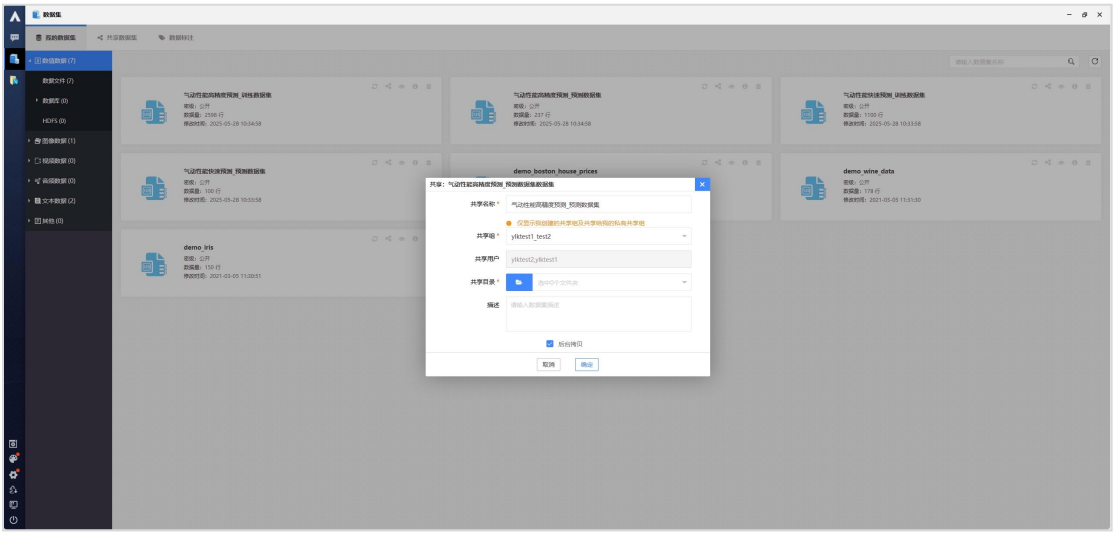
注意：进行共享操作前，需要先创建共享组。

2.1.7.1. 将我的数据集共享至共享数据集

共享“数据文件”类型的数据集，如下所示：

- “数值数据-数据文件”数据集
- “图像数据-数据文件”数据集
- “视频数据-数据文件”数据集
- “音频数据-数据文件”数据集
- “文本数据-数据文件”数据集
- “其他-TFrecord”数据集

以共享“数值数据-数据文件”数据集为例，步骤如下：依次点击“我的数据集”-“数值数据”-“数据文件”分类按钮，然后在右侧的工作区，点击数据集卡片上的“共享”图标按钮，此时会弹出“共享”窗口，如下图所示：



共享“数值数据-数据文件”数据集

图中每个参数的具体含义如下：

共享名称：共享后的数据集名称，输入任意符合命名规则的名称即可。

共享组：选择可见的共享组，选项从我的数据->共享数据区获取。

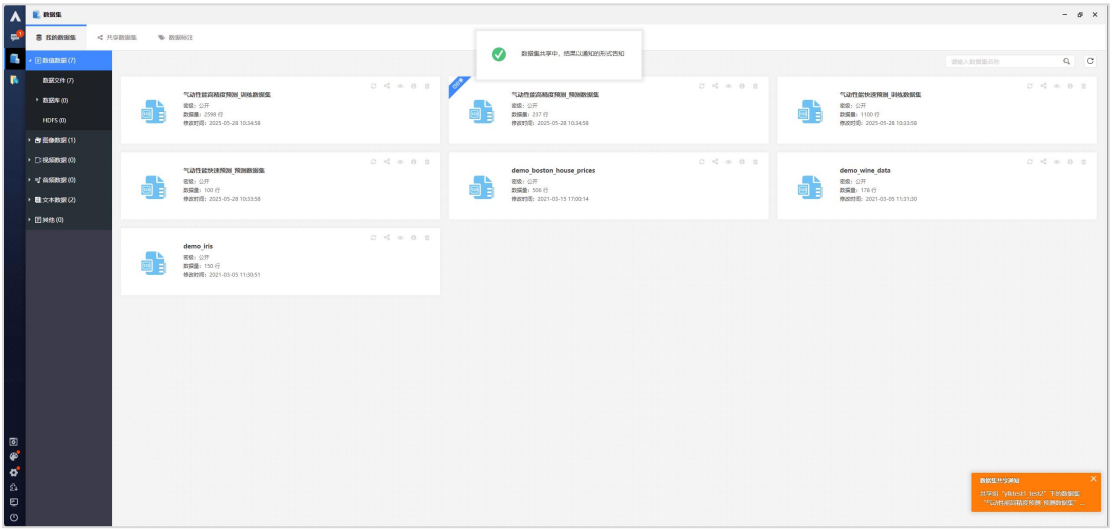
共享用户：选择共享组之后显示，显示为选择的共享组成员。

共享目录：选择共享组之后显示，只能在所选共享组的共享数据区选择文件夹，点击蓝色的“文件夹”图标按钮，然后在弹出的“选择共享目录”窗口中选择文件夹即可，用作存放源数据文件目录，只允许选择一个文件夹。

描述：共享后的数据集描述信息，可选。

后台拷贝：默认勾选，采用后台形式将数据集的数据文件拷贝至共享目录，当前共享页面退出；不勾选则采用前台拷贝方式，当前共享页面直至共享结束退出。

配置完成后，点击“确定”按钮，进行数据集共享，如下图所示：

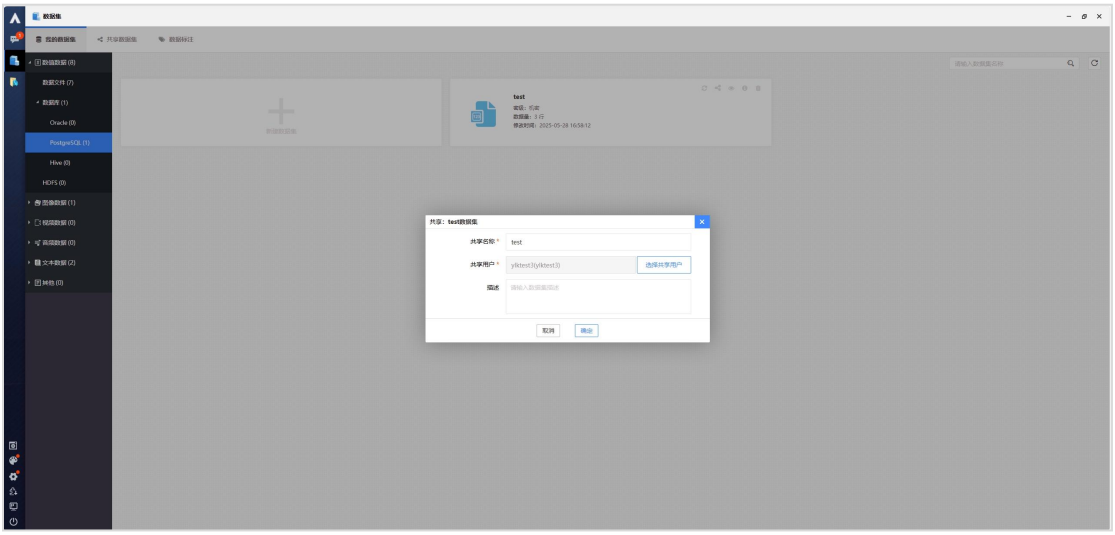


共享数据集

共享非“数据文件”类的数据集，如下所示：

- “数值数据-数据库”数据集
- “数值数据-HDFS”数据集
- “图像数据-HDFS”数据集
- “视频数据-HDFS”数据集
- “音频数据-HDFS”数据集
- “文本数据-HDFS”数据集

以共享“数值数据-数据库-PostgreSQL”数据集为例，操作步骤如下：依次点击“我的数据集”-“数值数据”-“数据库”-“PostgreSQL”分类按钮，然后在右侧的工作区，点击数据集卡片上的“共享”按钮，此时会弹出“共享”窗口，如下图所示：



共享“数值数据-数据库-PostgreSQL”数据集

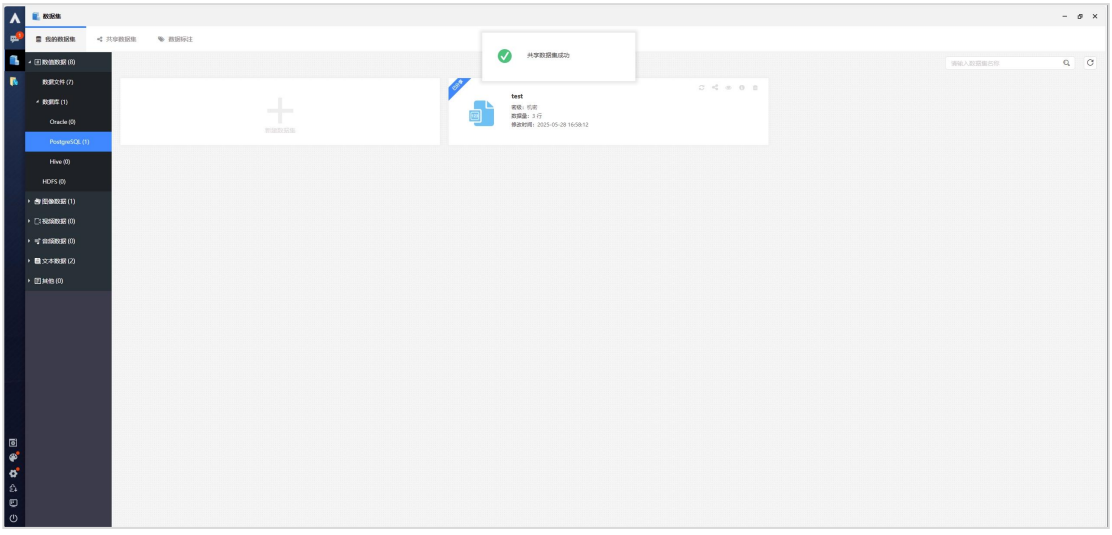
图中每个参数的具体含义如下：

共享名称：共享后的数据集名称，输入任意符合命名规则的名称即可。

共享用户：选择共享用户。

描述：共享后的数据集描述信息，可选。

配置完成后，点击“确定”按钮，共享数据集，如下图所示：



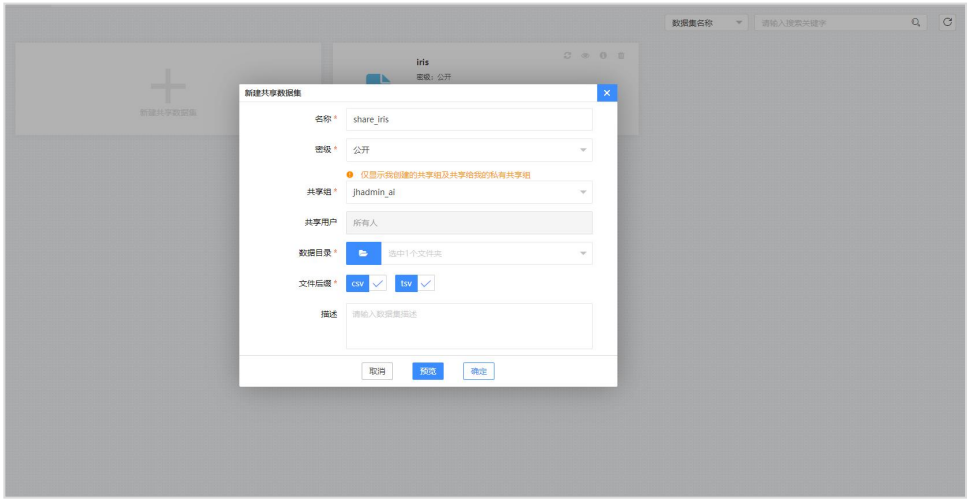
共享数据集

2.1.7.2. 新建共享数据集

新建数据文件类的共享数据集，如下所示：

- “数值数据-数据文件”数据集
- “图像数据-数据文件”数据集
- “视频数据-数据文件”数据集
- “音频数据-数据文件”数据集
- “文本数据-数据文件”数据集
- “其他-TFrecord”数据集

以新建“数值数据-数据文件”共享数据集为例，操作步骤如下：依次点击“共享数据集”-“数值数据”-“数据文件”分类按钮，然后在右侧的工作区，点击“新建共享数据集”卡片按钮，此时会弹出“新建共享数据集”窗口，如下图所示：



新建“数值数据-数据文件”共享数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

共享组：选择可见的共享组，选项从我的数据->共享数据区获取。

共享用户：选择共享组之后显示，显示为选择的共享组成员。

数据目录：选择共享组之后显示，只能在所选共享组的共享数据区选择数据目录，同我的数据集中数值数据->数据文件。

描述：输入该数据集的描述信息，可选。

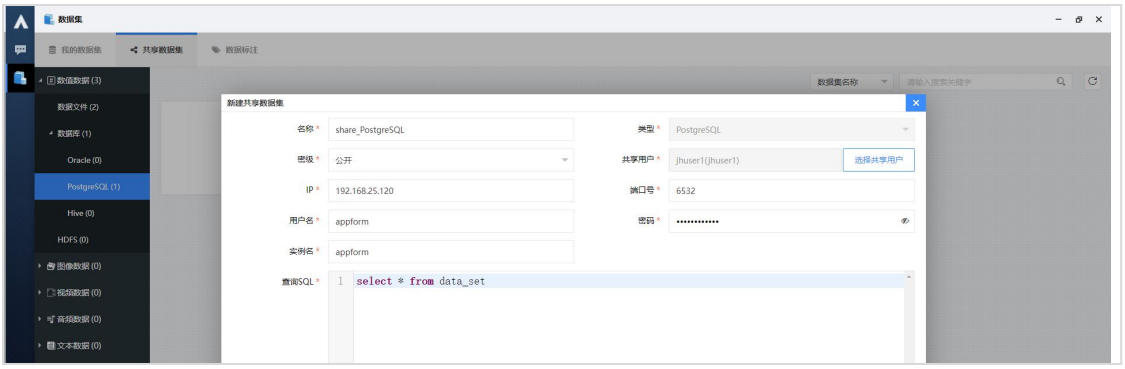
配置完成后，点击“确定”按钮，新建共享数据集。如下图所示：

新建共享数据集

共享非“数据文件”类的数据集，如下所示：

- “数值数据-数据库”数据集
- “数值数据-HDFS”数据集
- “图像数据-HDFS”数据集
- “视频数据-HDFS”数据集
- “音频数据-HDFS”数据集
- “文本数据-HDFS”数据集

以新建“数值数据-数据库-PostgreSQL”共享数据集为例，操作步骤如下：依次点击“共享数据集”-“数值数据”-“数据库”-“PostgreSQL”分类按钮，然后在右侧的工作区，点击“新建共享数据集”卡片按钮，此时会弹出“新建共享数据集”窗口，如下图所示：



新建“数值数据-数据库-PostgreSQL”共享数据集

图中每个参数的具体含义如下：

名称：数据集的名称，输入任意符合命名规则的名称即可。

类型：选择数据库的类型。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

共享用户：选择共享组之后显示，显示为选择的共享组成员。

IP：输入数据库所在机器的 IP。

端口号：输入数据库使用的端口号。

用户名：输入数据库的用户名。

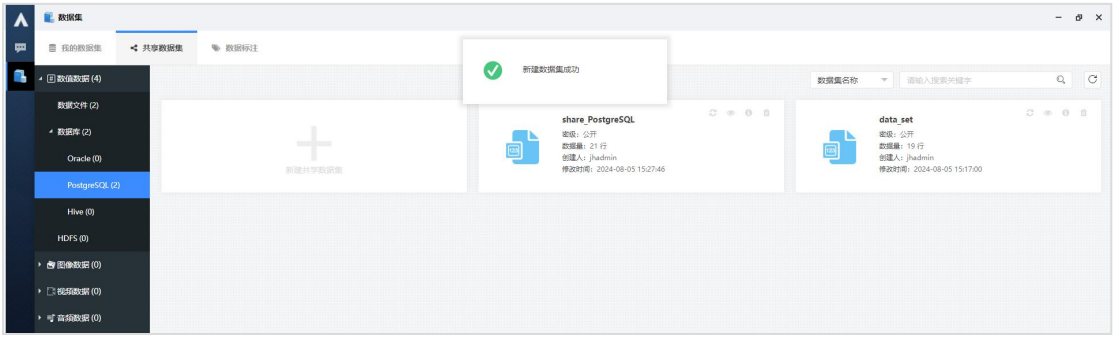
密码：输入数据库用户对应的密码。

实例名：输入数据库名。

查询 SQL：输入 SQL 查询语句（注意：不要在查询语句后面加分号，eg：select * from tbl）。

描述：输入该数据集的描述信息，可选。

配置完成后，点击“确定”按钮，新建共享数据集。如下图所示：



新建共享数据集

2.1.8. 数据标注

数据标注功能为用户提供了灵活、可扩展的在线数据标注工具。以下是数据标注的主要功能：

多数据类型支持：支持多种数据类型的标注，包括图像、文本、音频、视频等，使其适用于各种人工智能任务。

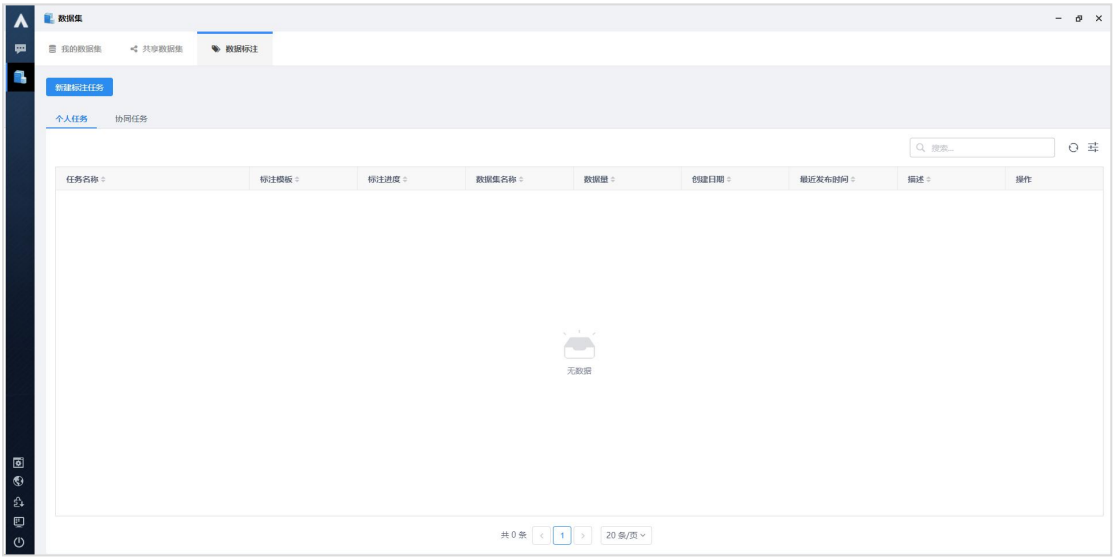
灵活的标注工具：提供丰富的标注工具，包括矩形框、多边形、点标注、文本标注等，以满足不同任务的标注需求。

可视化界面：具备直观的可视化界面，使标注过程更加用户友好，同时支持标注者在标注过程中实时预览标注效果。

自定义标签：用户可以定义和配置标签集，以适应不同项目的特定标注任务，实现个性化的标注需求。

协作与团队功能：提供协作功能，支持多人团队在同一数据集上进行标注工作。

标注格式导出：支持将标注后的数据以各种标准格式导出。



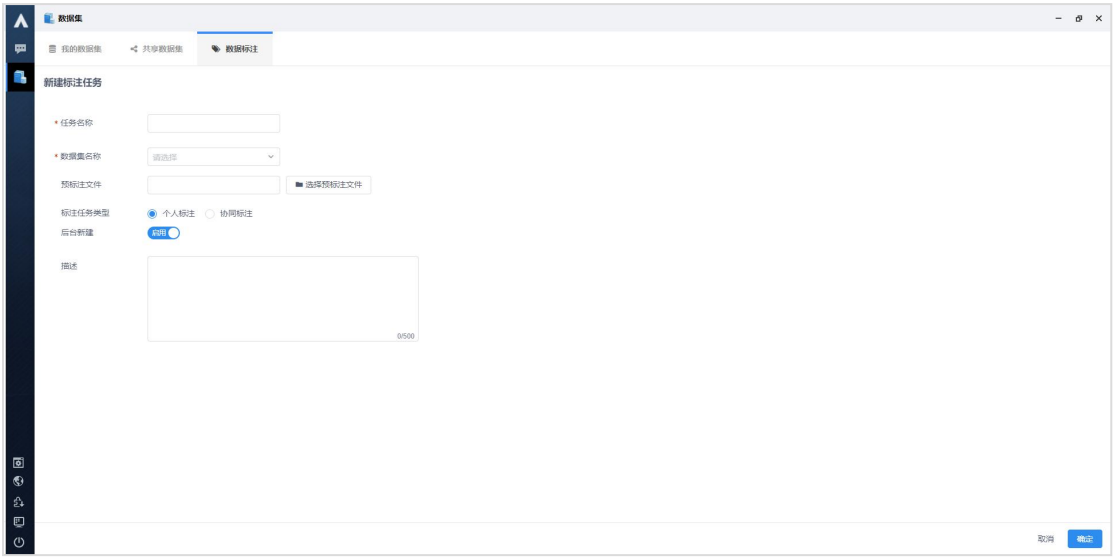
数据标注

2.1.8.1. 标注任务管理

2.1.8.1.1. 新建

您可以点击数据标注页面顶部的“新建标注任务”按钮，进入新建标注任务的页面。根据实际需要完成此页面的表单后，点击“确定”按钮即可完成标注任务的创建。

注意：“数值数据”和“其他”类型的数据集不支持标注。



新建标注任务

图中每个参数的具体含义如下：

任务名称：标注任务的名称，输入任意符合命名规则的名称即可。

数据集名称：选择已有数据集。

预标注文件：选择该数据集数据的标注文件，主要用于对已标注的标注数据，可持续标注。

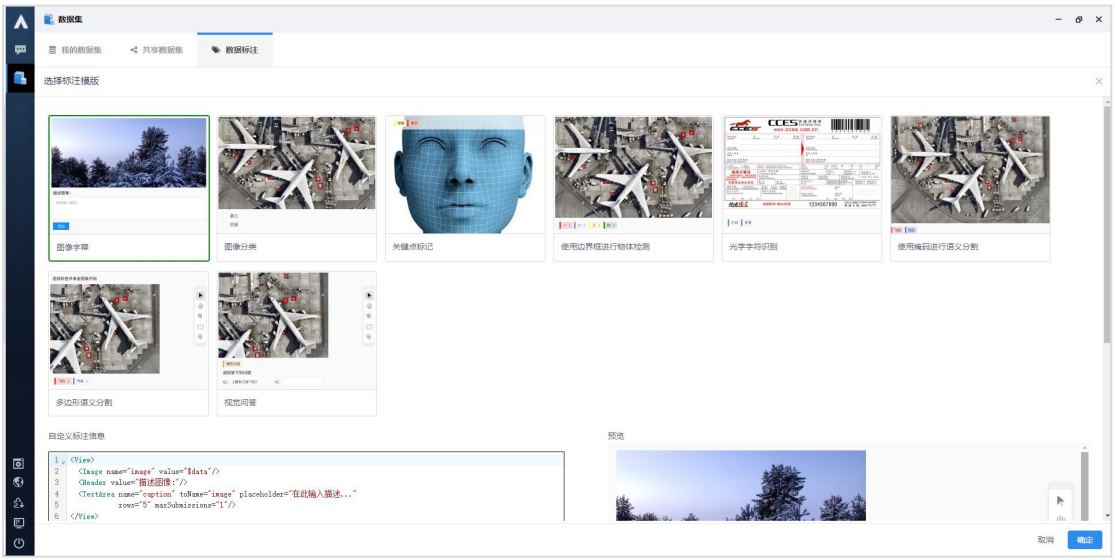
标注模板：根据当前选择数据集类型选择不同场景下的标注模板。

标注任务类型：可根据需要选择“个人标注”或“协同标注”。

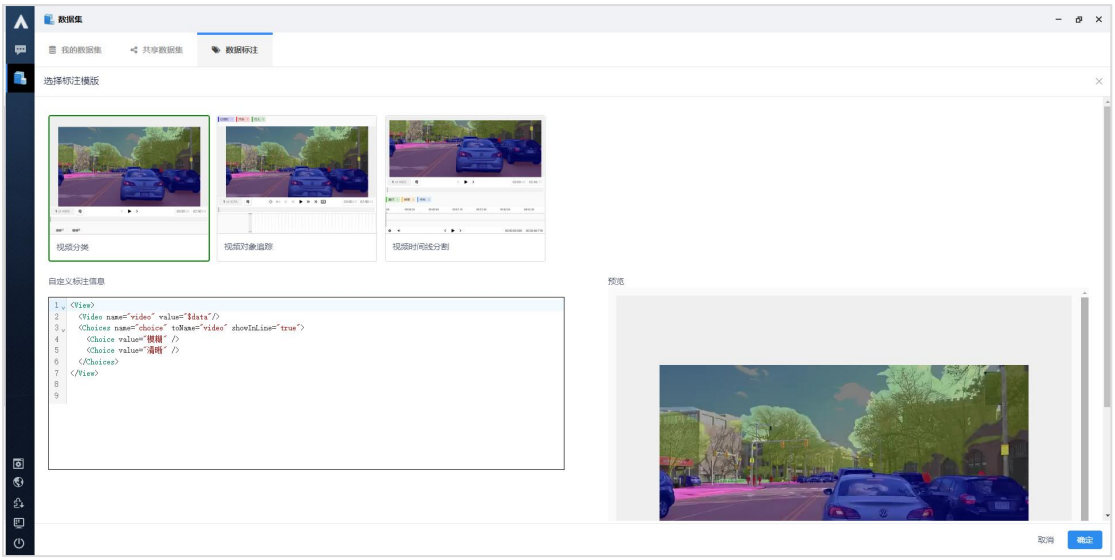
后台新建：默认开启，采用后台形式新建标注任务；新建成功后，通过右下角的气泡信息通知用户；关闭后则采用前台形式拷贝方式。

描述：输入该标注任务的描述信息，可选。

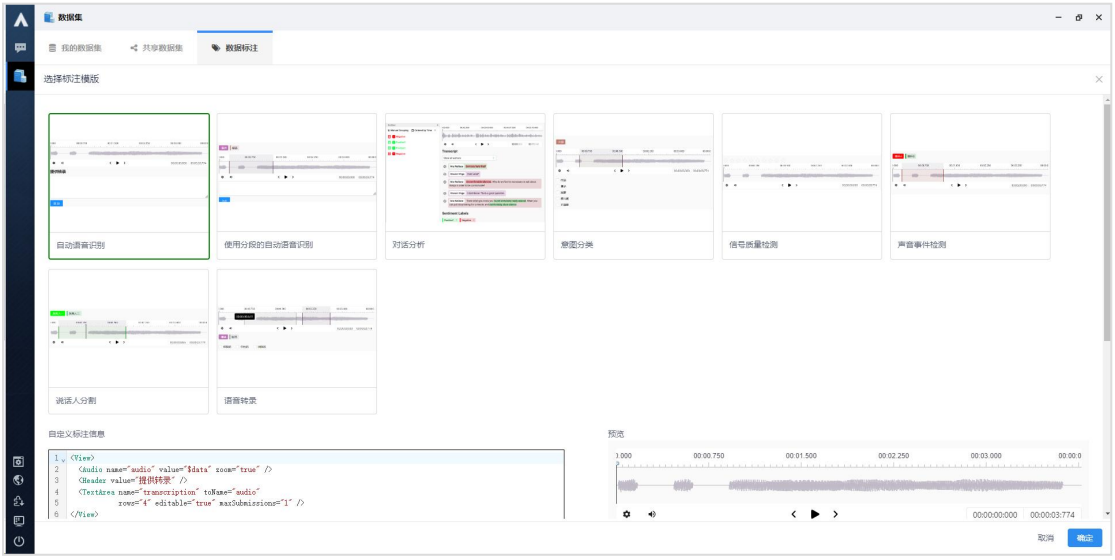
标注模板页面如下所示：



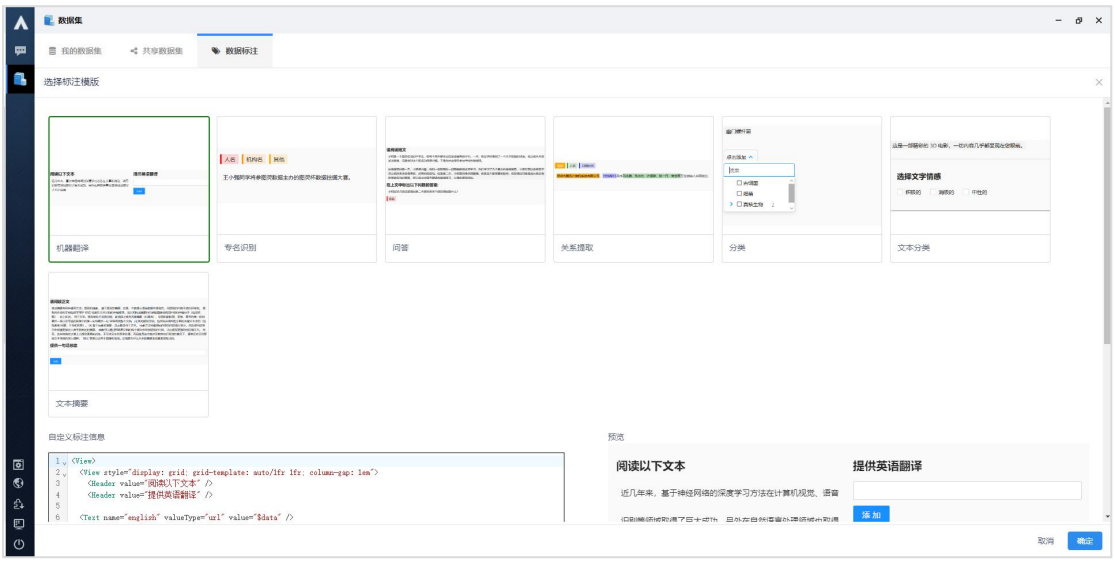
图像标注模板



视频标注模板



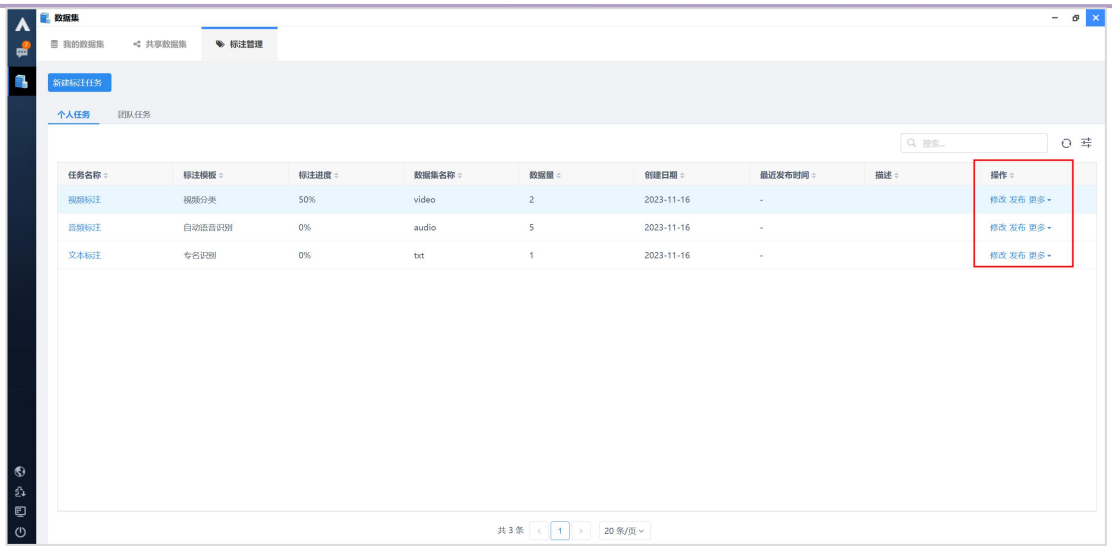
音频标注模板



文本标注模板

2.1.8.1.2. 修改、删除

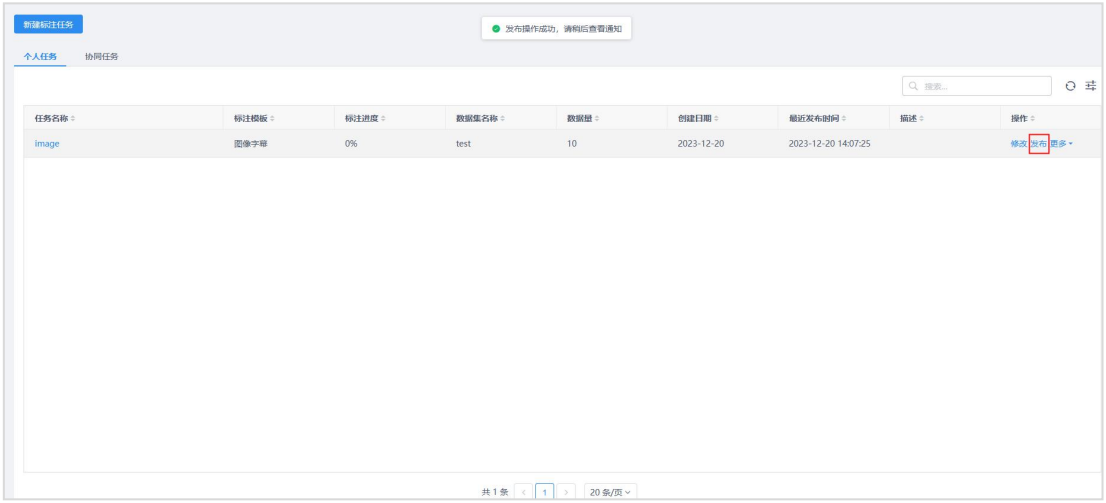
您可以对标注任务执行以下操作：修改、发布、导出和删除。其中，修改操作仅限于修改预标注文件、标注模板以及描述信息。导出操作是将标注数据导出至指定目录。发布操作是将标注数据发布至与标注数据同级的目录。最后，删除操作将彻底删除标注任务。



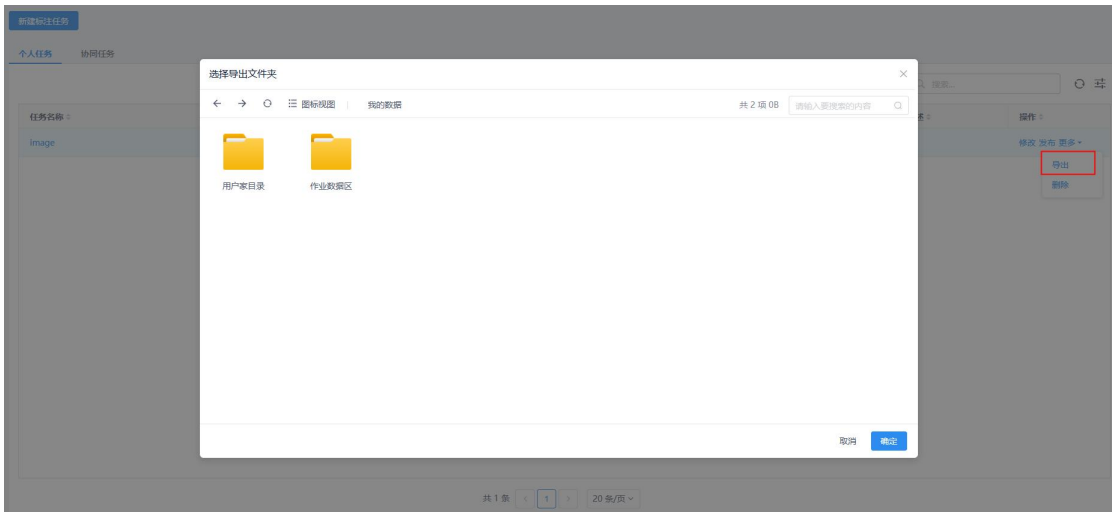
标注任务的操作

2.1.8.1.3. 发布、导出

对创建完成的标注任务进行发布和导出。



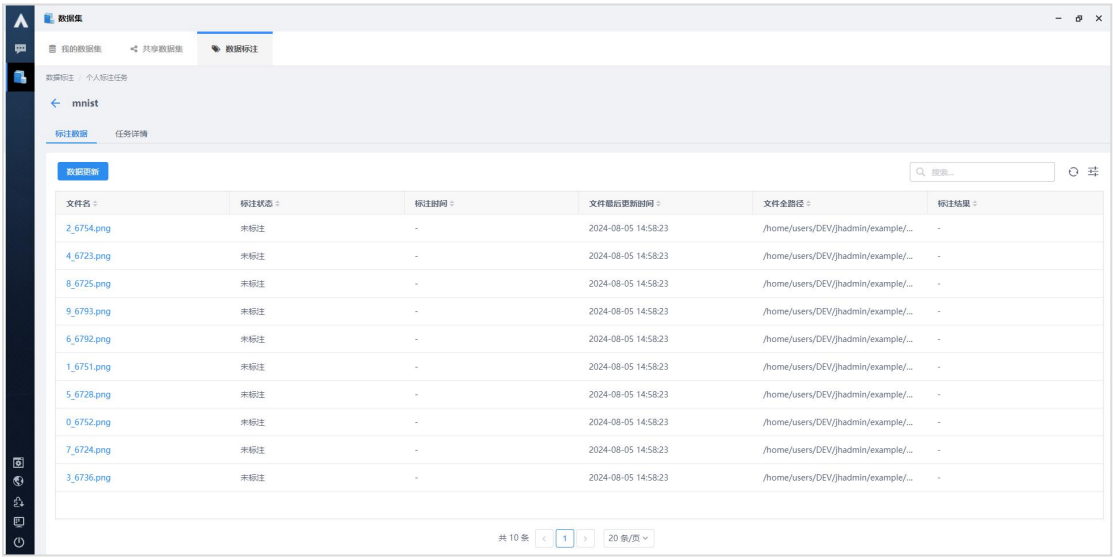
发布标注任务



导出标注任务

2.1.8.1.4. 任务数据

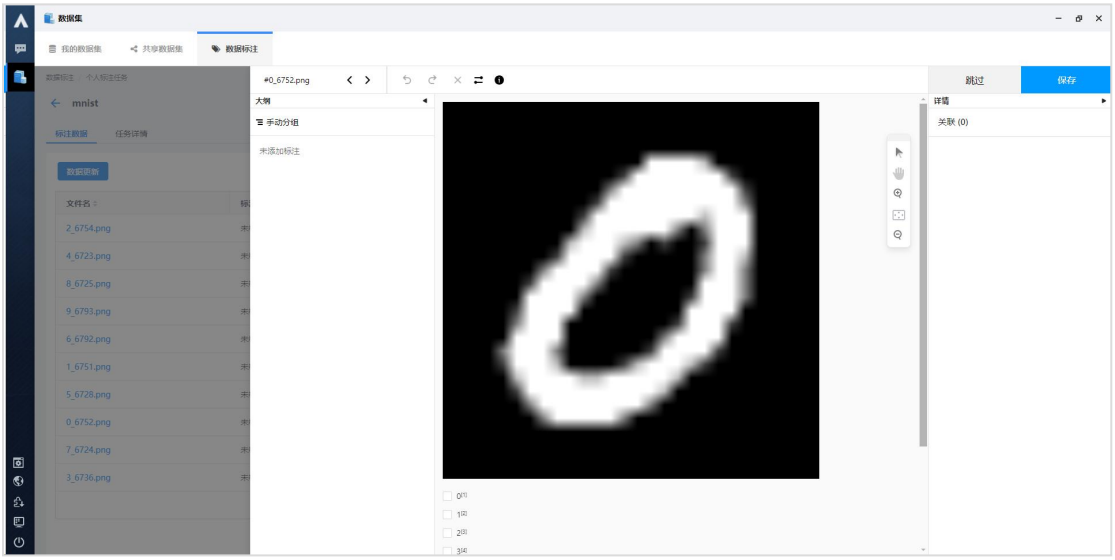
通过点击任务列表中任务名称，可以进入任务数据列表。



任务数据列表

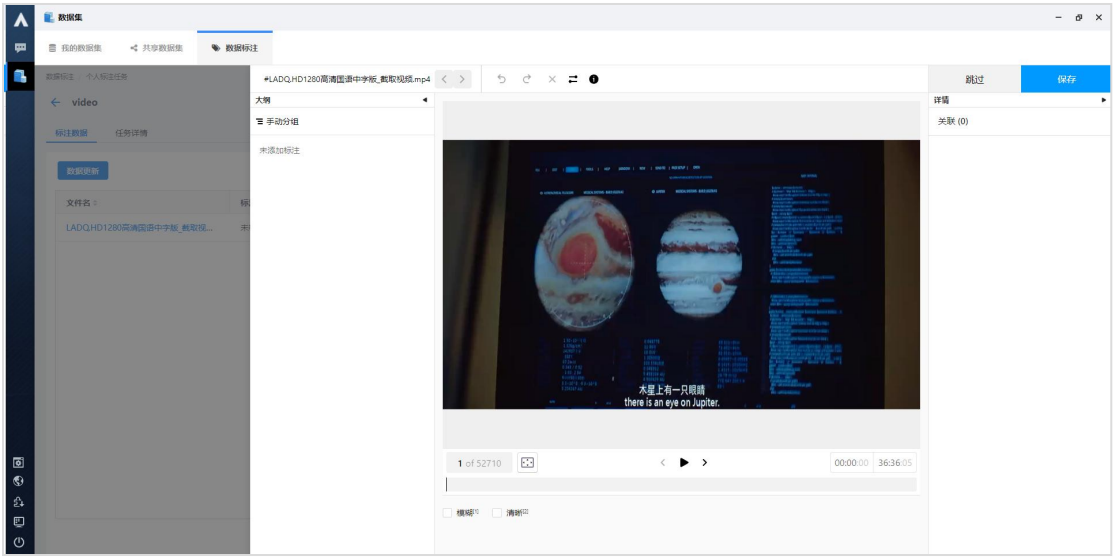
2.1.8.2. 标注数据类型

2.1.8.2.1. 图像标注



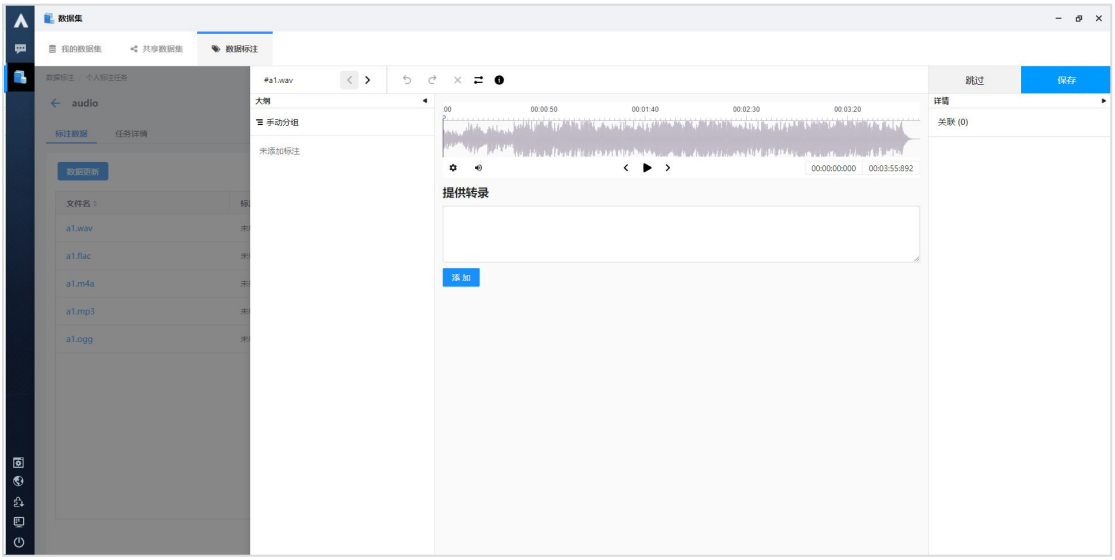
图像标注

2.1.8.2.2. 视频标注



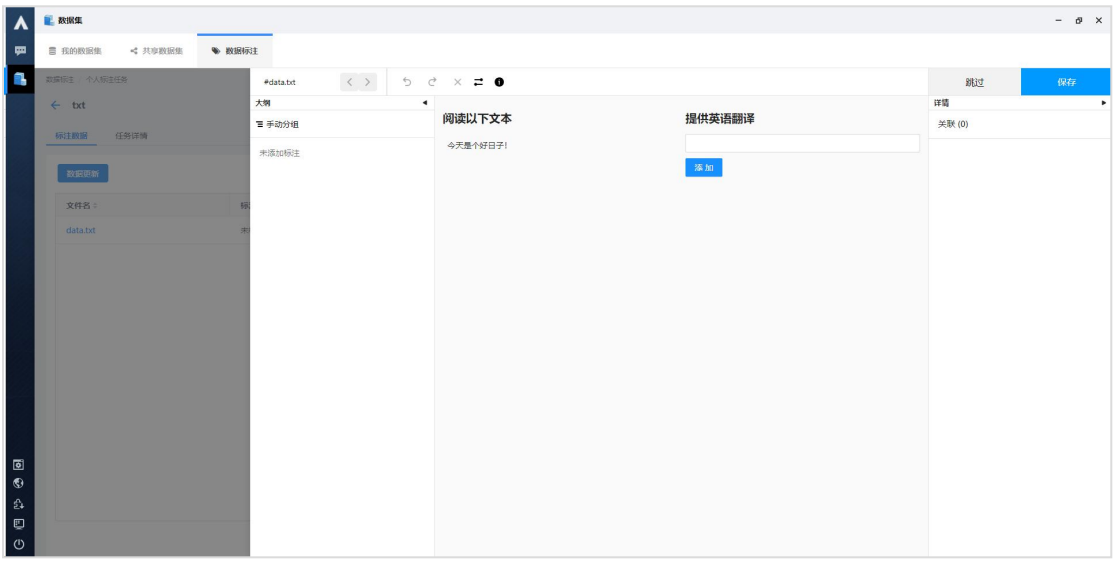
视频标注

2.1.8.2.3. 音频标注



音频标注

2.1.8.2.4. 文本标注



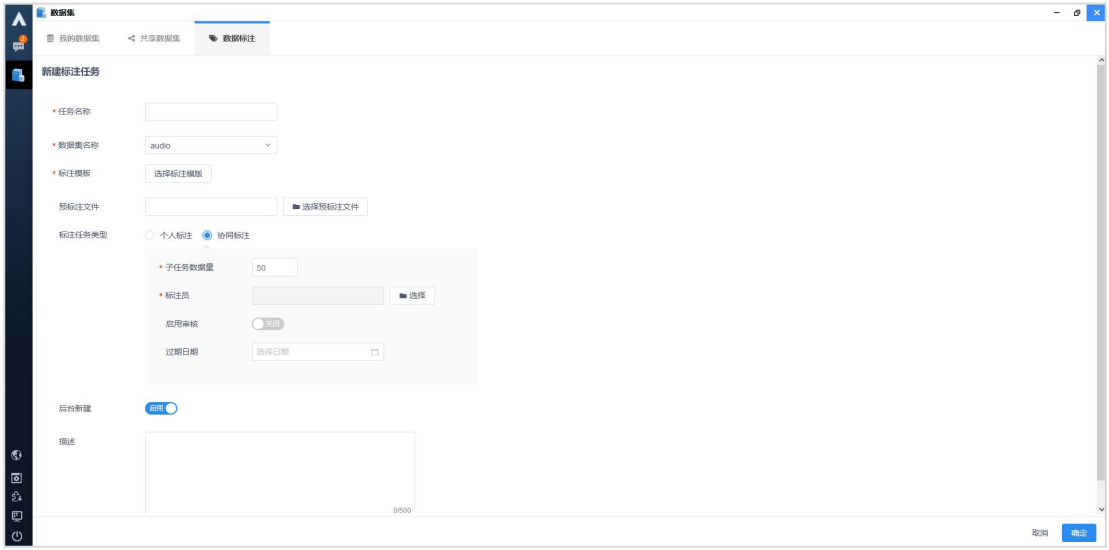
文本标注

2.1.8.3. 协同标注

对于一些大的数据集一个人无法完成所有标注，需要寻找其他人协助自己完成整个数据集的标注工作，此时需要创建协同标注任务，同时也可以设置质量审核人员来保证标注结果的质量。

2.1.8.3.1. 新建

您可以点击数据标注页面顶部的“新建标注任务”按钮，进入新建标注任务的页面。切换标注任务类型为“协同标注”。



创建协同标注任务

协同标注参数具体含义如下：

子任务数据量：协同任务会按照设置的子任务包数量进行拆分数据集，每个子任务在标注过程中独立操作。

标注员：标注任务可以派发的人员。

启用审核：是否开启标注审核这一流程，开启后标注员提交任务到审核人员，审核人员对标注结果进行检查。

审核员：对标注任务进行审核的人员。

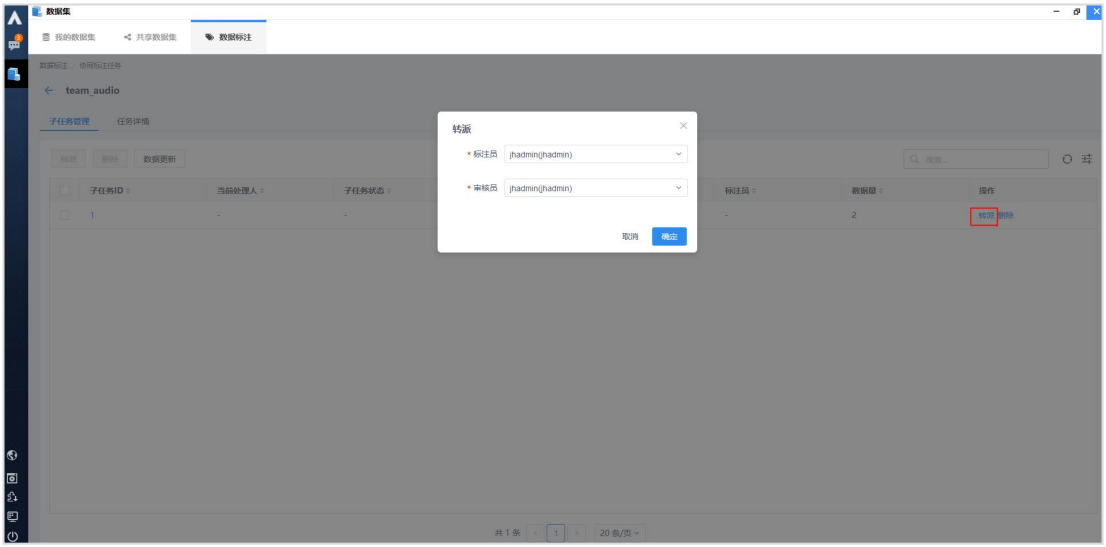
审核策略：支持两种形式，按百分比和按数量，标注员提交子任务后按此规

则抽取待检查的数据。

过期日期：整个标注任务的完成期限设置。

2.1.8.3.2. 转派任务

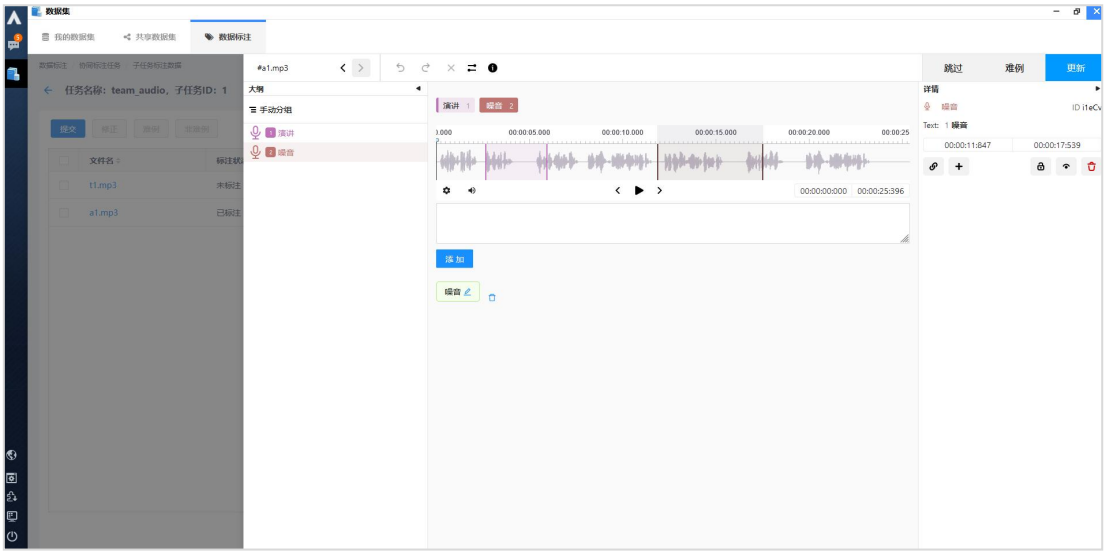
您可以对创建完成的协同标注任务进行转派操作，即将拆分后的若干个子任务分配到具体的标注人员。通过点击任务列表中任务名称，进入子任务列表，点击子任务的转派按钮即可分配到具体标注人员。



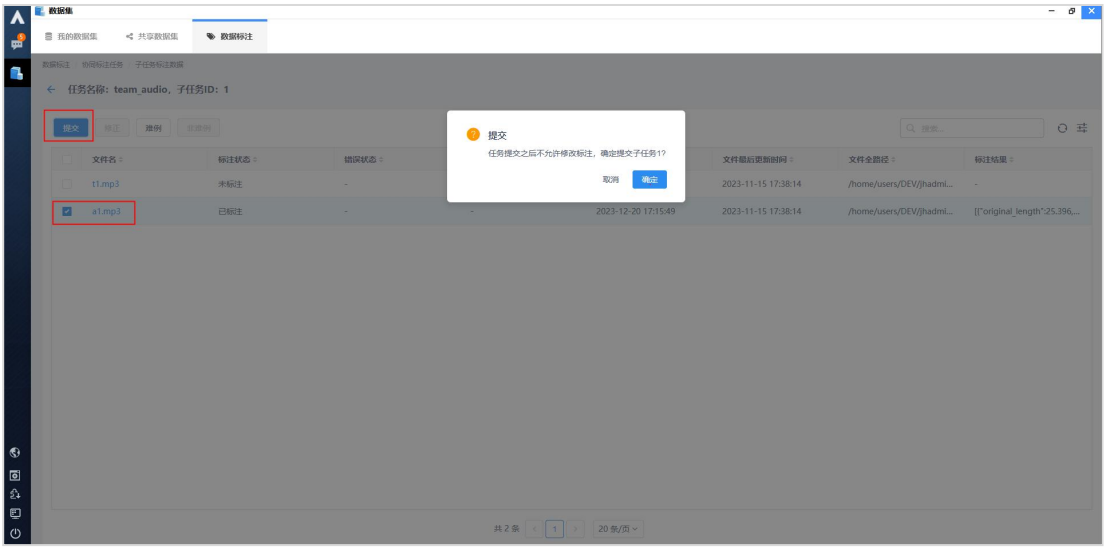
转派任务

2.1.8.3.3. 提交任务

当标注子任务被转派给具体的标注人员，标注员就可以开始标注和提交标注任务。



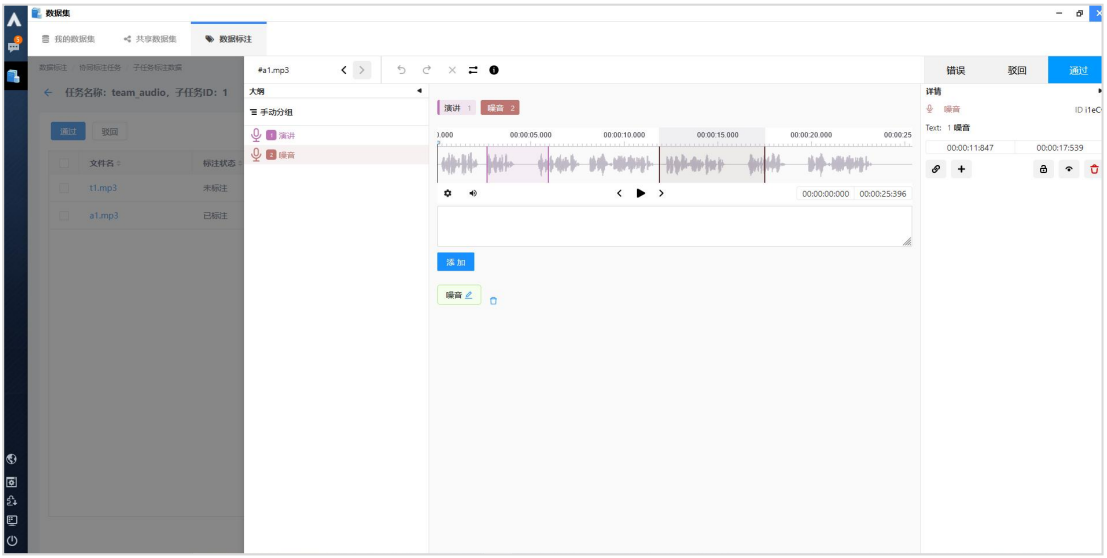
数据标注



提交标注任务

2.1.8.3.4. 审核任务

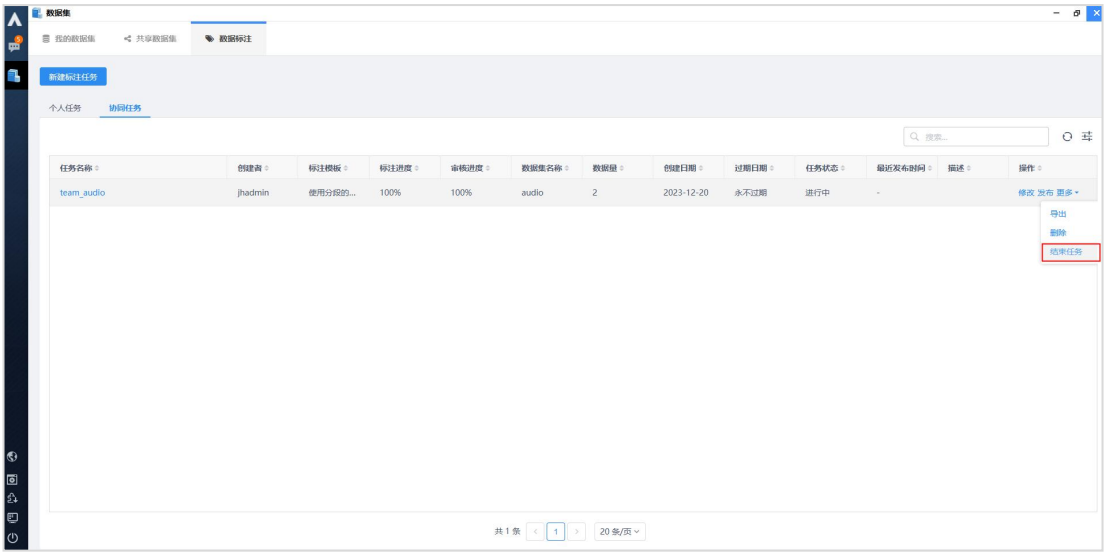
标注员提交任务后审核员进行审查。有错误、驳回、通过三种审核结果。



审核标注任务

2.1.8.3.5. 结束任务

标注任务创建者可以在任意阶段对标注任务进行结束操作，结束操作后标注员和审核员将不能对标注任务进行任何修改。



结束标注任务

2.2. 模型开发

模型开发模块集成了开发环境、方案设计、案例中心和实验管理，实现通过IDE 编辑代码训练模型和通过拖拽组件搭建方案流程训练模型，添加实验管理API 的作业，可在实验管理模块查看训练曲线。案例中心，包含了拖拽式搭建模型的完整示例。

2.2.1. 开发环境

开发环境支持用户使用集群资源创建编程开发环境，开发环境在用户进行AI 算法与模型开发时发挥关键作用。用户可通过 WEB 界面按需申请所需资源，迅速便捷地创建个性化的开发环境，在开发环境中进行代码编写和调试工作。系统内置了“JupyterLab”“VSCode”“RStudio”“桌面”和“Web 终端”五种环境类型。

开发环境提供了丰富的功能，包括：

AI 框架集成：集成多种主流 AI 框架，如 TensorFlow、PyTorch 等，并支持用户在开发环境内安装自定义 AI 框架，满足个性化开发需求。

多种算力规格资源选择：提供 CPU、GPU 等多种算力规格，用户可灵活选择适应任务的资源配置。新增 GPU 动态挂载卸载功能，用户可以根据任务需求动态调整 GPU 资源，提升资源利用效率。

资源监控：实时监控 GPU、CPU 和内存的使用情况，确保资源状态一目了然。

环境访问：提供多种访问方式，包括 Web 终端、Remote SSH 方式（通过本地 VSCode、PyCharm 插件连接），满足用户多样化的使用需求。容器桌面中新增 VSCode 和插件支持，用户可以直接在容器桌面中使用 VSCode 及其丰富的插件生态。同时，PyCharm 也增加了插件支持，进一步提升开发体验。

环境变更：用户能够在线调整开发环境，并保存开发环境为新的镜像。

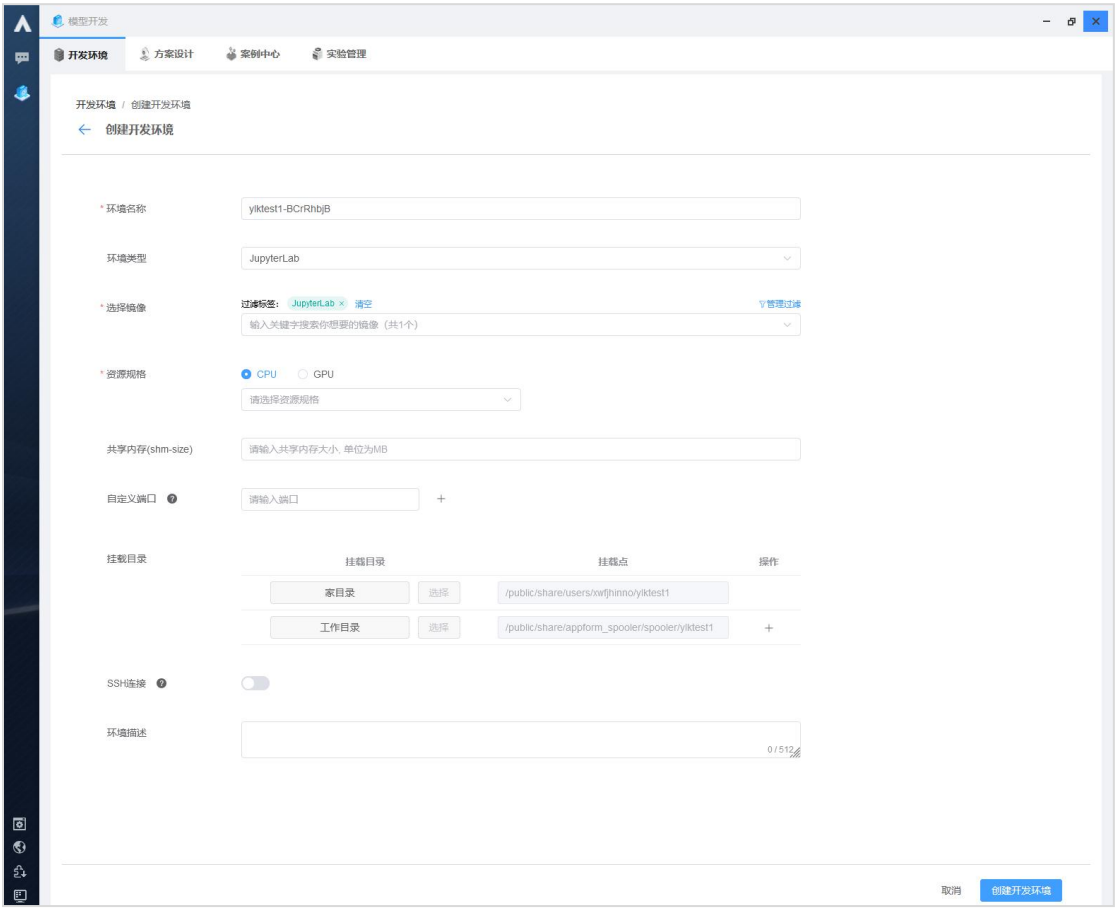
新增 RStudio 类型支持：开发环境新增了对 RStudio 的支持，满足 R 语言开发者的需求。

VSCode 类型支持智能编程插件：VSCode 类型环境现在支持智能编程插件，

如代码自动补全、语法检查、代码格式化等，帮助开发者提高编码效率和质量。

2.2.1.1. 新建开发环境

以新建“JupyterLab”类型的开发环境为例，操作步骤如下：点击“创建开发环境”按钮，弹出“创建开发环境”窗口，如下图所示：



新建开发环境

图中每个参数的具体含义如下：

环境名称：必填项，默认系统会自动回填一个可用的环境名称。

环境类型：选择创建环境的类型，支持五种类型的环境，如下所示：

- JupyterLab：该环境集成了 Jupyter 开发工具。
- VSCode：该环境集成了 VSCode 开发工具。
- RStudio：该环境集成了 RStudio 开发工具。

- 桌面：提供带 GUI 的操作系统环境。
- Web 终端：仅提供 Web 终端。

选择镜像：选择创建环境实例所用的镜像。不同环境类型的镜像，会根据镜像标签进行过滤，例如：创建 JupyterLab 类型的环境，默认只能选择包含“JupyterLab”标签的镜像。Web 终端支持所有镜像。如需订制和修改 JupyterLab、VSCode、RStudio 和桌面镜像，请基于 AI 平台提供的基础镜像进行修改。

资源规格：选择创建环境实例所需要的资源规格。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

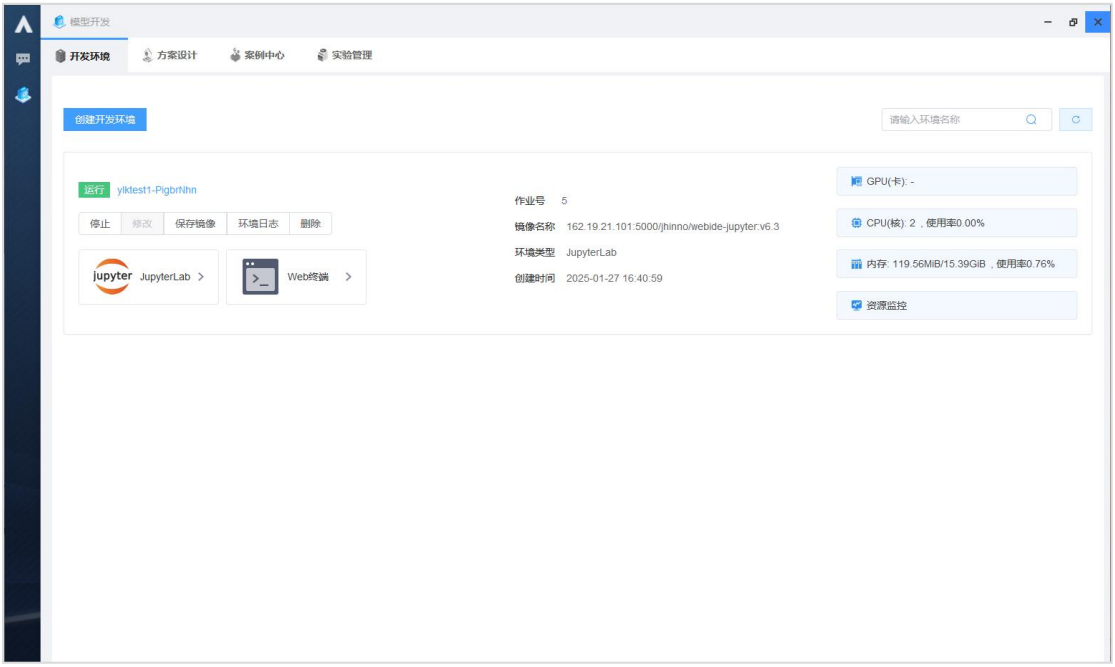
自定义端口：设置创建环境实例所用的端口。

挂载目录：默认会把“家目录”、“工作目录”挂载到容器中，也可以额外指定目录，挂载到开发环境的容器中。

SSH 连接：默认关闭。打开时，开发环境启动成功后，可以使用用户名 SSH 连接信息在集群内通过 ssh 访问该容器。

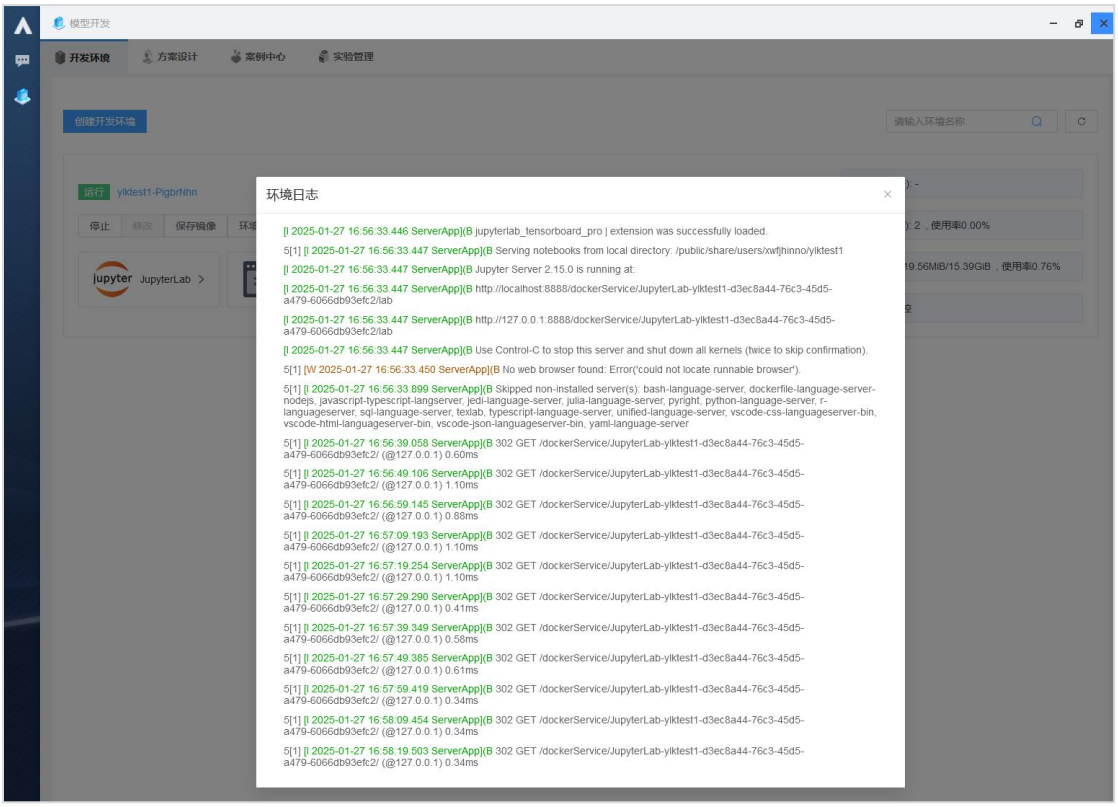
环境描述：输入环境描述信息，可选。

点击“创建开发环境”按钮，即可完成开发环境的创建，如下图所示：



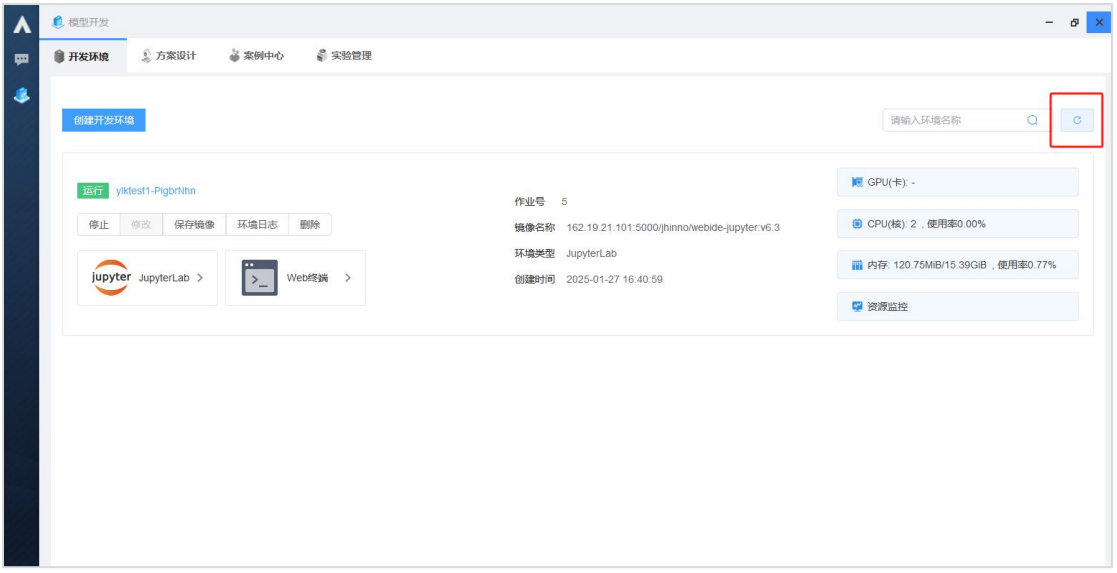
开发环境列表

点击“环境日志”按钮，查看环境实例启动情况，如下图所示：



环境日志

点击右上角的“刷新”按钮，手动更新环境实例的状态，如下图所示：

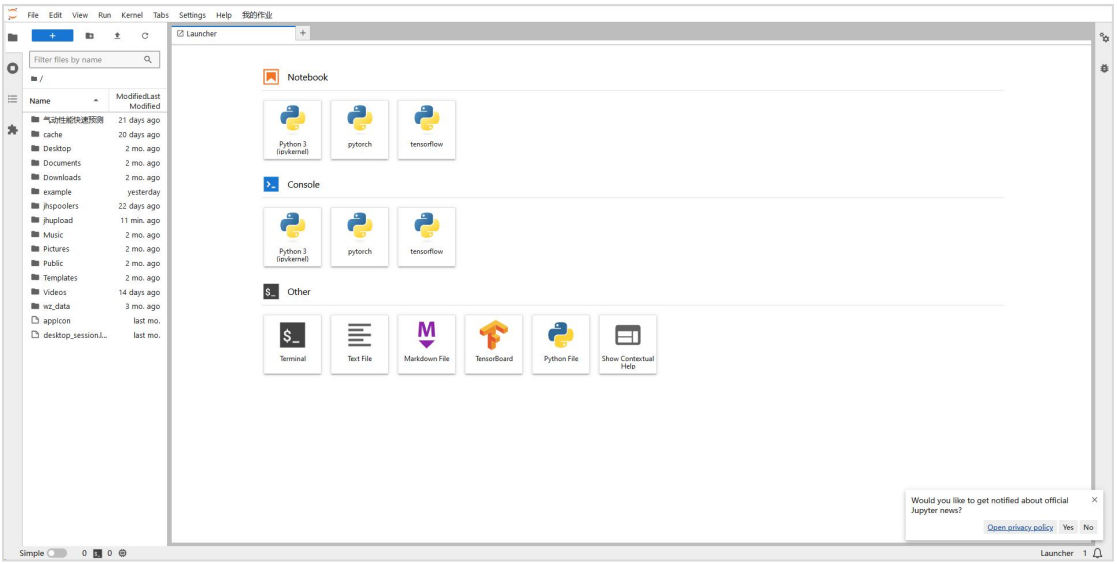


更新环境实例的状态

2.2.1.2. 使用开发环境

2.2.1.2.1. JupyterLab

点击“JupyterLab”类型的开发环境实例上的“Jupyter”卡片按钮，访问 Jupyter 服务，如下图所示：

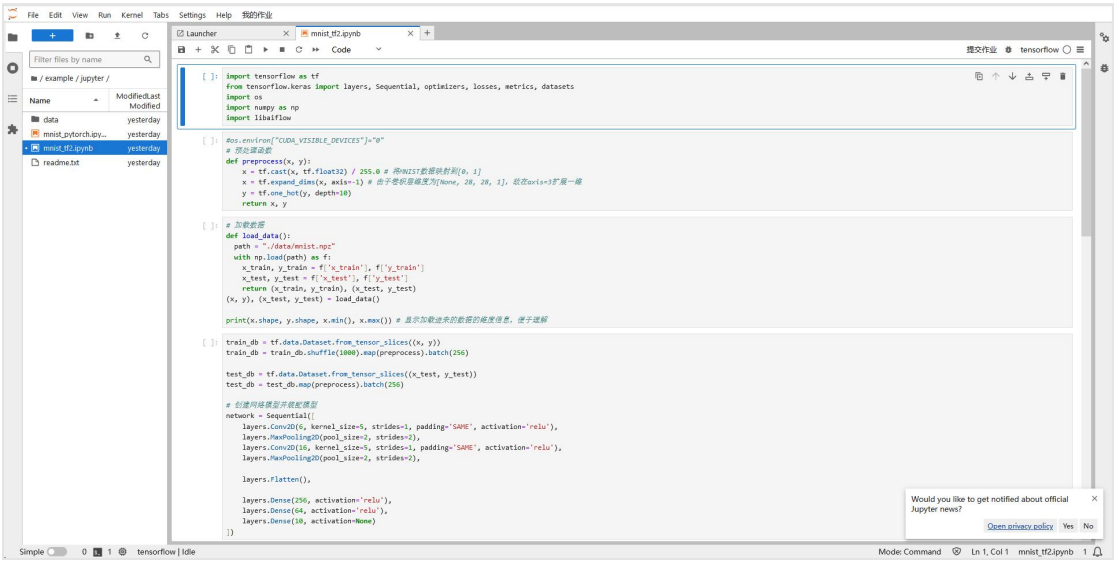


Jupyter 服务

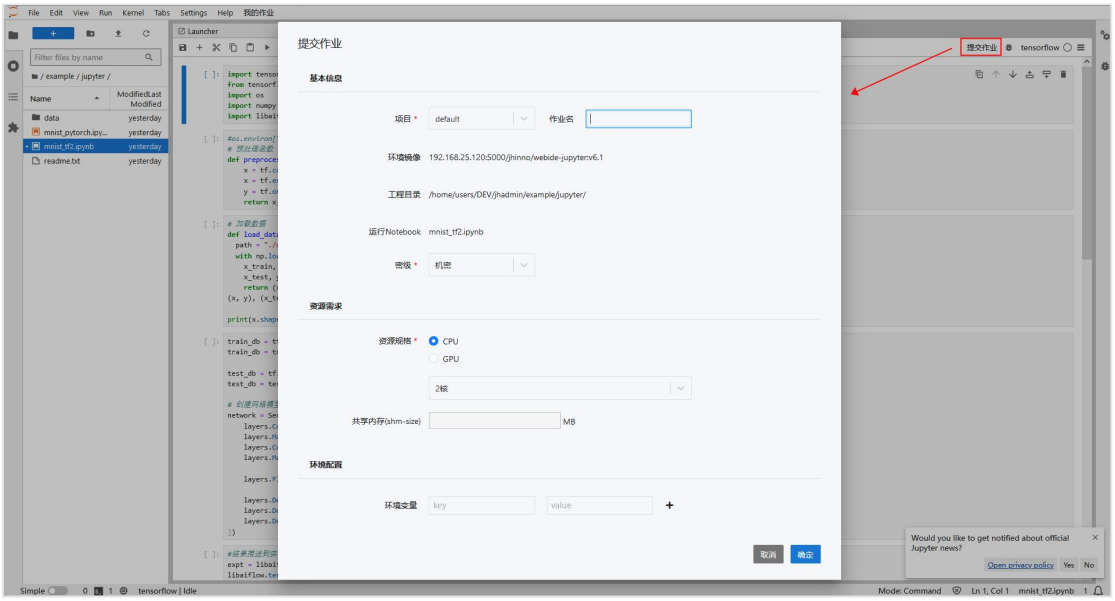
- Jupyter 插件

Jupyter 中内置 AI 作业插件，支持从 Jupyter 向集群中提交作业和管理作业。

打开“ipynb”格式的文件，如下图所示：



可以点击“提交作业”按钮，打开作业提交参数设置界面，如下图所示：



作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：jupyter_webide。

环境镜像：默认为创建当前环境实例时选择的镜像，也就是提交作业使用的镜像，此处镜像不可修改；

工程目录：ipynb 文件所在的目录，ipynb 运行所依赖的文件也应该在该目

录下，不可修改。

运行 Notebook：选中的 ipynb 文件，如：train.ipynb。

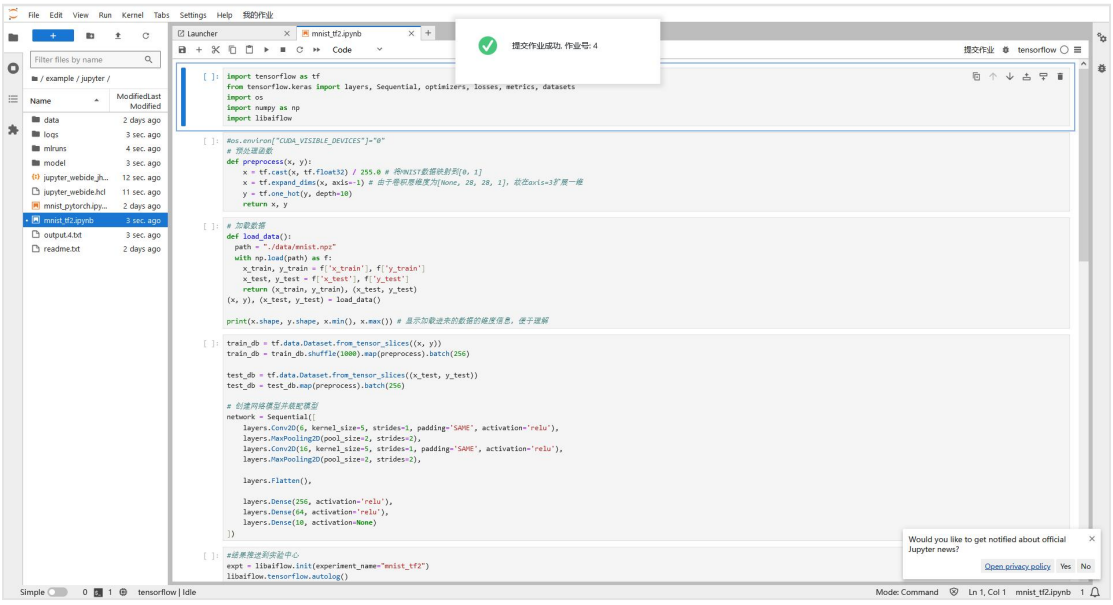
密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

资源规格：必填项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

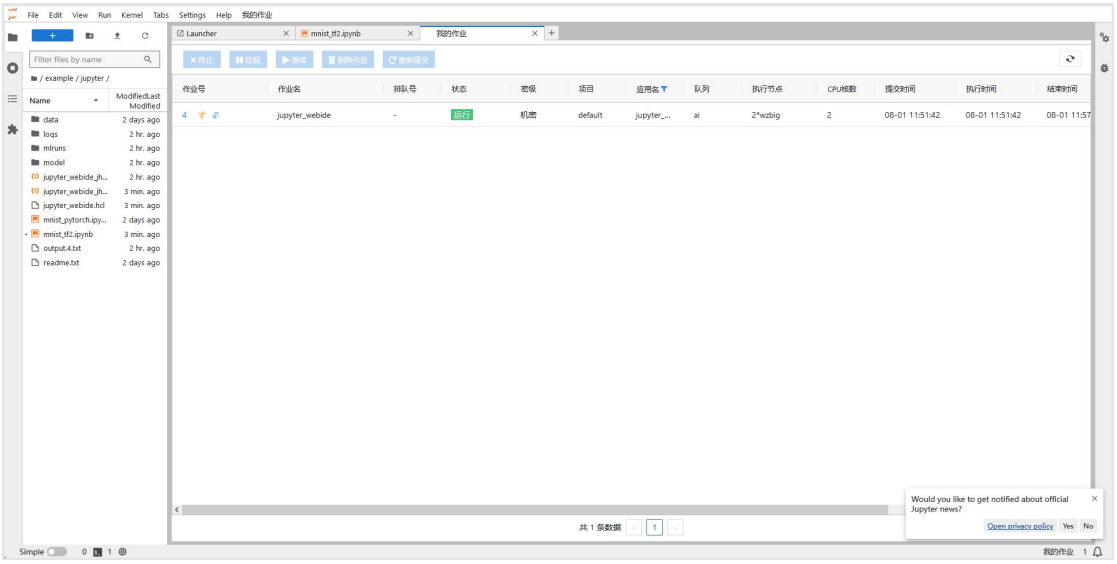
共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

环境变量：指定程序运行所需要的额外环境变量。

点击“确定”按钮，提交作业，如下图所示：

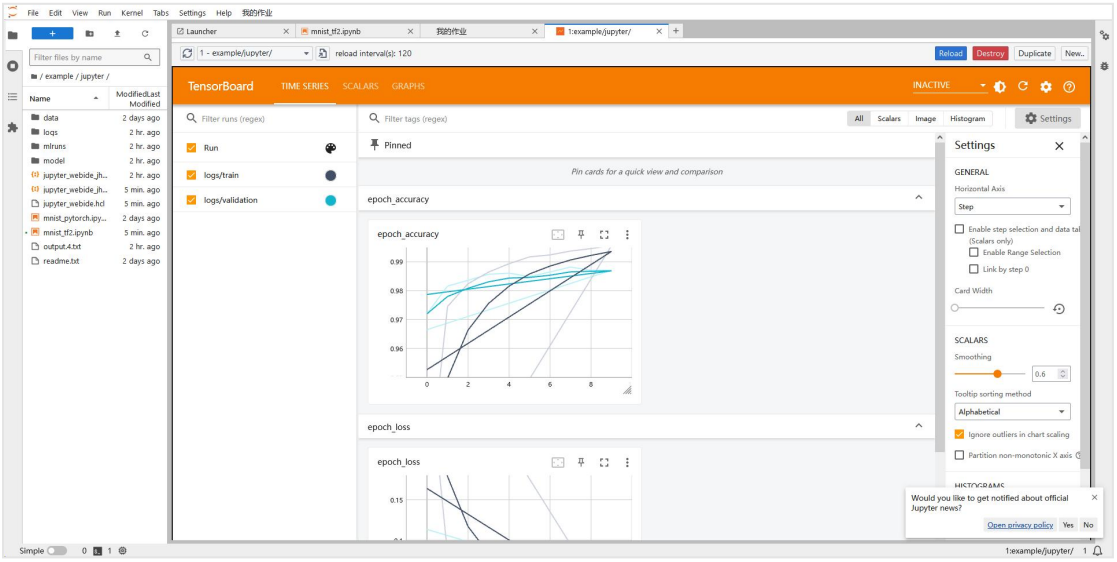


点击顶部工具栏处的“我的作业”按钮，打开我的作业界面，如下图所示：



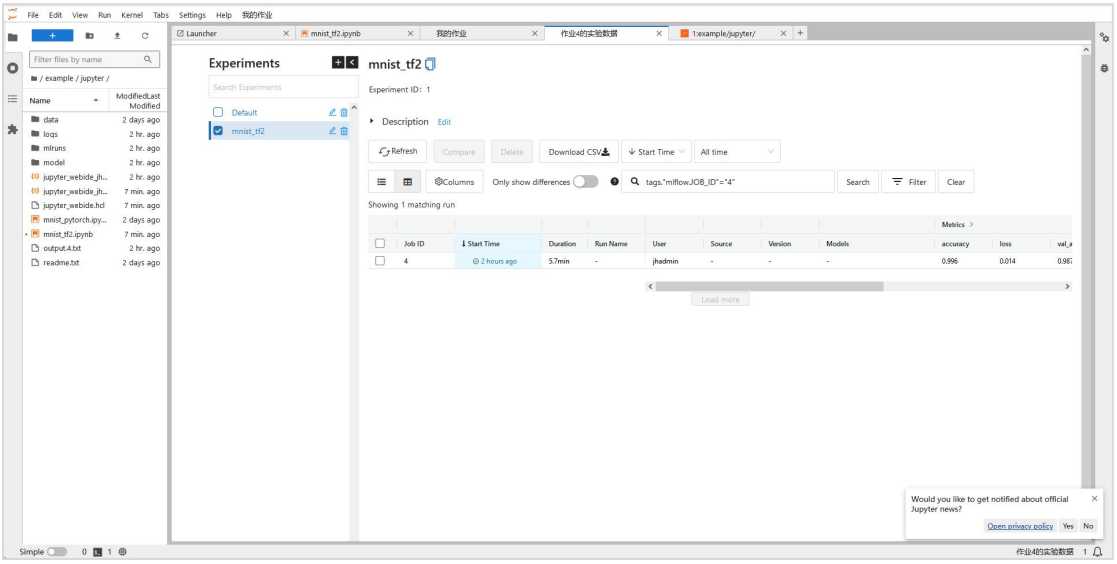
我的作业

点击作业实例上的“Tensorboard”图标按钮，访问 Tensorboard 服务查看模型数据，如下图所示：



访问 Tensorboard 服务

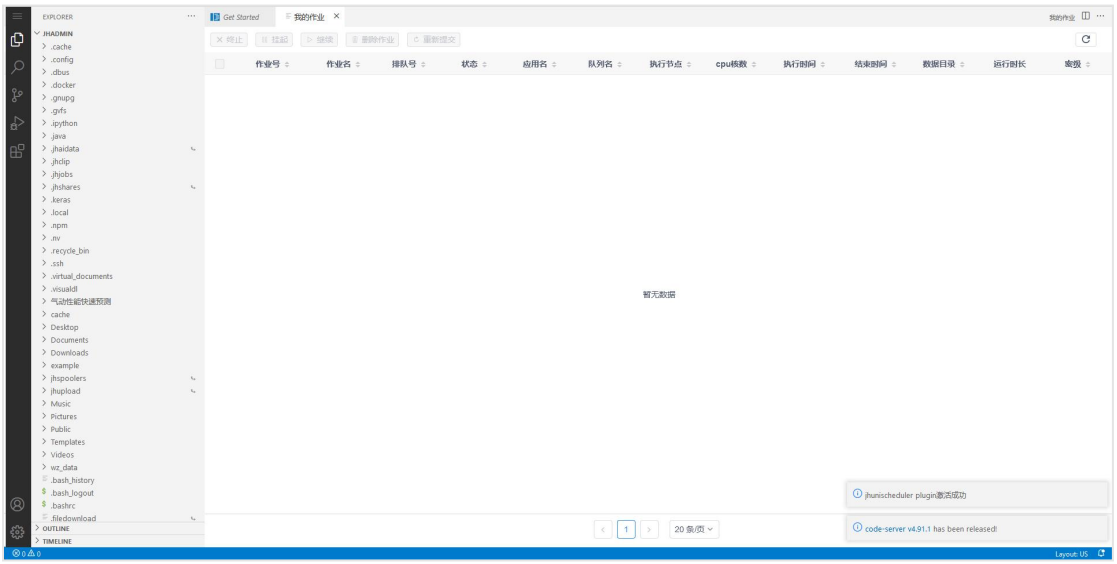
点击作业实例上的“实验管理”图标按钮，查看实验数据，如下图所示：



查看实验数据

2.2.1.2.2. VSCode

点击“VSCode”类型的开发环境实例上的“VSCode”卡片按钮，访问 VSCode 服务，如下图所示：



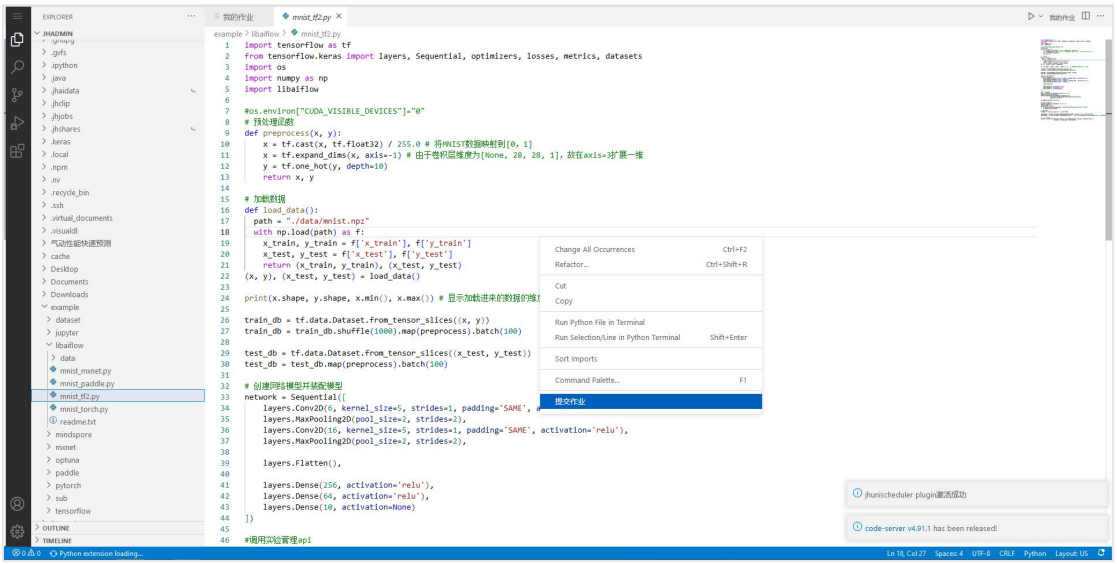
VSCode 初始界面

- VSCode 插件

VSCode 中内置智能编程插件和 AI 作业插件，支持从 VSCode 向集群中提交

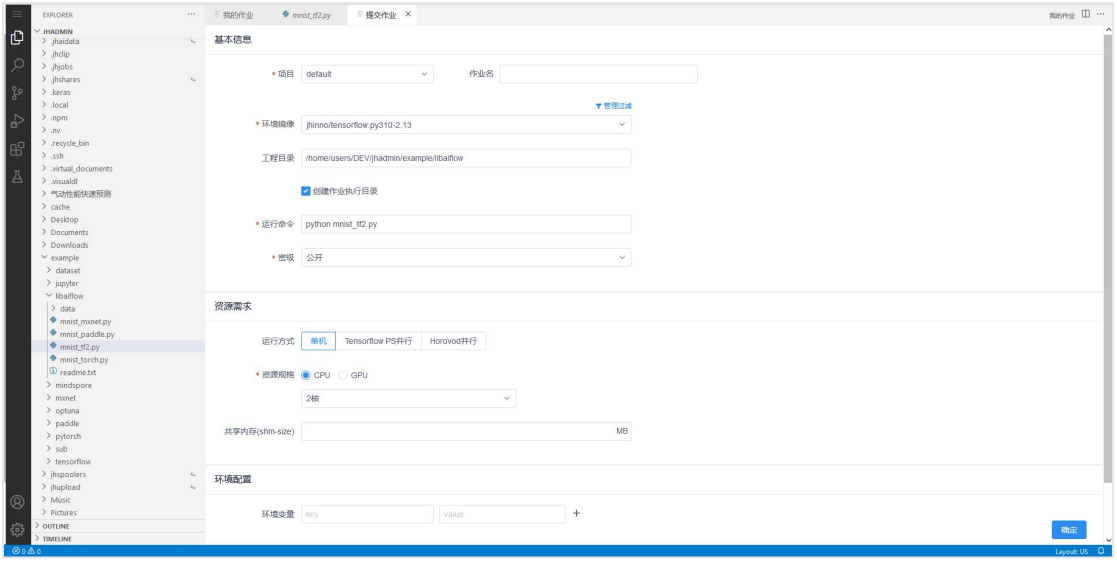
作业和管理作业。

选中或者打开文件，在右键菜单中选择“提交作业”，如下图所示：



右键菜单中的提交作业

点击“提交作业”右键菜单，打开作业提交参数设置界面，如下图所示：



作业提交页面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：vscode_webide。

环境镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适

的镜像。

工程目录：指定代码的工程目录，默认回填运行程序的父目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

运行命令：必填项，指定运行命令，包括程序的入口文件和程序参数等，例如：python train.py --batch=64 --data-path="/data/minst_test"。

密级：管理员开启密级功能后，显示此选项。

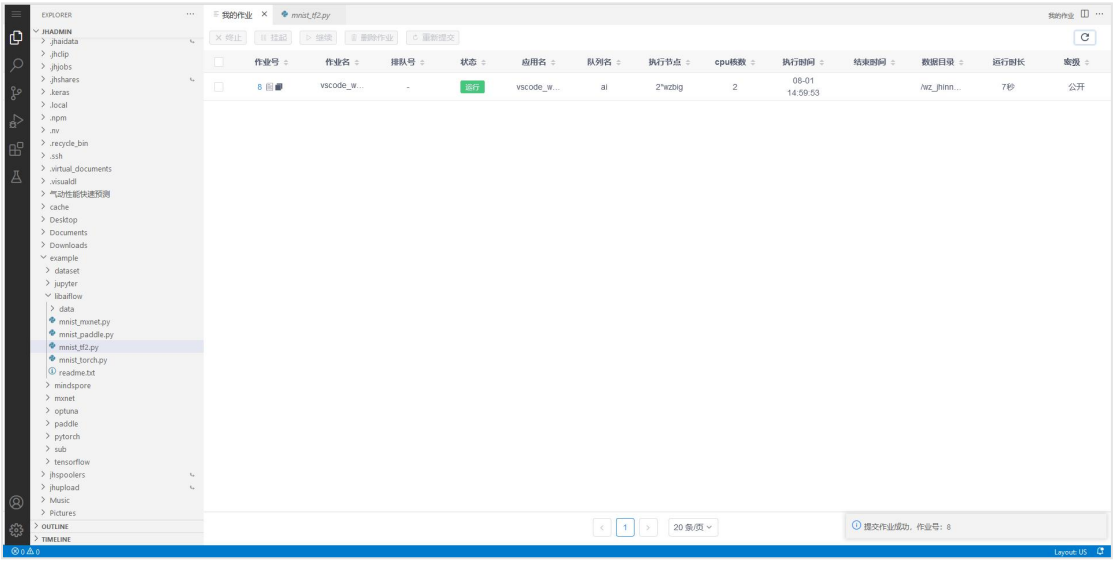
运行方式：选择作业运行方式，如下所示：

- 单机：默认选择单机，仅在单节点上运行训练程序。
  - 资源规格：必填项，选择运行程序所需要的资源配置。可以选择CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。
- Tensorflow PS 并行：提交以 Parameter Server 方式实现的tensorflow 训练代码，支持跨节点，需填写并行相关参数，如下所示：
  - Server 数：指定参数服务的数量。
  - Worker 数：指定训练服务的数量。
  - 资源规格：必填项，选择运行程序所需要的资源配置。可以选择CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。
- Horovod 并行：提交以 Horovod 实现的并行 AI 训练程序， 需要指定Horovod 相关参数。Horovod 并行模式需要模型训练代码以 Horovod 库的并行方式进行编码，如下所示：
  - 并行实例数：指定并行训练的实例数量。
  - 资源规格：必填项，选择运行程序所需要的资源配置。可以选择CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

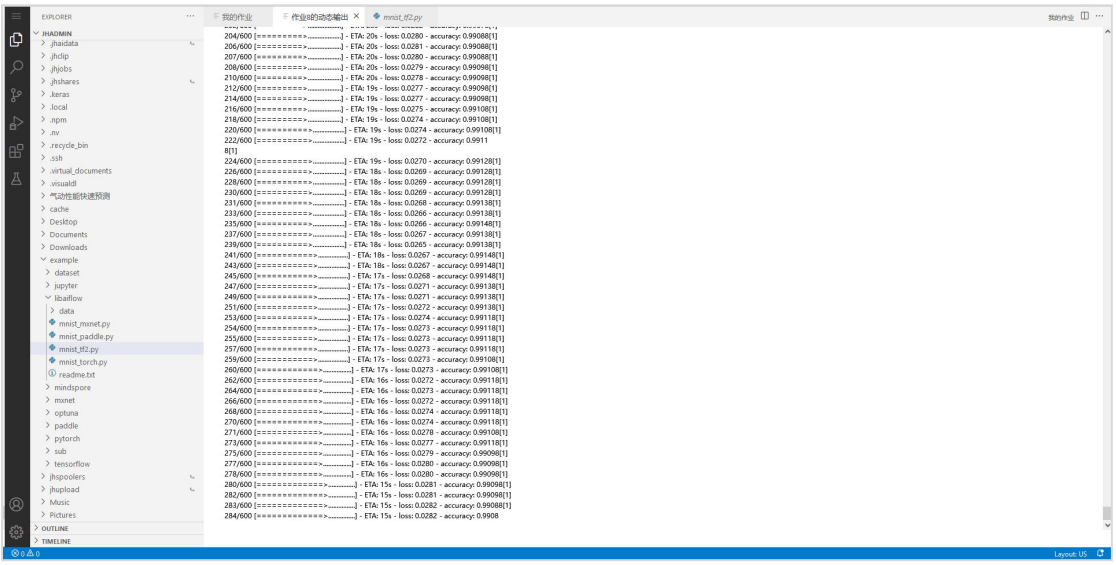
共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

环境变量：指定程序运行所需要的额外环境变量。

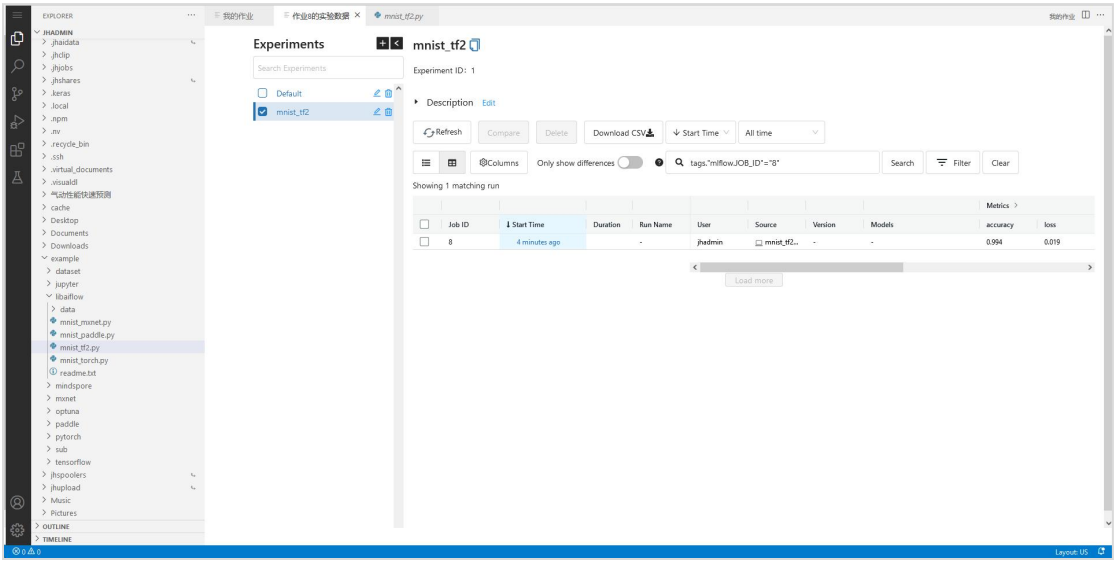
在“提交作业”页面中完成基本信息、资源需求和环境配置相关信息的填写和选择后，点击“确定”按钮，提交作业。作业提交后，会自动打开“我的作业”页面，方便用户在此查看提交作业的信息，如下图所示：



点击作业实例上的“动态输出”图标按钮，查看作业日志的动态输出，如下图所示：



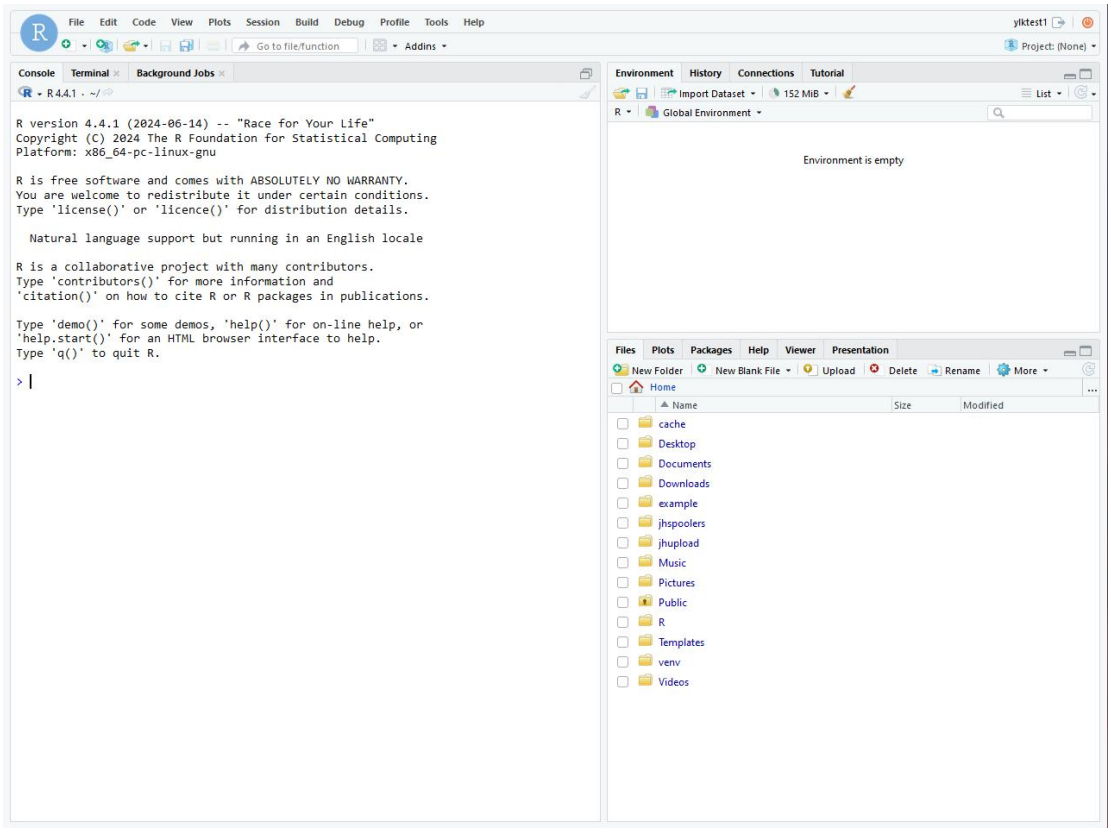
点击作业实例上的“实验管理”图标按钮，查看实验数据，如下图所示：



查看实验数据

2.2.1.2.3. RStudio

点击“RStudio”类型的开发环境实例上的“RStudio”卡片按钮，访问 RStudio 服务，如下图所示：

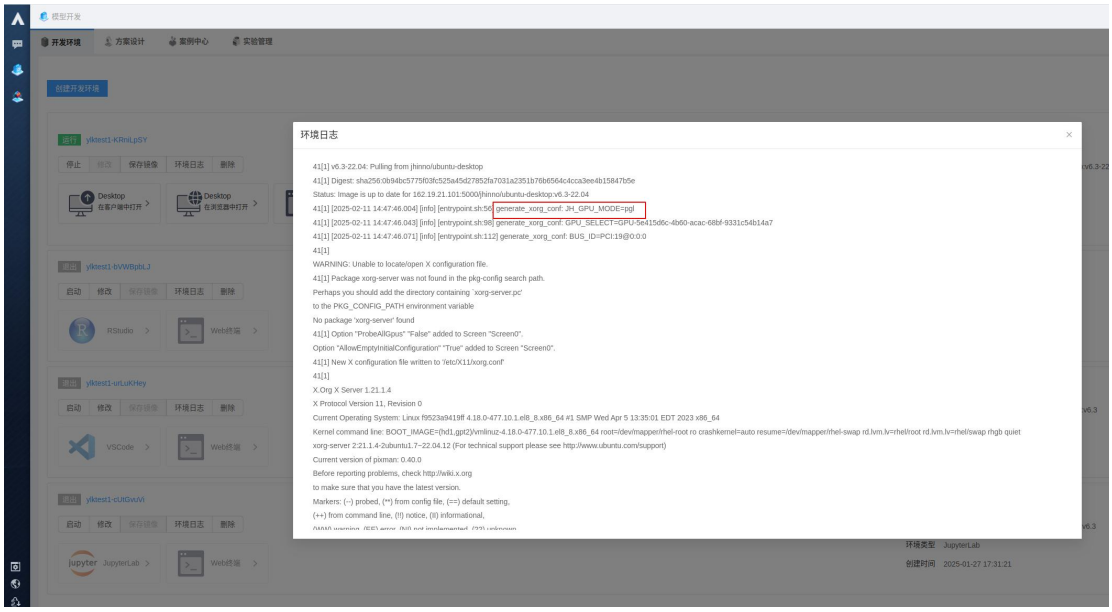


RStudio 服务

2.2.1.2.4. 桌面

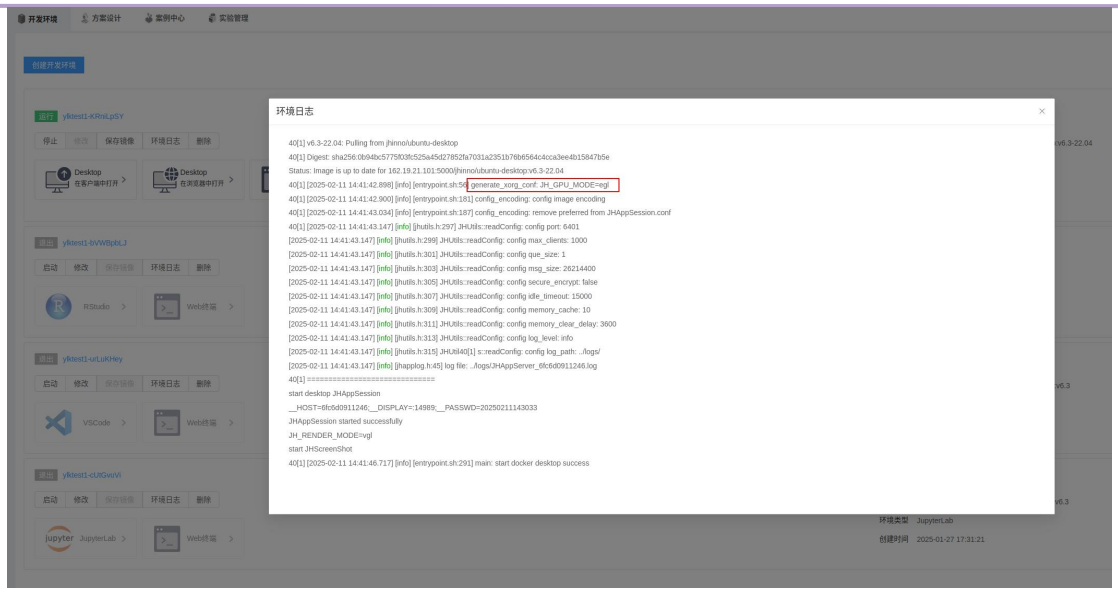
桌面类型的开发环境，提供 CentOS/Ubuntu 系统带 GUI 图像桌面的容器开发环境，并且内置了 Conda，Pycharm IDE 和 VSCode 及插件，用户可以在 GUI 桌面中打开其他图形程序，如果创建环境时选择了 GPU，在环境内支持使用 3D 加速和 CUDA 计算能力。开发环境的容器桌面支持 PGL 容器 GPU 直通模式和 EGL 容器 GPU 共享模式，提高容器内 GPU 3D 渲染性能和兼容性。

创建桌面类型的开发环境时，选择独占的资源规格，增加环境变量 JH_GPU_MODE=pgl 显卡直通模式。如下图所示：



显卡直通模式

选择共享的资源规格，增加环境变量 JH_GPU_MODE=egl，如下图所示：



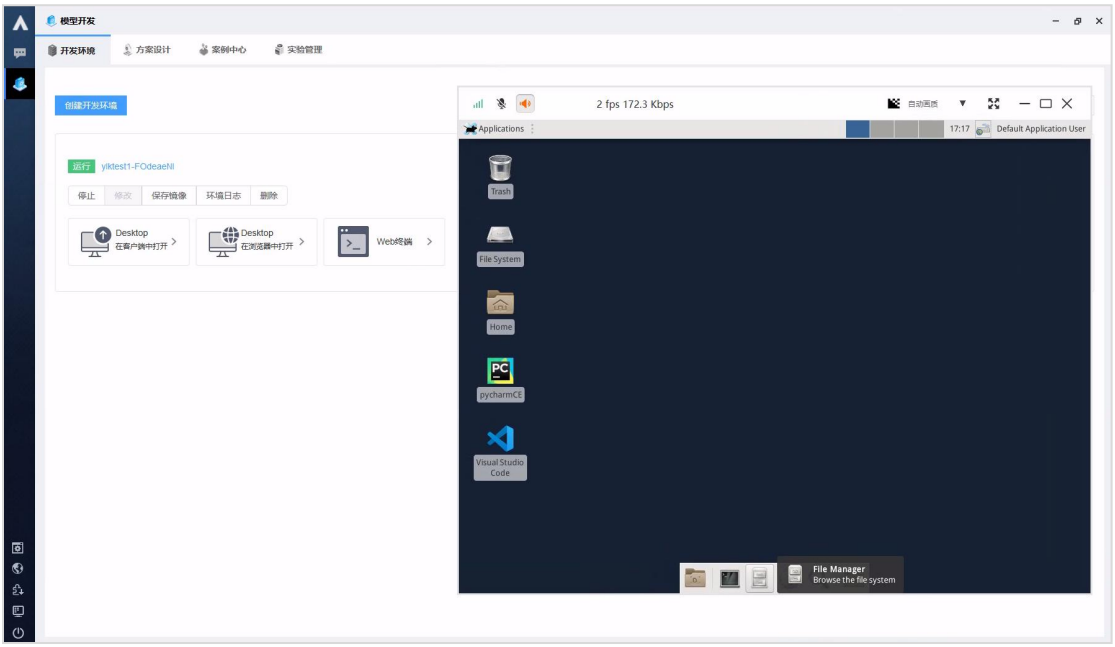
EGL 容器模式

桌面类型的开发环境创建完成后，不仅可以通过点击“桌面客户端打开”卡片按钮，以客户端访问桌面，还可以通过点击“桌面 WEB 打开”卡片按钮，以 WEB 方式访问桌面，如下图所示：

如果以客户端方式，访问桌面，需要先安装客户端软件，如下图所示：

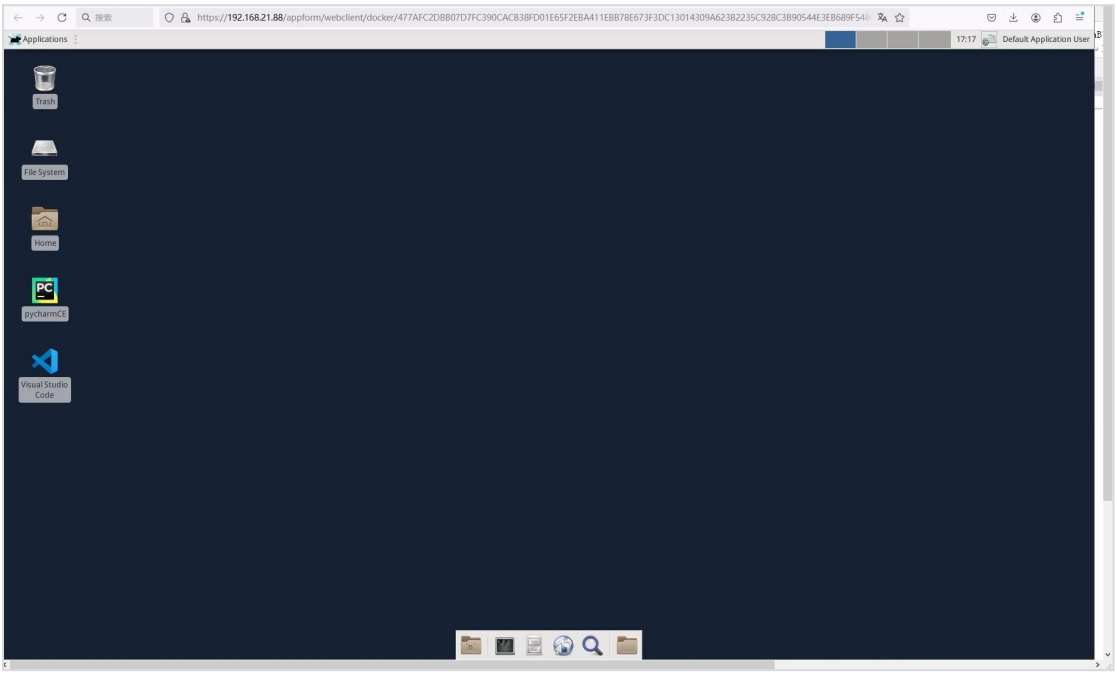
以客户端方式访问桌面

已安装客户端软件并重启浏览器后，点击“桌面 客户端打开”卡片按钮，以客户端方式访问桌面，如下图所示：



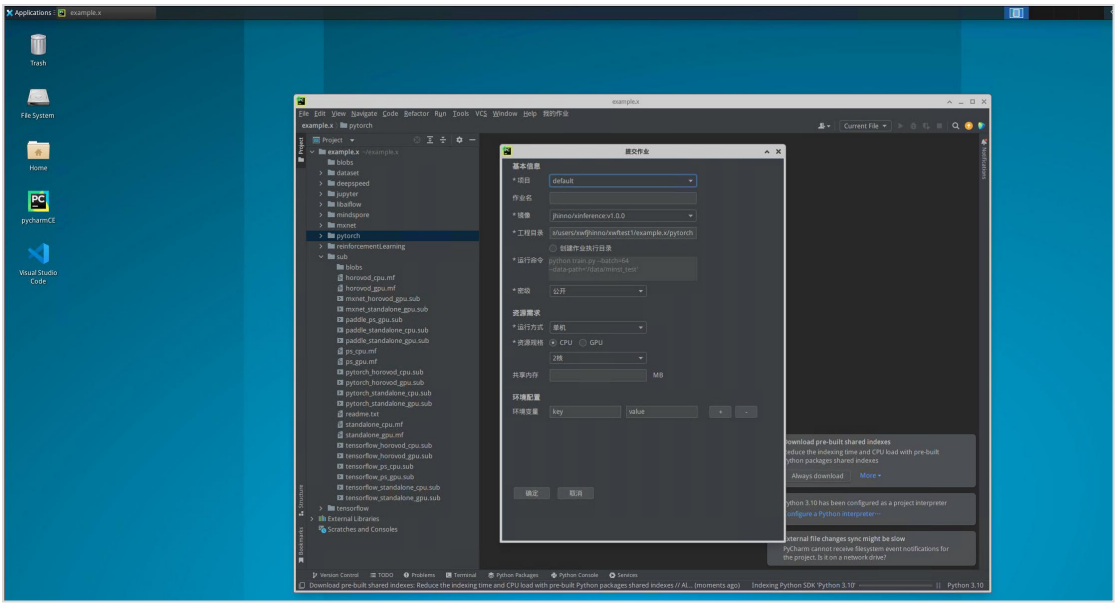
以客户端方式访问桌面

点击“桌面 WEB 打开”卡片按钮，以 WEB 方式访问桌面，如下图所示：

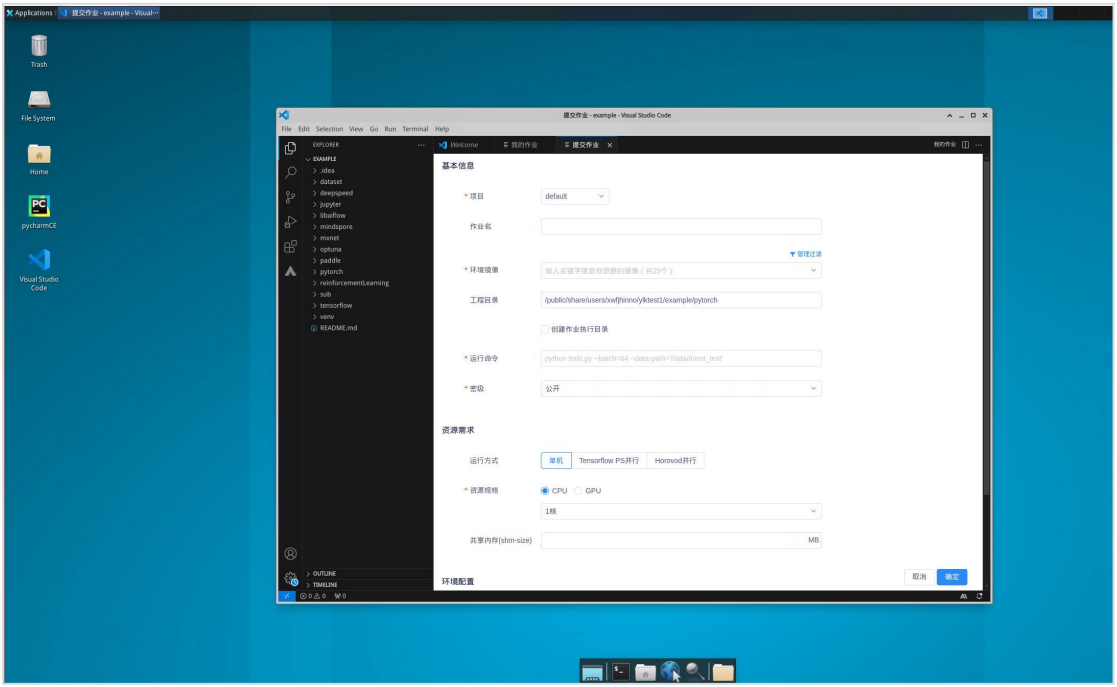


以 WEB 方式访问桌面

桌面内置 Pycharm IDE 和 VSCode 及插件，插件主要功能包括作业提交、我的作业以及对我的作业进行管理。如下图所示：



桌面 Pycharm 及插件

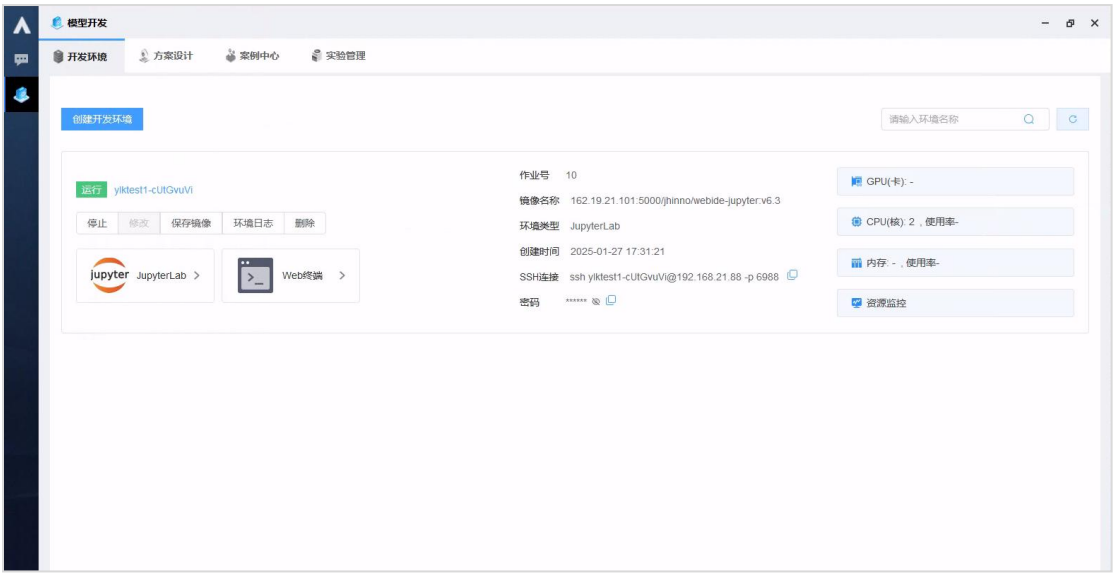


桌面 VSCode 及插件

2.2.1.2.5. 使用 SSH 远程连接开发环境

JupyterLab，VSCode，RStudio 和桌面类型的开发环境，默认支持 SSH 连接功能且对应的内置镜像默认集成了 sshd 服务。而 Web 终端类型的开发环境，需要选择不仅支持 sshd 服务（详情见：附录六），还打上了 SSH 标签的镜像，才能正常使用 SSH 连接功能。

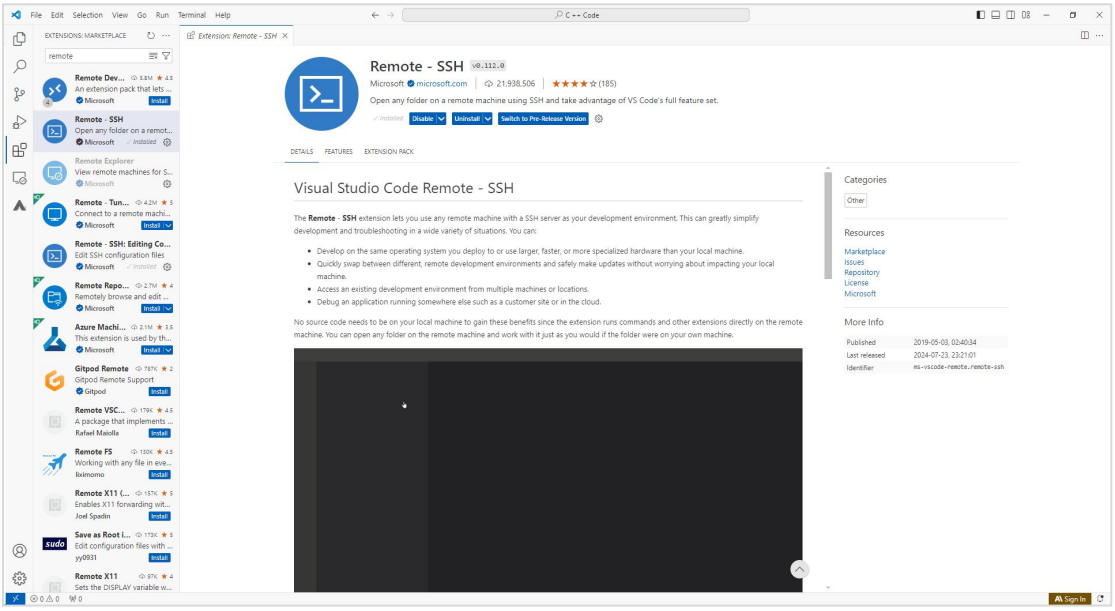
新建开发环境时，如果开启了 SSH 连接功能，在开发环境实例上会显示 SSH 连接信息。使用本地的 SSH 连接工具，复制 SSH 连接和密码，就可以连接到开发环境中。如果已存在且尚未开启 SSH 连接功能的开发环境实例，需要使用 SSH 连接功能，可以通过“修改”开发环境实例，开启 SSH 连接功能。修改后重启开发环境，当开发环境处于“运行中”状态时，即可看到 SSH 连接信息，如下图所示：



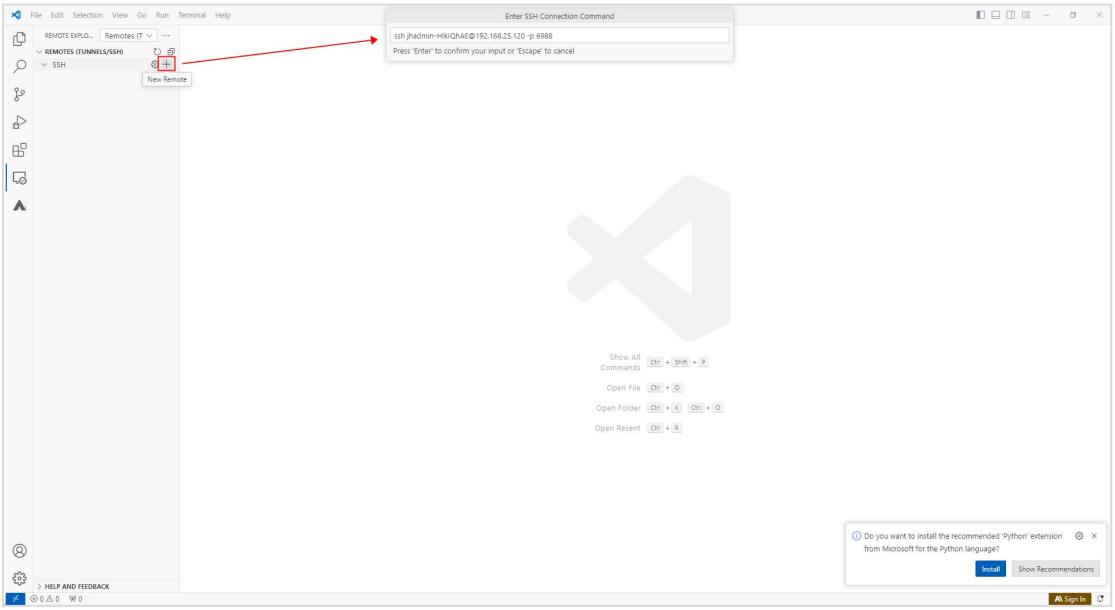
开发环境实例的连接信息

1）通过 IDE 连接开发环境

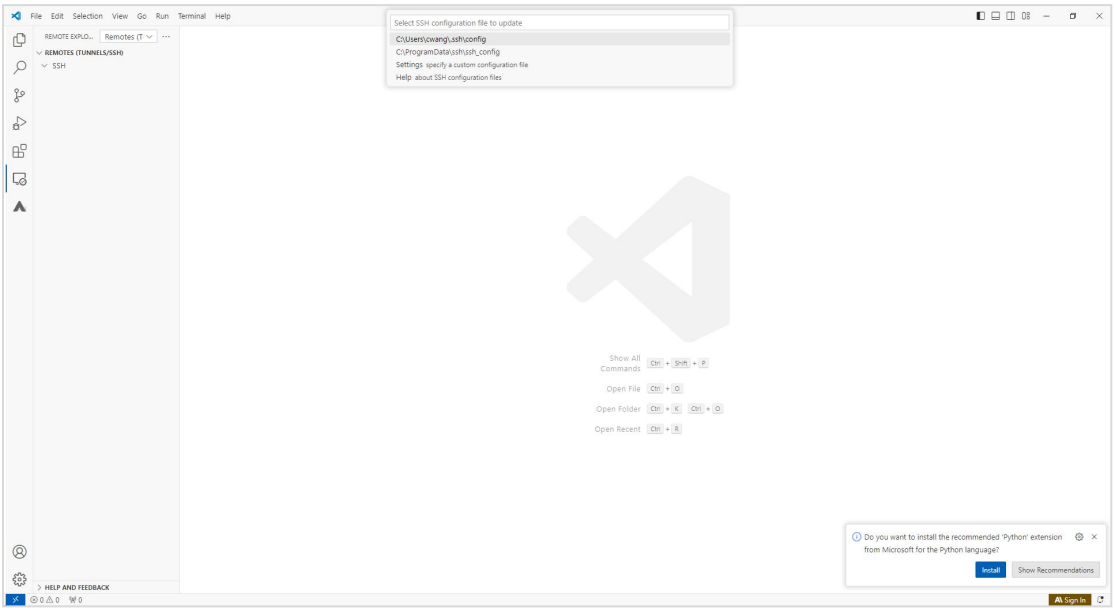
通过 IDE 连接开发环境，以“VSCode”IDE 为例，需要安装 remote 插件连接开发环境，如下图所示：



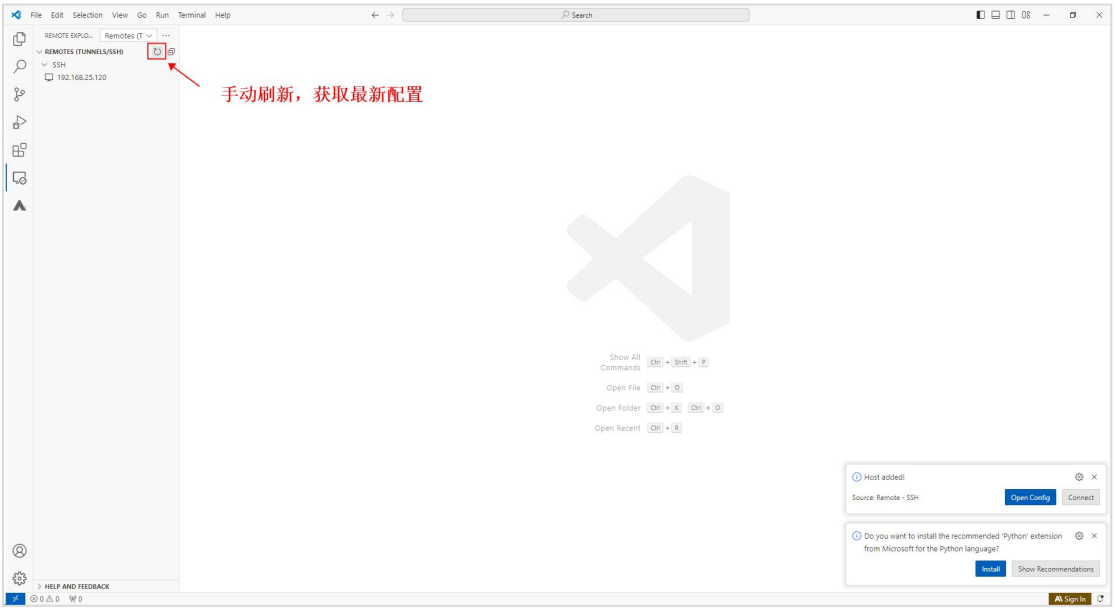
在 VSCode 左侧工具栏中点击“Remote Explorer”图标按钮，打开 Remote Explorer 扩展栏。点击“New Remote”按钮，弹出“Enter SSH Connection Command”弹框，如下图所示：



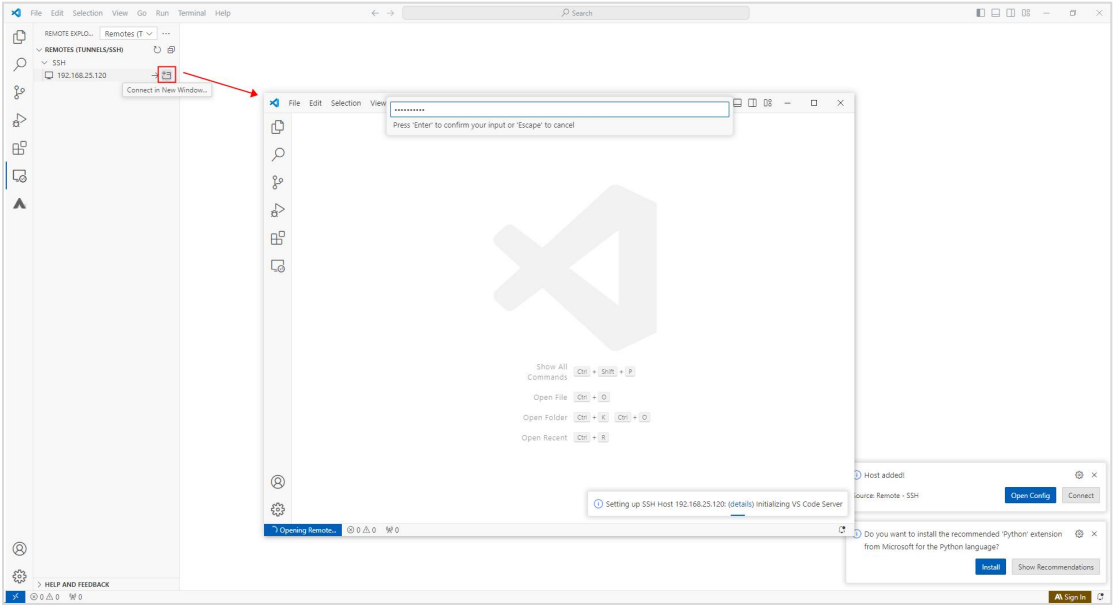
输入 SSH 连接信息后，点击“回车”键，弹出“Select SSH configuration file to update”弹框，如下所示：



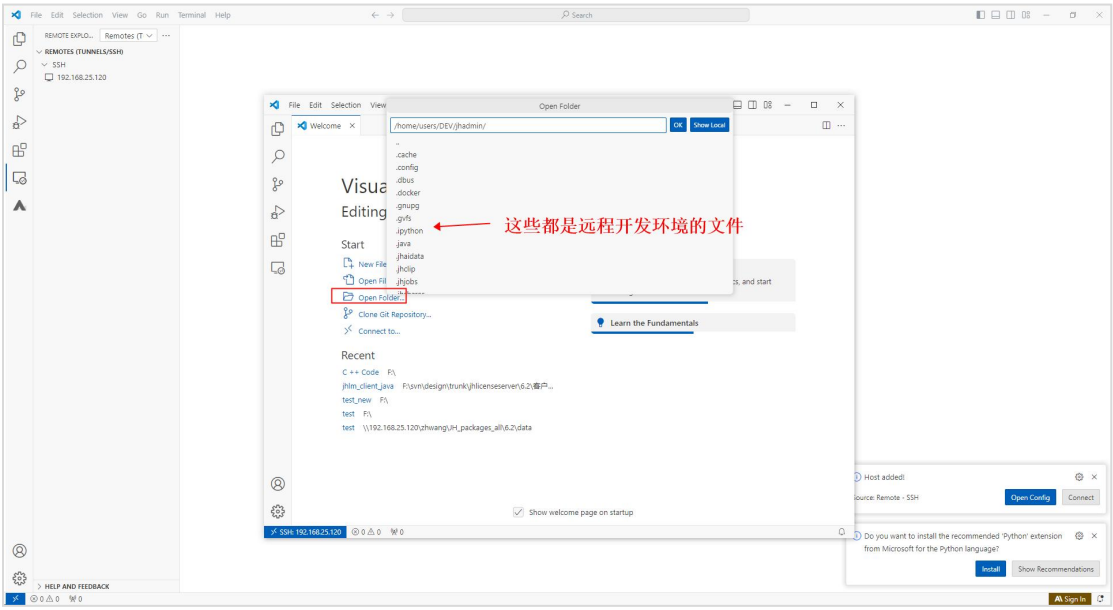
选择一个需要更新的 SSH 配置文件后，需要手动点击“刷新”按钮，获取最新的 SSH 配置，如下图所示：



点击“Connect in New Window”按钮，在新窗口连接开发环境，如下所示



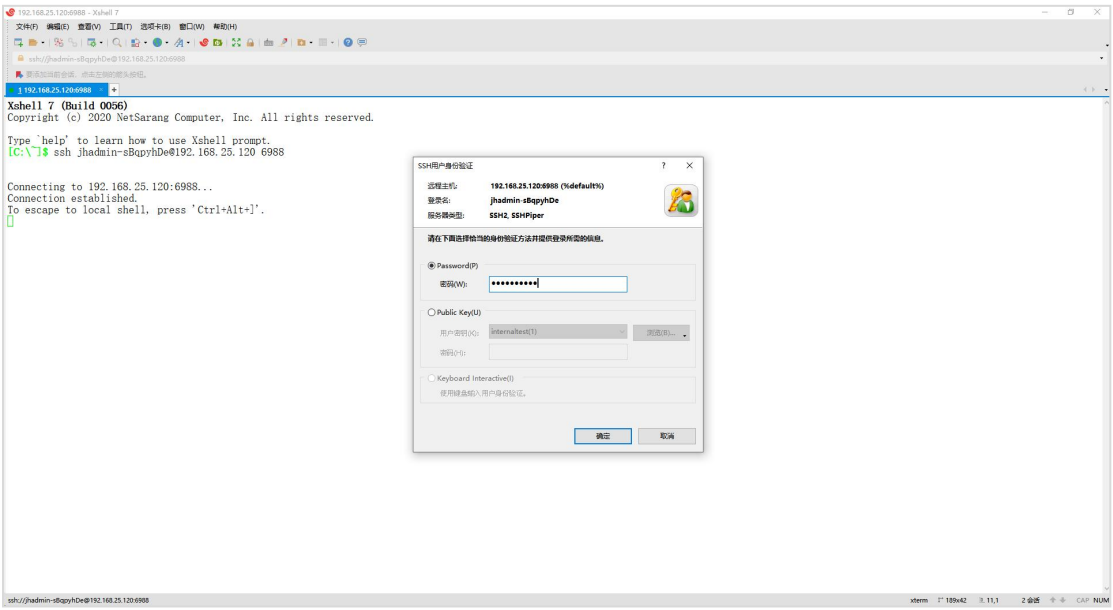
输入密码后，点击“回车”键，连接开发环境，如下图所示：



VSCode IDE 连接开发环境

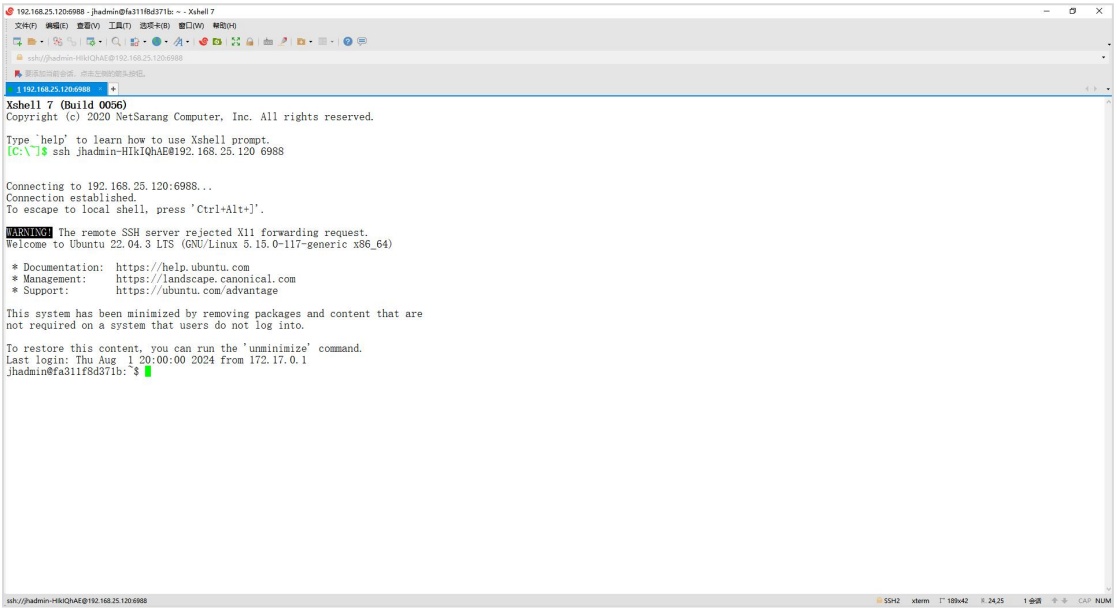
2）通过 SSH 工具连接开发环境

通过 SSH 工具连接开发环境，以“Xshell”SSH 工具为例，如下图所示：



注：Xshell 不支持-p 参数，所以连接时，不直接应用开发环境的 SSH 连接信息：ssh jhadmin-HIkIQhAE@192.168.25.120 -p 6988，需要手动删除“-p”，进行连接：ssh jhadmin-HIkIQhAE@192.168.25.120 6988。

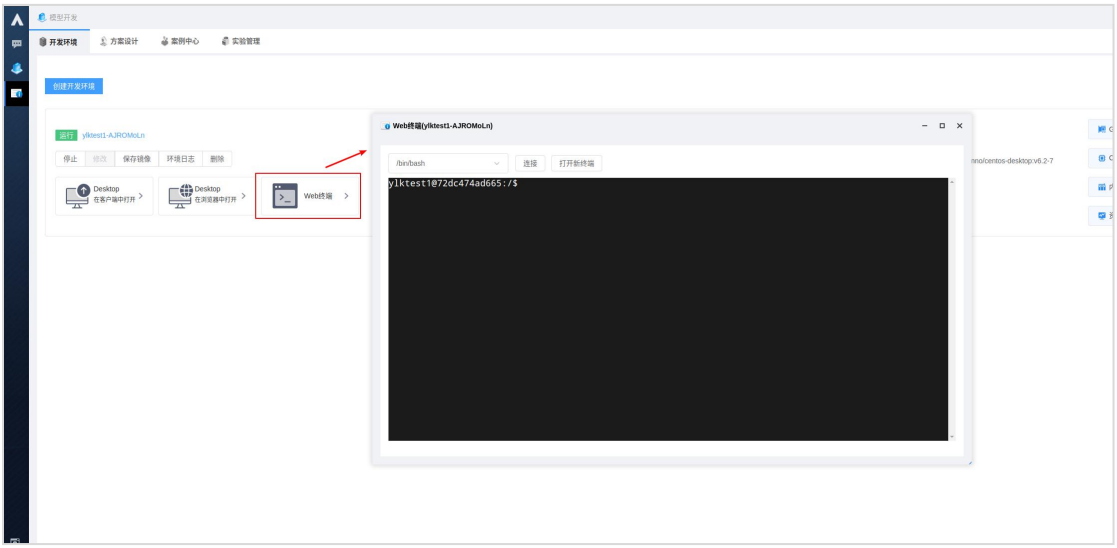
输入密码后，点击“回车”键，进行连接，如下图所示：



SSH 终端连接

2.2.1.2.6. 命令行提交管理作业

开发环境容器中集成了调度命令行接口，可以通过终端命令行方式提交作业和管理作业，如下图所示：



WEB 终端

以单机 Tensorflow 训练程序为例，提交脚本如下所示：

```
#!/bin/sh
```

```
#BSUB -J tensorflow_standalone_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf standalone_cpu.mf
#BSUB -o output.%J.txt

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type standalone --image
192.168.0.153:5000/jhinno/tensorflow:py310-2.13  --cmd='python
mnist_tf2.py' --workdir=${HOME}/example/tensorflow
```

tensorflow_standalone_cpu.sub

```
[host:all slots:1]
```

standalone_cpu.mf

准备好脚本后通过以下命令提交：

```
jsub < tensorflow_standalone_cpu.sub
```

查询作业为例，执行命令如下：

```
jhadmin@c47a7d1c9e79:~$ jjobs
JOBID      USER     STAT  QUEUE     FROM_HOST EXEC_HOST JOB_NAME     SUBMIT_TIME    ORDER
88         jhadmin  RUN   ai_excl   jhai110   2*jhai110 *in-BxfsjVtB Feb 05 02:22   -
93         jhadmin  RUN   ai_excl   jhai110   2*jhai110 *ard-jhadmin Feb 06 02:02   -
94         jhadmin  RUN   ai_excl   jhai110   2*jhai110 *in-SwJCdgGU Feb 06 02:04   -
102        jhadmin  RUN   ai_excl   jhai110   jhxihost* *in-KEtjGBOe Feb 06 08:03   -
jhadmin@c47a7d1c9e79:~$
```

注意：在 RStudio 开发环境的“Web 终端”中看到的用户是 root，需要执行 su - userx 切换成普通用户，才能提交资源管理与调度作业。

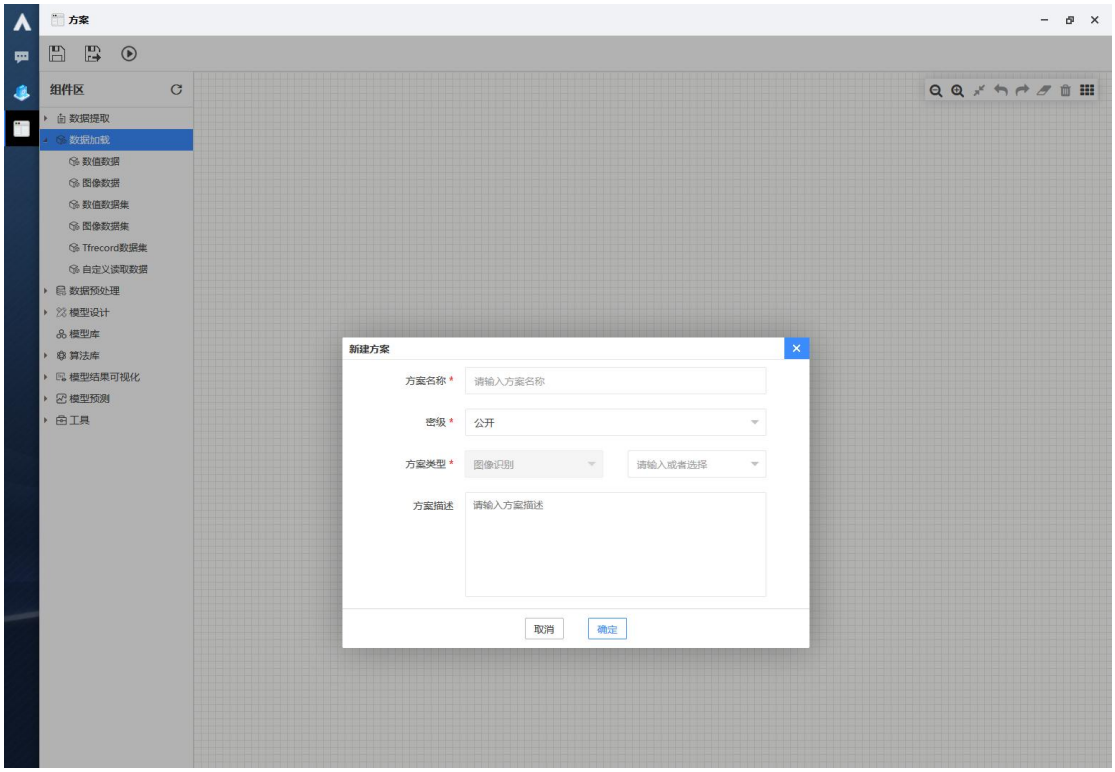
2.2.2. 方案设计

方案设计提供人工智能的流程可视化设计，方案提供了数据接入、数据处理、机器学习和深度学习等组件，通过拖拽组件和连线的方式可以实现数据处理和人工智能建模、训练、评估等复杂流程。

2.2.2.1. 构建方案设计流程

2.2.2.1.1. 新建

在模型开发应用选择方案设计，点击“新建方案”卡片，弹出新建方案的表单页，如下图所示：



新建方案

图中每个参数的具体含义如下：

方案名称：用户自定义，但是不能与现有方案名称一致。

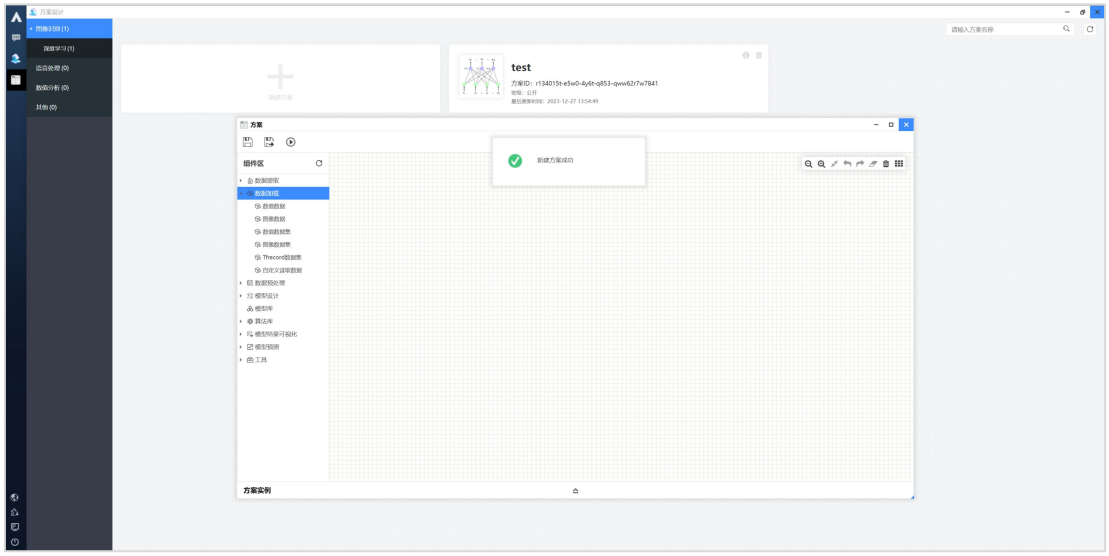
密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

方案类型：默认为已选的方案类型，也可以自定义方案类型。

方案描述：用于描述方案设计的相关内容，选填。

2.2.2.1.2. 设计

填写完成新建方案的表单后，点击“确定”按钮，进入方案设计页面。

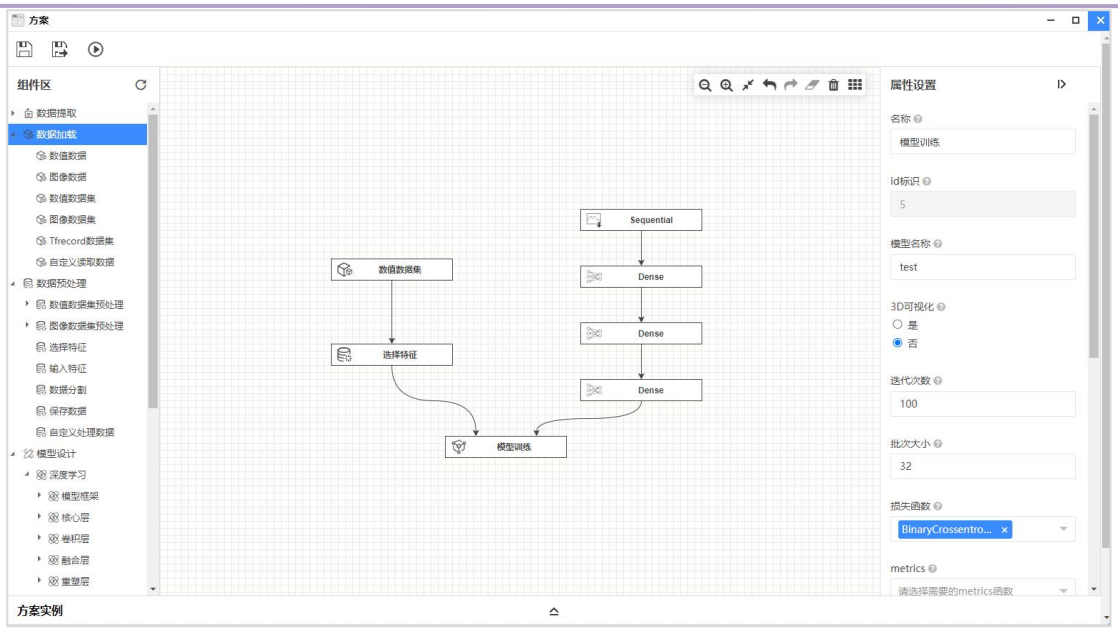


方案设计页面

“方案设计”左侧区域为组件区，组件区又根据组件属性分为“数据提取”，“数据加载”，“数据预处理”，“模型设计”，“模型库”，“算法库”，“模型结果可视化”，“模型预测”和“工具”。每一个组件都是一个模块的封装。

“方案设计”右侧区域为属性区，组件拖拽到画布区域后，属性设置面板就会从画布右侧滑出，用户可自行配置参数。当属性设置有误或未设置属性时，设计区的组件右侧会显示黄色叹号的图标，当鼠标移至此图标上时会显示出错误信息，必须纠正错误后才能够开始运行此方案。

画布右上角的工具栏可对画布区域和画布中的组件进行快捷操作，功能包含缩小、放大、实际比例、后退、前进、删除和清空。其中“删除”按钮仅会删除画布中选中的任意组件，“清空”则会删除整个画布中的内容，长按鼠标右键可拖动整个画布。



画布工具及组件属性配置

自定义函数管理

在自定义函数管理中可以创建和管理“损失函数”和“Metrics 函数”，选择相应的分类并点击新建卡片即可新建自定义函数。

创建成功的自定义函数会以卡片的形式显示在右侧区域中，卡片中的信息包括函数名称、ID 和创建时间，当鼠标停留在卡片右上角的“描述”按钮处时，会显示此函数的描述信息，点击卡片右上角的删除按钮即可删除当前自定义函数，删除操作不可恢复。

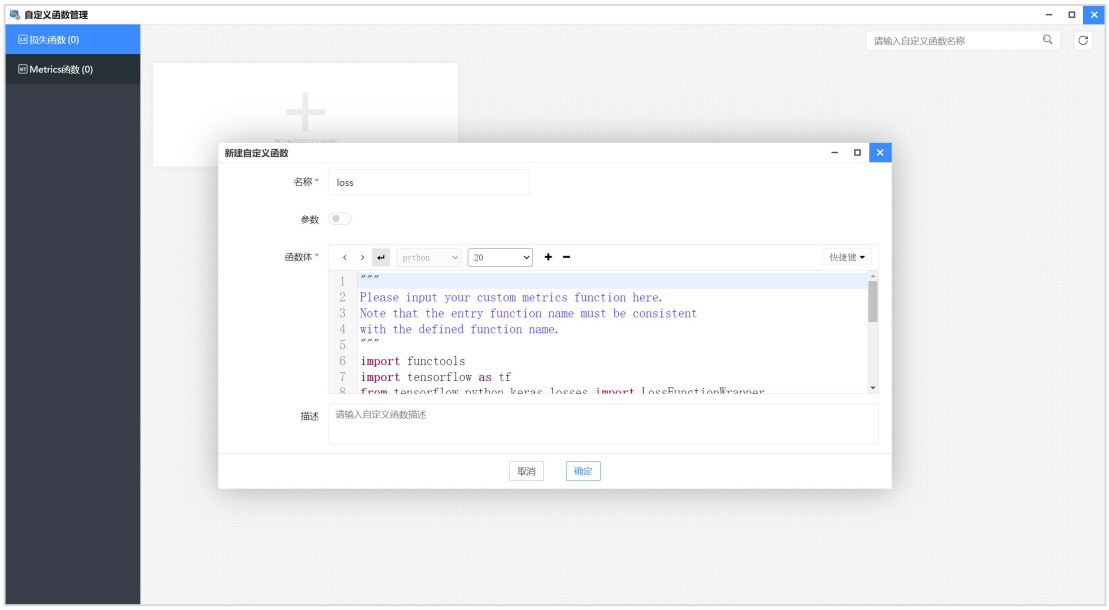
注意：

(1) 在此创建的自定义函数可在方案设计中被引用。

(2) 组件区域添加“模型评估”按钮，在案例详情页面可产生评估结果页面。

➢ 损失函数

进入“自定义函数管理”页面，选择左侧“损失函数”，点击“新建自定义函数”卡片，弹出新建窗口，如下图所示：



自定义损失函数

输入名称，并根据函数是否包含参数选择是否打开“参数”开关，在函数体部分输入自定义函数的内容，在描述部分输入对函数的描述或者备注信息，点击“确定”按钮，即可完成自定义损失函数的创建。

注意：函数“名称”的命名需要和函数入口名称一致。

损失函数内容如下：

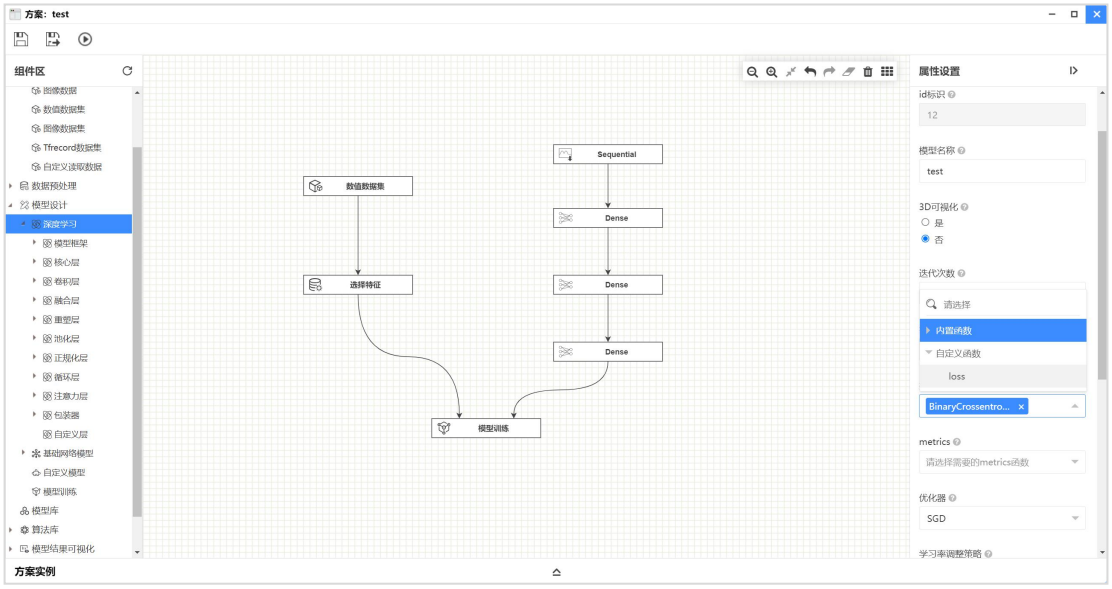
```
"""
Please input your custom metrics function here.
Note that the entry function name must be consistent
with the defined function name.
"""
import functools
import tensorflow as tf
from tensorflow.python.keras.losses import LossFunctionWrapper


def scaled_mean_square_error(y_true, y_pred, mean=0., std=1.):
    y_pred = (tf.convert_to_tensor(y_pred) - mean) / std
    y_true = (tf.cast(y_true, y_pred.dtype) - mean) / std
```

```
    return tf.keras.metrics.mean_squared_error(y_true, y_pred)


class my_loss_1(LossFunctionWrapper):
    def __init__(self, mean=0., std=1.,
name='scaled_mean_square_error'):
        super(my_loss_1,
self).__init__(functools.partial(scaled_mean_square_error, mean=mean,
std=std), name=name)
```

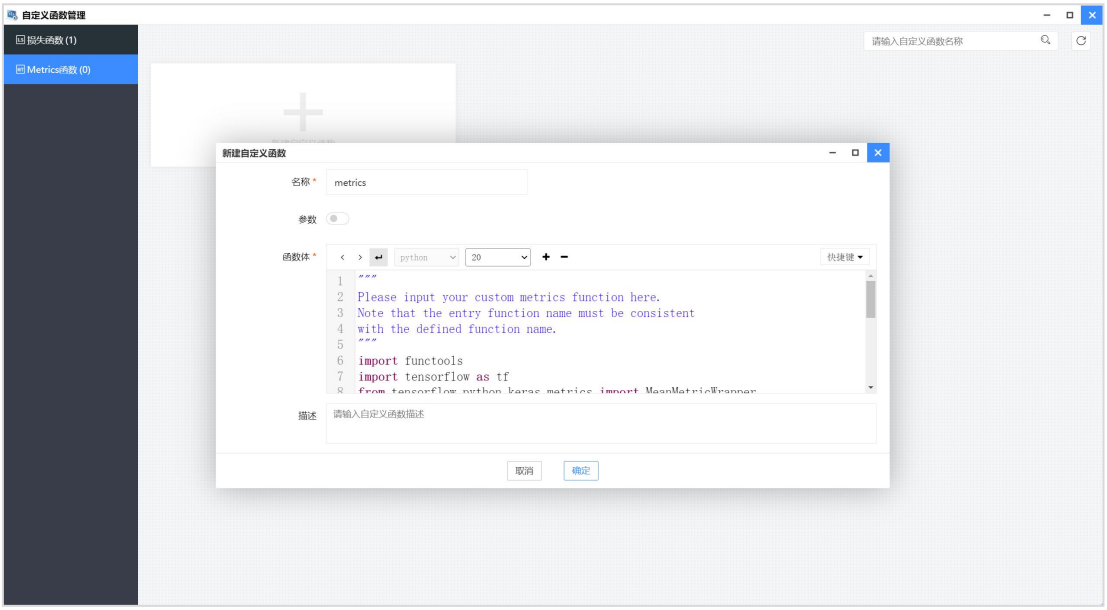
损失函数创建完成后在方案设计中训练深度学习方案结构时可以引用此函数，此函数会在自定义函数中显示。



属性区显示损失函数

➢ Metrics 函数

选择左侧“Metrics 函数”，点击“新建自定义函数”卡片，弹出新建窗口，如下图所示：



自定义 Metrics 函数

输入名称，根据函数是否包含参数选择是否打开“参数”开关，在函数体部分输入自定义函数的内容，在描述部分输入对函数的描述或者备注信息，点击“确定”按钮，即可完成自定义 metrics 函数的创建。

注意：函数“名称”的命名需要和函数入口名称一致。

metrics 函数内容如下：

```
"""
Please input your custom metrics function here.
Note that the entry function name must be consistent
with the defined function name.
"""
import functools
import tensorflow as tf
from tensorflow.python.keras.metrics import MeanMetricWrapper


def mean_scaled_relative_error(y_true, y_pred, epsilon=1.):
    y_pred = tf.convert_to_tensor(y_pred)
    y_true = tf.cast(y_true, y_pred.dtype)
```

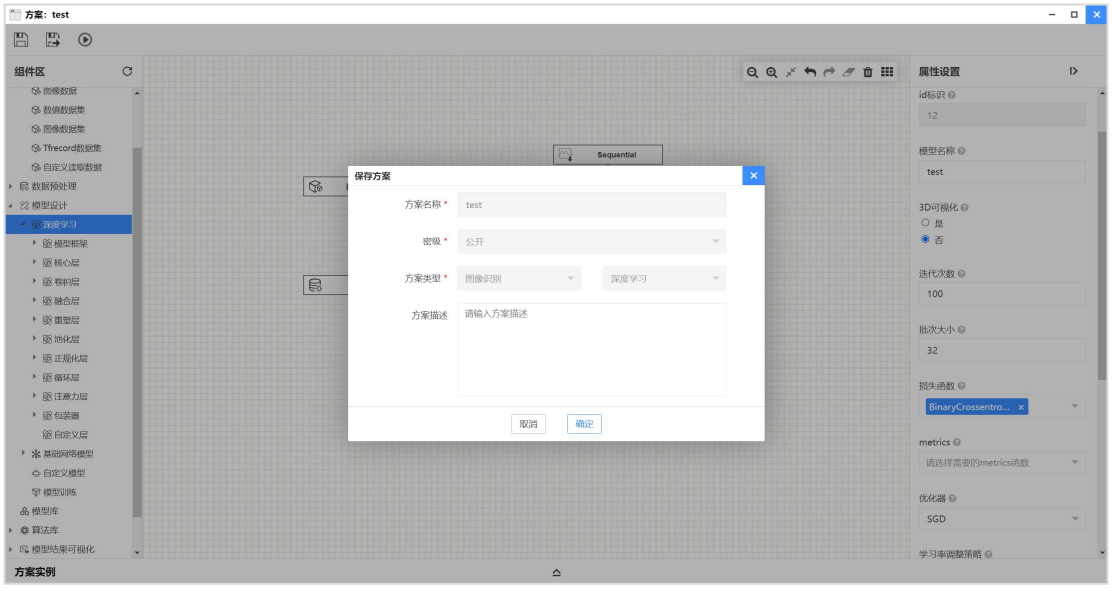
```
    return tf.keras.backend.mean(tf.abs(y_pred - y_true) /
(tf.abs(y_true) + epsilon), axis=-1)


 class my_metrics_1(MeanMetricWrapper):
    def __init__(self, epsilon=0.1,
name='mean_scaled_relative_error', dtype=None):
        super(my_metrics_1,
self).__init__(functools.partial(mean_scaled_relative_error,
epsilon=epsilon), name, dtype=dtype)
```

2.2.2.2. 操作

2.2.2.2.1. 方案保存

点击工具栏中的“保存”按钮，即可将方案描述和画布上的方案设计保存至数据库，当再次打开该方案时，画布上会显示最近一次保存的方案。“保存方案”页面，如下图所示：



保存方案页面

图中每个参数的具体含义如下：

模型名称：任何状态下均不可修改。

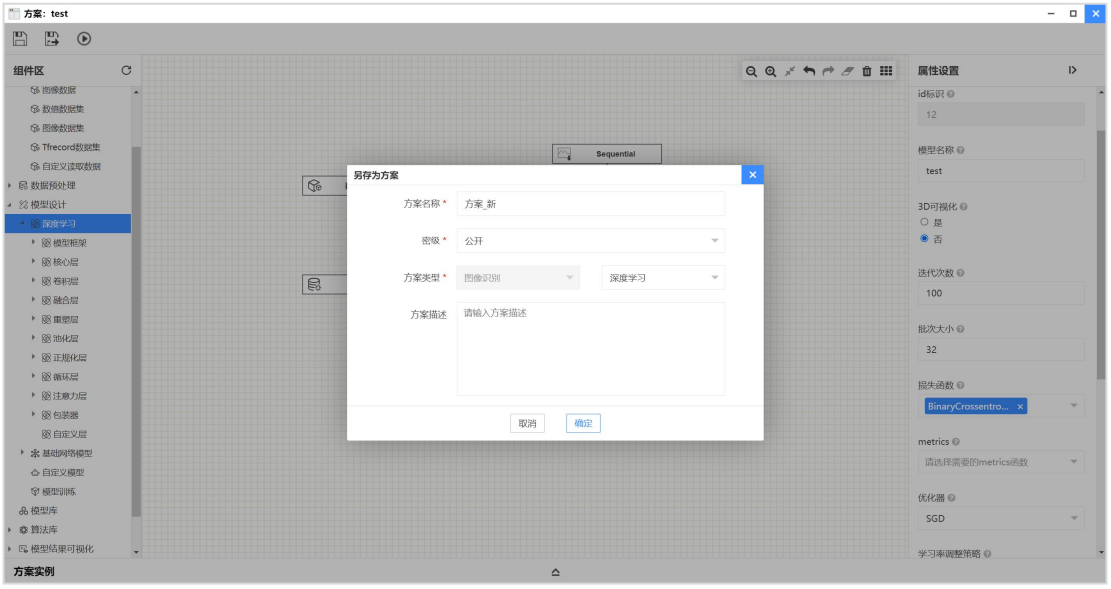
密级：任何状态下均不可修改。

模型类型：任何状态下均不可修改。

模型描述：用于描述方案的相关内容，选填。

2.2.2.2.2. 方案另存为

点击工具栏中的“另存为”按钮，即可将方案另存为一个新的方案，方案可以是已保存的和未保存的。“另存为方案”页面。如下图所示：



另存为方案页面

图中每个参数的具体含义如下：

方案名称：用户自定义，但是不能与现有的模型名称一致，必填。

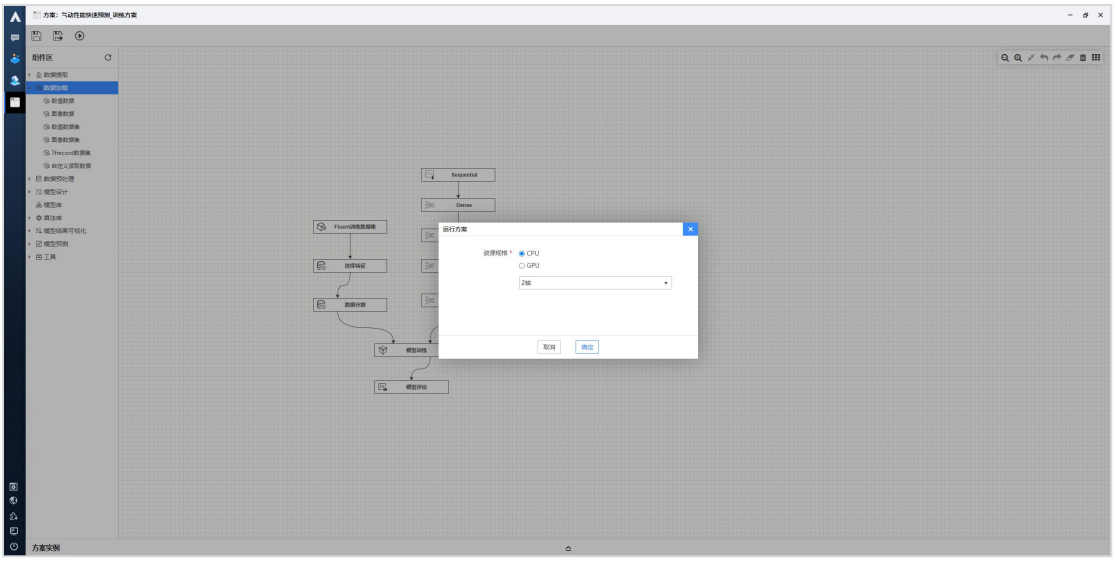
密级：对于另存的方案可根据需要修改密级。

方案类型：默认为当前方案的方案类型，可以选择，也可以自定义方案类型，必填。

方案描述：用于描述方案的相关内容，选填。

2.2.2.2.3. 方案运行

方案设计完成后，点击“运行”按钮，弹出“运行方案”窗口，选择训练方案需要使用的资源，资源选择后，点击“确定”按钮，即可开始训练。如下图所示：



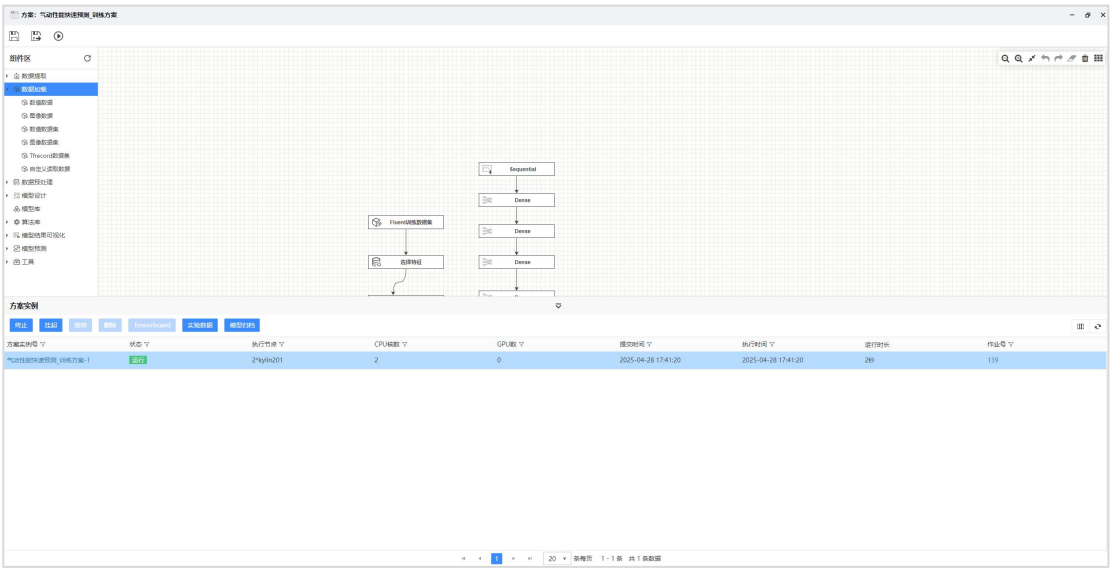
资源配置页面

点击运行按钮，打开选择资源页面，同时进行方案保存。选择资源后，点击确定按钮，即可开始训练方案。

注意：在方案设计页面中，点击“运行”按钮时，错误和问题对应如下：

- 画布中不存在组件时，提示“组件不能为空”。
- 组件的锚点没有连线、必填项属性为空时，提示“请检查异常组件并清除异常”。

开始训练方案后，会自动打开底部方案实例滑块，如下图所示：

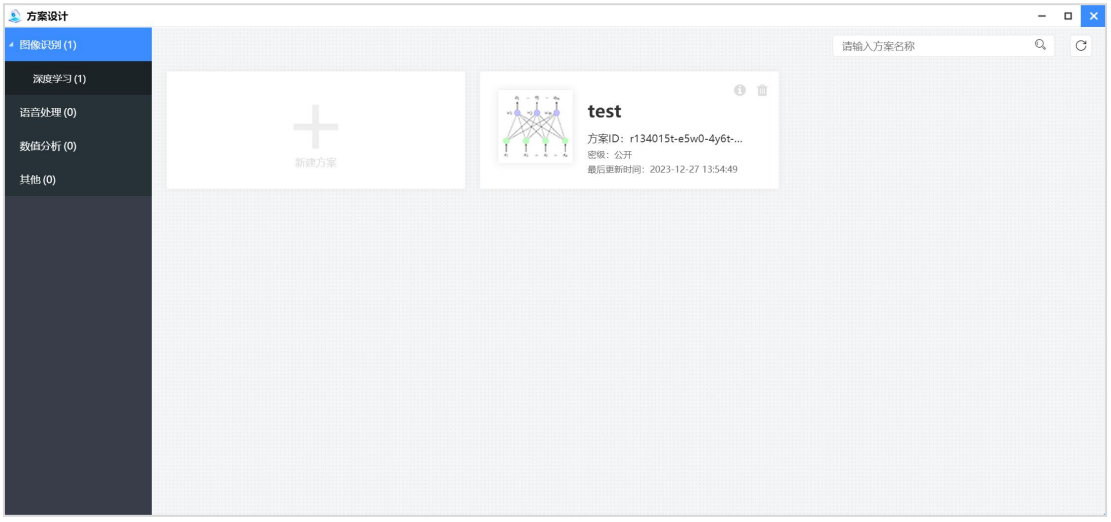


方案实例滑块

底部方案实例滑块除了只能显示该方案的所有实例外，功能操作与方案实例一致，具体操作详情请看“方案实例”章节。

2.2.2.3. 管理

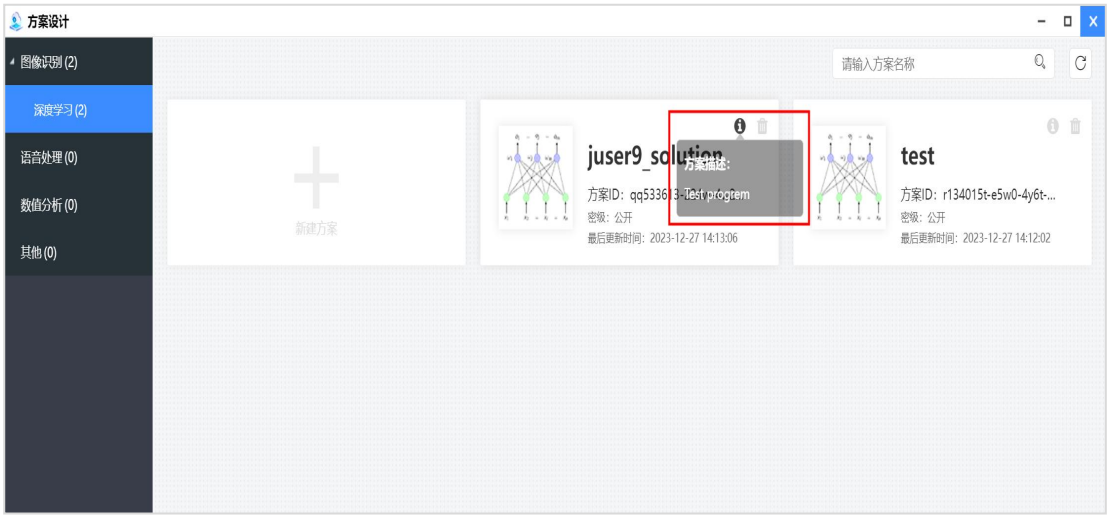
方案新建成功后，在方案设计应用中会自动生成了一个方案卡片，卡片内容包含方案名称、方案 ID、最后更新时间，点击方案卡片的右上角功能图标，支持对方案进行“发布”，“查看描述”和“删除”操作。



方案设计管理

2.2.2.3.1. 查看描述

鼠标移至方案卡片上的“描述”按钮时，可以查看方案的描述信息，如下图所示：



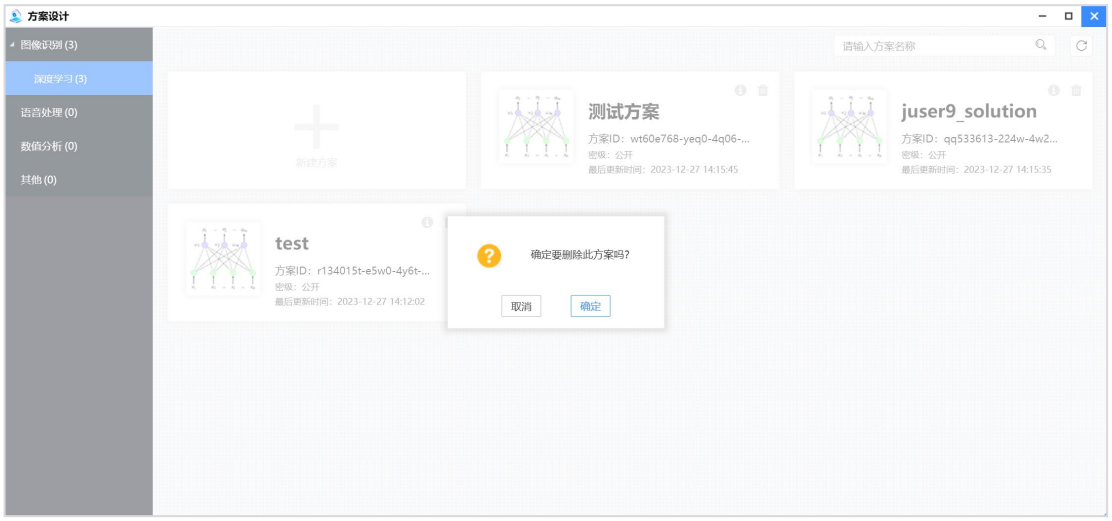
方案描述信息

点击“描述”的图标按钮，可在弹出窗口中修改方案的描述信息，如下图所示：

修改方案描述

2.2.2.3.2. 删除

点击“删除”按钮，可以把该方案卡片删除。



删除方案

2.2.2.4. 方案实例

方案实例中以列表的形式记录了所有方案的历史运行记录，用户可以通过实例号、名称等筛选功能快速过滤实例。

用户可在方案实例应用中查看自己的所有实例，或在具体的方案设计页面中，通过点击画布下方的方案实例滑块，在展开的面板中查看当前方案实例的信息，亦可对其进行终止、挂起、模型归档、tensorboard 等操作。

方案实例列表

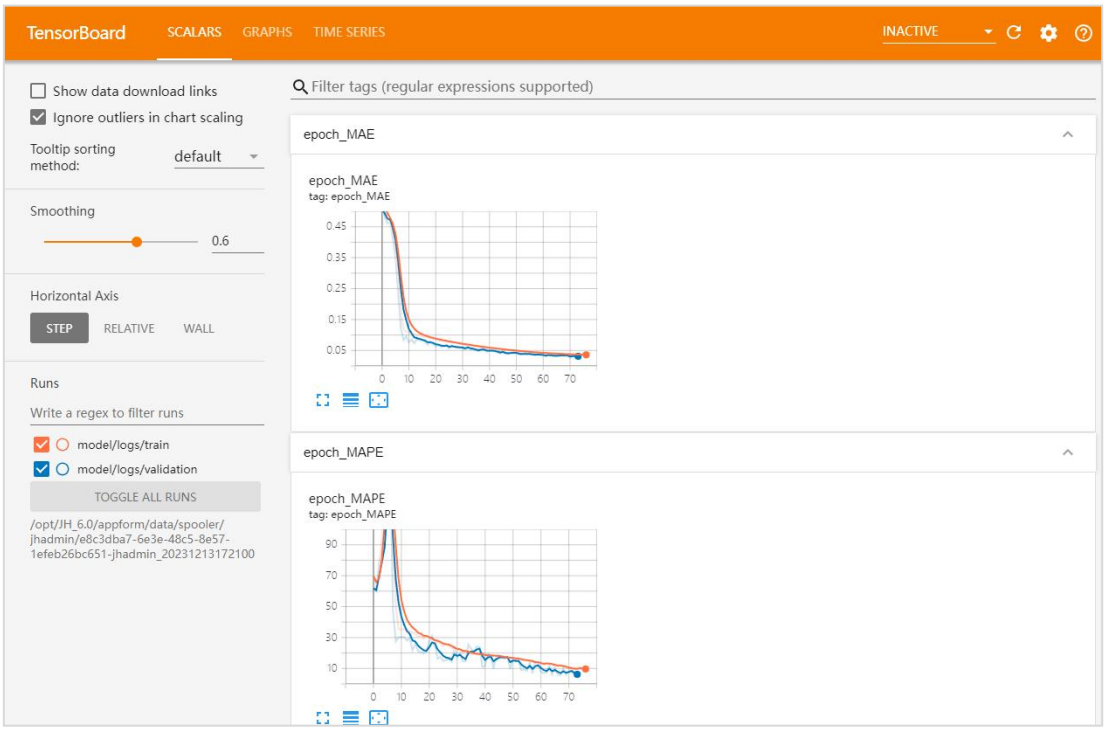
- 功能按钮：

方案实例支持对方案实例进行“终止”“挂起”“继续”“删除”、Tensorboard、实验数据和模型归档操作。

当深度学习的方案运行完成后，点击“Tensorboard”按钮可以以图表形式

查看当前实例运行结果。

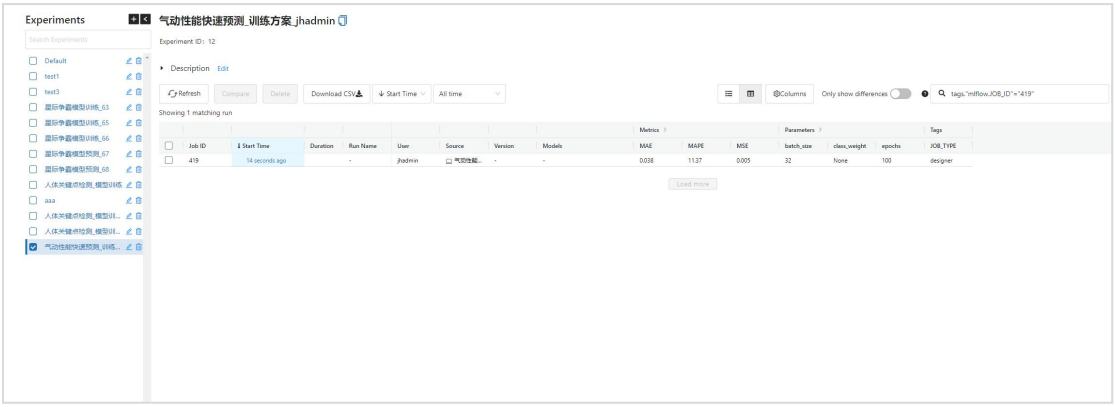
方案实例 tensorboard 按钮



Tensorboard

模型训练的方案运行过程中，点击“实验数据”跳转到实验管理页面，查看该方案的训练过程参数。

方案实例实验数据按钮

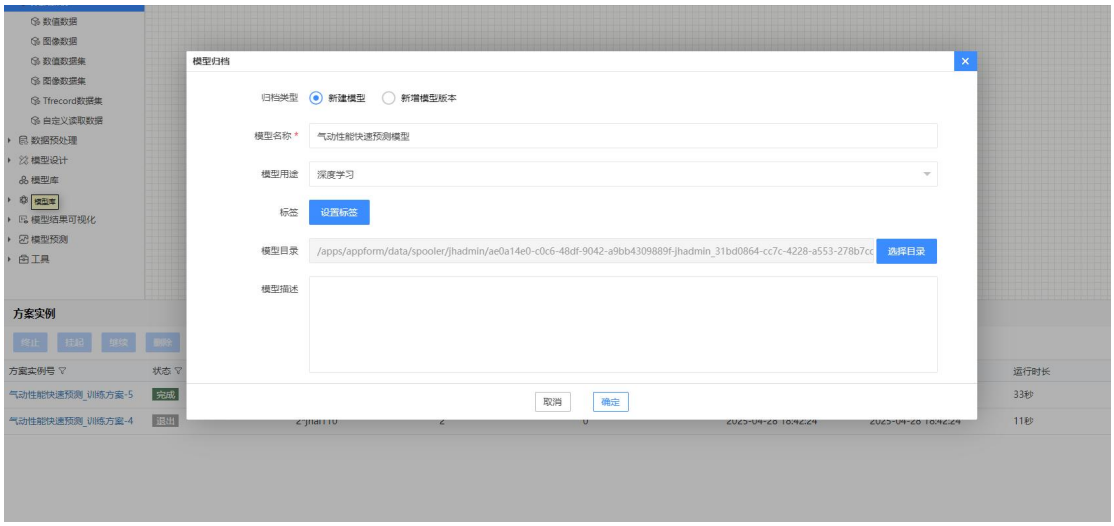


实验数据

模型训练方案运行完成后，点击“模型归档”按钮归档生成的模型到模型库中。

方案实例模型归档按钮

“模型归档”页面如下：



模型归档

归档类型：分为新建模型和新增模型版本。

当选择“新建模型”时，设置参数如下所示：

模型归档类型为新增模型

模型名称：输入归档模型的模型名称。

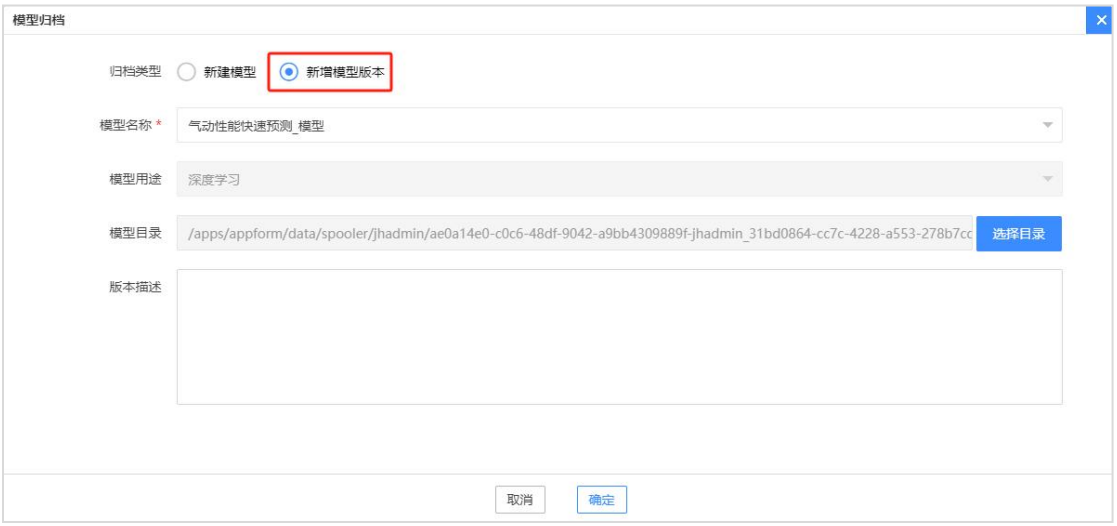
模型用途：选择模型用途，选择机器学习或者深度学习。

标签：设置模型的标签，标签可选内容和模型库标签内容一致。

模型目录：默认回填作业运行 spooler 目录，也可以选择 spooler 目录下的子目录。

模型描述：输入归档模型的描述信息。

当选择归档类型选择“新增模型版本”时，模型归档页面如下：



模型归档类型为新增模型版本

模型名称：选择模型名称，模型名称来自模型库中所有的机器学习模型和深度学习模型。

模型用途：模型用途会根据模型名称自动回填模型名称相应的模型用途。

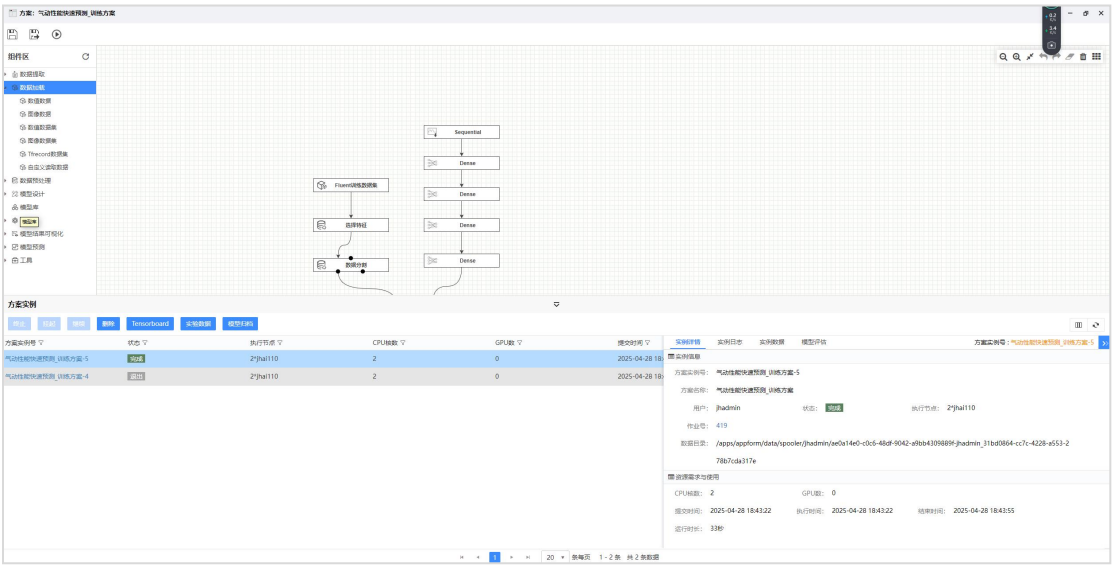
模型目录：默认回填作业运行 spooler 目录，也可以选择 spooler 目录下的某个子目录。

版本描述：添加对新增版本模型的描述信息。

- 方案实例滑块：

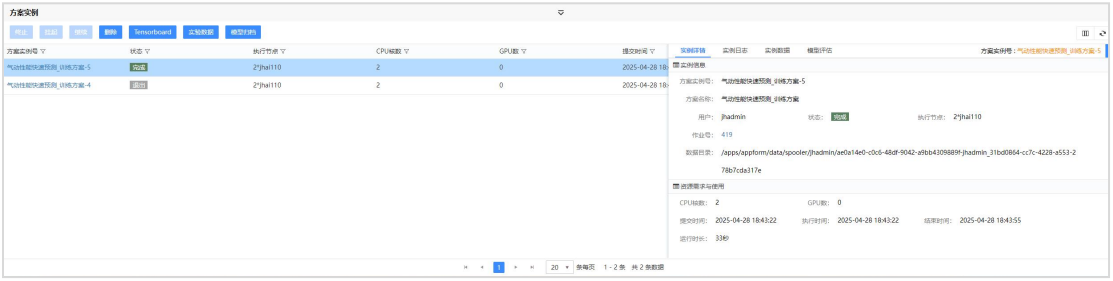
点击“方案实例号”或者双击方案实例列表中的某条记录，会打开右侧滑块。滑块有三个标签页，分别为实例详情、实例日志和实例数据。

当组件中包含“模型评估”组件，且方案运行成功，滑块会有四个标签页，分别为实力详情、实例日志、示例数据和模型评估。如下图所示：



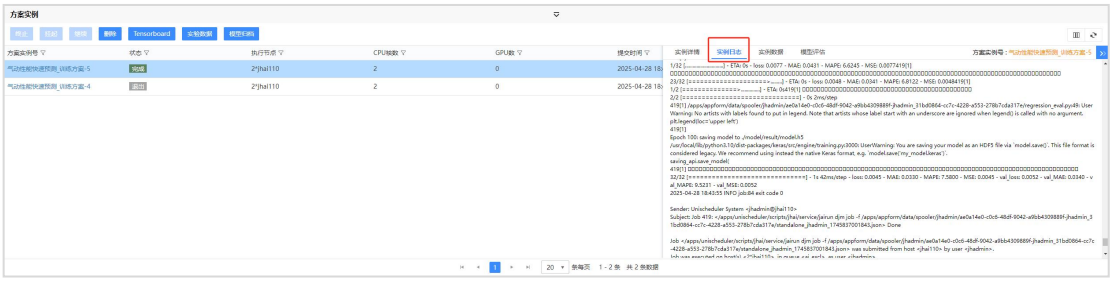
方案设计

实例详情：显示方案实例的基本信息、资源信息。点击作业号，可以查看该实例关联的作业信息。实例详情中包含方案实例关联的数据集信息，包括数据集名称、数据集类型、字符集、分隔符以及数据保存目录。实例详情中包含方案实例关联的模型信息，实例详情中会显示模型信息，包括模型名称、模型类型、超参数(仅深度学习模型)。



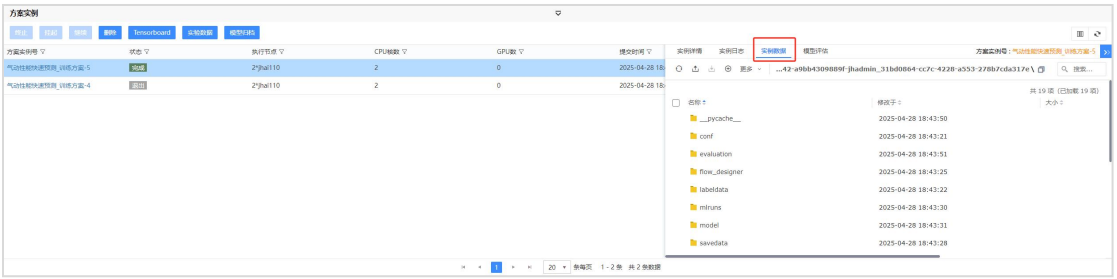
方案实例滑块

实例日志：点击实例日志按钮，打开实例日志界面。方案实例的状态处于正在运行时，该实例日志会实时输出。方案实例的状态处于完成时，会显示该方案实例的所有日志。



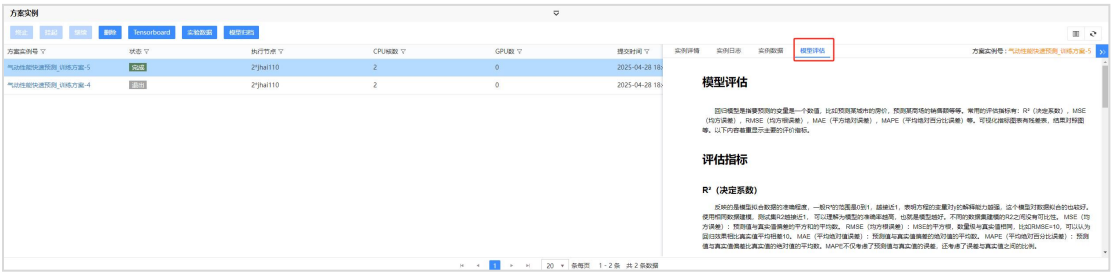
方案实例日志

实例数据：点击实例数据按钮，打开实例数据界面，显示该方案实例包含的所有相关文件。



方案实例数据

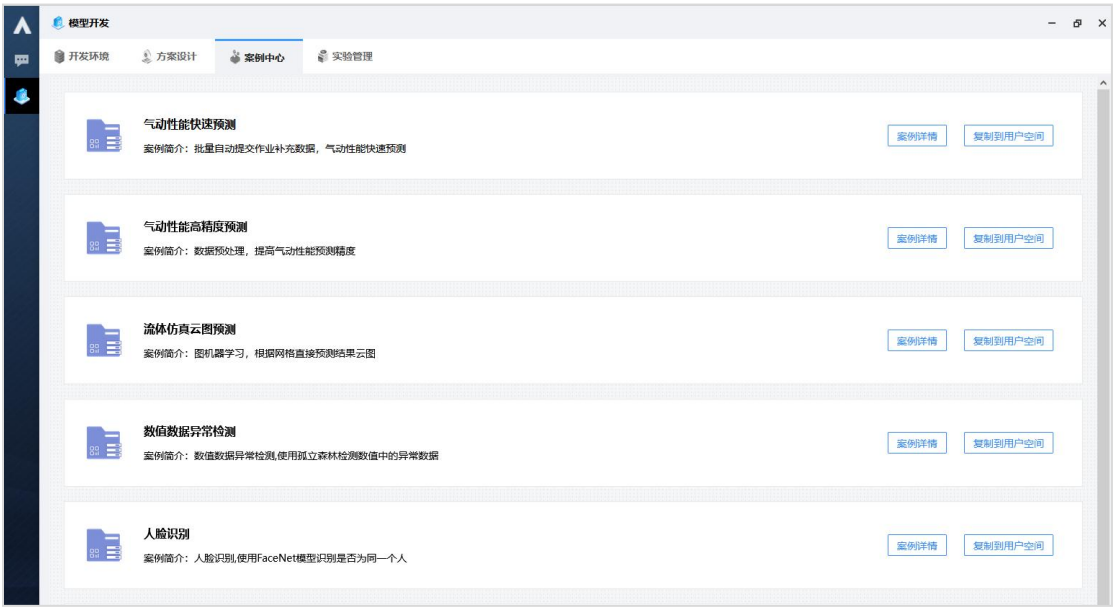
模型评估：点击模型评估按钮，打开模型评估界面，显示该模型的评估结果，如下图所示：



方案模型评估

2.2.3. 案例中心

案例中心内置了一些AI案例，方便用户了解AI平台功能的基本使用和流程，用户可以复制案例中心的内置案例到自己的用户空间，查看和运行相关案例。桌面双击“模型开发”应用，选择“案例中心”。如下图所示：



案例中心

2.2.3.1. 案例详情

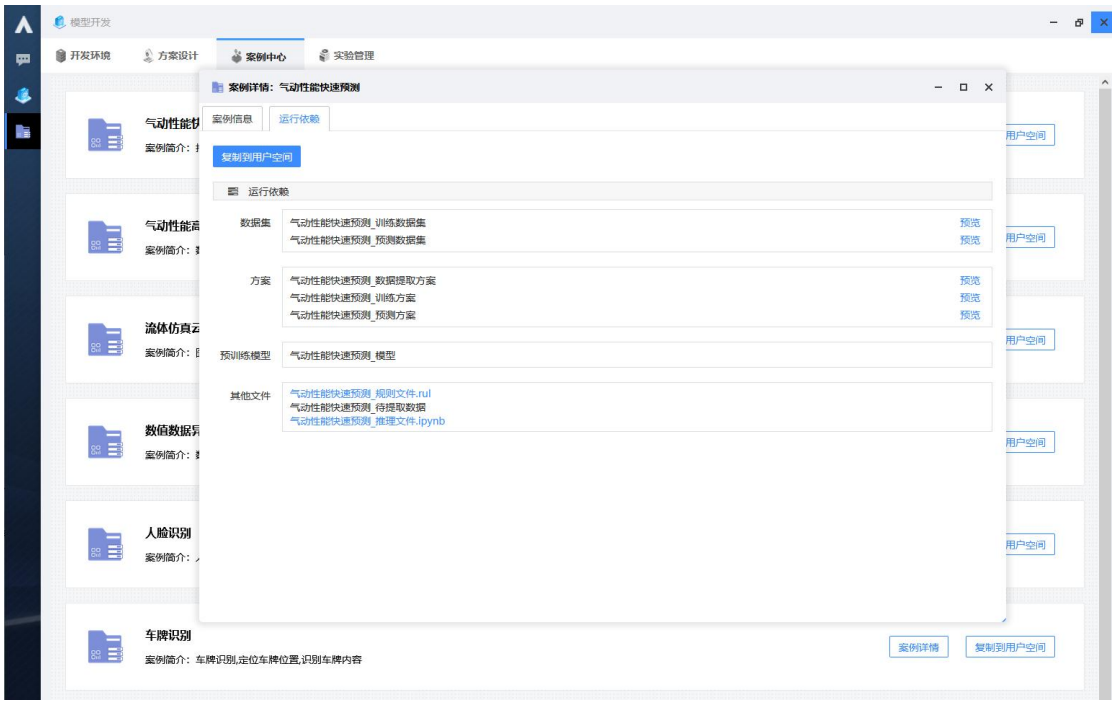
目前案例中心中集成了 10 个案例，包括：气动性能快速预测、气动性能高精度预测、流体仿真云图预测、数值数据异常检测、人脸识别、车牌识别、火灾检测、行人轮廓检测、人体关键点检测和强化学习。

案例详情展示了案例的详细介绍和使用方法。以“气动性能快速预测”案例为例，操作步骤如下所示：点击“气动性能快速预测”案例卡片上的“案例详情”按钮，弹出“案例详情”窗口，如下图所示：

案例详情

案例信息界面详细描述了案例的模型背景、数据准备、模型训练、模型效果和硬件要求，用户可以按照该步骤使用案例。

切换至“运行依赖”标签页，用户可以查看案例所需要的数据、方案、模型等依赖。如下图所示：

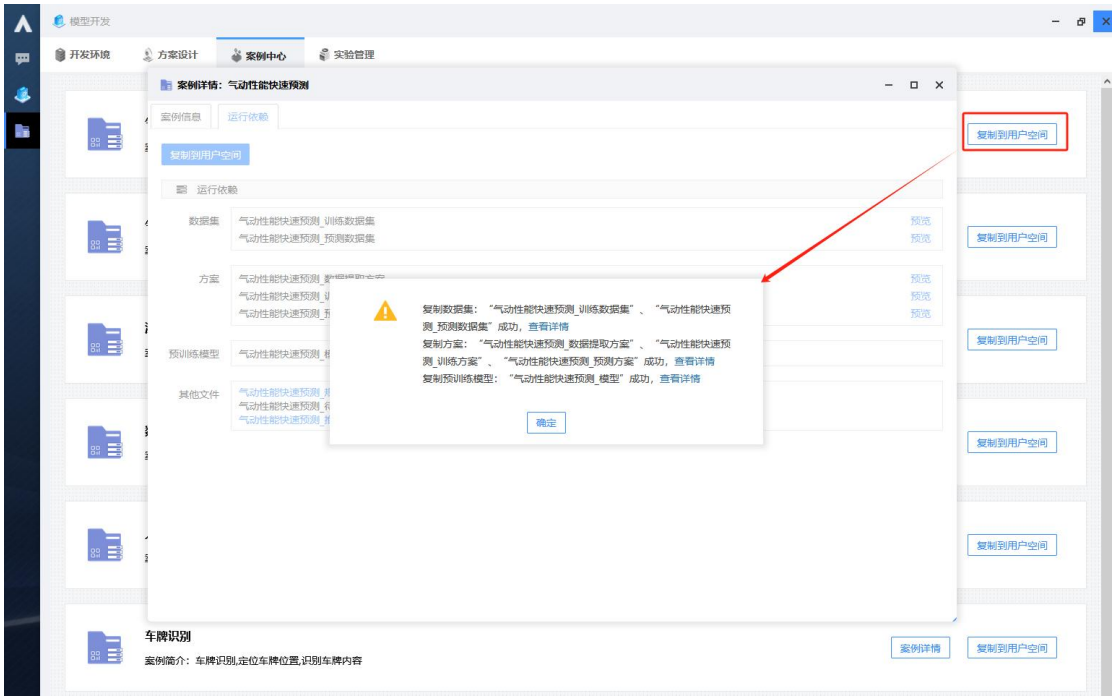


运行依赖页面

其中，“预览”按钮，可以查看数据集的数据或者方案的方案结构；“其他文件”框中提供了推理文件，支持下载。

2.2.3.2. 案例使用

点击“复制到用户空间”按钮，可以将案例依赖的所有数据文件复制到用户自己的环境中。以“气动性能快速预测”案例为例，操作步骤如下所示：点击“气动性能快速预测”案例卡片右边的“复制到用户空间”按钮，将案例复制到自己的空间，如下图所示：

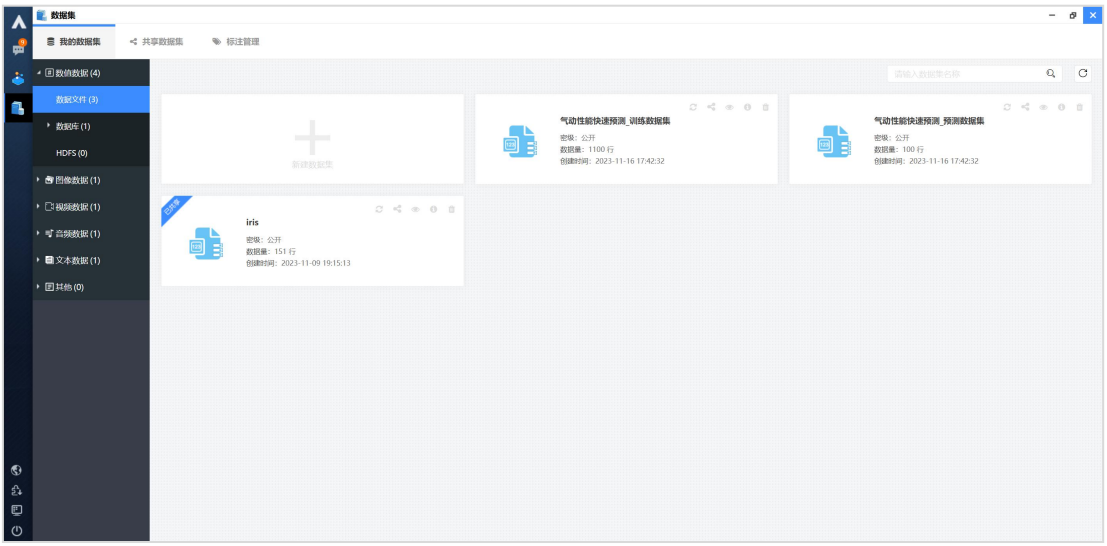


复制到用户空间

注意：案例的每次复制都会覆盖用户本地的同名数据集、方案和模型。因此用户可以将修改过的方案进行另存为，防止被覆盖导致的丢失。

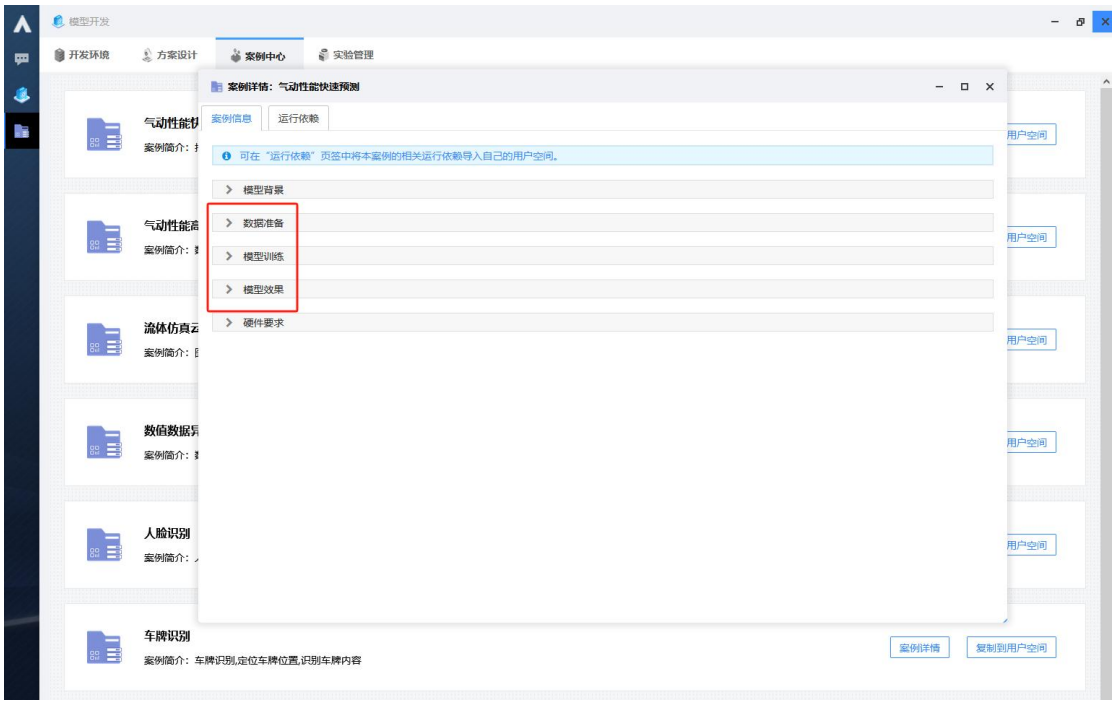
可以点击提示中的“查看详情”跳转链接，查看复制过来的数据集、方案或者模型。以查看复制过来的数据集为例，操作步骤如下所示：点击复制数据集项的“查看详情”跳转链接，可以在“数据集管理”-“我的数据集”中，查看数

据集信息，如下图所示：

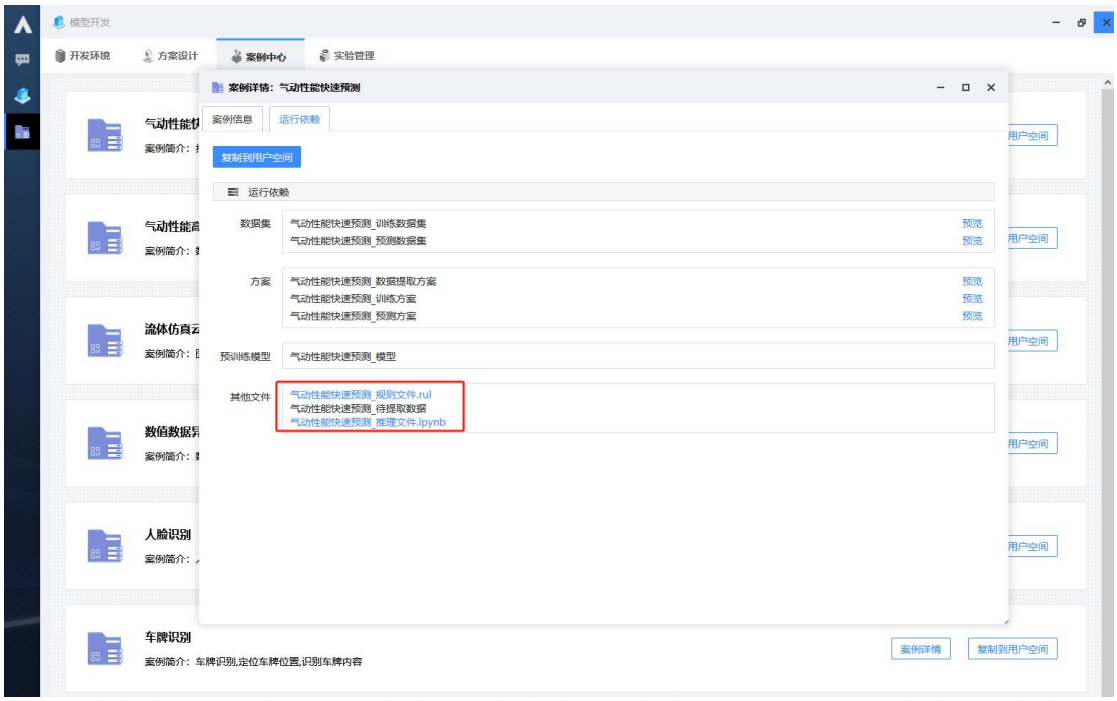


查看数据集信息

根据“案例详情”中的步骤：“数据准备”-“模型训练”-“模型效果”，依次准备数据、训练数据、部署服务、推理数据即可，推理文件在“运行依赖”中下载，如下图所示：



查看案例信息



推理文件下载

2.2.4. 实验管理

实验管理为人工智能平台提供跟踪、比较和记录模型训练指标的工具，训练指标包括训练的参数，超参数，评估结果，模型文件等，用户可以通过实现管理界面查看和对比多次实验的数据。

涉及的一些关键字：

实验（experiment）：相当于项目名，所有的调参都是针对这个实验。

运行（Run）：实验的一次运行记录。

参数(parameter)：实验运行的参数、超参数。

指标(metric)：实验运行产生的指标结果。

文件(artifact)：记录**实验运行相关的**文件，可以包括代码、日志或保存的模型。

2.2.4.1. 示例程序

如果需要使用实验管理的功能，需要在代码中集成 libaiflow API，目前仅支持 Python。

自动记录：Tensorflow 和 Pytorch 框架，支持通过 libaiflow.tensorflow.autolog 和 libaiflow.pytorch.autolog 提供的 Callback 机制实现自动收集 tensorflow 和 pytorch 实验的参数，指标等多种实验训练数据。下面展示 autolog 自动收集数据的关键代码，（完整样例代码参考附录四样例一）。

```
import libaiflow
expt = libaiflow.init(experiment_name="mnist_tf2_cpu")
libaiflow.tensorflow.autolog()
```

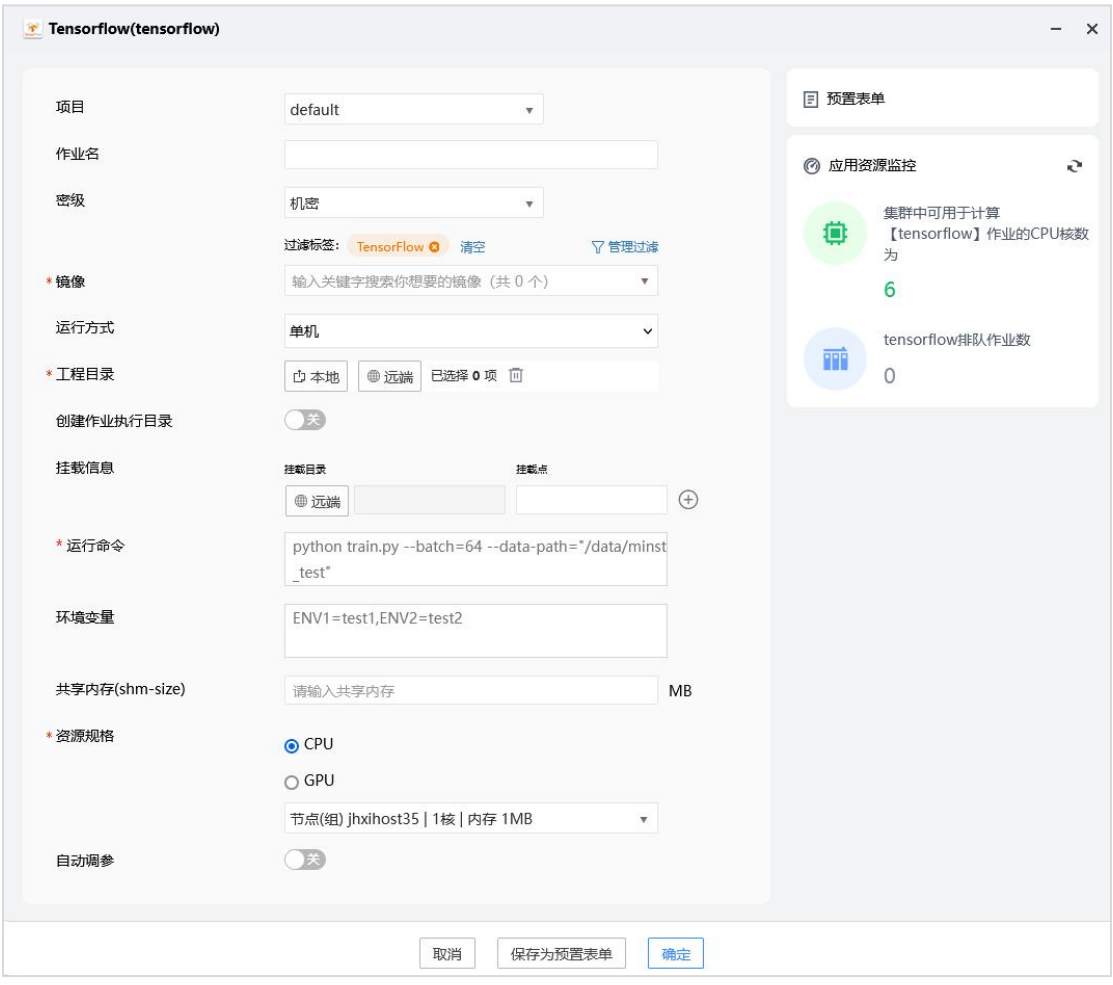
手动记录：通过调用指标和超参数相关的 api 记录和收集数据（完整样例代码参考附录四样例二）。

```
import libaiflow
expt = libaiflow.init(experiment_name="pytorch-sample")
……
with libaiflow.start_run(experiment_id=expt.experiment_id):
      lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
      lr.fit(train_x, train_y)
      predicted_qualities = lr.predict(test_x)
      (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)
      print("Elasticnet model (alpha=%f, l1_ratio=%f):" % (alpha,
l1_ratio))
      print("  RMSE: %s" % rmse)
      print("  MAE: %s" % mae)
      print("  R2: %s" % r2)
      libaiflow.log_param("alpha", alpha)
      libaiflow.log_param("l1_ratio", l1_ratio)
```

```
    libaiflow.log_metric("rmse", rmse)
    libaiflow.log_metric("r2", r2)
    libaiflow.log_metric("mae", mae)
```

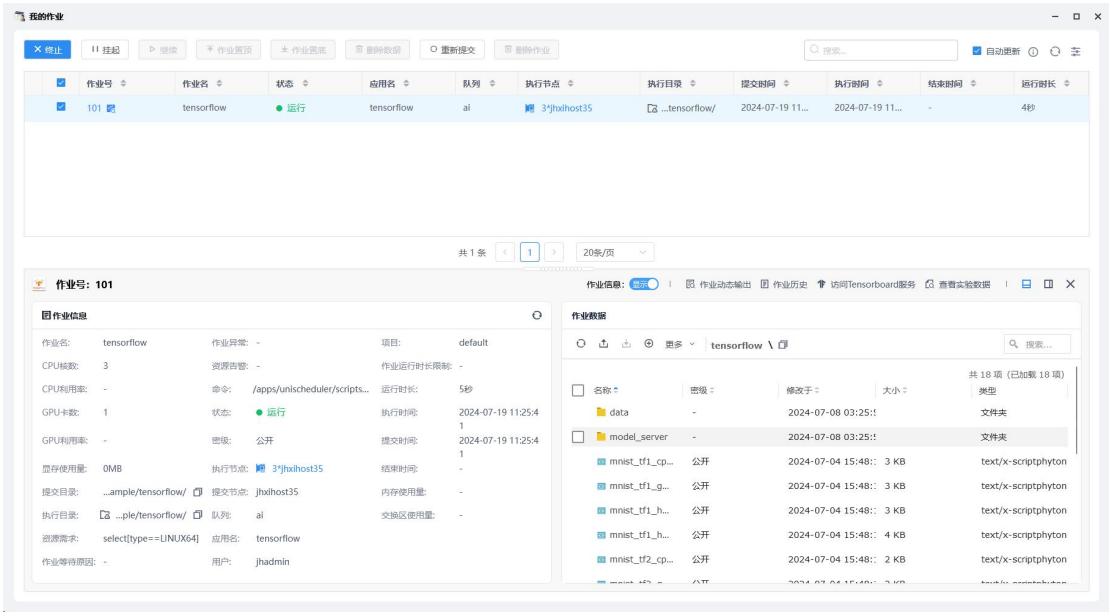
2.2.4.1.1. 提交 AI 作业

人工智能的内置训练框架镜像中已经默认安装 Libaiflow SDK，如果用户的模型训练代码中，已经集成了 libaiflowSDK，可以通过 Web 页面提交 AI 作业进行模型训练。以提交 Tensorflow 作业为例，如下图所示：



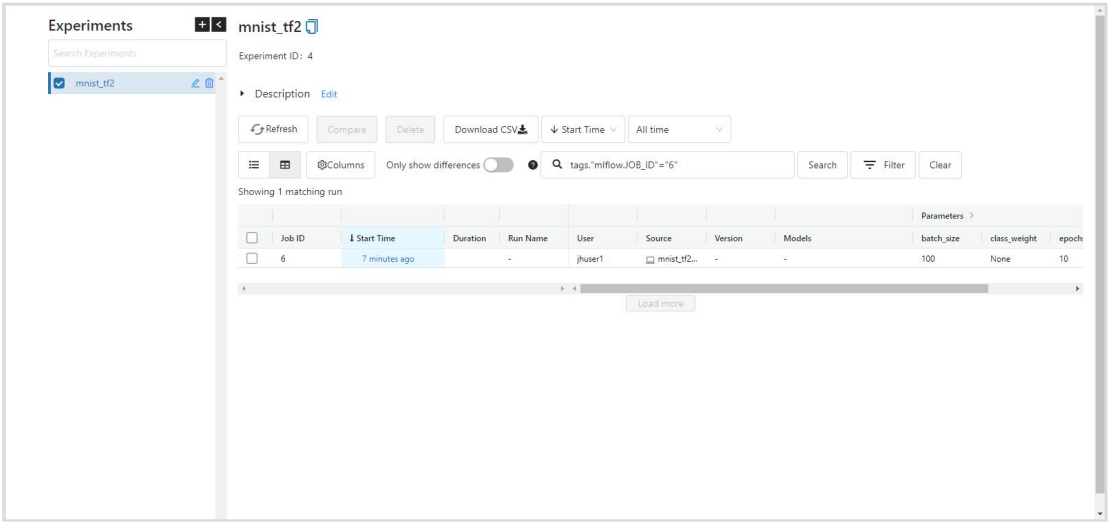
提交 AI 作业

Tensorflow 作业提交后，会自动打开“我的作业”页面，方便用户在此查看提交作业的信息，如下图所示：



注意：仅当模型训练代码中调用了 libaiflow SDK，才能查看实验数据。

点击“查看实验数据”按钮，可以查看实验数据，如下图所示：



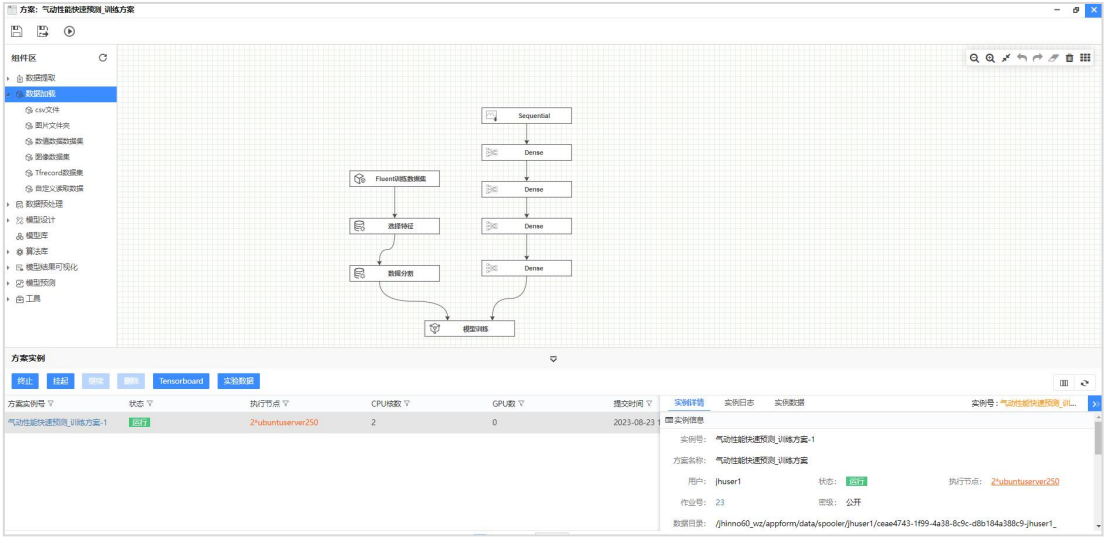
查看实验数据

2.2.4.1.2. 可视化建模

使用可视化建模方式构建的方案，已默认集成了 Libaiflow SDK，会自动记录相关的参数和指标数据

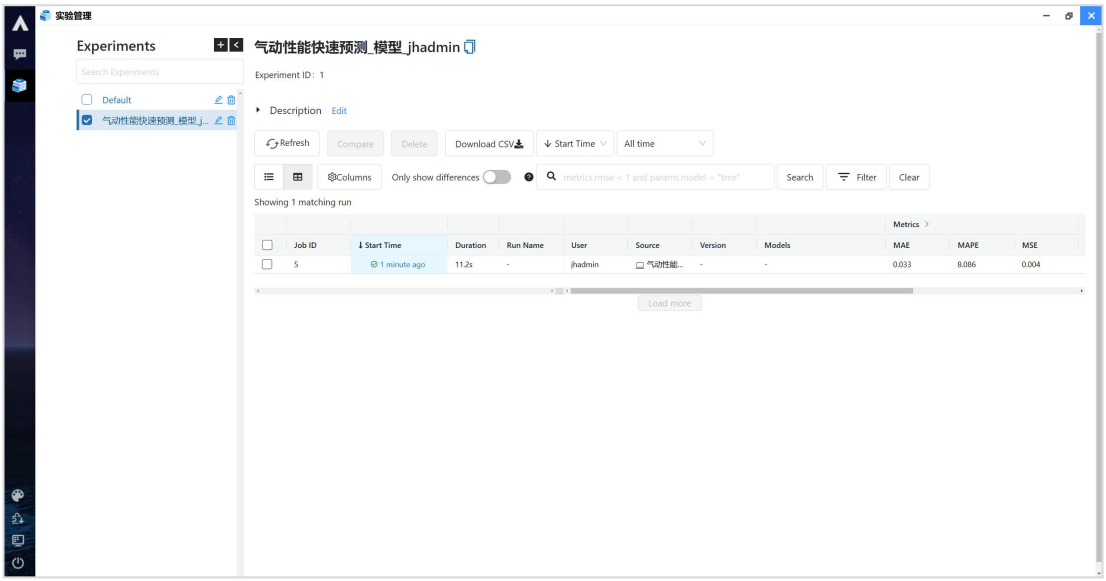
打开方案设计管理，进入方案设计器界面，构建模型结构流程、训练模型，

如下图所示：



运行方案

点击“实验数据”，可以打开实验管理页面，如下图所示：



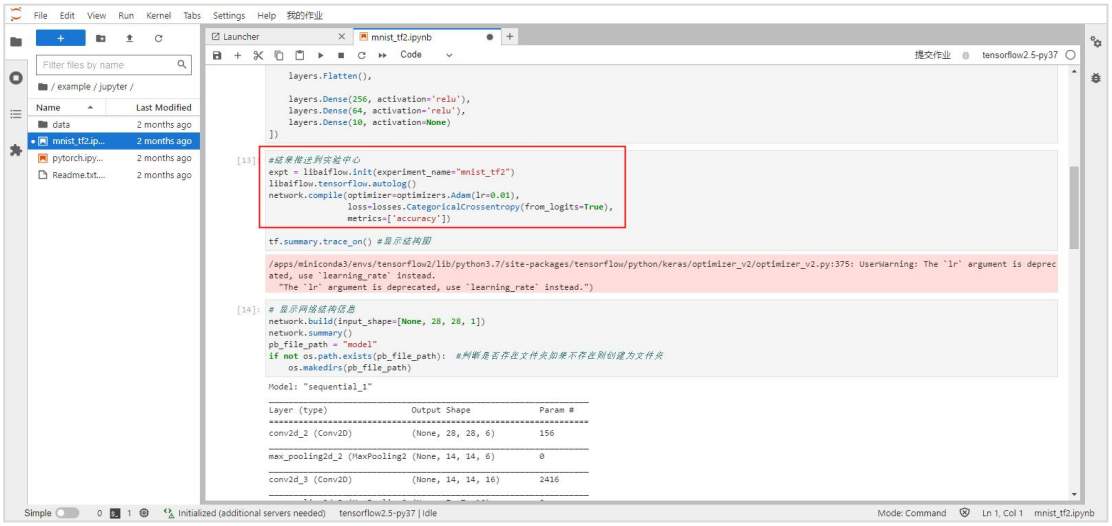
查看实验数据

2.2.4.1.3. WebIDE

WebIDE 类型的开发环境已经默认安装 libaiflow SDK，在 WebIDE 中运行集成了 libaiflow SDK 的代码时，会记录相关的实验数据，

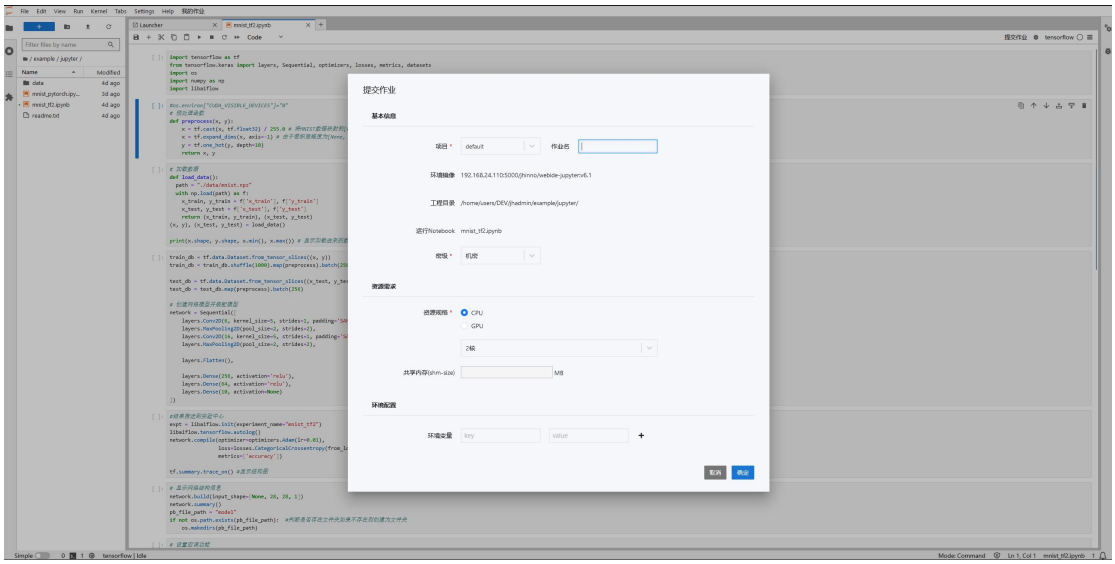
访问 Jupyter 服务，运行集成了 libaiflow SDK 的代码，会推送参数和指标

到实验管理。如下图所示：



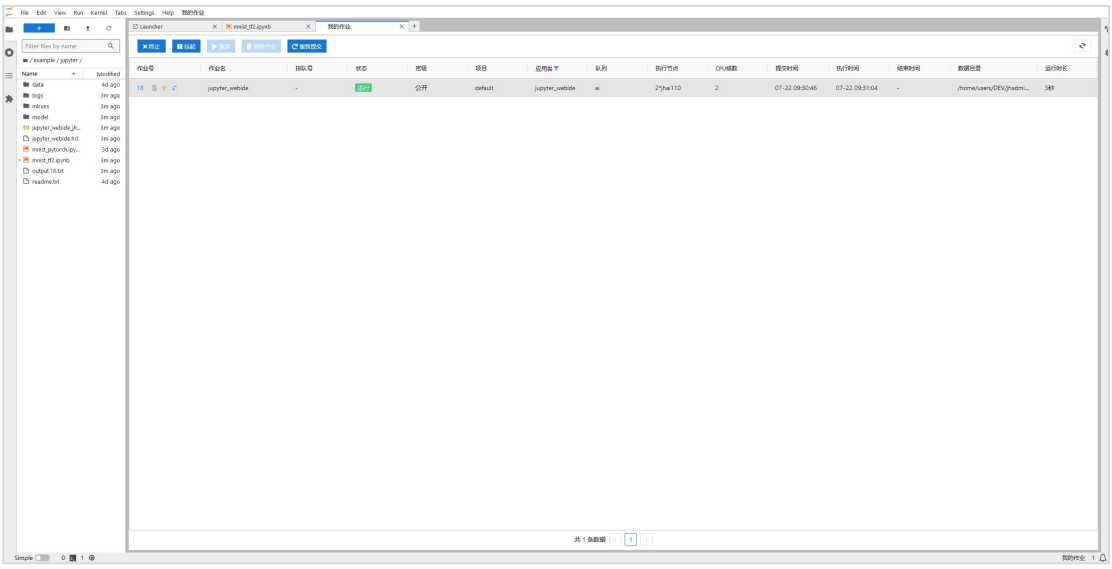
“Jupyter”服务

点击“提交作业”按钮，提交作业，如下图所示：



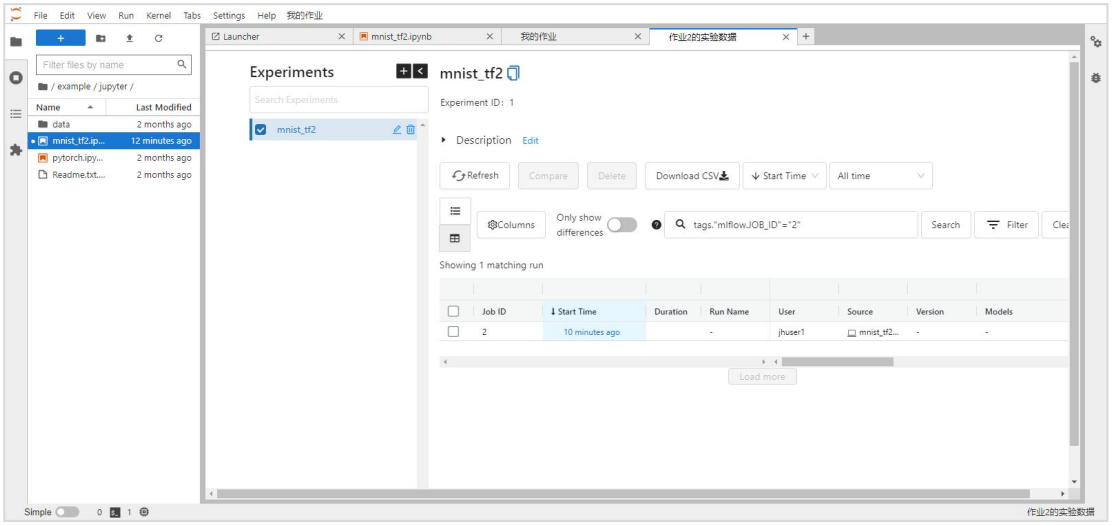
提交作业

点击“我的作业”按钮，打开我的作业 tab 页，如下图所示：



我的作业

点击作业实例上的“实验管理”图标按钮，查看实验数据，如下图所示：



查看实验数据

2.3. AI 模型训练

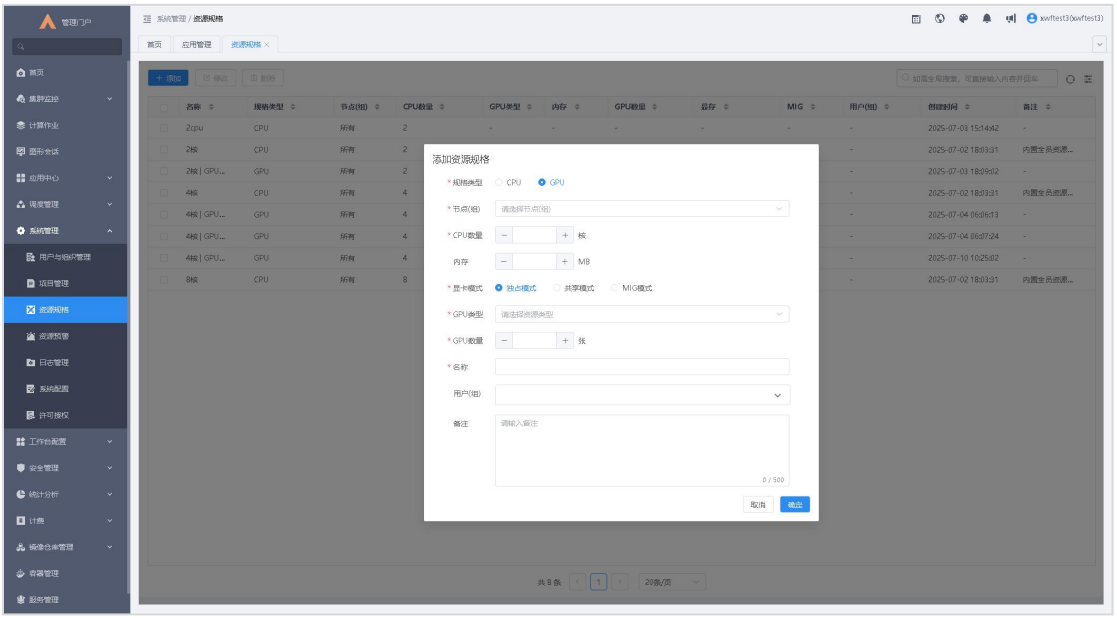
人工智能平台集成了主流深度学习框架，包括 Tensorflow、Pytorch、Mxnet、PaddlePaddle、MindSpore 和 DeepSpeed，并为用户提供了直观的任务提交 Web 页面和命令行方式提交。

通过 Web 门户，用户能够方便地提交训练作业，灵活指定训练所需的资源规

格，包括 CPU、内存、GPU、GPU 类型。用户提交完成后可以在作业管理中，实时查看训练进度，训练日志，以及访问 Tensorboard、VisualDL、MindInsight 等可视化服务。并且平台支持多种框架的并行训练，同时为用户提供了对不同资源规格的选择，包括单机单 GPU、单机多 GPU，以及多机多 CPU。

在创建资源规格时，支持指定独占、共享和 MIG 三种模式，允许指定资源的提交节点（组），更加细粒度的控制资源分配情况，并且对于不同的资源可以指定部分用户（组）可见。

在管理门户中，选择“系统管理”下的“资源规格”来创建，如下图所示：



添加 AI 资源规格

资源规格参数：

规格类型：选 CPU 或 GPU 主导资源分配，区分计算任务类型。

节点（组）：任务运行的节点分组，决定调度目标。可在调度管理下的节点与节点组中创建。

CPU 数量：需要申请的 CPU 核心数。

内存：以 MB 为单位的内存容量，满足程序运行需求。

显卡模式：独占（独享 GPU）、共享（多任务共用）、MIG（切分 GPU 为独立实例），适配不同 GPU 复用需求。

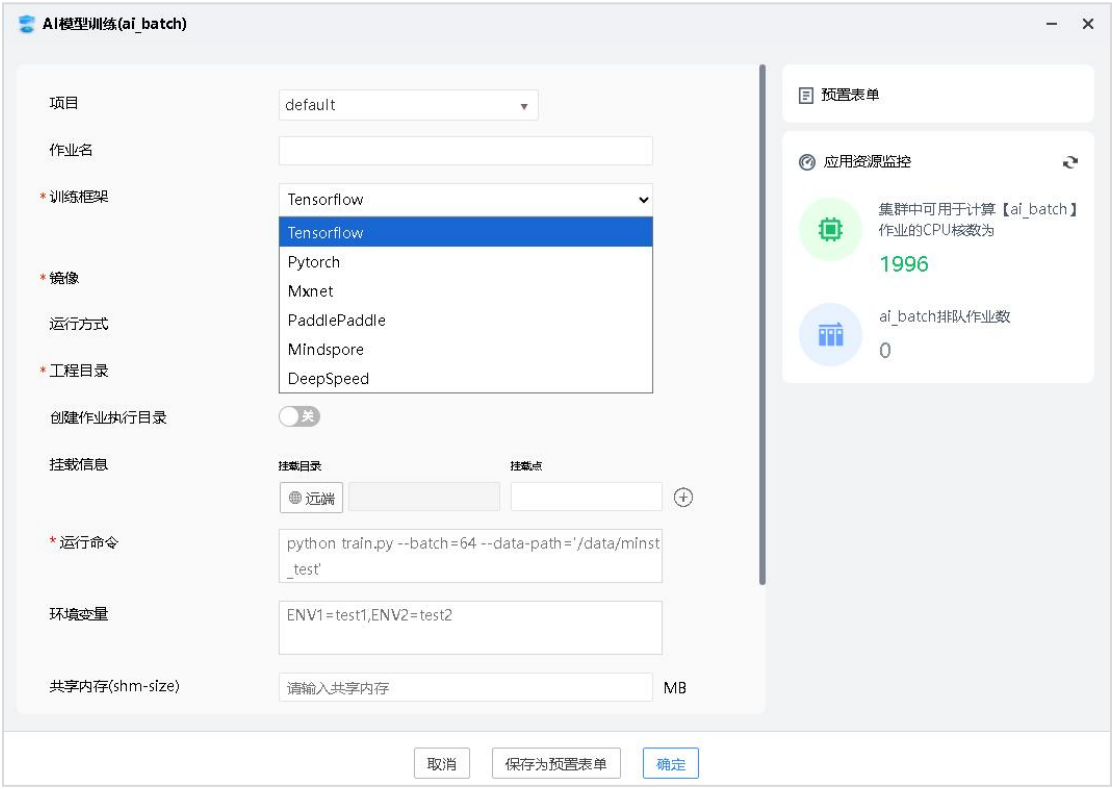
GPU 类型：具体 GPU 型号，匹配任务对算力、特性的要求。

GPU 数量：申请的 GPU 卡数，支撑多卡并行任务。

名称：自定义标识，方便区分、管理资源申请。

用户（组）：资源归属的用户/组，做权限与配额管控。

点击“AI 模型训练（ai_batch）”桌面图标，AI 模型训练参数设置界面，从训练框下拉列表中选择训练框架，切换到不同类型训练框架，如下图所示：



AI 模型训练

命令行提交作业，用户在 sub 脚本中设置队列、资源、镜像等参数，如下所示：

```
#!/bin/sh


#BSUB -J tensorflow_standalone_cpu（作业名称）
#BSUB -q ai（队列名称）
#BSUB -app jhai_command （应用名称）
#BSUB -mf standalone_cpu.mf （machine 文件，用于设置 cpu 数，slot 数和 gpu 数量）
#BSUB -o output.%J.txt （输出日志名称）
```

```
 ${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type standalone --image
192.168.0.153:5000/jhinno/tensorflow:py310-2.13 --cmd='python
mnist_tf2.py' --workdir=${HOME}/example/tensorflow （运行命令）
```

machine 文件示例如下：

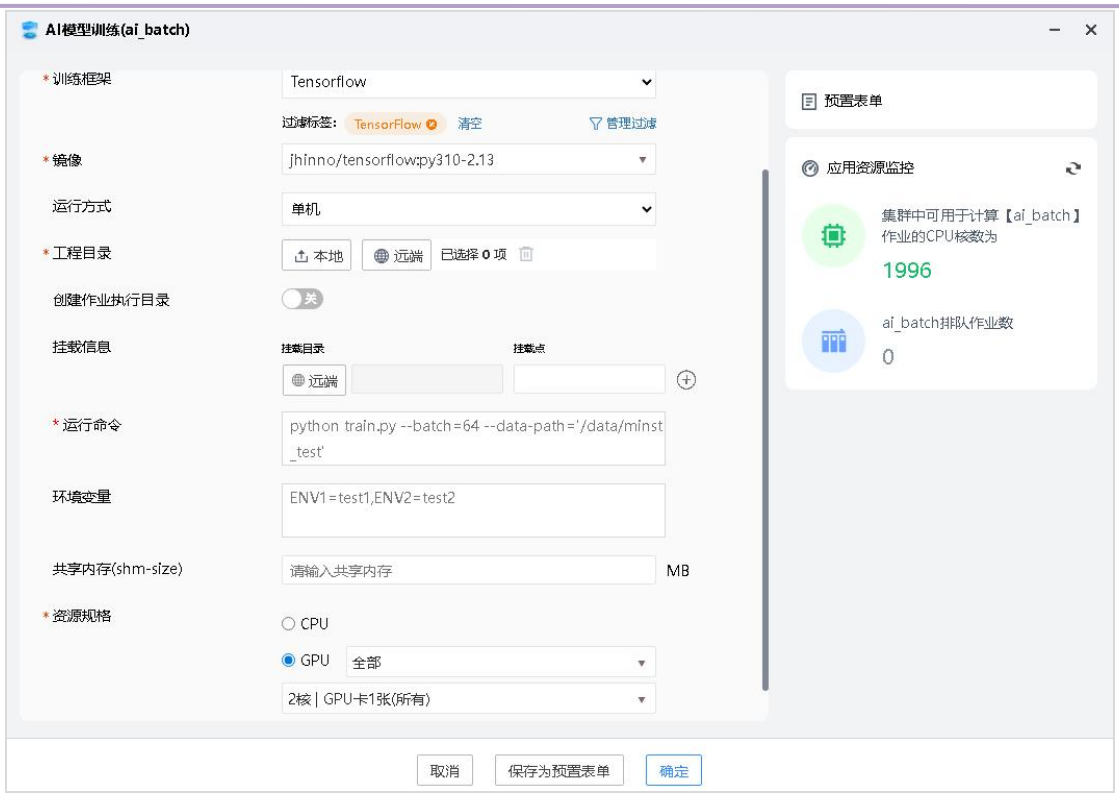
```
[host:all slots:1 gpu:1] （host 运行节点名称，slots 数，gpu 数）
```

2.3.1. Tensorflow

2.3.1.1. 可视化方式提交

2.3.1.1.1. 单机模式

点击“AI 模型训练”桌面图标，打开作业提交参数设置界面，训练框架下拉列表选择“Tensorflow”，如下图所示：



Tensorflow 单机模式作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：tensorflow。

密级：管理员开启密级功能后，显示此选项，默认为用户密级。

训练框架：必选项，下拉列表中选择“Tensorflow”模型训练的框架。

镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像，下拉列表中默认仅显示“我的镜像”中标签为“tensorflow”的镜像。

运行方式：默认选择单机，仅在单节点上运行训练程序。

工程目录：必选项，指定代码的工程目录，可以从本地上传或选择服务端已存在的目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

挂载信息：可以添加额外的挂载目录，需要设置挂载目录和挂载点。

1） 挂载目录：指定挂载额外的目录到运行容器中，例如：可以选择挂

载数据集。

2） 挂载点：挂载目录在容器中对应的目录，未填写时挂载点和挂载目录的路径保持一致。

运行命令：必填项，指定运行命令，包括程序的入口文件和程序参数等，例如：python train.py --batch=64 --data-path="/data/minst_test"。

环境变量：指定程序运行所需要的额外环境变量。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

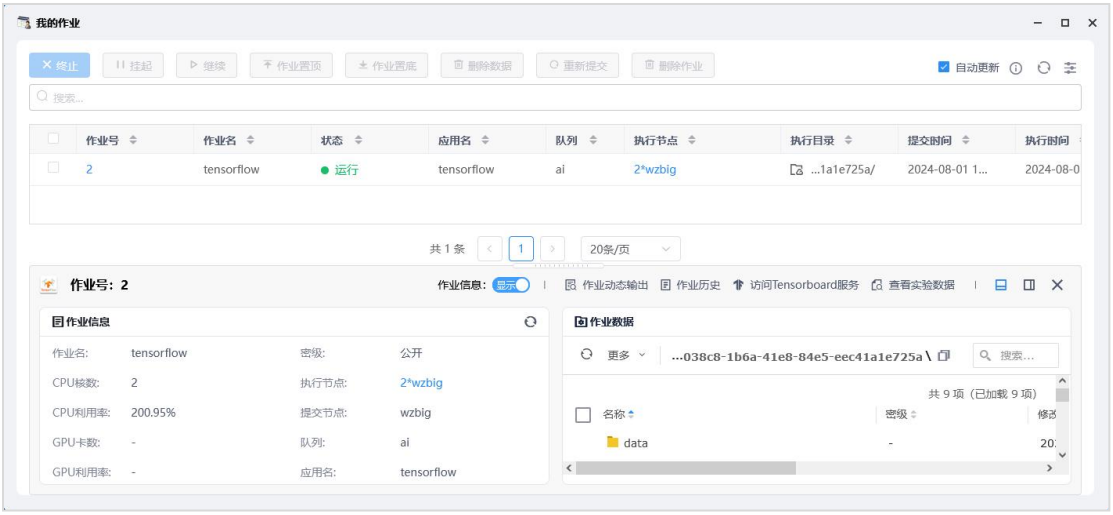
资源规格：必选项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

自动调参：是否启用自动调参功能，具体使用方式见本章节的“自动调参”模块。

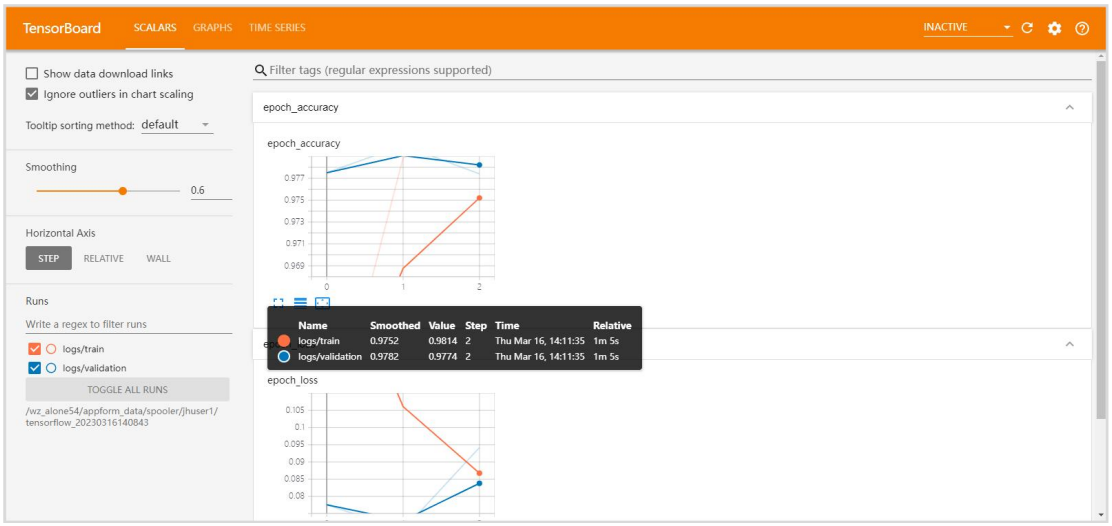
预置表单：显示已保存的预置表单列表，可以将某个表单“设为默认表单”，或者删除表单。

应用资源监控：显示当前集群中可用于当前计算应用的 CPU 核数和当前应用排队的作业数。

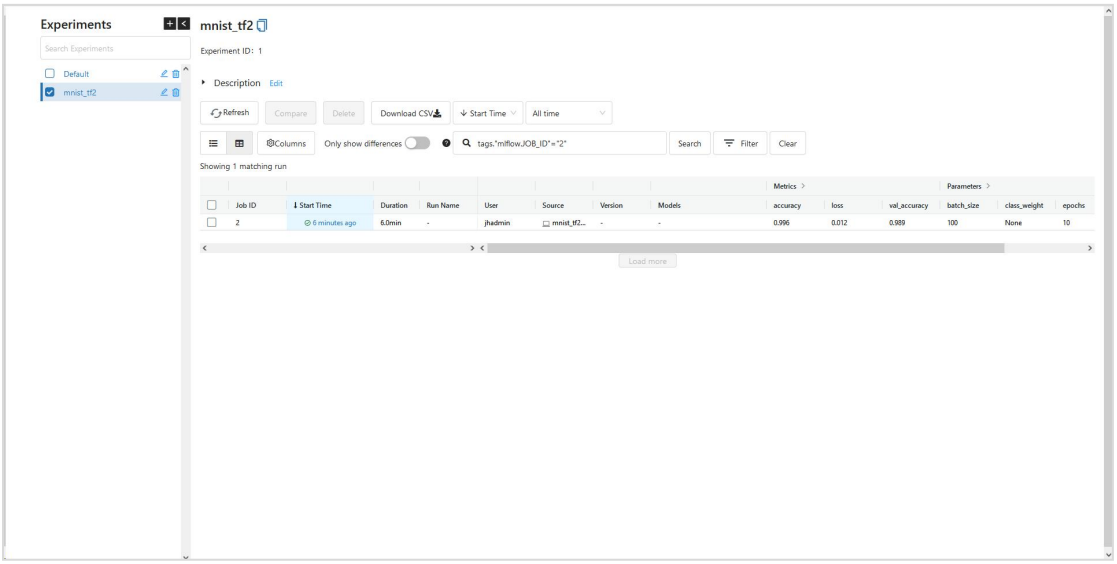
Tensorflow 作业提交后，会自动打开“我的作业”页面，方便用户在此查看提交作业的信息，如下图所示：



点击作业详情右上角“访问 Tensorboard 服务”按钮，可以查看训练情况，如下图所示：



如果运行程序中集成了实验管理 api，点击作业详情右上角“查看实验数据”按钮，可以查看实验数据，如下图所示：

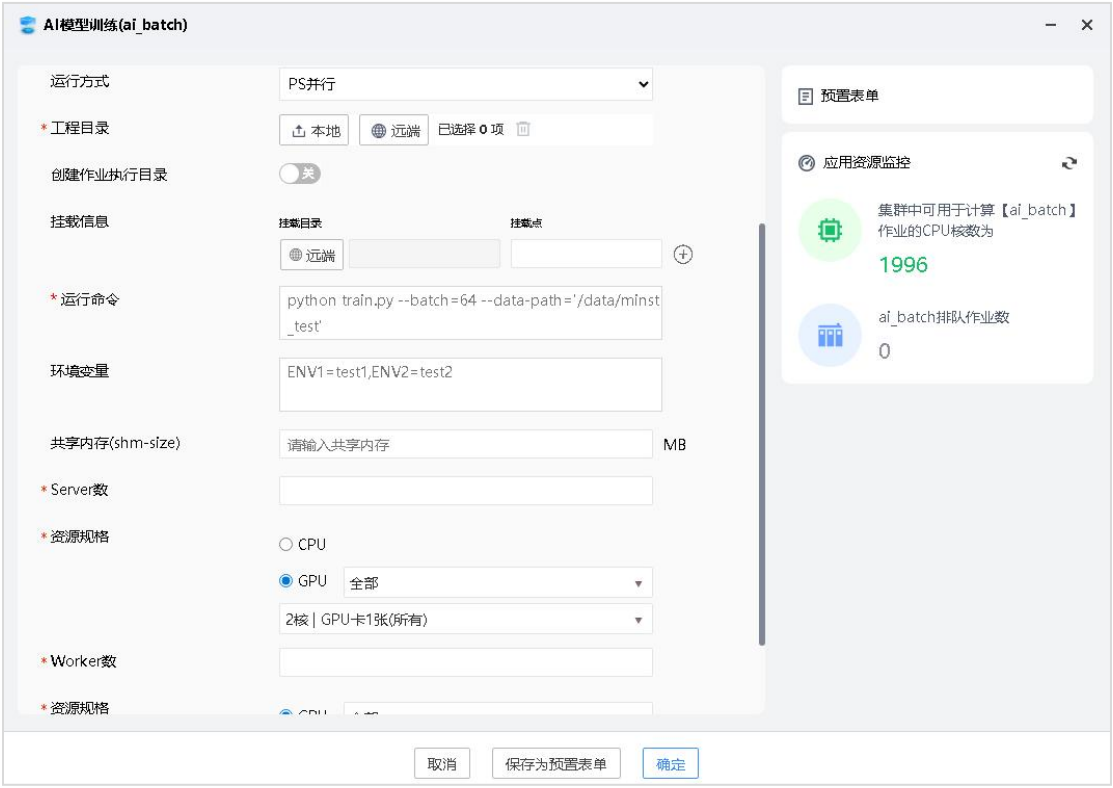


2.3.1.1.2. PS 并行模式

提交以 Parameter Server 方式实现的训练代码，支持跨节点，需填写并行相关参数，如下所示：

- **Server 数：**指定参数服务的数量。
- **Worker 数：**指定训练服务的数量。

- **资源规格：**分别指定每个 Server 和 Worker 使用的资源。

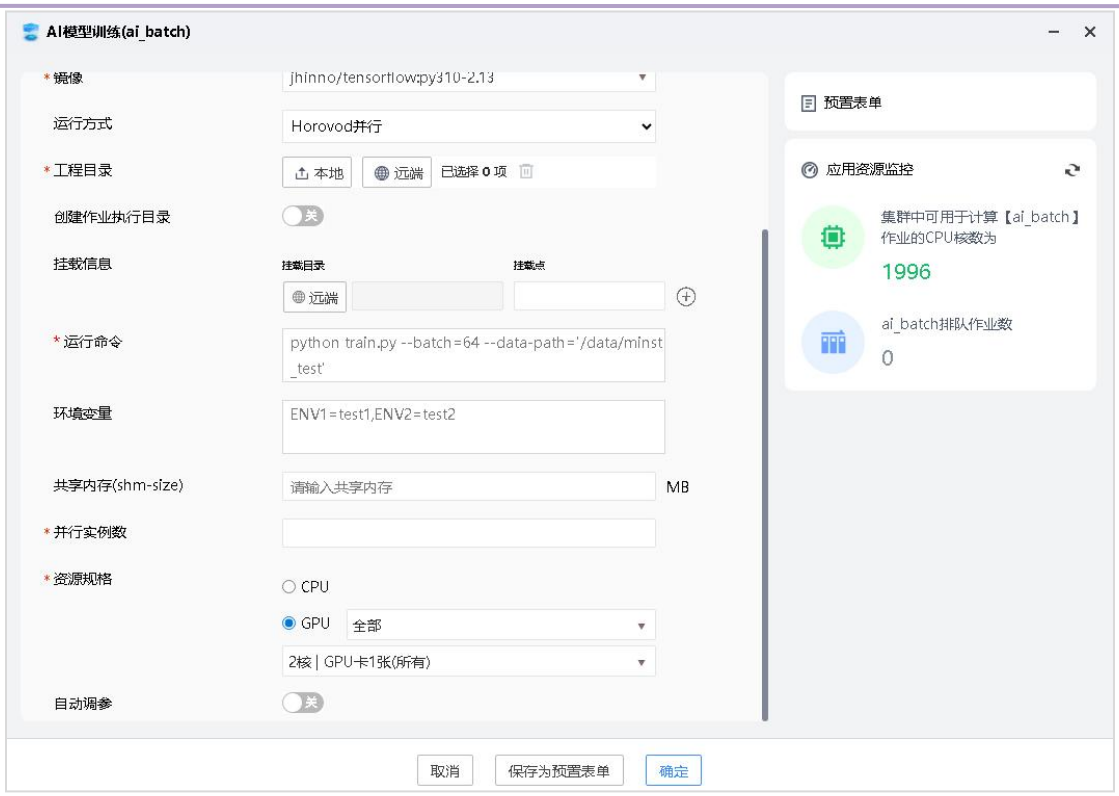


Tensorflow PS 并行模式作业提交界面

2.3.1.1.3. Horovod 并行模式

提交以 Horovod 实现的并行 AI 训练程序，并指定并行实例数和每个实例的资源规格。如下所示：

- **并行实例数：**指定并行训练的实例数量。
- **资源规格：**指定每个并行实例所使用的计算资源。



Tensorflow Horovod 并行模式作业提交页面

2.3.1.2. 命令行方式提交

2.3.1.2.1. 单机模式

通过终端命令行方式提交单机 Tensorflow 训练程序，提交脚本如下所示：

```
#!/bin/sh


#BSUB -J tensorflow_standalone_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf standalone_cpu.mf
#BSUB -o output.%J.txt
```

```
 ${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type standalone --image
192.168.0.153:5000/jhinno/tensorflow:py310-2.13  --cmd='python
mnist_tf2.py' --workdir=${HOME}/example/tensorflow
```

tensorflow_standalone_cpu.sub

```
[host:all slots:1]
```

standalone_cpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < tensorflow_standalone_cpu.sub
```

2.3.1.2.2. PS 并行模式

通过终端命令行方式提交以 Parameter Server 方式实现的 Tensorflow 训练代码，支持跨节点，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J tensorflow_ps_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf ps_cpu.mf
#BSUB -o output.%J.txt
#BSUB -port 2

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type ps --image 192.168.0.153:5000/jhinno/tensorflow:py310-2.13
--cmd='python tensorflow2_ps.py' --workdir=${HOME}/example/tensorflow
```

tensorflow_ps_cpu.sub

```
1*[host:all slots:2 tag:"server"]
1*[host:all slots:2 tag:"worker"]
```

ps_cpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < tensorflow_ps.sub
```

2.3.1.2.3. Horovod 并行模式

通过终端命令行方式提交以 Horovod 实现的并行 Tensorflow 训练程序，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J tensorflow_horovod_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf horovod_cpu.mf
#BSUB -o output.%J.txt
#BSUB -port 2

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type horovod --image
192.168.0.153:5000/jhinno/tensorflow:py310-2.13 --cmd='python
mnist_tf2_horovod.py' --workdir=${HOME}/example/tensorflow
```

tensorflow_horovod_cpu.sub

```
2*[host:all slots:2]
```

horovod_cpu.mf

准备好脚本后通过以下命令提交：

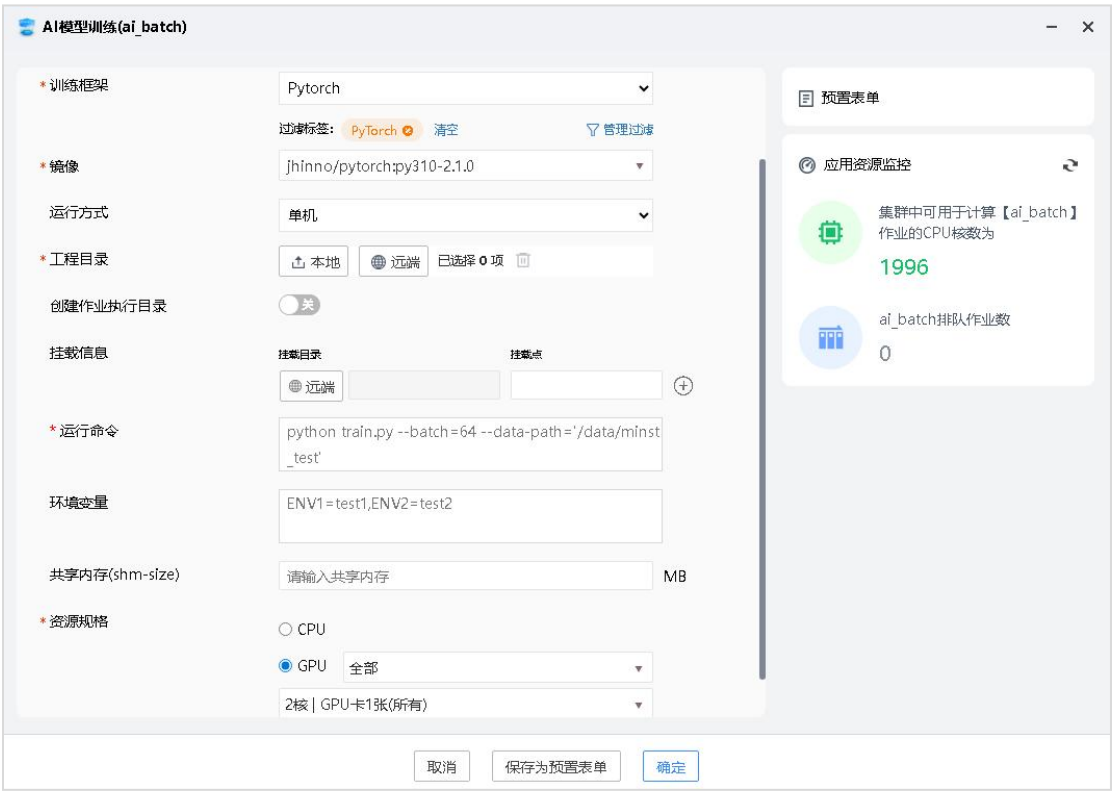
```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < tensorflow_horovod_cpu.sub
```

2.3.2. Pytorch

2.3.2.1. 可视化方式提交

2.3.2.1.1. 单机模式

点击“AI 模型训练”桌面图标，打开作业提交参数设置界面，训练框架下拉列表选择“Pytorch”，如下图所示：



Pytorch 单机模式作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：pytorch。

密级：管理员开启密级功能后，显示此选项，默认为用户密级。

训练框架：必选项，下拉列表中选择“Pytorch”模型训练的框架。

镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像，下拉列表中默认仅显示“我的镜像”中标签为“pytorch”的镜像。

运行方式：默认选择单机，仅在单节点上运行训练程序。

工程目录：必选项，指定代码的工程目录，可以从本地上传或选择服务端已存在的目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

挂载信息：可以添加额外的挂载目录，需要设置挂载目录和挂载点。

1) **挂载目录**：指定挂载额外的目录到运行容器中，例如：可以选择挂载数据集。
2) **挂载点**：挂载目录在容器中对应的目录，未填写时挂载点和挂载目录路径保持一致。

运行命令：必填项，指定运行命令，包括程序的入口文件和程序参数等，例如：python train.py --batch=64 --data-path="/data/minst_test"。

环境变量：指定程序运行所需要的额外环境变量。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

资源规格：必选项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

自动调参：是否启用自动调参功能，具体使用方式见“自动调参”模块。

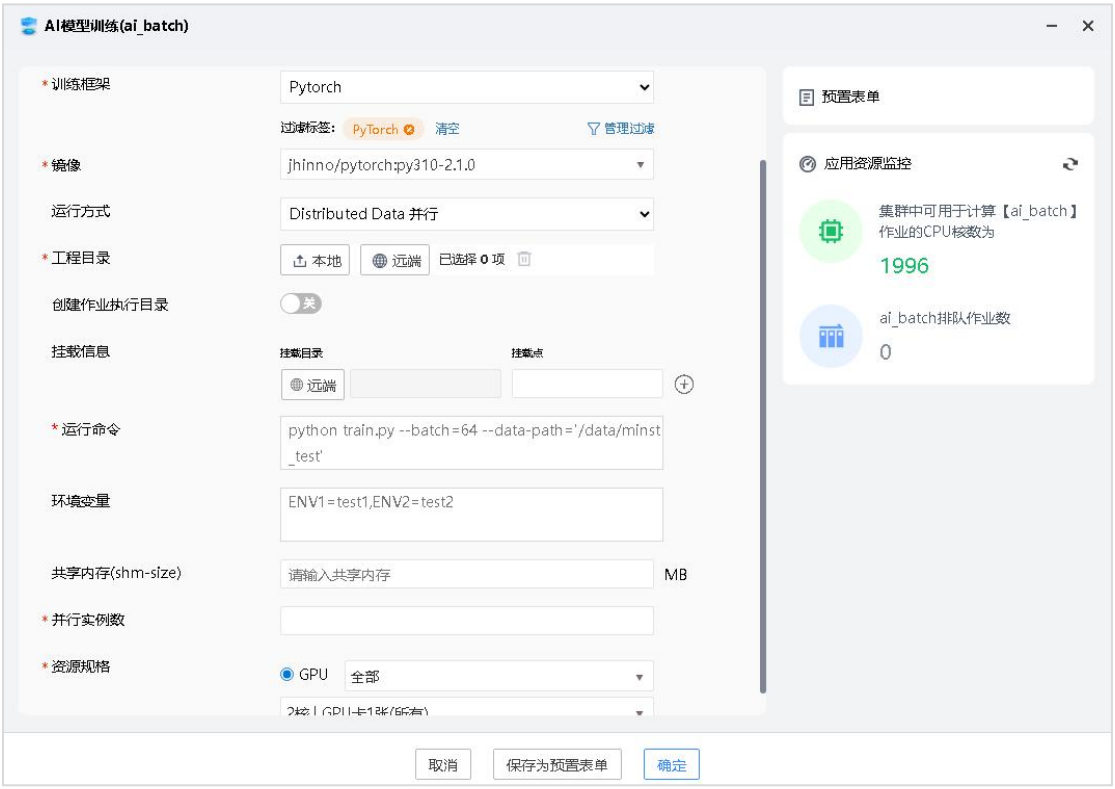
预置表单：显示已保存的预置表单列表，可以将某个表单“设为默认表单”，或者删除表单。

应用资源监控：显示当前集群中可用于当前计算应用的 CPU 核数和当前应用排队的作业数。

2.3.2.1.2. Distributed Data 并行(DDP)模式

提交以 Distributed Data Parallel(DDP)实现的分布式数据并行 AI 训练程序，并指定并行实例数和每个实例的资源规格，如下所示：

- **并行实例数：**指定并行训练的实例数量。
- **资源规格：**指定每个并行实例所使用的计算资源。

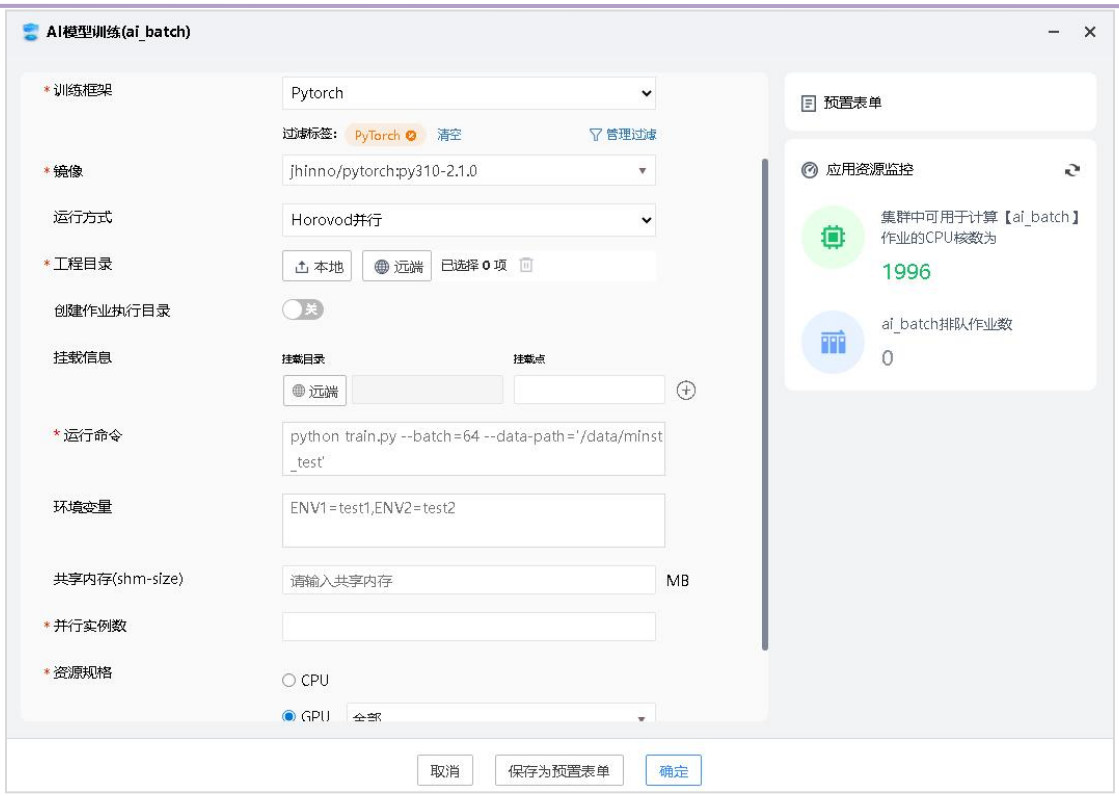


Pytorch DDP 模式作业提交界面

2.3.2.1.3. Horovod 并行模式

提交以 Horovod 实现的并行 AI 训练程序，并指定并行实例数和每个实例的资源规格，如下所示：

- **并行实例数：**指定并行训练的实例数量。
- **资源规格：**指定每个并行实例所使用的计算资源。



Pytorch Horovod 并行模式作业提交界面

2.3.2.2. 命令行方式提交

2.3.2.2.1. 单机模式

通过终端命令行方式提交 Pytorch 单机训练程序，提交脚本如下所示：

```
#!/bin/sh


#BSUB -J pytorch_standalone_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf standalone_cpu.mf
#BSUB -o output.%J.txt


${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
```

```
--job-type standalone --image
192.168.0.153:5000/jhinno/pytorch:py310-2.1.0 --cmd='python
mnist_torch.py' --workdir=${HOME}/example/pytorch
```

pytorch_standalone_cpu.sub

```
[host:all slots:1]
```

standalone_cpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < pytorch_standalone_cpu.sub
```

2.3.2.2.2. Horovod 并行模式

通过终端命令行方式提交以 Horovod 实现的 Pytorch 并行训练程序，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J pytorch_horovod_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf horovod_cpu.mf
#BSUB -o output.%J.txt
#BSUB -port 2

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type horovod --image
192.168.0.153:5000/jhinno/pytorch:py310-2.1.0 --cmd='python
mnist_torch_horovod.py' --workdir=${HOME}/example/pytorch
```

pytorch_horovod_cpu.sub

```
2*[host:all slots:2]
```

horovod_cpu.mf

准备好脚本后通过以下命令提交：

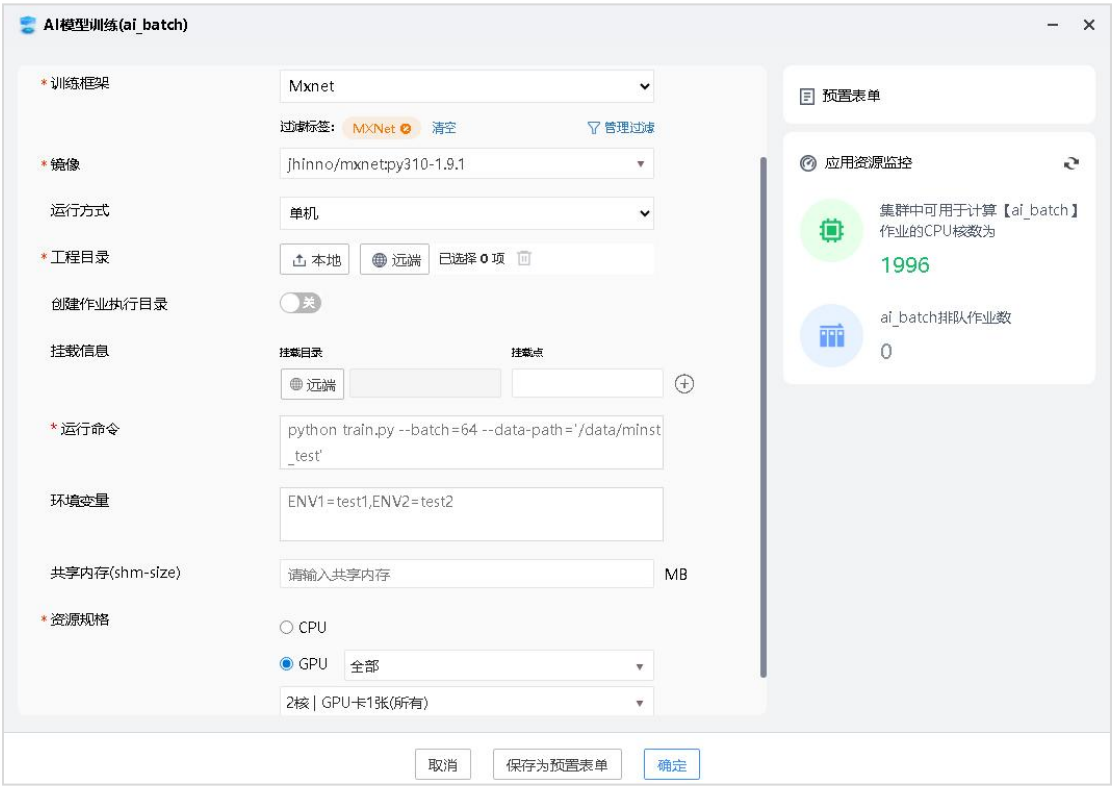
```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < pytorch_horovod_cpu.sub
```

2.3.3. Mxnet

2.3.3.1. 可视化方式提交

2.3.3.1.1. 单机模式

点击“AI 模型训练”桌面图标，打开作业提交参数设置界面，训练框架下拉列表选择“Mxnet”，如下图所示：



Mxnet 单机模式作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：mxnet。

密级：管理员开启密级功能后，显示此选项，默认为用户密级。

训练框架：必选项，下拉列表中选择“Mxnet”模型训练的框架。

镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像，下拉列表中默认仅显示“我的镜像”中标签为“mxnet”的镜像。

运行方式：默认选择单机，仅在单节点上运行训练程序。

工程目录：必选项，指定代码的工程目录，可以从本地上传或选择服务端已存在的目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

挂载信息：可以添加额外的挂载目录，需要设置挂载目录和挂载点。

1) **挂载目录：**指定挂载额外的目录到运行容器中，例如：可以选择挂载数据集。
2) **挂载点：**挂载目录在容器中对应的目录，未填写时挂载点和挂载目录路径保持一致。

运行命令：必填项，指定运行命令，包括程序的入口文件和程序参数等，例如：python train.py --batch=64 --data-path="/data/minst_test"。

环境变量：指定程序运行所需要的额外环境变量。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

资源规格：必选项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

自动调参：是否启用自动调参功能，具体使用方式见“自动调参”模块。

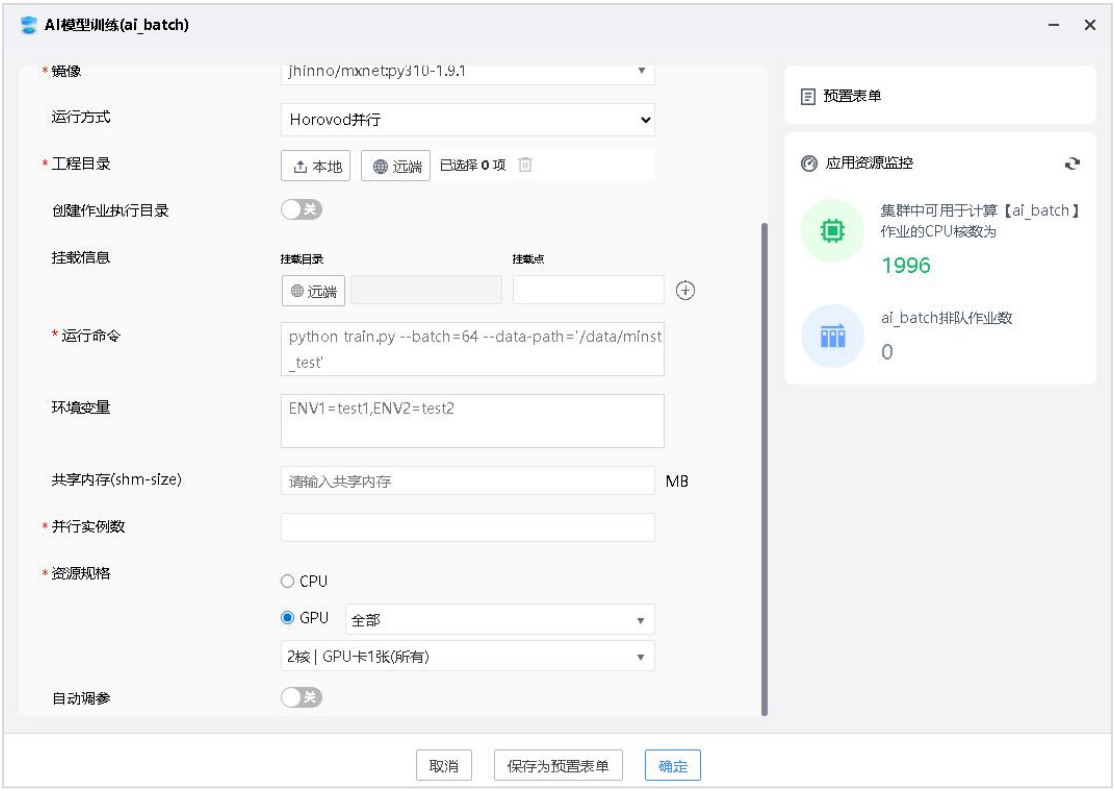
预置表单：显示已保存的预置表单列表，可以将某个表单“设为默认表单”，或者删除表单。

应用资源监控：显示当前集群中可用于当前计算应用的 CPU 核数和当前应用排队的作业数。

2.3.3.1.2. Horovod 并行模式

提交以 Horovod 实现的并行 AI 训练程序，并指定并行实例数和每个实例的资源规格，如下所示：

- **并行实例数：**指定并行训练的实例数量。
- **资源规格：**指定每个并行实例所使用的资源。



Mxnet Horovod 并行模式作业提交页面

2.3.3.2. 命令行方式提交

2.3.3.2.1. 单机模式

通过终端命令行方式提交单机 Mxnet 训练程序，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J mxnet_standalone_gpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf standalone_gpu.mf
#BSUB -o output.%J.txt

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type standalone --image
192.168.0.153:5000/jhinno/mxnet:py310-1.9.1  --cmd='python
mnist_mxnet.py' --workdir=${HOME}/example/mxnet
```

mxnet_standalone_gpu.sub

```
[host:all slots:1 gpu:1]
```

standalone_gpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < mxnet_standalone_gpu.sub
```

2.3.3.2.2. Horovod 并行模式

通过终端命令行方式提交以 Horovod 实现的并行 Mxnet 训练程序，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J mxnent_horovod_gpu
#BSUB -q ai_excl
```

```
#BSUB -app jhai_command
#BSUB -mf horovod_gpu.mf
#BSUB -o output.%J.txt
#BSUB -port 2

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type horovod --image 192.168.0.153:5000/jhinno/mxnet:py310-1.9.1
--cmd='python mnist_mxnet_horovod.py' --workdir=${HOME}/example/mxnet
```

mxnet_horovod_gpu.sub

```
2*[host:all slots:1 gpu:1]
```

horovod_gpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < mxnet_horovod_gpu.sub
```

2.3.4. PaddlePaddle

PaddlePaddle（百度飞桨）是由百度推出的开源深度学习平台，旨在为开发者提供灵活、高效的工具和库，以构建各种深度学习模型。以下是 PaddlePaddle 的一些主要特点和优势：

全面的深度学习支持：PaddlePaddle 支持多种深度学习任务，包括图像识别、自然语言处理、语音识别等。它提供了丰富的预训练模型和预处理工具，使开发者能够更轻松地构建自己的模型。

灵活的动态图和静态图：PaddlePaddle 支持动态图和静态图的混合编程。这使得开发者可以选择适合其任务和习惯的编程方式，从而更好地平衡灵活性和性能。

分布式训练：PaddlePaddle 提供了强大的分布式训练能力，支持在多个设备和多个节点上进行模型训练，以应对大规模数据和复杂模型的挑战。

多端部署：PaddlePaddle 支持模型在不同平台上的部署，包括移动端、嵌入式设备和服务器。这使得开发者能够将训练好的模型应用到各种应用场景中。

易用性：PaddlePaddle 注重开发者体验，提供了简洁、清晰的 API 和文档。同时，它还集成了可视化工具，帮助开发者更好地理解和调试他们的模型。

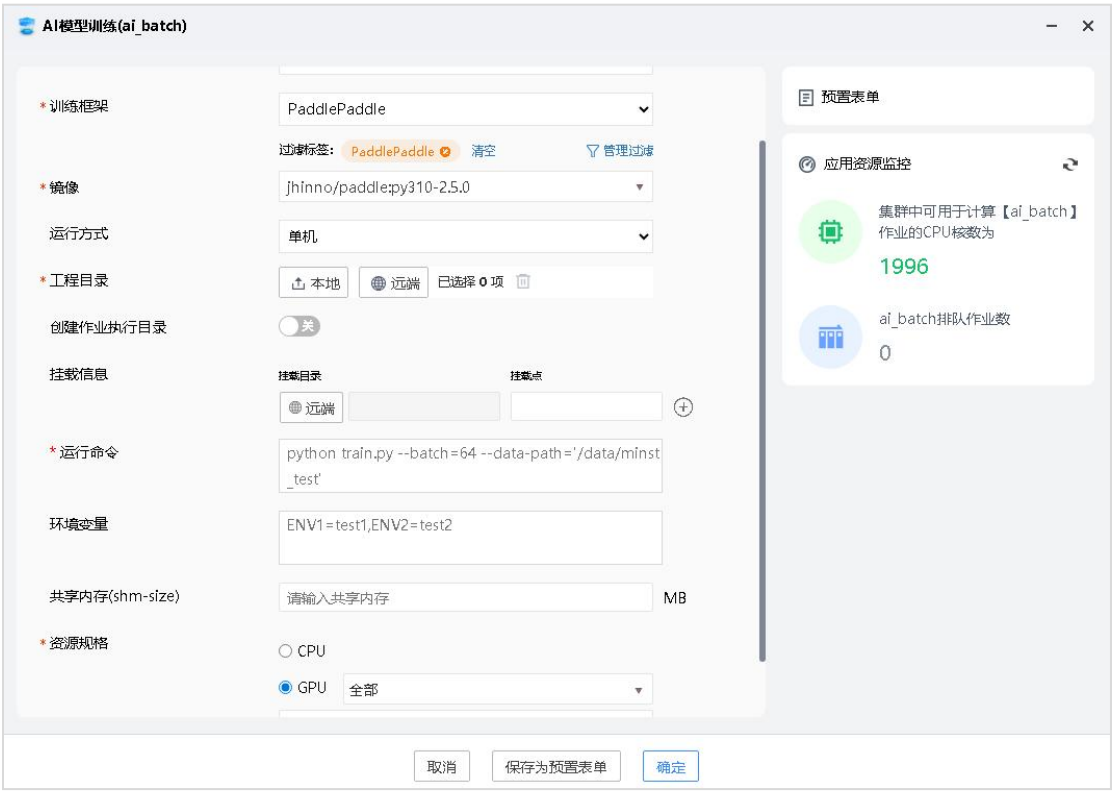
总体而言，PaddlePaddle 是一款强大而全面的深度学习平台，适用于各种规模和领域的深度学习项目。

2.3.4.1. 可视化方式提交

2.3.4.1.1. 单机模式

3）单机单卡

点击“AI 模型训练”桌面图标，打开作业提交参数设置界面，训练框架下拉列表选择“PaddlePaddle”，如下图所示：



PaddlePaddle 单机模式作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：PaddlePaddle。

密级：管理员开启密级功能后，显示此选项，默认为用户密级。

训练框架：必选项，下拉列表中选择“PaddlePaddle”模型训练的框架。

镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像，下拉列表中默认仅显示“我的镜像”中标签为“PaddlePaddle”的镜像。

运行方式：默认选择单机，仅在单节点上运行训练程序。

工程目录：必选项，指定代码的工程目录，可以从本地上传或选择服务端已存在的目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

挂载信息：可以添加额外的挂载目录，需要设置挂载目录和挂载点。

1) **挂载目录：**指定挂载额外的目录到运行容器中，例如：可以选择挂载数据集。
2) **挂载点：**挂载目录在容器中对应的目录，未填写时挂载点和挂载目录路径保持一致。

运行命令：必填项，指定运行命令，包括程序的入口文件和程序参数等，例如：python train_demo.py。

环境变量：指定程序运行所需要的额外环境变量。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

资源规格：必选项，选择运行程序所需要的资源配置。可以选择 CPU 资源组的规格列表，也可以选择 GPU 资源组的规格列表。

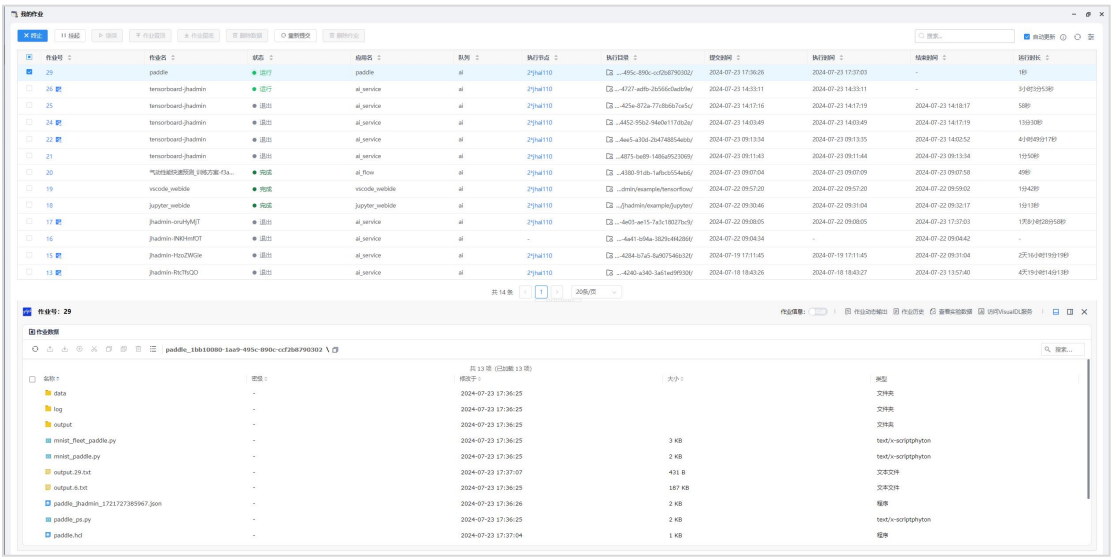
自动调参：是否启用自动调参功能，具体使用方式见“自动调参”模块。

预置表单：显示已保存的预置表单列表，可以将某个表单“设为默认表单”，或者删除表单。

应用资源监控：显示当前集群中可用于当前计算应用的 CPU 核数和当前应用

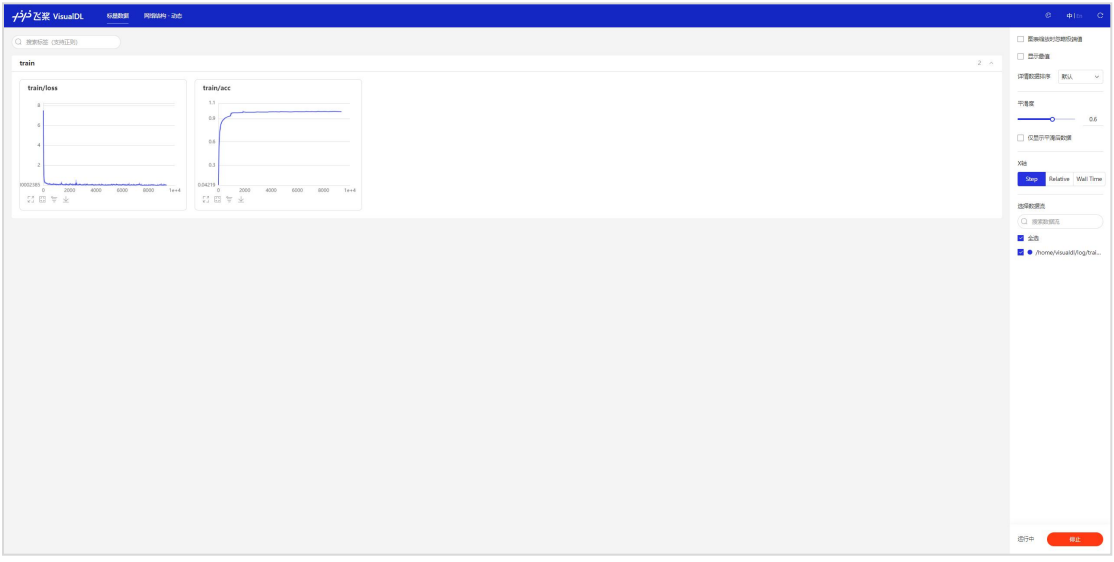
排队的作业数。

PaddlePaddle 作业提交后，会自动打开“我的作业”页面，方便用户在此查看提交作业的信息，如下图所示：



作业信息查看

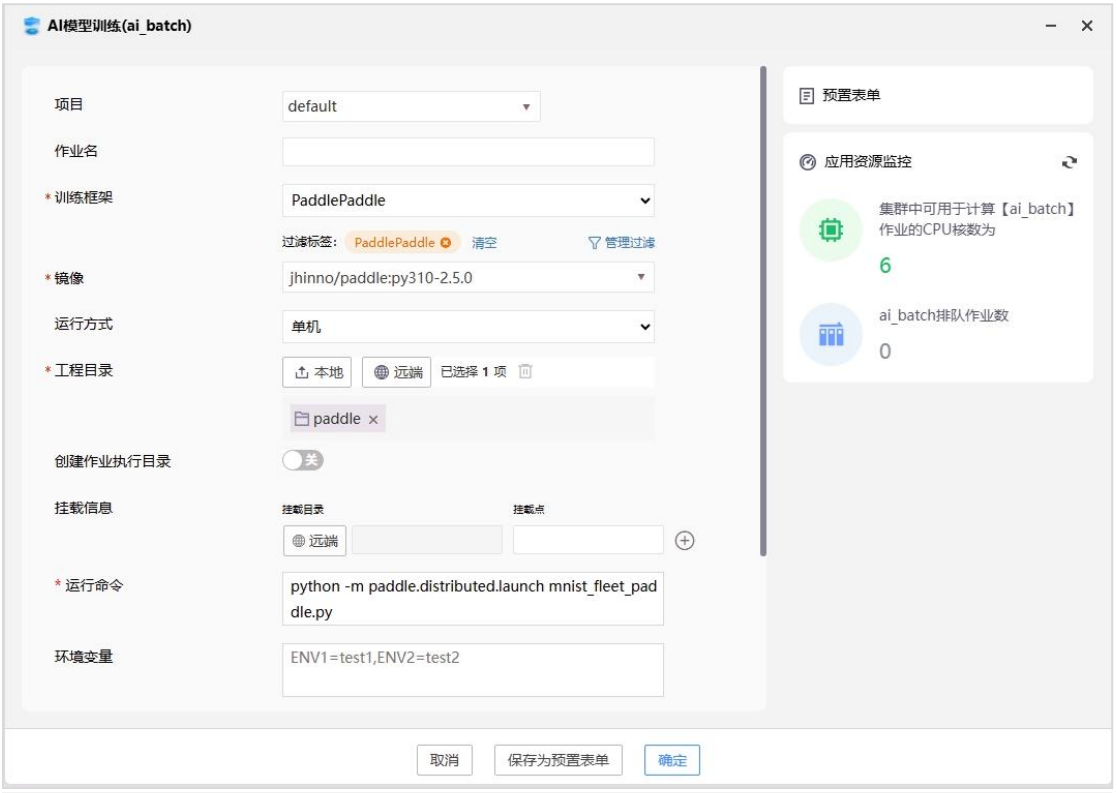
点击作业滑块右上角的“访问 VisualDL 服务”按钮，可以查看训练情况，如下图所示：

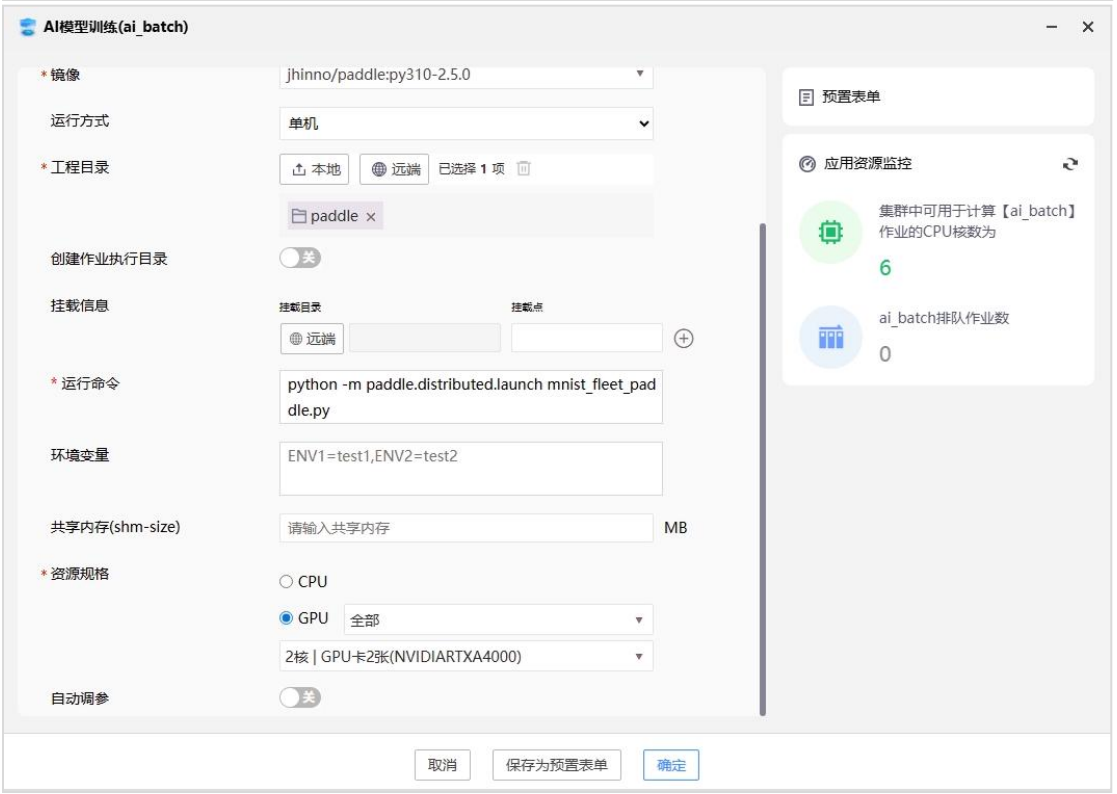


访问 VisudlDL 服务

4）单机多卡

提交单机多卡 PaddlePaddle 作业，需要在运行命令处额外配置“-m paddle.distributed.launch”参数，例如：python -m paddle.distributed.launch train.py，如下图所示：



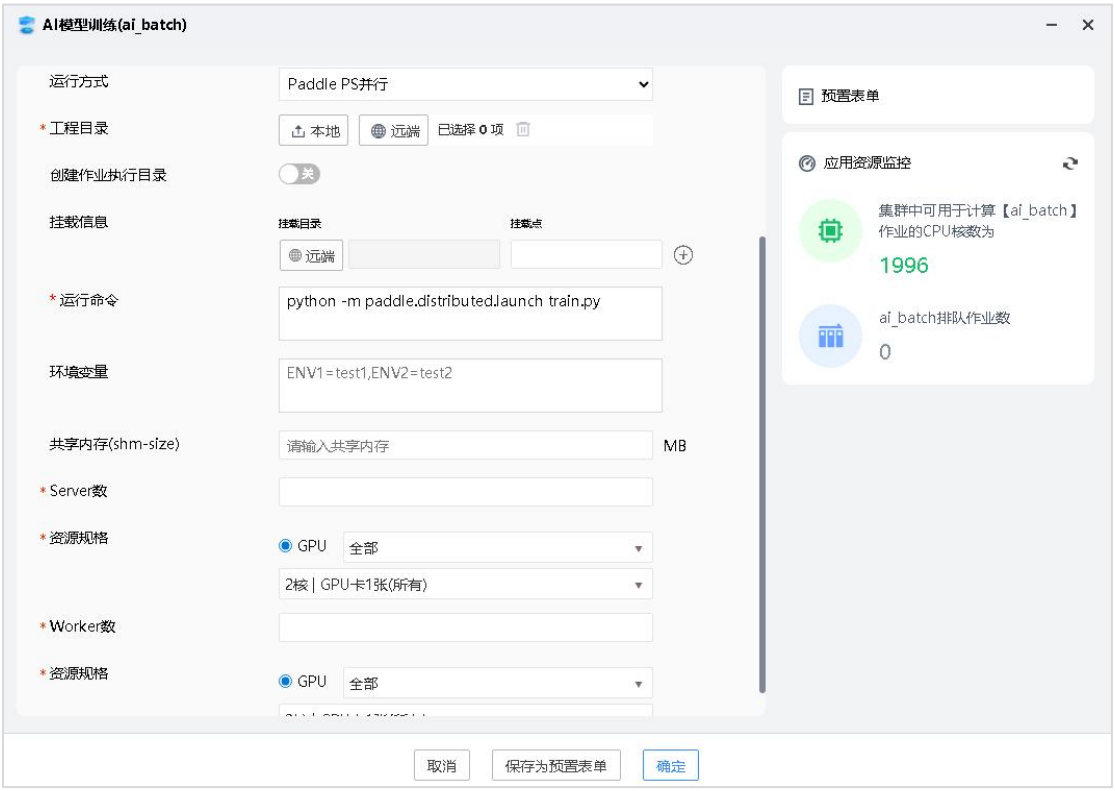


PaddlePaddle 单机多卡模式作业提交界面

2.3.4.1.2. Paddle PS 并行模式

提交以 Paddle 框架 Parameter Server 方式实现的训练代码，支持跨节点，需填写并行相关参数，如下所示：

- **Server 数：**指定参数服务的数量。
- **Worker 数：**指定训练服务数量。
- **资源规格：**分别指定每个 Server 和 Worker 使用的资源。



Paddle PS 并行模式作业提交页面

2.3.4.2. 命令行方式提交

2.3.4.2.1. 单机模式

通过终端命令行方式提交单机 PaddlePaddle 训练程序，提交脚本如下所示：

```
#!/bin/sh
```

```
#BSUB -J paddle_standalone_cpu
#BSUB -q ai_excl
#BSUB -app jhai_command
#BSUB -mf standalone_cpu.mf
#BSUB -o output.%J.txt

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type standalone --image
192.168.0.153:5000/jhinno/paddle:py310-2.5.0  --cmd='python
mnist_paddle.py' --workdir=${HOME}/example/paddle
```

paddle_standalone_cpu.sub

```
[host:all slots:1]
```

standalone_cpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < paddle_standalone_cpu.sub
```

2.3.4.2.2. Paddle PS 并行模式

通过终端命令行方式提交 Paddle PS 模式 PaddlePaddle 训练程序，提交脚本如下所示：

```
#!/bin/sh

#BSUB -J paddle_ps_gpu
#BSUB -q ai_excl
```

```
#BSUB -app jhai_command
#BSUB -mf ps_gpu.mf
#BSUB -o output.%J.txt
#BSUB -port 2

${JHSCHEDULER_TOP}/scripts/jhai/service/jairun djm command
--job-type ps --image 192.168.0.153:5000/jhinno/paddle:py310-2.5.0
--cmd='python paddle_ps.py' --workdir=${HOME}/example/paddle
```

paddle_ps_gpu.sub

```
1*[host:all slots:2 tag:"server"]
1*[host:all slots:2 gpu:1 tag:"worker"]
```

ps_gpu.mf

准备好脚本后通过以下命令提交：

```
source ${JHSCHEDULER_TOP}/etc/profile.unischeduler
jsub < paddle_ps_gpu.sub
```

2.3.5. MindSpore

MindSpore 是华为推出的开源深度学习框架，旨在提供灵活、高性能、易用的深度学习开发平台。它支持端到云的全场景 AI 应用开发，并具有以下特点：

自动并行：MindSpore 能够自动分析网络结构并生成相应的并行计算图，提高训练性能。

全场景：适用于各种场景，包括设备、边缘和云端，支持多种硬件架构。

算子融合：通过算子融合技术，降低内存占用，提高计算效率。

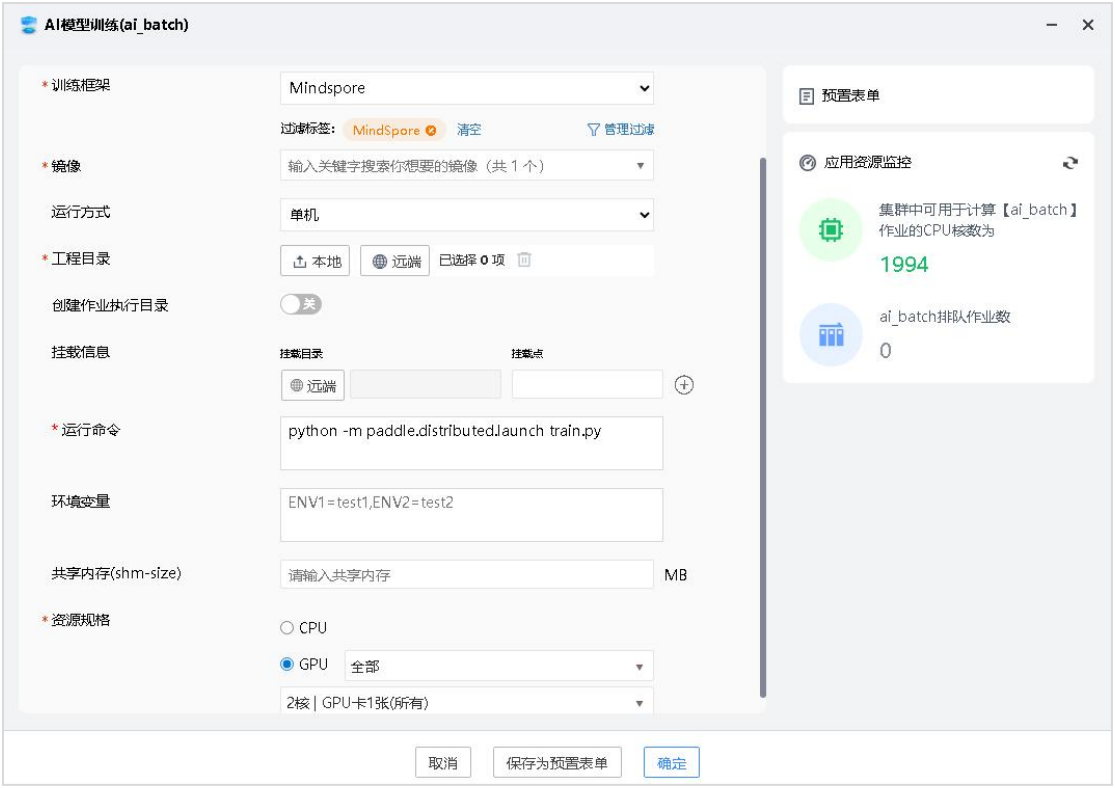
模型保护：提供了模型保护技术，有助于保护模型的安全性和隐私性。

易用性：设计简单直观，提供了友好的 API 和丰富的文档，降低开发者的学习成本。

2.3.5.1. 可视化方式提交

2.3.5.1.1. 单机模式

点击“AI 模型训练”桌面图标，打开作业提交参数设置界面，训练框架下拉列表选择“MindSpore”，如下图所示：



MindSpore 单机模式作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：mindspore。

密级：管理员开启密级功能后，显示此选项，默认为用户密级。

训练框架：必选项，下拉列表中选择“Mindspore”模型训练的框架。

镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像，下拉列表中默认仅显示“我的镜像”中标签为“MindSpore”的镜像。

运行方式：默认选择单机，仅在单节点上运行训练程序。

工程目录：必选项，指定代码的工程目录，可以从本地上传或选择服务端已存在的目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

挂载信息：可以添加额外的挂载目录，需要设置挂载目录和挂载点。

1) **挂载目录：**指定挂载额外的目录到运行容器中，例如：可以选择挂载数据集。
2) **挂载点：**挂载目录在容器中对应的目录，未填写时挂载点和挂载目录路径保持一致。

运行命令：指定运行命令，包括程序的入口文件和程序参数等，例如：python train.py --data-path=/data --batch=54。

环境变量：指定程序运行所需要的额外环境变量。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

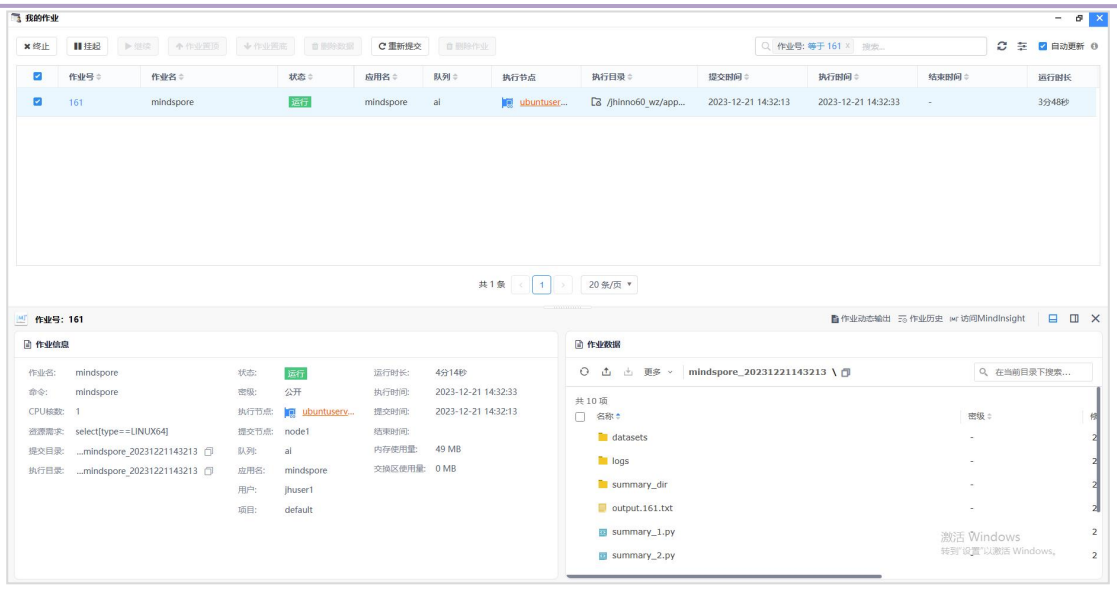
资源规格：必选项，选择运行程序所需要的资源配置。只能选择 GPU 资源组的规格列表。

自动调参：是否启用自动调参功能，具体使用方式见“自动调参”模块。

预置表单：显示已保存的预置表单列表，可以将某个表单“设为默认表单”，或者删除表单。

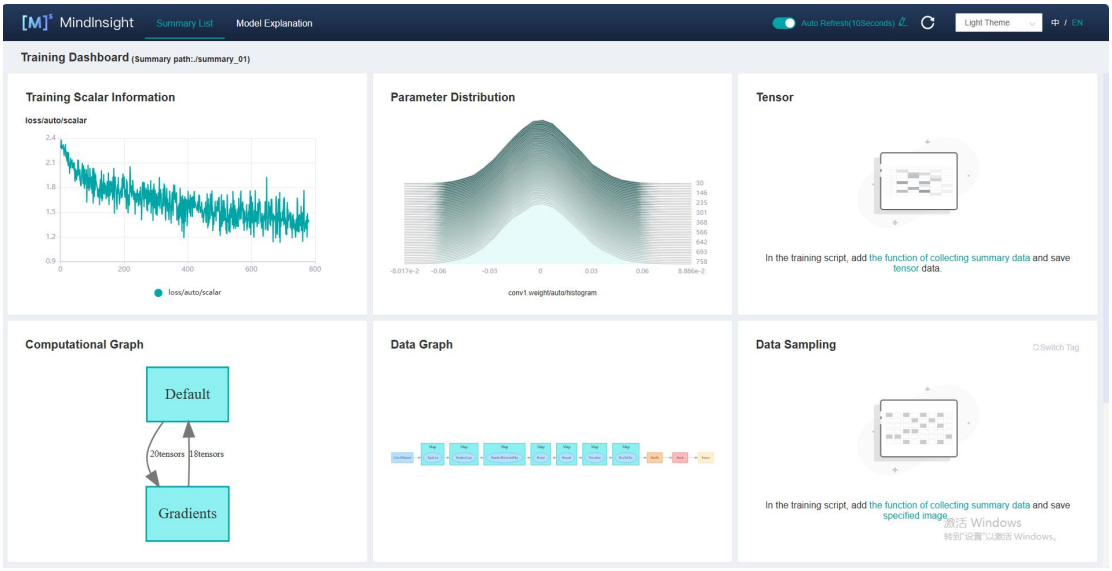
应用资源监控：显示当前集群中可用于当前计算应用的 CPU 核数和当前应用排队的作业数。

MindSpore 作业提交后，会自动打开“我的作业”页面，方便用户在此查看提交作业的信息，如下图所示：



作业信息查看

点击作业滑块右上角的“访问 MindInsight”按钮，可以查看训练情况，如下图所示：

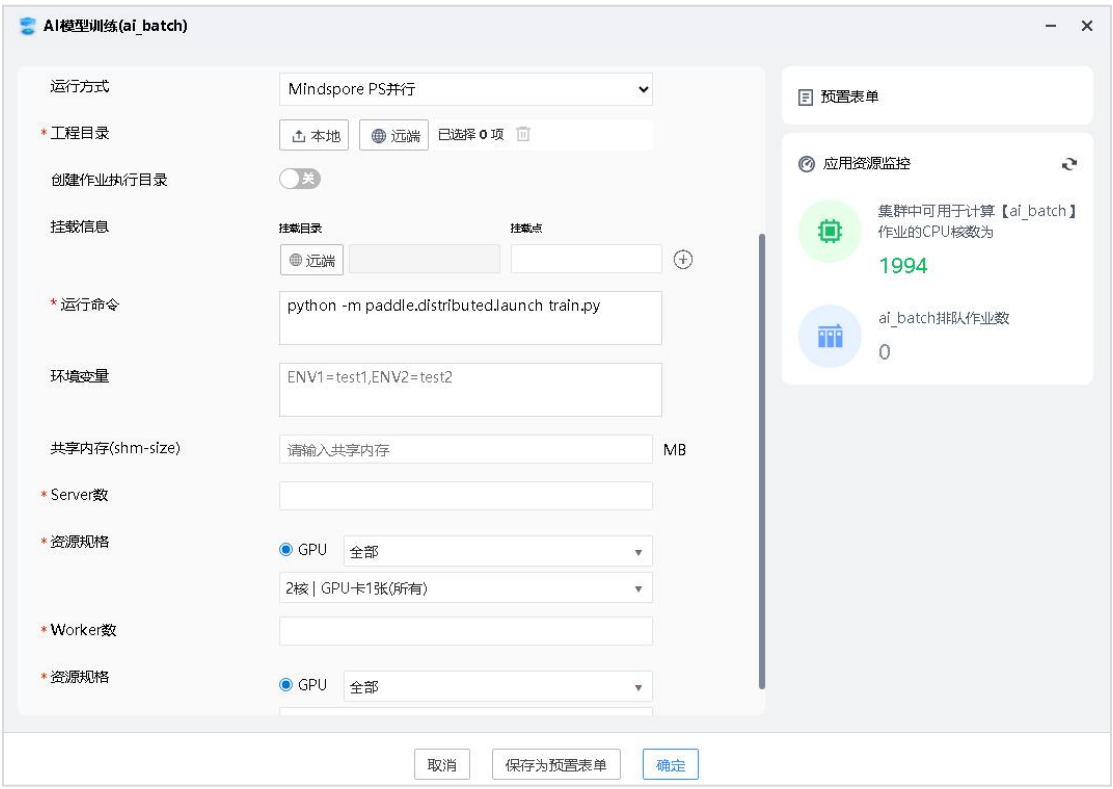


访问 MindInsight

2.3.5.1.2. MindSpore PS 并行模式

提交以 MindSpore 框架 Parameter Server 方式实现的训练代码，支持跨节点，需填写并行相关参数，如下所示：

- **Server 数：**指定参数服务的数量。
- **Worker 数：**指定训练服务的数量。
- **资源规格：**分别指定每个 Server 和 Worker 使用的资源。

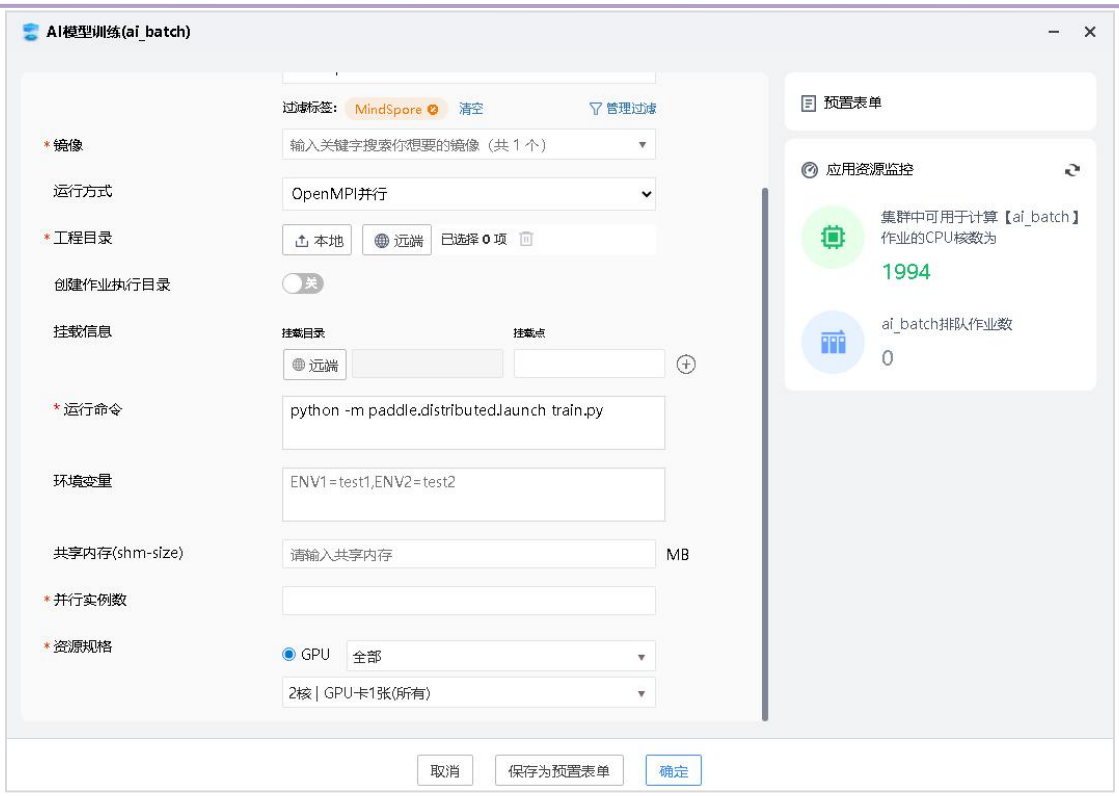


MindSpore PS 并行模式作业提交页面

2.3.5.1.3. Open MPI 并行模式

提交以 Open MPI 实现的并行 AI 训练程序， 并指定并行实例数和每个实例的资源规格。如下所示：

- **并行实例数：**指定并行训练的实例数量。
- **资源规格：**指定每个并行实例使用的计算资源。



MindSpore Open MPI 并行模式作业提交页面

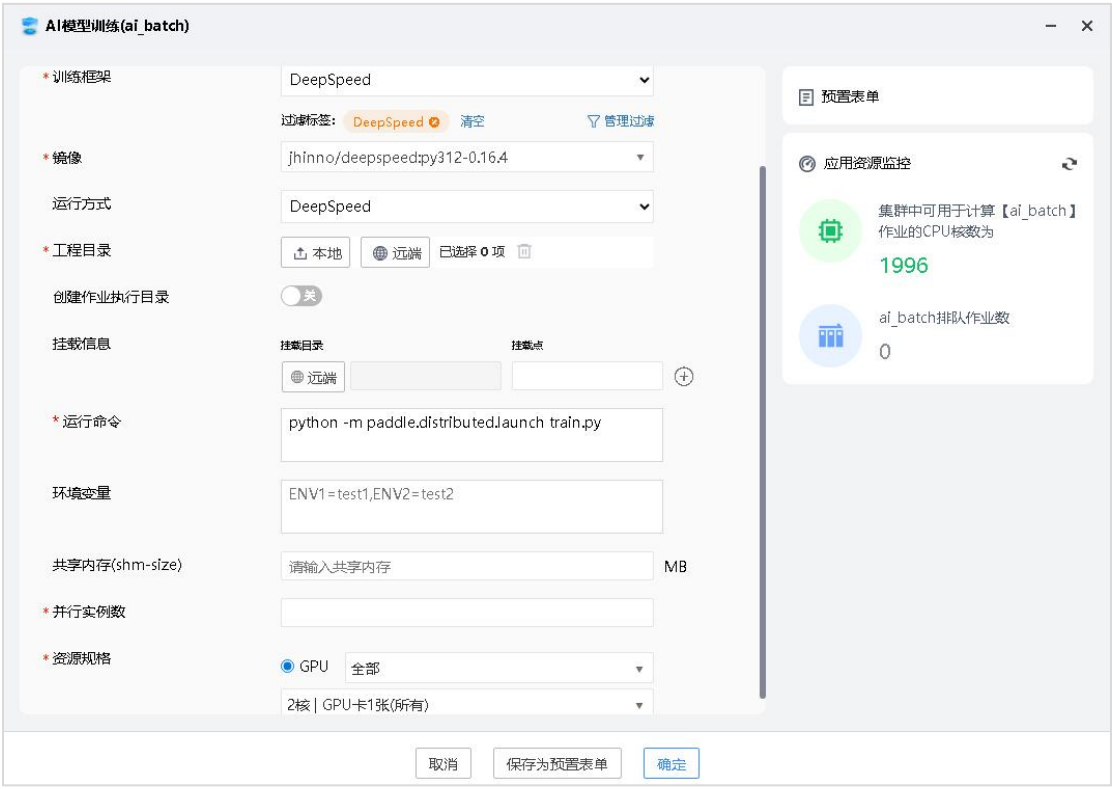
2.3.6. DeepSpeed

DeepSpeed 是由微软开发的开源深度学习优化库，旨在提高大规模模型训练的效率和可扩展性。通过创新的并行化策略、内存优化技术（如 ZeRO）及混合精度训练，DeepSpeed 显著提升了训练速度并降低了资源需求。

2.3.6.1. 可视化方式提交

2.3.6.1.1. DeepSpeed 并行模式

点击“AI 模型训练”桌面图标，打开作业提交参数设置界面，训练框架下拉列表选择“DeepSpeed”，如下图所示：



DeepSpeed 并行模式作业提交界面

作业提交参数：

项目：指定项目，默认为：default。

作业名：指定作业名，默认为：deepspeed。

密级：管理员开启密级功能后，显示此选项，默认为用户密级。

训练框架：必选项，下拉列表中选择“DeepSpeed”模型训练的框架。

镜像：必选项，选择程序运行环境的镜像，需要根据运行程序选择合适的镜像，下拉列表中默认仅显示“我的镜像”中标签为“DeepSpeed”的镜像。

运行方式：默认选择 DeepSpeed 并行运行训练程序。

工程目录：必选项，指定代码的工程目录，可以从本地上传或选择服务端已存在的目录。

创建作业执行目录：默认为关，直接在工程目录中运行程序。如果手动开启此功能，提交作业后，会在“作业数据区”中创建一个临时执行目录，将工程目录中的文件拷贝到临时执行目录后，运行程序。

挂载信息：可以添加额外的挂载目录，需要设置挂载目录和挂载点。

1) **挂载目录：**指定挂载额外的目录到运行容器中，例如：可以选择挂

载数据集。

2） **挂载点：**挂载目录在容器中对应的目录，未填写时挂载点和挂载目录路径保持一致。

运行命令：指定运行命令，包括程序的入口文件和程序参数等，例如：python train.py --data-path=/data --batch=54。

环境变量：指定程序运行所需要的额外环境变量。

共享内存(shm-size)：设置作业运行容器的共享内存，默认为作业运行所在节点的/etc/docker/daemon.json 配置文件中配置 default-shm-size 参数的值。

并行实例数：提交以 DeepSpeed 实现的并行 AI 训练程序，指定并行训练的实例数量。

资源规格：必选项，指定每个并行实例所使用的计算资源。只能选择 GPU 资源组的规格列表。

自动调参：是否启用自动调参功能，具体使用方式见“自动调参”模块。

预置表单：显示已保存的预置表单列表，可以将某个表单“设为默认表单”，或者删除表单。

应用资源监控：显示当前集群中可用于当前计算应用的 CPU 核数和当前应用排队的作业数。

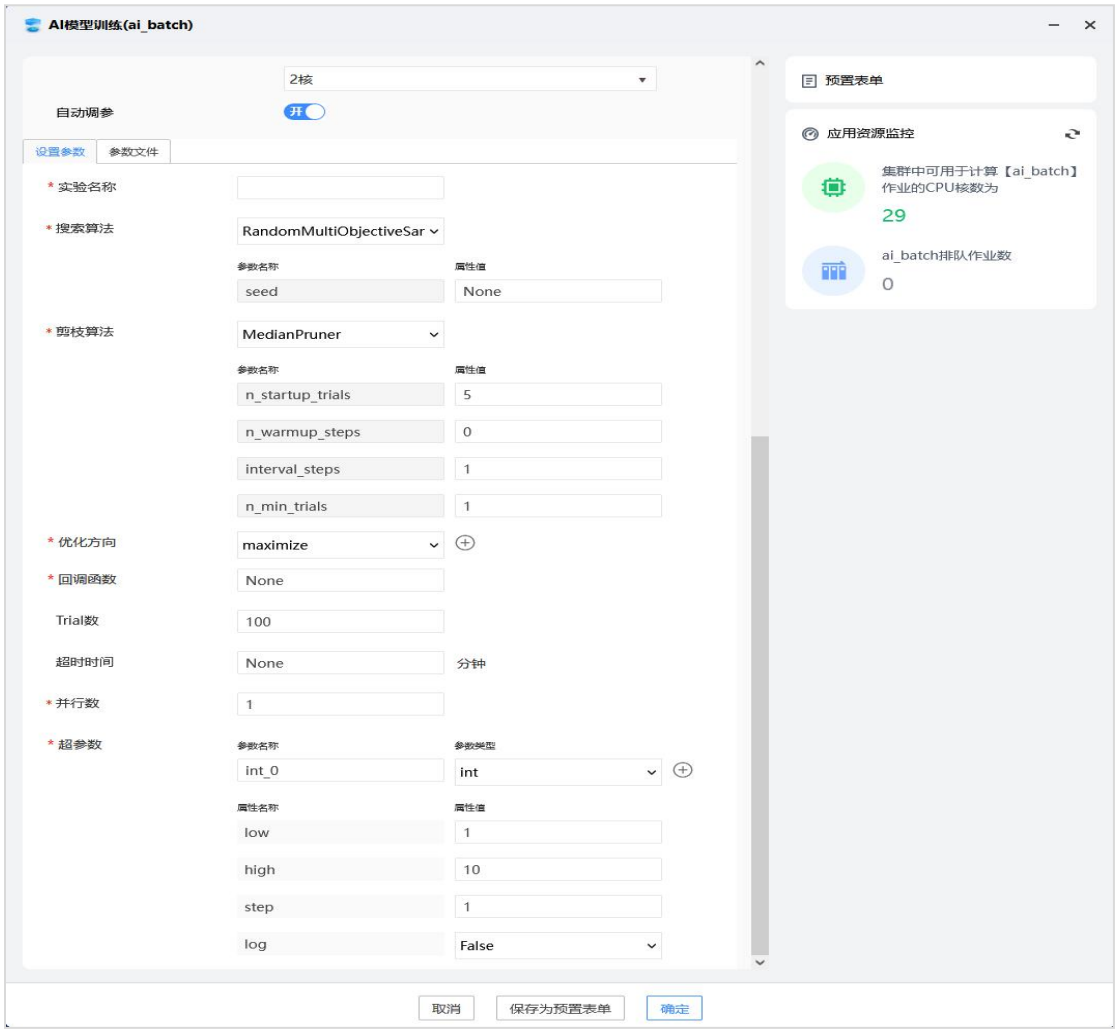
2.3.7. 自动调参

人工智能软件平台在作业提交模块引入了智能的自动调参功能，旨在解决训练模型过程中繁琐的参数设置和结果统计的问题。传统方式下，每次实验都需要手动设定参数，对于性能较差的模型，更需要手动中止，这不仅工作量大，还容易导致计算资源的浪费。自动调参的引入通过设置试验次数和实验时间，能够及时中止性能较差的模型，实现了对计算资源的有效利用。

此外，自动调参还具备自动保存实验参数和结果的功能。在每次实验结束后，系统会自动记录并保存实验的关键参数和结果，最终提供最优参数组合，从而显著减少了训练模型所需的时间。这一智能调优功能为用户提供了更高效、便捷的超参数搜索体验，使得模型优化过程更加智能和可控。

2.3.7.1. 设置参数

打开自动调参开关，展示自动调参配置，如下所示：



设置参数

在“设置参数”标签页中，可查看设置自动调参的所有参数，如下所示：

实验名称：必填项，作业训练过程中创建实验的名称。

搜索算法：人工智能平台中集成的搜索算法。搜索算法主要有“RandomMultiObjectiveSampler”，“NSGAIIMultiObjectiveSampler”，“MOTPEMultiObjectiveSampler”，“TPESampler”，“QMCSampler”，“MOTPESampler”，“RandomSampler”，“GridSampler”，“CmaEsSampler”和“BruteForceSampler”，搜索算法的参数解释见附件八搜索算法。

剪枝算法：选择是否对实验中的试验进行剪枝，当选择“NopPruner”时，表示不剪枝；其他剪枝算法“PercentilePruner”，“HyperbandPruner”，“MedianPruner”，“PatientPruner”，“ThresholdPruner”，“SuccessiveHalvingPruner”的剪枝的过程中具体剪枝参数，见附件八剪枝算法。

优化方向：优化方向可选参数有“minimize”和 “maximize”。当实验的优化目标有多个时，需要设置多个优化方向。

回调函数：自定义回调函数名称，回调函数的个数应该与优化方向个数相同。此处设置的回调函数名称，可以作为实验管理中查看指标的索引。

Trial 个数：设置实验所有的 Trial 个数，默认为 100 个。当参数的组合数量小于 100 时，运行 Trial 为实际组合数。

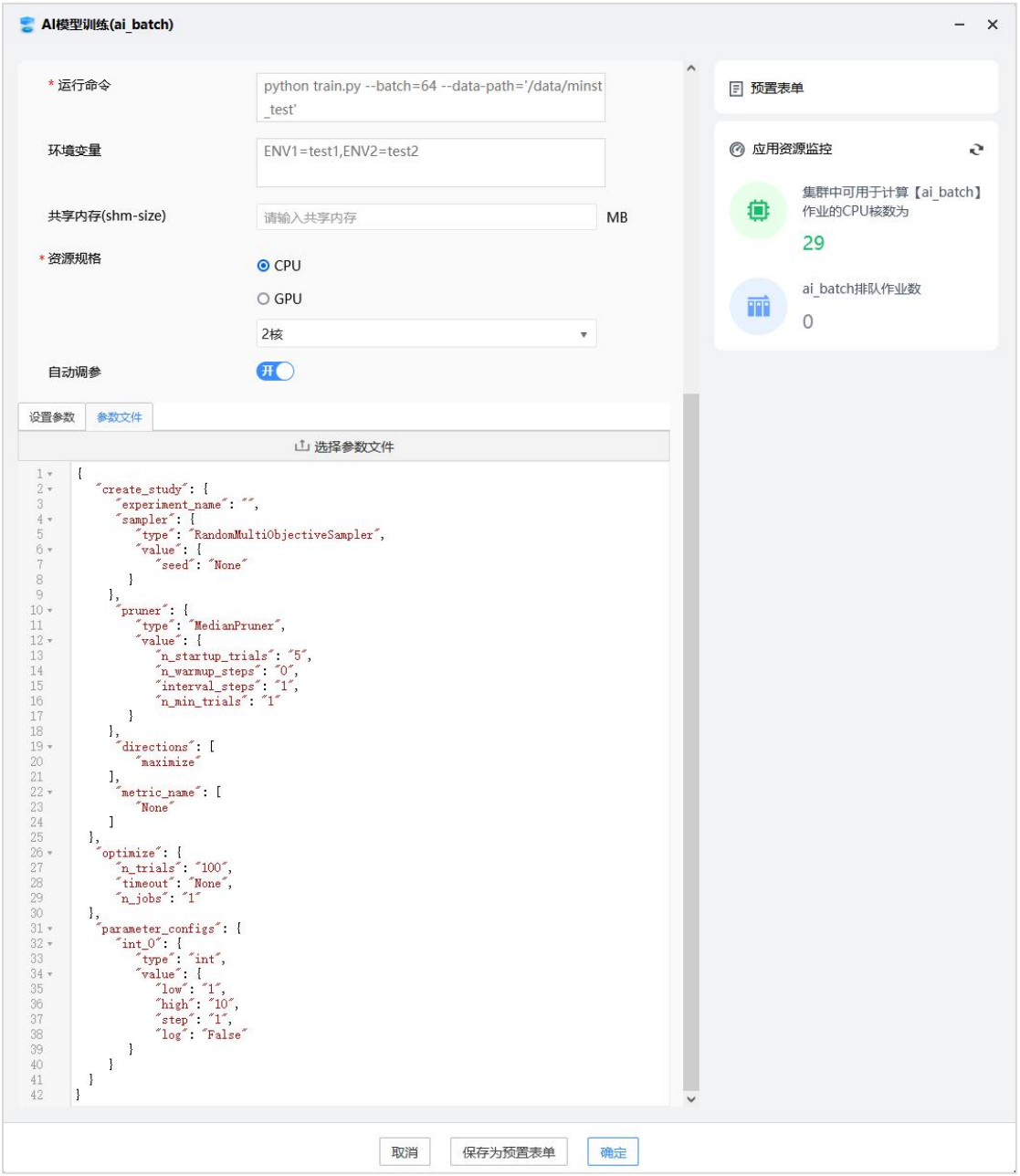
超时时间：本地实验运行的时长，单位为分钟，默认不限制运行时间。

并行数：同时启动的线程数，默认为 1 个。当并行数大于当前环境的核数时，实际并行数为当前的核数。

超参数：设置调参参数和参数搜索范围，可添加的参数不限个数。参数名，调参参数的名称；参数类型，包括 int，float，uniform，loguniform，discrete_uniform 和 categorical。

2.3.7.2. 参数文件

点击“参数文件”标签页按钮，切换至参数文件界面。点击“选择参数文件”按钮，可以上传参数文件，如下图所示：



上传文件的格式，如下所示：

```
{
  "create_study": {
    "experiment_name": "",
    "sampler": {
      "type": "RandomMultiObjectiveSampler",
      "value": {
        "seed": "None"
```

```
      }
    },
    "pruner": {
      "type": "MedianPruner",
      "value": {
        "n_startup_trials": "5",
        "n_warmup_steps": "0",
        "interval_steps": "1",
        "n_min_trials": "1"
      }
    },
    "directions": [
      "maximize"
    ],
    "metric_name": [
      "None"
    ]
  },
  "optimize": {
    "n_trials": "100",
    "timeout": "None",
    "n_jobs": "1"
  },
  "parameter_configs": {
    "int_0": {
      "type": "int",
      "value": {
        "low": "1",
        "high": "10",
```

```
        "step": "1",
        "log": "False"
      }
    }
  }
}
```

自动调参的参数：

experiment_name：对应设置参数中的实验名称；

sampler：对应设置参数中的搜索算法；

pruner：对应设置参数中的剪枝算法；

directions：对应设置参数中的优化方向；

metric_name：对应设置参数中的回调函数；

n_trials：对应设置参数中的 trial 个数；

timeout：对应设置参数中的超时时间；

n_jobs：对应设置参数中的并行数；

parameter_configs：对应设置参数中的超参数。

2.3.7.3. 脚本 API 设置

为了实现自动调参功能，除了设置上述参数或者文件之外，需要在运行脚本中添加相关内容：

（1）添加“parse_arg”模型，作为参数入口，如下所示：

```
def parse_arg(args=None):
    """parse arguments"""
    parser = argparse.ArgumentParser(description='tensorflow MNIST Example')
    parser.add_argument('--filters', type=int, default=32,
                        help='number of the filters')
```

```
    parser.add_argument('--kernel_size', type=int, default=2,
                        help='number of the kernel size')
    parser.add_argument('--strides', type=int, default=1,
                        help='strides size')
    parser.add_argument('--activation', type=str, default='softmax',
                        help='activation function')
    parser.add_argument('--learning_rate', type=float, default=0.01,
                        help='learning rate (default: 0.01)')
    parser.add_argument('--trial', type=object,
default=optuna.trial._trial.Trial,
                        help='trial')
    args = parser.parse_args(args)
    return args
```

参数名称应该与超参数中设置的参数保持一致，另外需要额外添加以下内容。

```
 Import optuna
 parser.add_argument('--trial', type=object,
default=optuna.trial._trial.Trial,
                        help='trial')
```

（2）如果设置自动剪枝算法，需要添加相应框架的回调，如下所示：

```
 from libaiflow.intergration.callback import TFKerasPruningCallback
    callbacks = [
        tf.keras.callbacks.TensorBoard(log_dir=logdir),

tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_prefix,verbose
= 0,save_best_only=True),
        tf.keras.callbacks.EarlyStopping(patience=3),
        TFKerasPruningCallback(args.trial, 'val_accuracy'), #自动调
参剪枝回调
```

```
    ]
```

每个框架具有不同的剪枝回调，具体说明见附件八回调函数。

（3）脚本返回值的个数及目标方向和优化方向保持一致，此处的返回值也就是回调函数显示的内容。整个脚本如下所示：

```
import argparse
import os
import tensorflow as tf
from libaiflow.intergration.callback import TFKerasPruningCallback
import optuna


def parse_arg(args=None):
    parser = argparse.ArgumentParser(description='tensorflow MNIST Example')
    parser.add_argument('--filters', type=int, default=32,
                        help='number of the filters')
    parser.add_argument('--kernel_size', type=int, default=2,
                        help='number of the kernel size')
    parser.add_argument('--strides', type=int, default=1,
                        help='strides size')
    parser.add_argument('--activation', type=str, default='softmax',
                        help='activation function')
    parser.add_argument('--learning_rate', type=float, default=0.01,
                        help='learning rate (default: 0.01)')
parser.add_argument('--trial', type=object,
default=optuna.trial._trial.Trial,help='trial')
    args = parser.parse_args(args)
    return args


def load_data():
```

```
    path = "./data/mnist.npz"
    import numpy as np
    with np.load(path) as f:
      x_train, y_train = f['x_train'], f['y_train']
      x_test, y_test = f['x_test'], f['y_test']
      return (x_train, y_train), (x_test, y_test)

  def main(args):
      from tensorflow.keras.backend import clear_session
      from tensorflow.keras.datasets import mnist
      from tensorflow.keras.layers import Conv2D, Dense, Flatten
      from tensorflow.keras.models import Sequential
      from tensorflow.keras.optimizers import RMSprop
      clear_session()
      N_TRAIN_EXAMPLES = 3000
      N_VALID_EXAMPLES = 1000
      BATCHSIZE = 128
      CLASSES = 10
      EPOCHS = 10

      (x_train, y_train), (x_valid, y_valid) = load_data()
      img_x, img_y = x_train.shape[1], x_train.shape[2]
      x_train = x_train.reshape(-1, img_x, img_y,
1)[:N_TRAIN_EXAMPLES].astype("float32") / 255
      x_valid = x_valid.reshape(-1, img_x, img_y,
1)[:N_VALID_EXAMPLES].astype("float32") / 255
      y_train = y_train[:N_TRAIN_EXAMPLES]
      y_valid = y_valid[:N_VALID_EXAMPLES]
      input_shape = (img_x, img_y, 1)
```

```
    model = Sequential()
    model.add(
        Conv2D(
            filters=args.filters,kernel_size=args.kernel_size,
            strides=args.strides,activation=args.activation,
            input_shape=input_shape,
        )
    )
    model.add(Flatten())
    model.add(Dense(CLASSES, activation="softmax"))
    model.compile(
        loss="sparse_categorical_crossentropy",
        optimizer=RMSprop(learning_rate=args.learning_rate),
        metrics=["accuracy"],
    )
    checkpoint_dir = 'model'
 if not os.path.exists(checkpoint_dir):
    os.makedirs(checkpoint_dir)
    logdir = os.path.join('logs', str(args.trial.number))
    checkpoint_prefix = os.path.join(checkpoint_dir,
str(args.trial.number))
    callbacks = [
        tf.keras.callbacks.TensorBoard(log_dir=logdir),

tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_prefix,verbose
= 0,save_best_only=True),
        tf.keras.callbacks.EarlyStopping(patience=3),
```

```
        TFKerasPruningCallback(args.trial, 'val_accuracy'), #自动调参剪枝回调
    ]
    model.fit(
        x_train,
        y_train,
        validation_data=(x_valid, y_valid),
        shuffle=True,
        batch_size=BATCHSIZE,
        epochs=EPOCHS,
        verbose=False,
        callbacks=callbacks,
    )
    score = model.evaluate(x_valid, y_valid, verbose=0)
    return score[1]

if __name__ == '__main__':
    args = parse_arg()
    main(args)
```

2.4. 强化学习

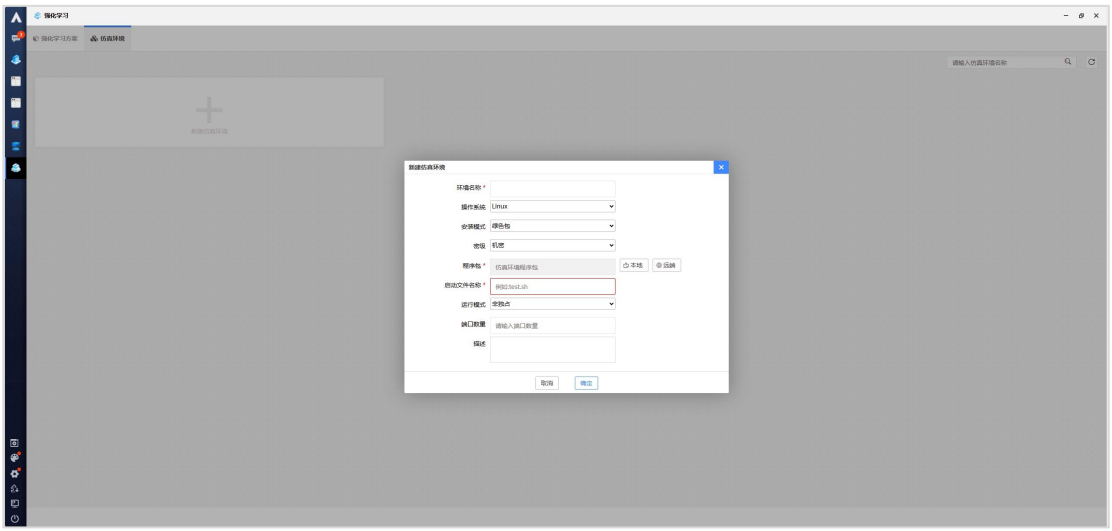
强化学习（Reinforcement Learning, RL）正逐渐成为推动技术进步的关键力量。它通过模拟试错学习的过程，使智能体能够在复杂环境中自主学习并做出最优决策。为了满足日益增长的市场需求，现将 Ray RLlib 的强化学习功能集成平台，旨在简化 RL 模型的开发、训练和部署流程。用户登录应用门户后，点击桌面“强化学习”图标，进入强化学习页面。

2.4.1. 构建方案设计流程

2.4.1.1. 新建仿真环境

在强化学习应用选择仿真环境，点击“新建仿真环境”卡片，弹出新建方案的表单页。

当安装模式选择“绿色包”，表单页如下图所示：



新建绿色包仿真环境

图中每个参数的具体含义如下：

环境名称：用户自定义，但是不能与现有仿真环境名称一致。

安装模式：仿真环境的安装模式，分为绿色包和已安装环境，默认为绿色包。

密级：管理员开启密级功能后，显示此选项，默认为机密，用于数据安全、保密。

程序包：支持 zip、tar、tar.gz、tar.bz2、tar.xz 结尾的压缩包，可本地或者远端上传，必填项。

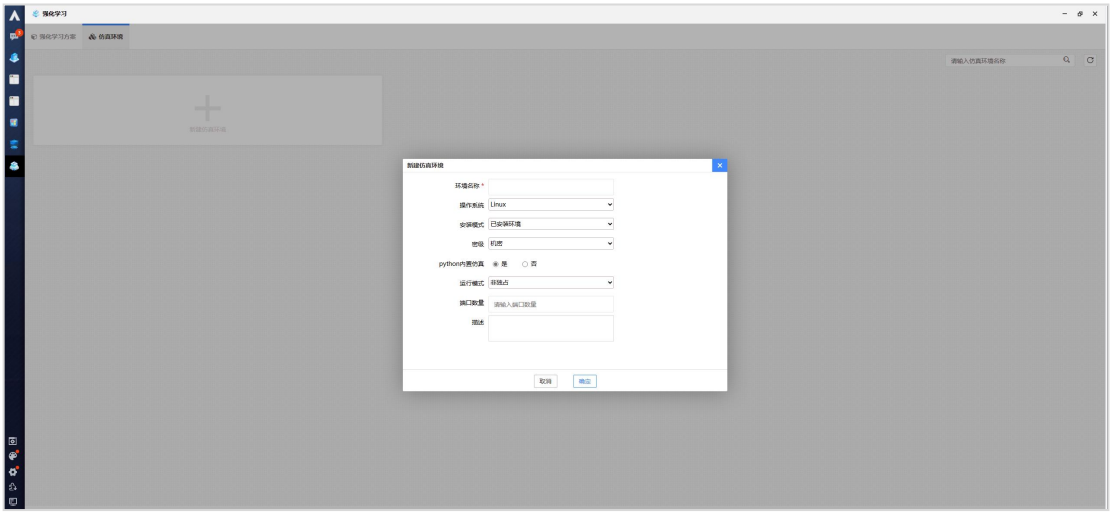
启动文件名称：用户自定义，支持 bat、exe、cmd、sh 结尾的文件全称，必填项。

运行模式：仿真环境的运行模式，默认为非独占。

端口数量：仿真环境需要的端口数量，非必填项，端口数只能输入 1-5 的正整数。

描述：用于描述仿真环境的相关内容，选填。

当安装模式选择“已安装环境”，表单页如下图所示：



新建已安装仿真环境

图中每个参数的具体含义如下：

环境名称：用户自定义，但是不能与现有仿真环境名称一致。

安装模式：仿真环境的安装模式，分为绿色包和已安装环境，默认为绿色包。

密级：管理员开启密级功能后，显示此选项，默认为机密，用于数据安全、保密。

python 内置仿真：仿真环境是否是 python 内置仿真，如果选择是，页面参数不变；如果选择否，增加“启动文件绝对路径”参数。

启动文件绝对路径：用户自定义，输入启动文件的绝对路径，例如：/home/jhadmin/test.sh，必填项。

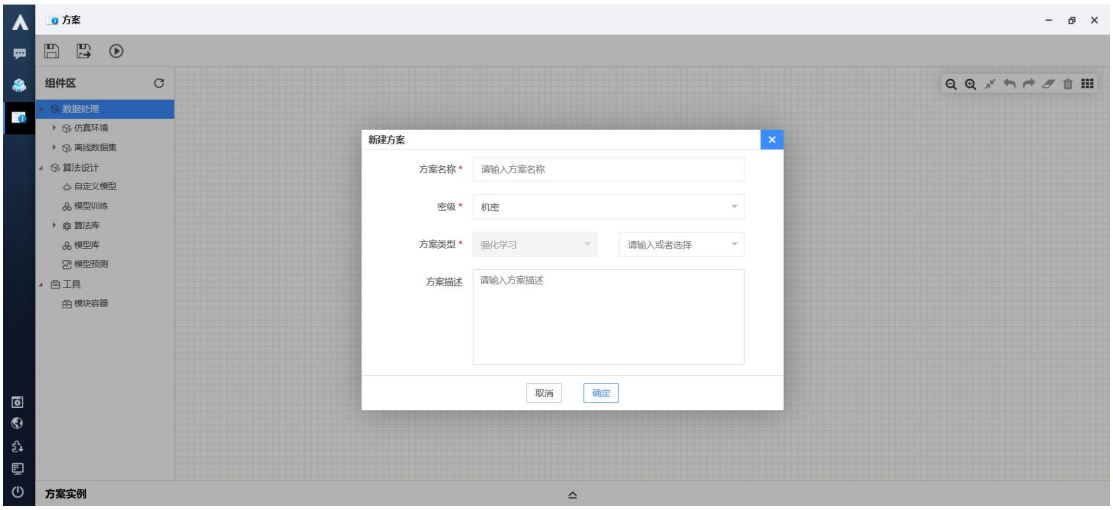
运行模式：仿真环境的运行模式，默认为非独占。

端口数量：仿真环境需要的端口数量，非必填项，端口数只能输入 1-5 的正整数。

描述：用于描述仿真环境的相关内容，选填。

2.4.1.2. 新建设计方案

在强化学习应用选择强化学习方案，点击“新建方案”卡片，弹出新建方案的表单页，如下图所示：



新建强化学习方案

图中每个参数的具体含义如下：

方案名称：用户自定义，但是不能与现有强化学习方案名称一致。

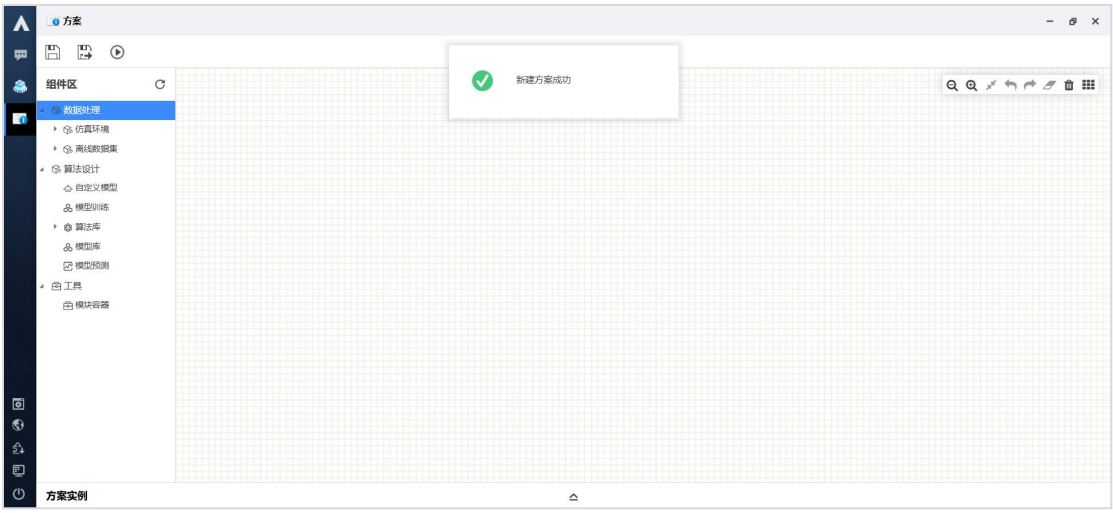
密级：管理员开启密级功能后，显示此选项，默认为机密，用于数据安全、保密。

方案类型：默认为已选的方案类型，也可以自定义方案类型。

方案描述：用于描述方案设计的相关内容，选填。

2.4.1.3. 可视化设计

填写完成新建强化学习方案表单后，点击“确定”按钮，进入方案设计页面。

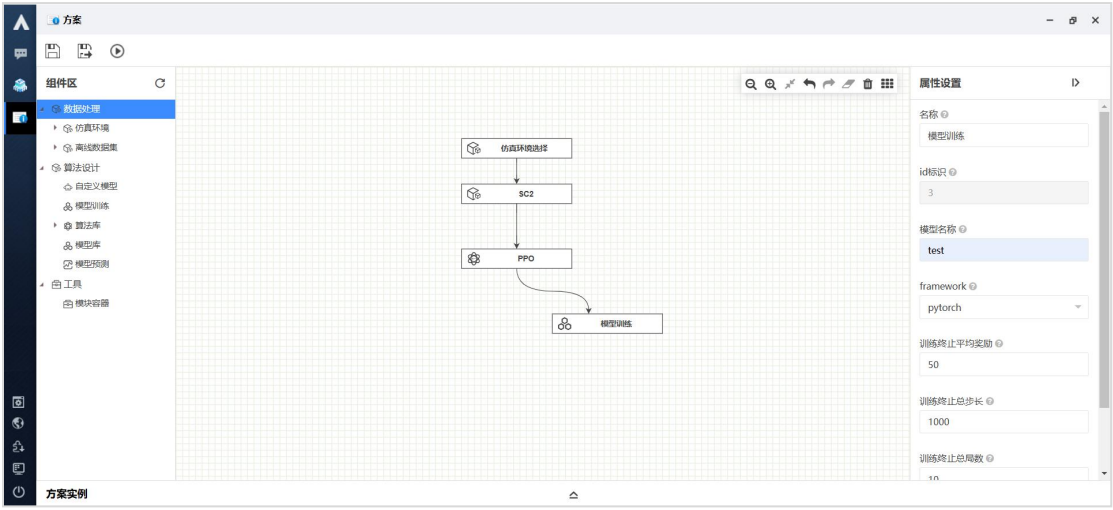


强化学习方案设计页面

“强化学习方案设计”左侧区域为组件区，组件区又根据组件属性分为“数据处理”，“算法设计”和“工具”。每一个组件都是一个模块的封装。

“强化学习方案设计”右侧区域为属性区，组件拖拽到画布区域后，属性设置面板就会从画布右侧滑出，用户可自行配置参数。当属性设置有误或未设置属性时，设计区的组件右侧会显示黄色叹号的图标，当鼠标移至此图标上时会显示出错误信息，必须纠正错误后才能够开始运行此方案。

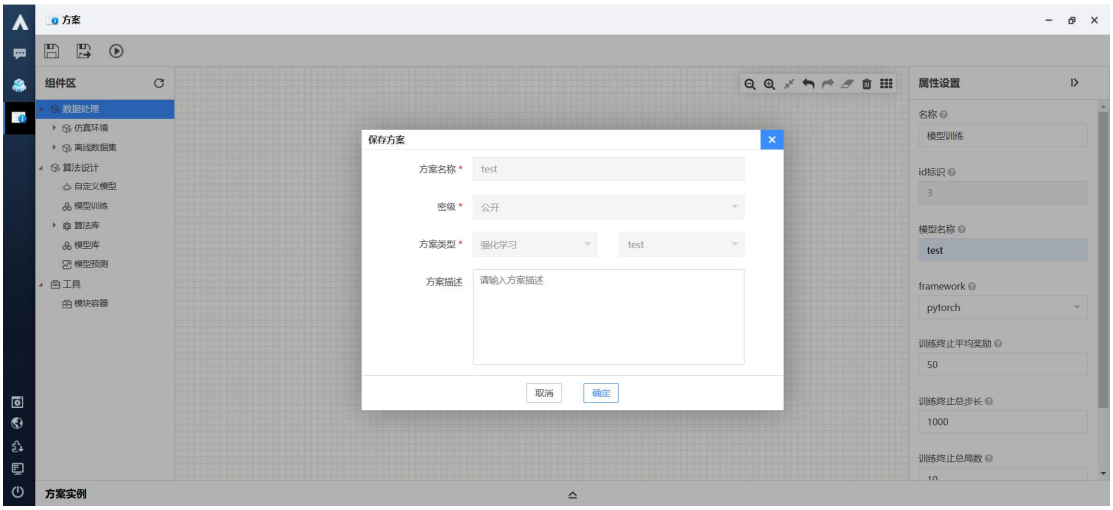
画布右上角的工具栏可对画布区域和画布中的组件进行快捷操作，功能包含缩小、放大、实际比例、后退、前进、删除和清空。其中“删除”按钮仅会删除画布中选中的任意组件，“清空”则会删除整个画布中的内容，长按鼠标右键可拖动整个画布。



2.4.2. 操作

2.4.2.1. 方案保存

点击工具栏中的“保存”按钮，即可将方案描述和画布上的强化学习方案设计保存至数据库，当再次打开该方案时，画布上会显示最近一次保存的强化学习方案。“保存方案”页面，如下图所示：



保存方案页面

图中每个参数的具体含义如下：

方案名称：任何状态下均不可修改。

密级：任何状态下均不可修改。

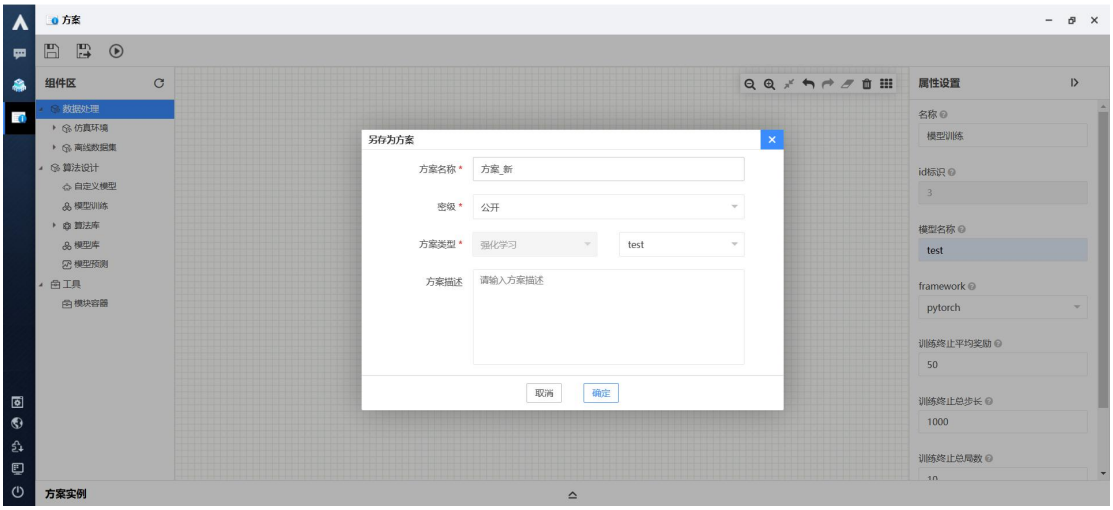
方案类型：任何状态下均不可修改。

方案描述：用于描述方案的相关内容，选填。

2.4.2.2. 方案另存为

点击工具栏中的“另存为”按钮，即可将方案另存为一个新的方案，方案可

以是已保存的和未保存的。“另存为方案”页面。如下图所示：



另存为方案页面

图中每个参数的具体含义如下：

方案名称：用户自定义，但是不能与现有的方案名称一致，必填。

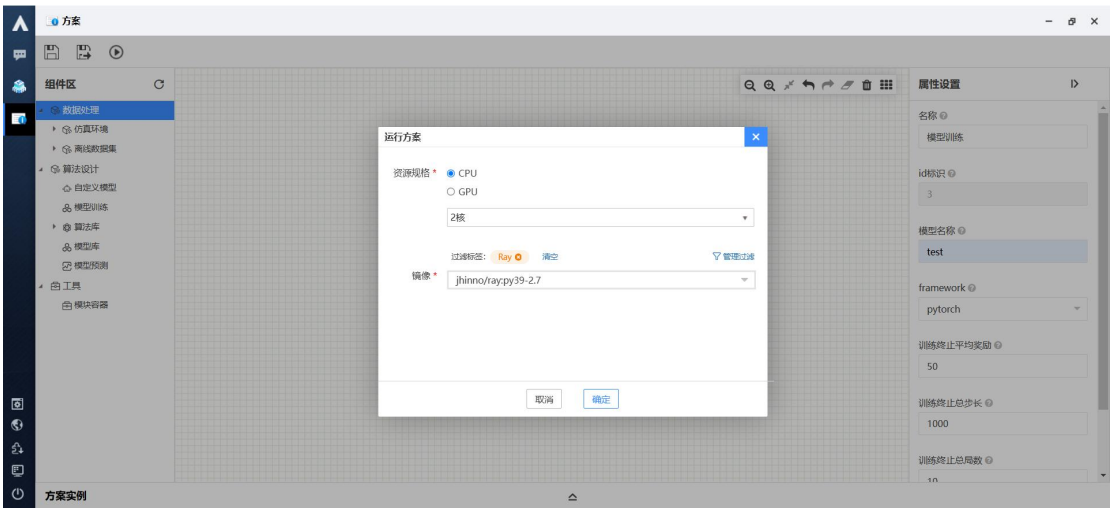
密级：对于另存的方案可根据需要修改密级。

方案类型：默认为当前方案的方案类型，可以选择，也可以自定义方案类型，必填。

方案描述：用于描述方案的相关内容，选填。

2.4.2.3. 方案运行

方案设计完成后，点击“运行”按钮，弹出“运行方案”窗口，选择训练方案需要使用的资源，资源选择后，选择方案使用的镜像，点击“确定”按钮，即可开始训练。如下图所示：



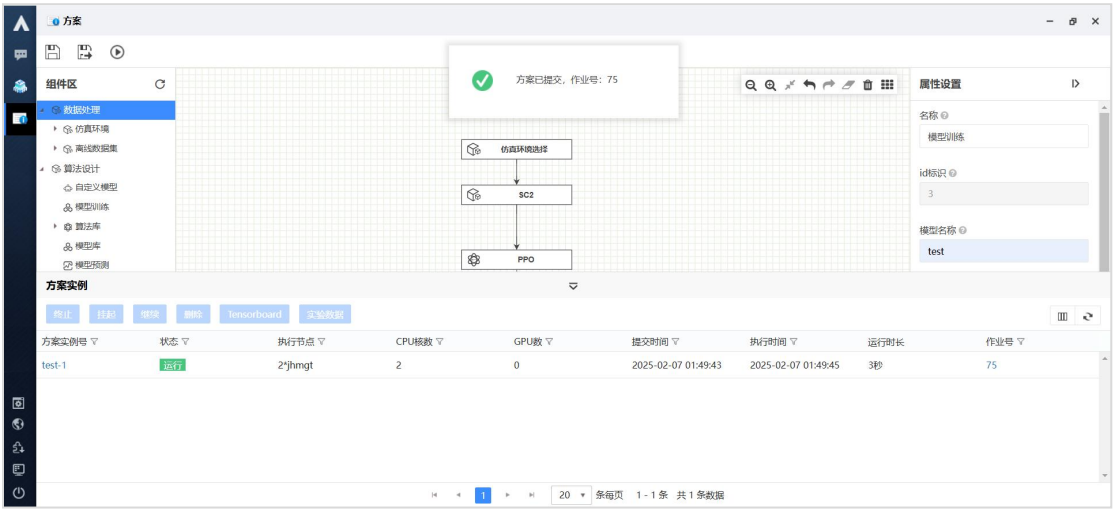
资源配置页面

点击运行按钮，打开选择资源页面，同时进行方案保存。选择资源后，点击确定按钮，即可开始训练方案。

注意：在方案设计页面中，点击“运行”按钮时，错误和问题对应如下：

- 画布中不存在组件时，提示“组件不能为空”。
- 组件的锚点没有连线、必填项属性为空时，提示“请检查异常组件并清除异常”。

开始训练方案后，会自动打开底部方案实例滑块，如下图所示：

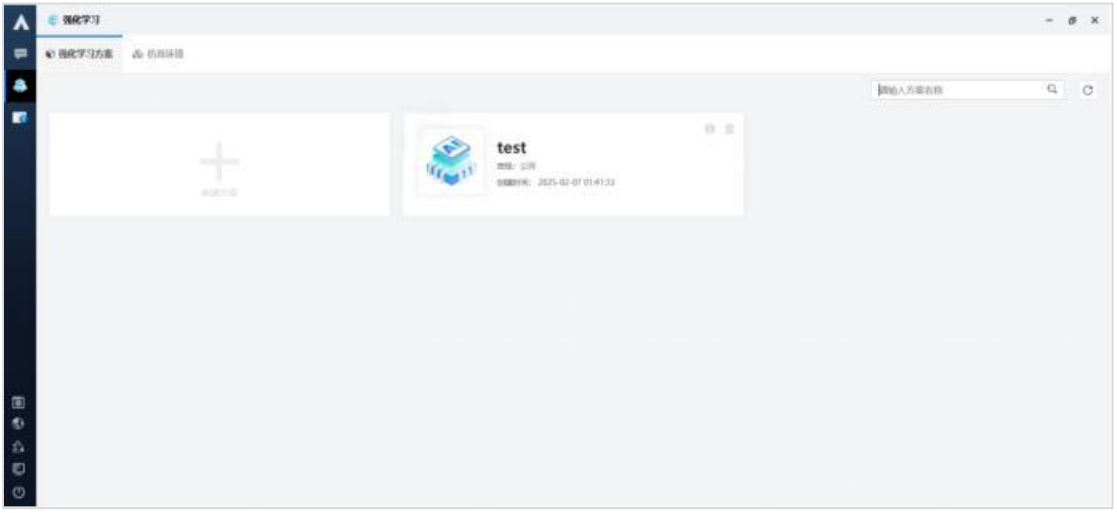


方案实例滑块

底部方案实例滑块除了只能显示该方案的所有实例外，功能操作与方案实例一致，具体操作详情请看“方案实例”章节。

2.4.3. 管理

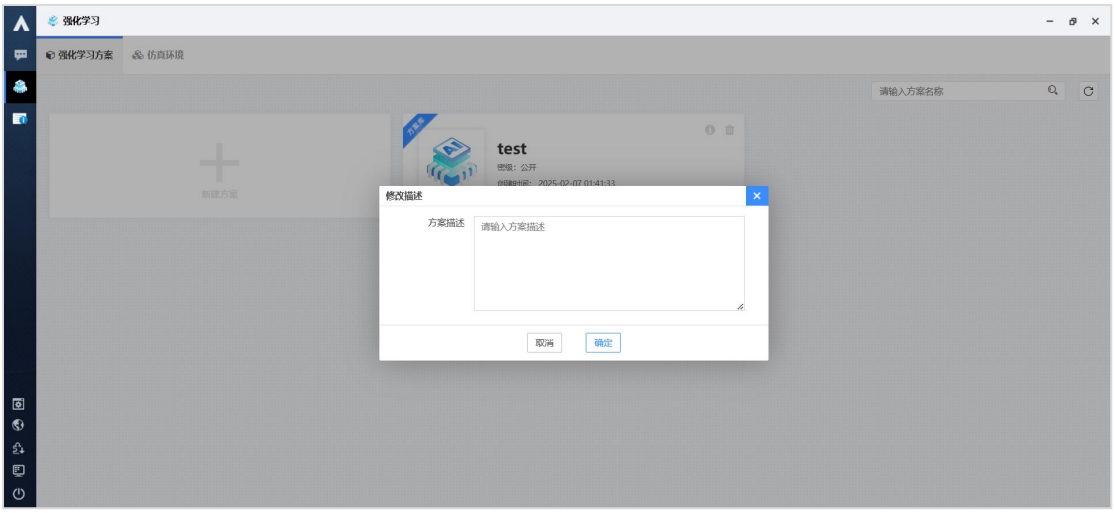
方案新建成功后，在强化学习方案中会自动生成了一个方案卡片，卡片内容包含方案名称、方案 ID、最后更新时间，点击方案卡片的右上角功能图标，支持对方案进行“修改描述”和“删除”操作。



方案设计管理

2.4.3.1. 修改描述

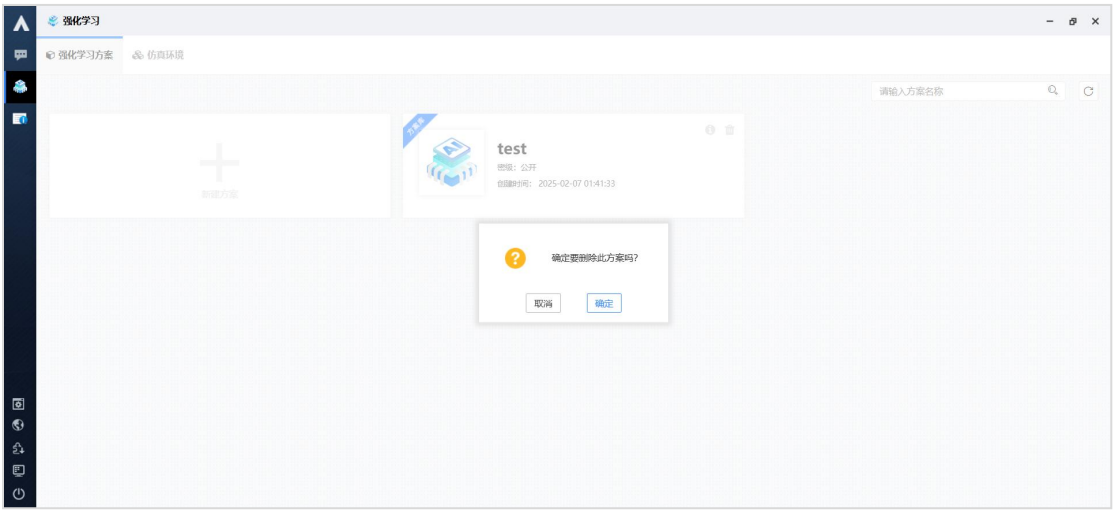
点击“描述”的图标按钮，可在弹出窗口中修改方案的描述信息，如下图所示：



修改方案描述

2.4.3.2. 删除

点击“删除”按钮，可以把该方案卡片删除。

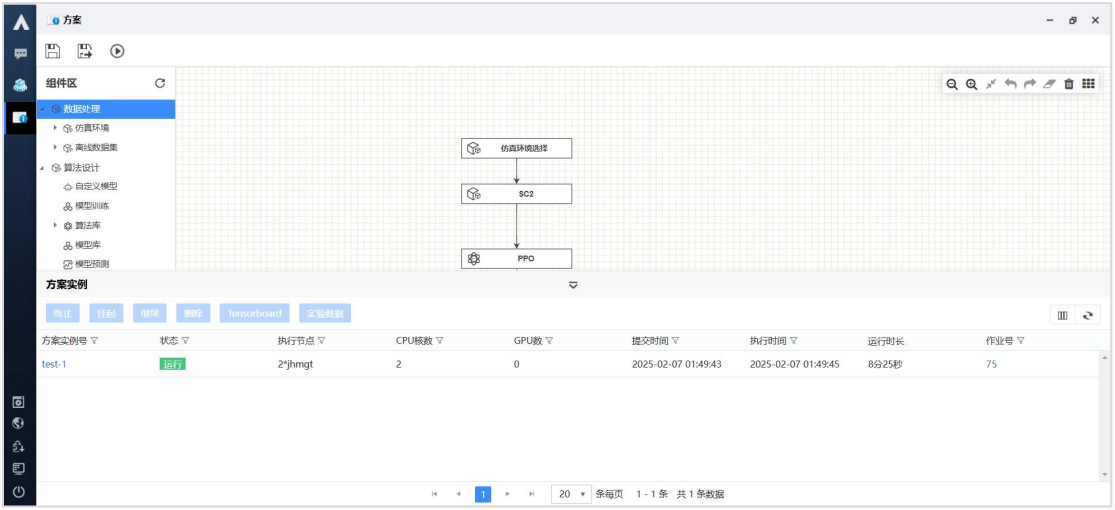


删除方案

2.4.4. 方案实例

方案实例中以列表的形式记录了所有方案的历史运行记录，用户可以通过实例号、名称等筛选功能快速过滤实例。

用户可在方案实例应用中查看自己的所有实例，或在具体的方案设计页面中，通过点击画布下方的方案实例滑块，在展开的面板中查看当前方案实例的信息，亦可对其进行终止、挂起等操作。

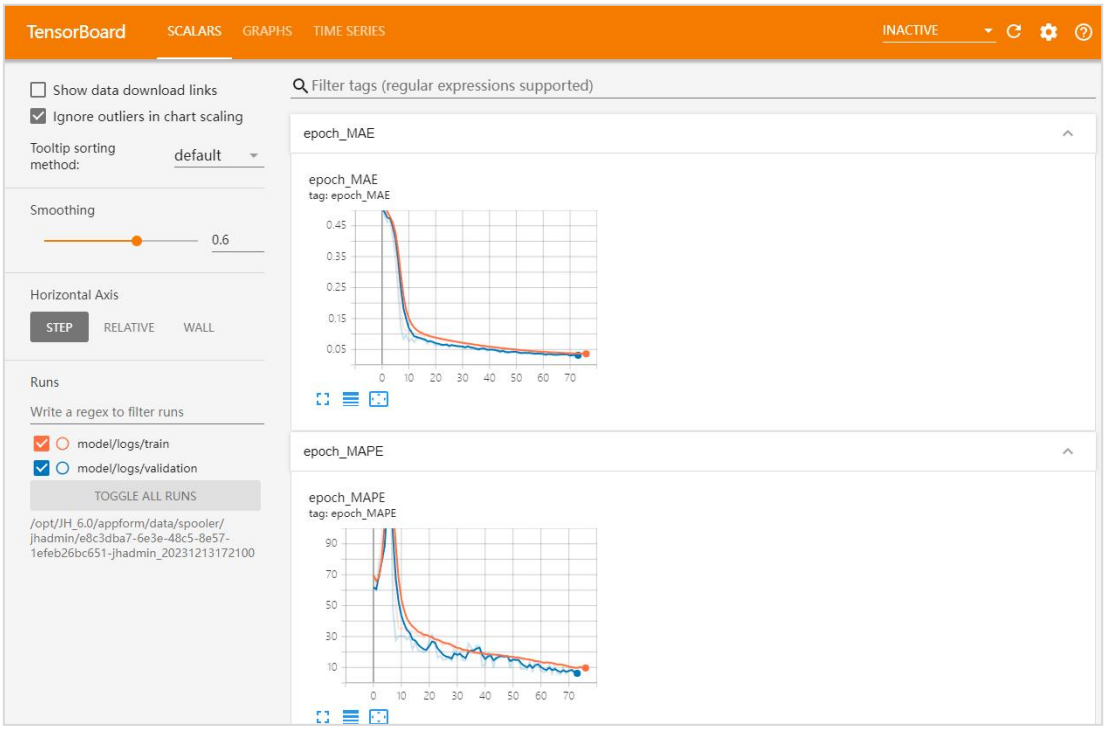


方案实例列表

➢ 功能按钮：

方案实例支持对方案实例进行“终止”“挂起”“继续”“删除”以及打开 Tensorboard 操作。

当强化学习的方案运行完成后，点击“Tensorboard”按钮可以以图表形式查看当前实例运行结果。

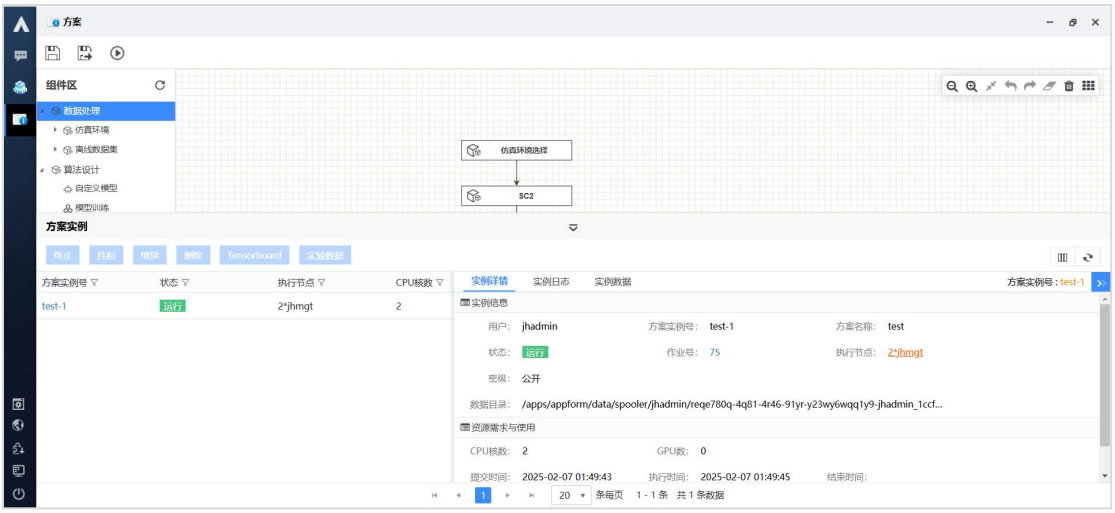


Tensorboard

➢ 方案实例滑块：

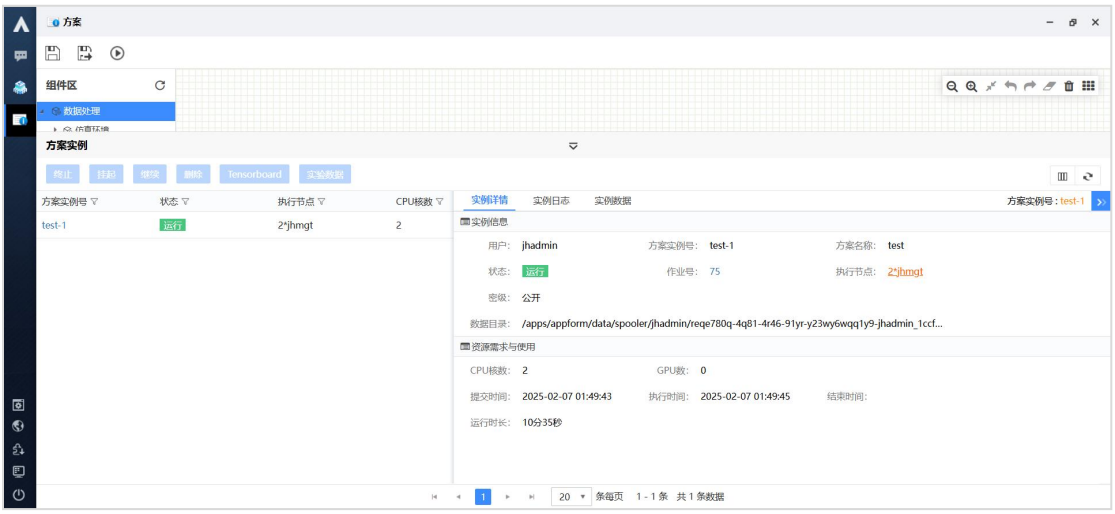
点击“方案实例号”或者双击方案实例列表中的某条记录，会打开右侧滑块。滑块有三个标签页，分别为实例详情、实例日志和实例数据。

当组件中包含“模型评估”组件，且方案运行成功，滑块会有四个标签页，分别为实例详情、实例日志、实例数据。如下图所示：



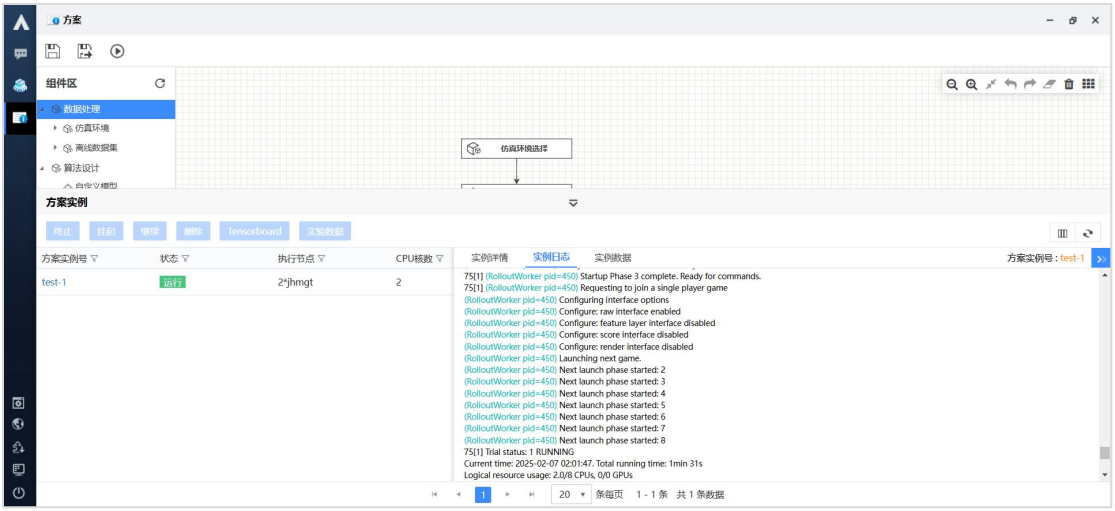
方案设计

实例详情：显示方案实例的基本信息、资源信息。点击作业号，可以查看该实例关联的作业信息。实例详情中包含方案实例关联的数据保存目录。



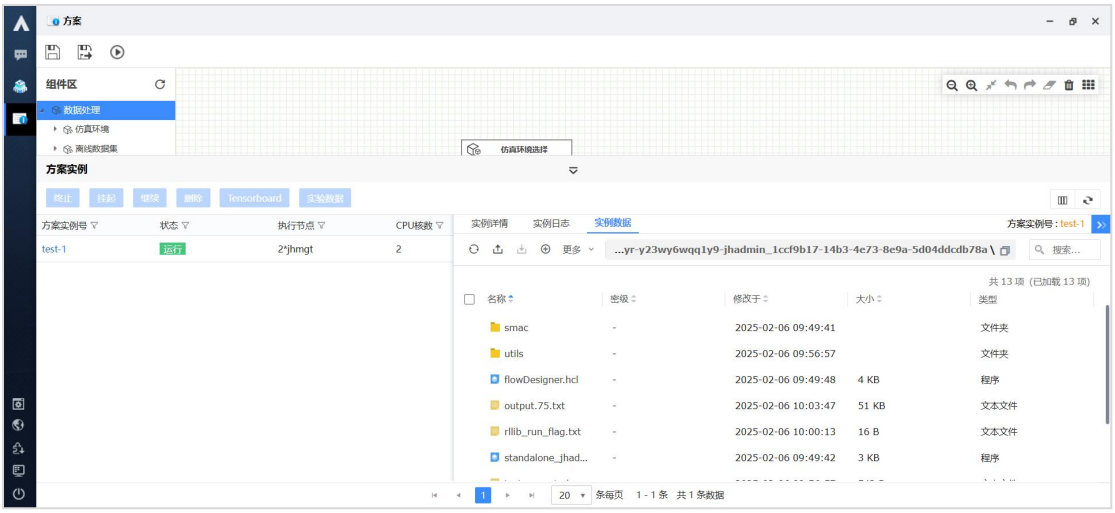
方案实例滑块

实例日志：点击实例日志按钮，打开实例日志界面。方案实例的状态处于正在运行时，该实例日志会实时输出。方案实例的状态处于完成时，会显示该方案实例的所有日志。



方案实例日志

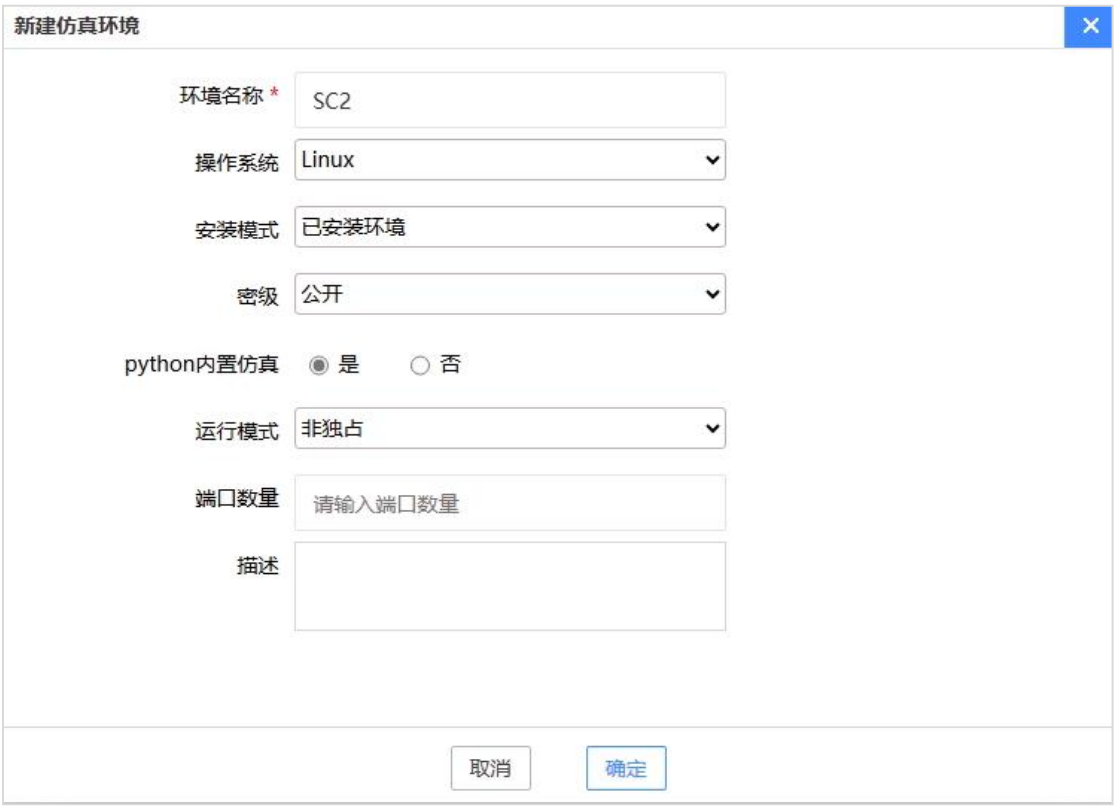
实例数据：点击实例数据按钮，打开实例数据界面，显示该方案实例包含的所有相关文件。



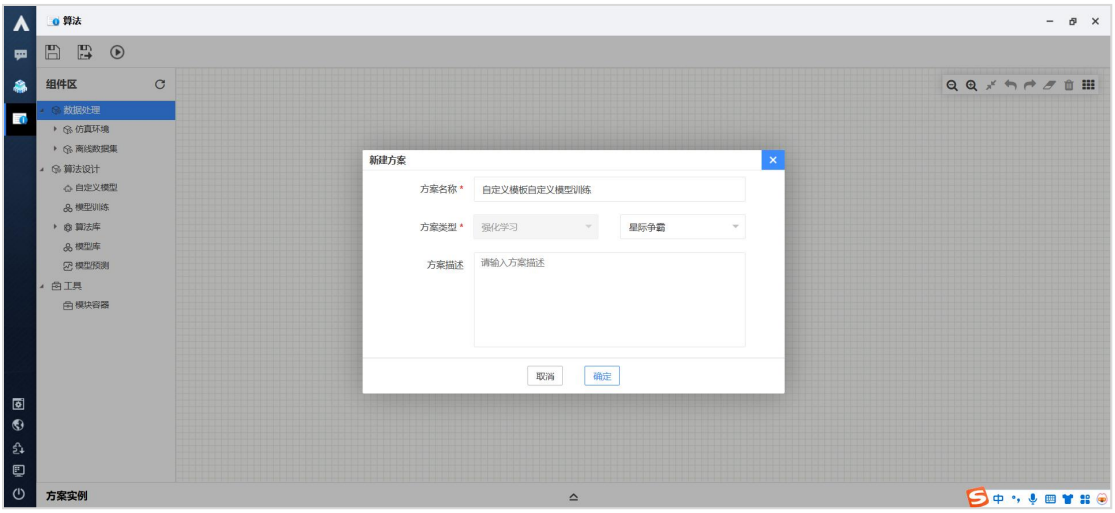
方案实例数据

2.4.5. 使用自定义开发模板，自定义模型进行训练/测试

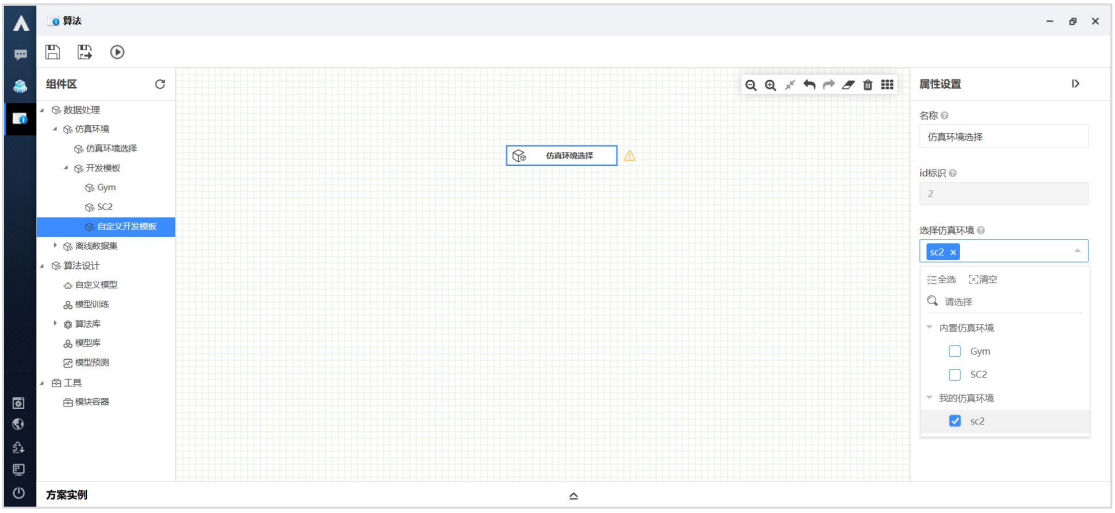
- 新建 python 内置仿真环境



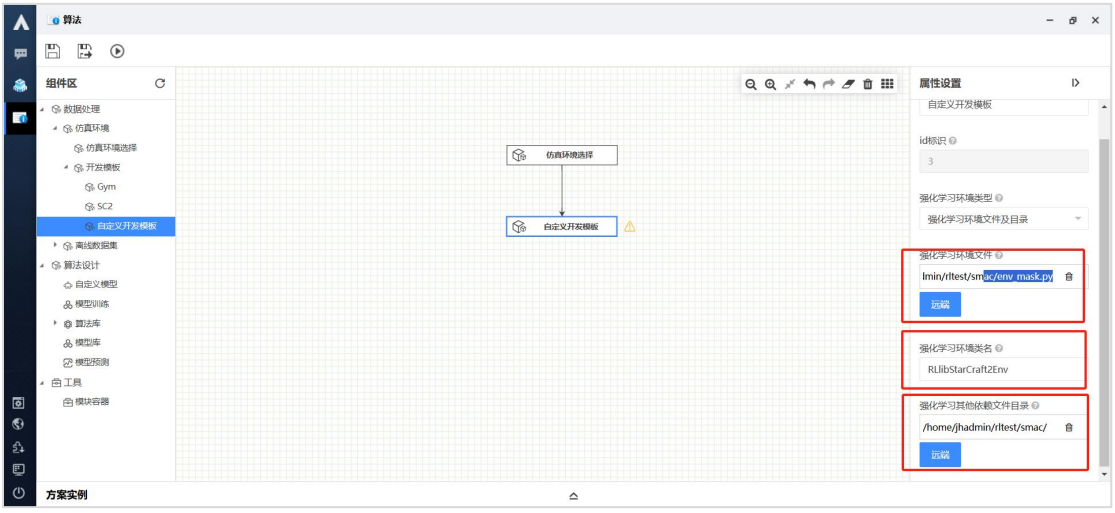
- 新建强化学习方案



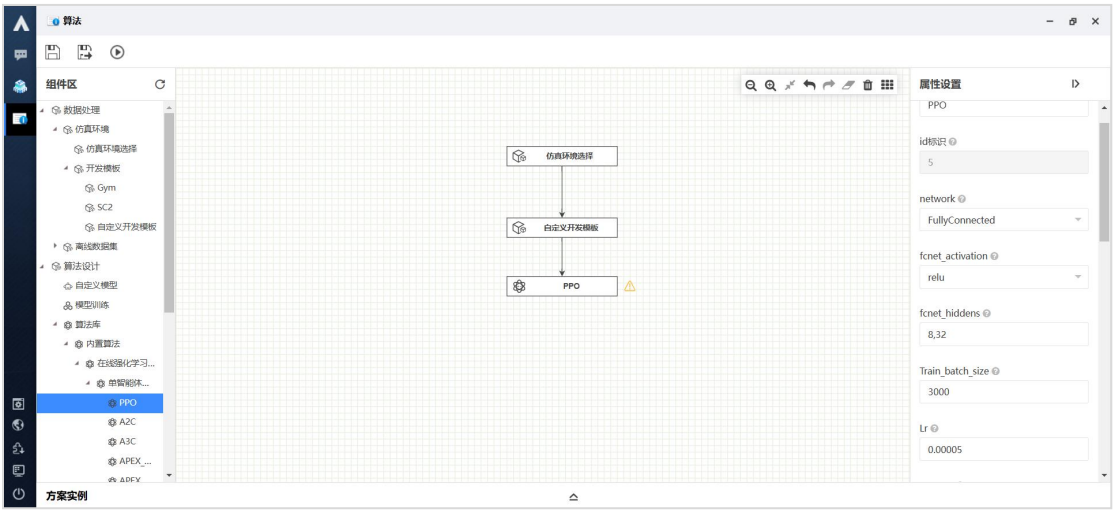
➢ 选择“仿真环境选择”组件，我的仿真环境中选中新增的仿真环境



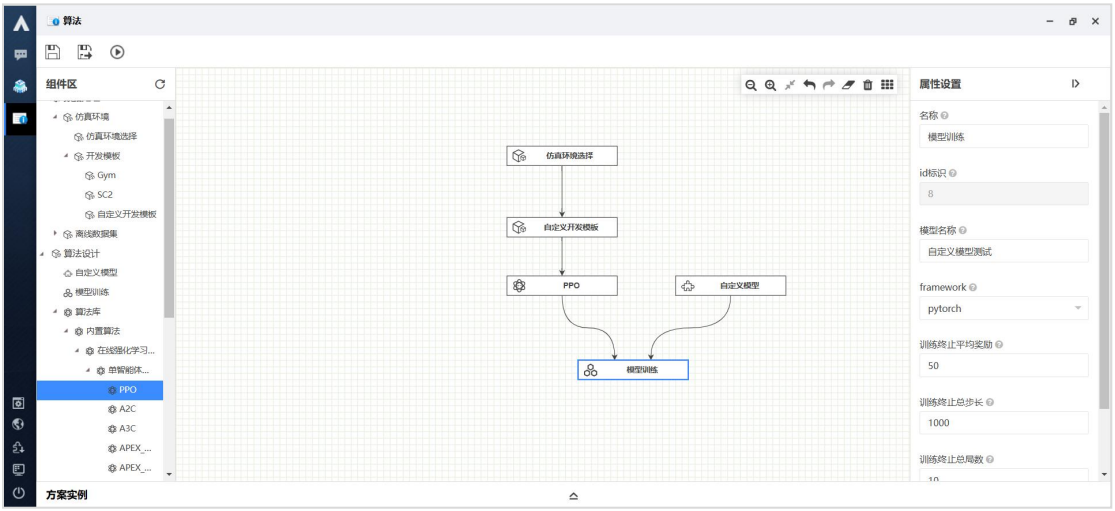
➢ 选择“自定义开发模板”组件，上传强化学习环境及目录



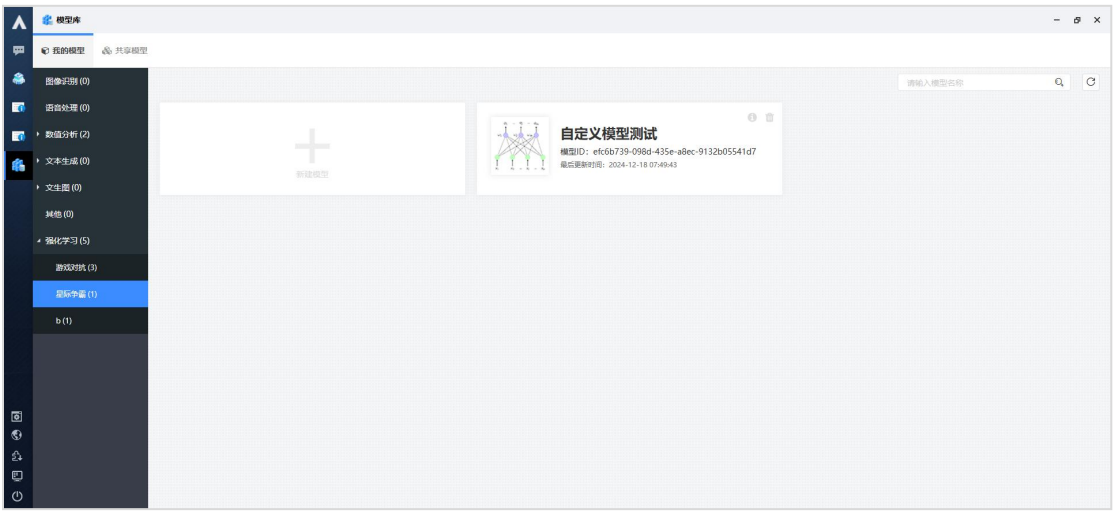
➢ 选择算法库-在线强化学习算法-单智能体算法-PPO



➢ 选择自定义模型组件，选择模型训练

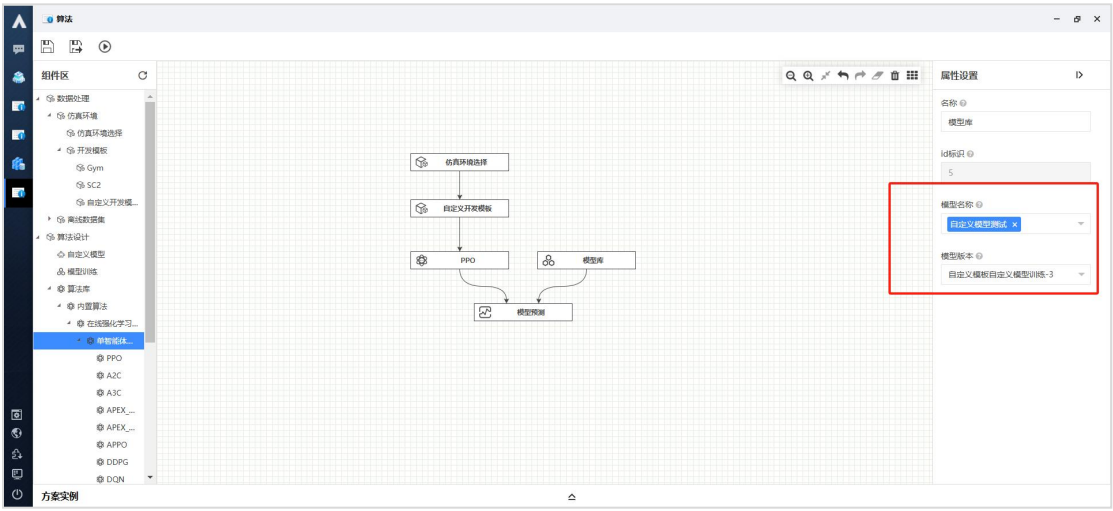


➢ 训练完成后，模型库生成强化学习模型



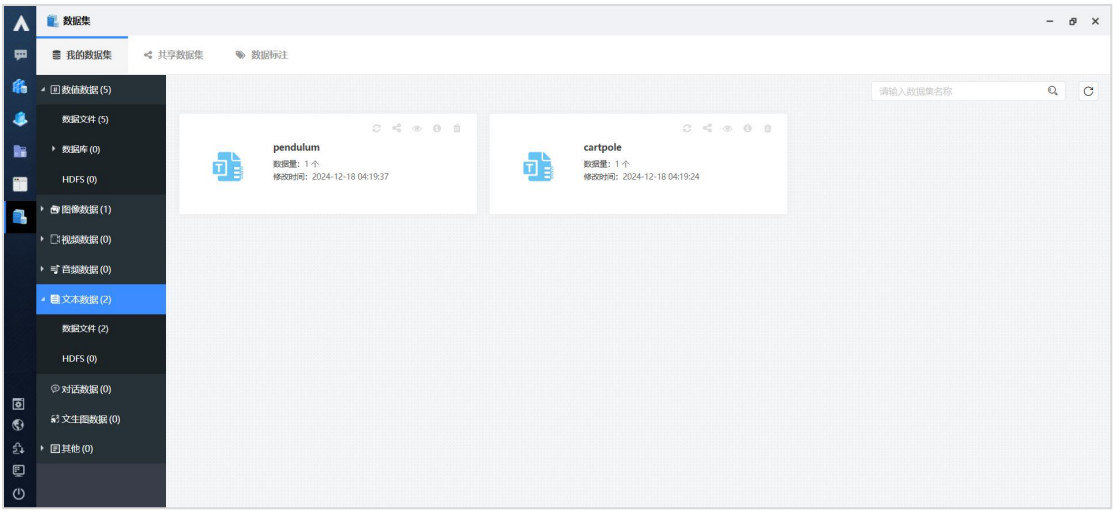
➢ 和上述步骤类似，加载已生成的模型库模型选择模型预测组件，进行强

化学习模型预测

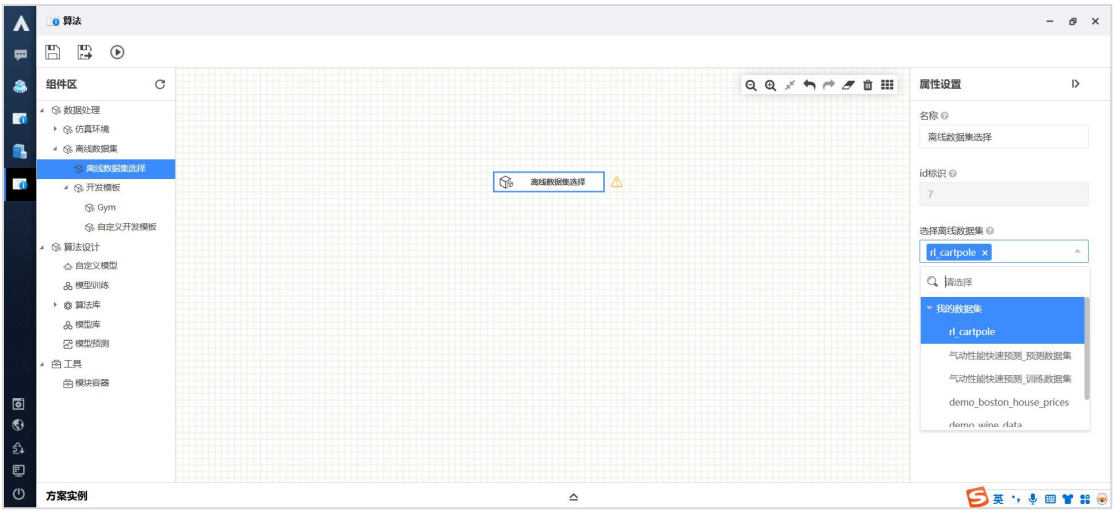


2.4.6. 内置离线数据集进行模型训练、模型预测

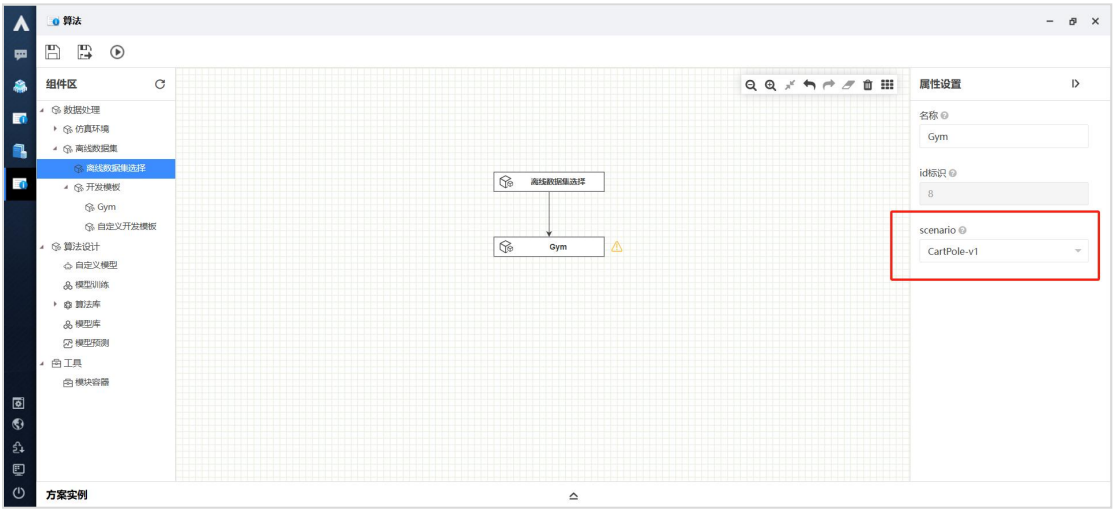
- 新建数据集-文本数据-数据文件



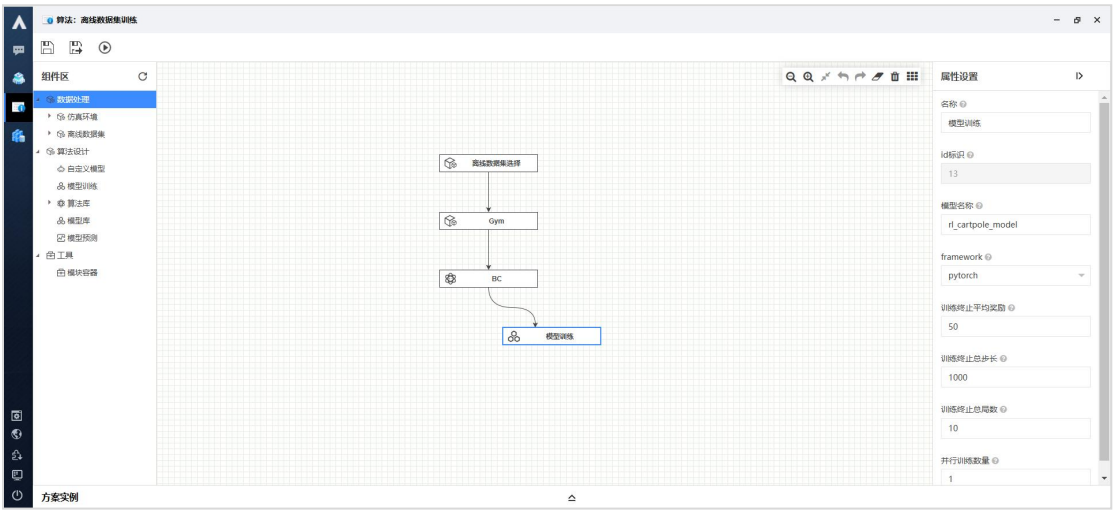
- 选择“离线数据集”组件，我的离线数据集中选中新增的离线数据集



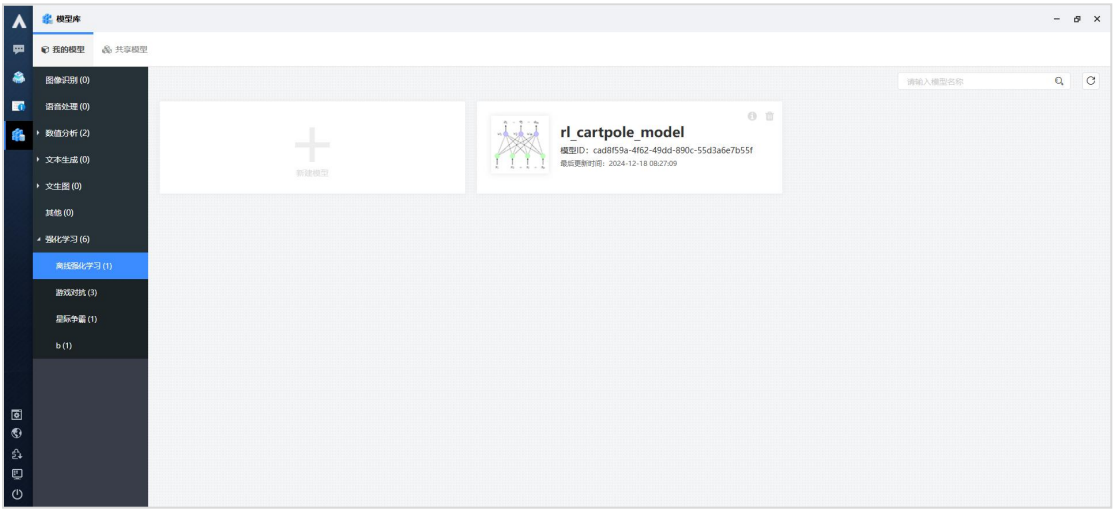
➢ 选择开发模板-Gym，注意右侧 scenario 应与数据集相对应。



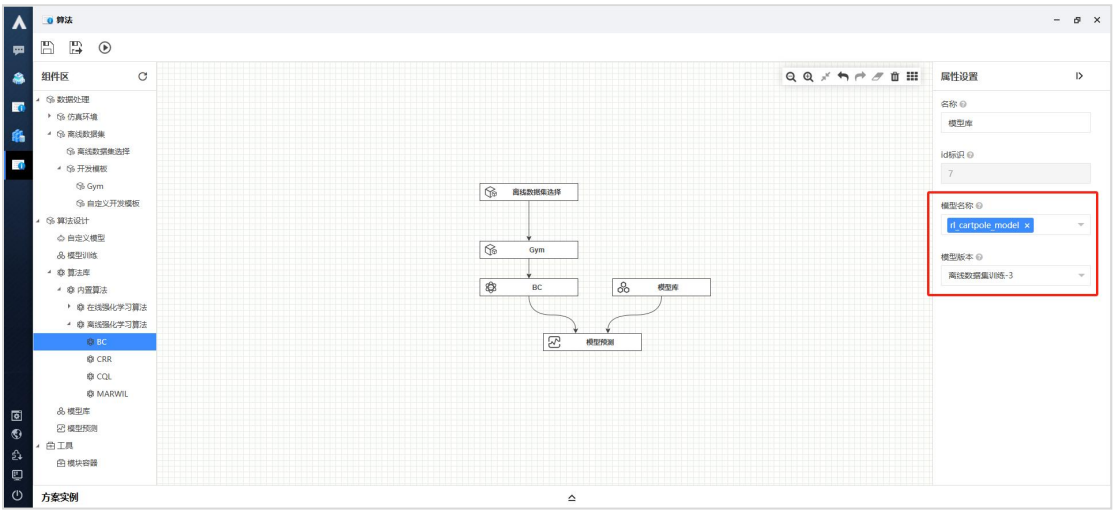
➢ 选择算法库-离线强化学习算法-BC，选择模型训练（离线强化学习算法不支持自定义模型训练）



- 训练完成后，在模型库生成离线强化学习模型

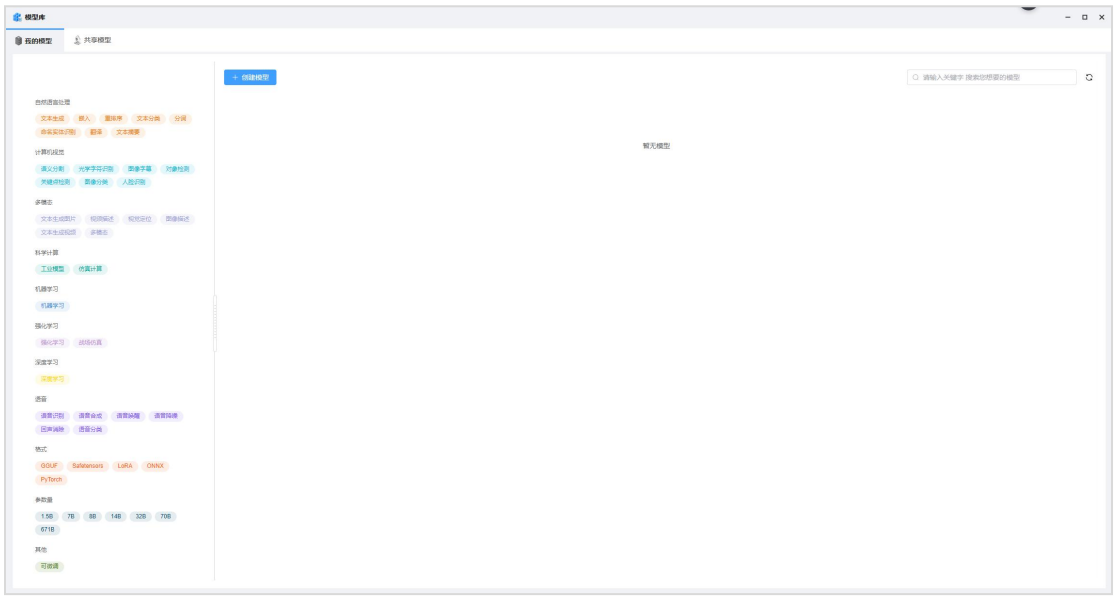


- 和上述步骤类似，加载已生成的模型库模型选择模型预测组件，进行强化学习模型预测



2.5. 模型库

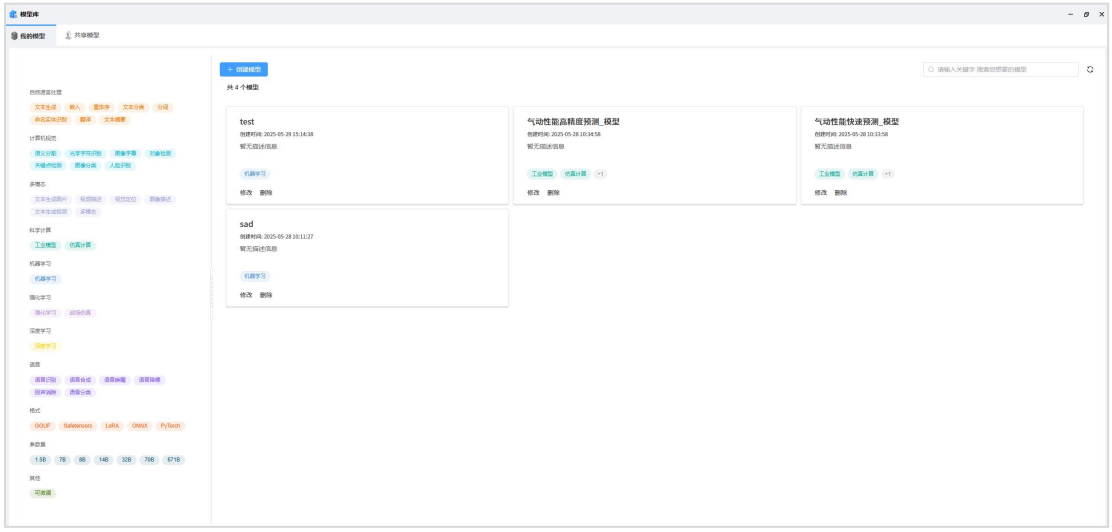
模型库，包含了平台中归档产生和创建模型产生的所有模型。通过模型库模块，可以实现模型版本管理、模型部署、模型转换和模型共享。模型库中包括两个页签：我的模型、共享模型。“我的模型”用于存放用户自己创建的模型，“共享模型”用于存放内置模型、其他用户共享的模型以及用户自己共享的模型。界面如下图所示：



模型库

2.5.1. 我的模型

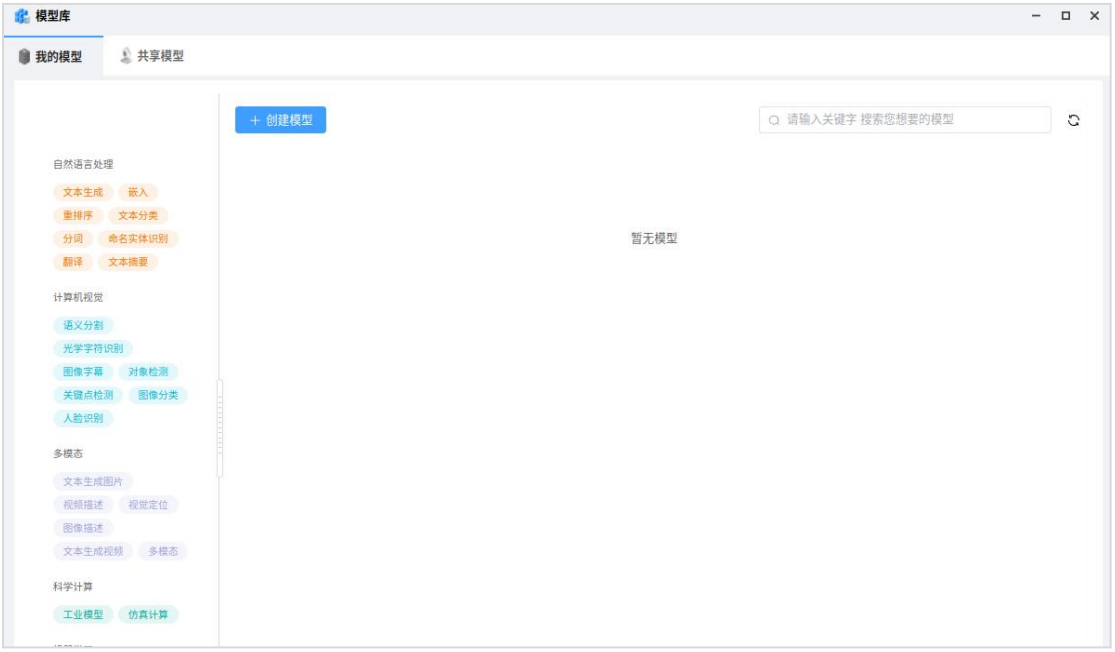
在“模型库”中点击“我的模型”按钮，可以展示我的模型页面，界面如下图所示：



我的模型

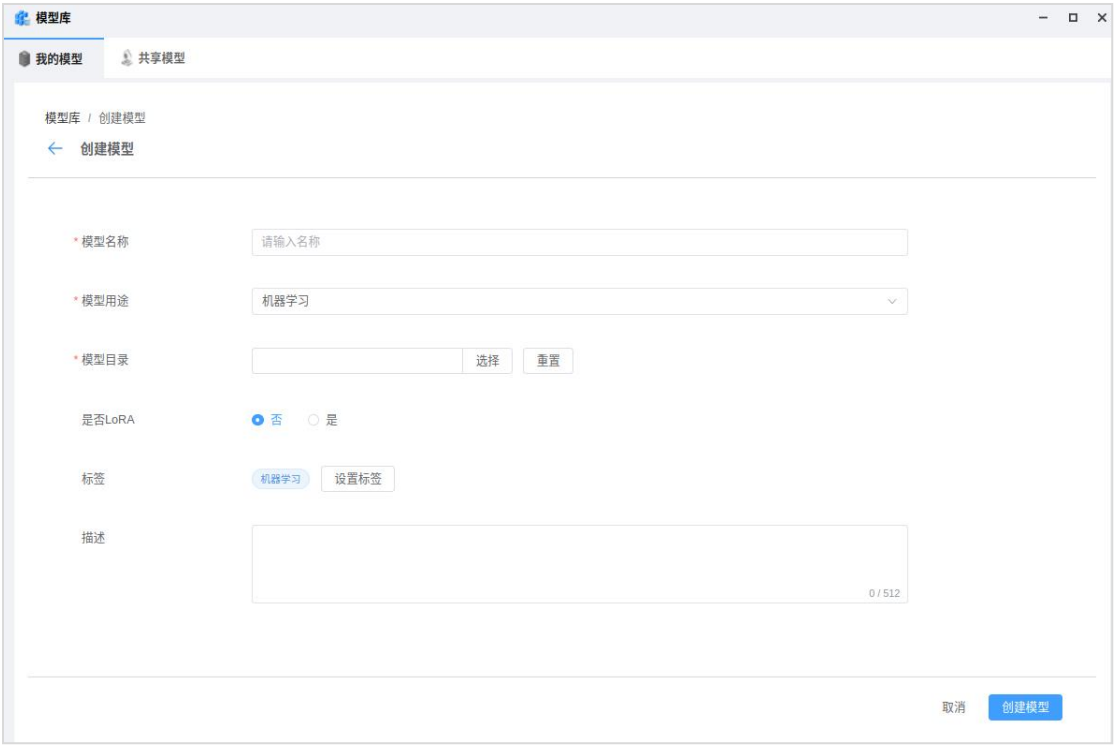
2.5.2. 创建模型

在我的模型左侧展示了所有可以设置的标签，右侧为我的模型列表区域。点击“创建模型”按钮，开始创建模型。如下图所示：



在模型库创建模型

点击“创建模型”后，跳转到“创建模型”页面，如下图所示：



新建模型页面

创建模型页面所有参数解释如下：

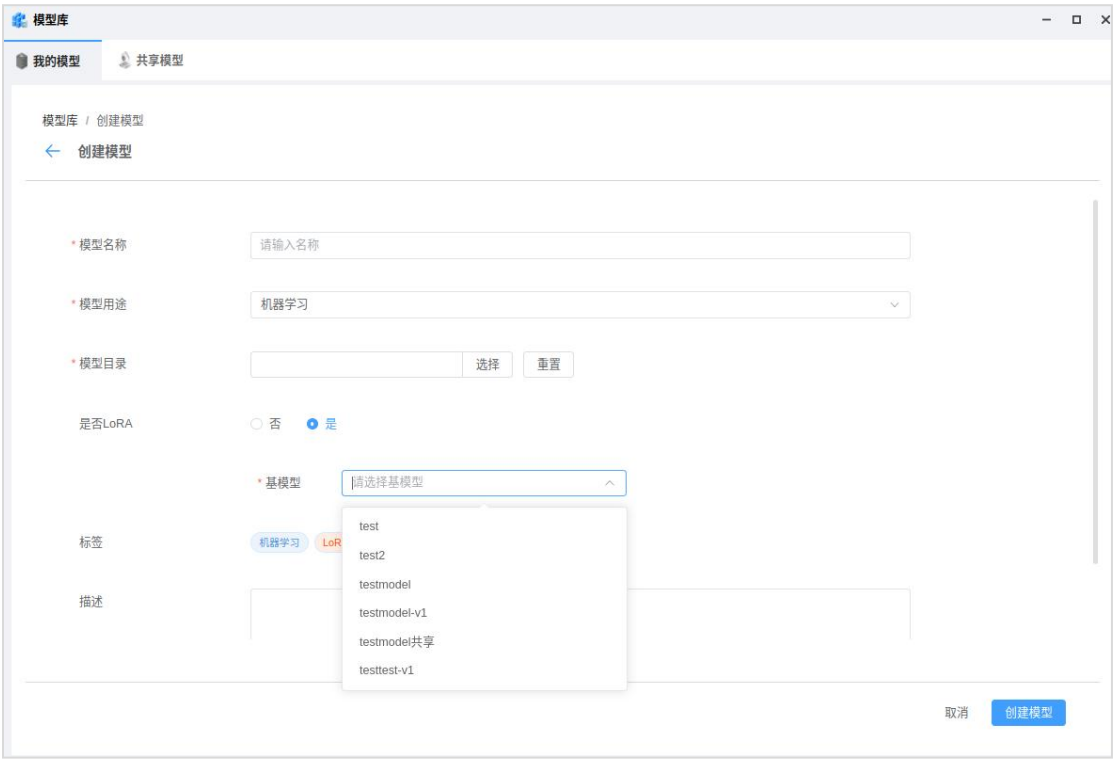
模型名称：用户自定义，但是不能与现有模型名称一致，必填项。

模型用途：选择模型用途用于为部署模型推荐合适的镜像。模型用途可选有：文本生成、文本生成图片、嵌入、重排序、多模态、机器学习、强化学习、深度学习和其他。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

模型目录：选择模型文件所在的目录，必填项。选中“重置”后清空选择内容。

是否 LoRA：选择模型是否是 LoRA 模型，如果选择“是”，默认会展示“基模型”参数，其选择框中会列出平台中所有可用的 LoRA 模型供选择。此时，“是否 LoRA”上方的“模型目录”需要选择 LoRA 文件目录。



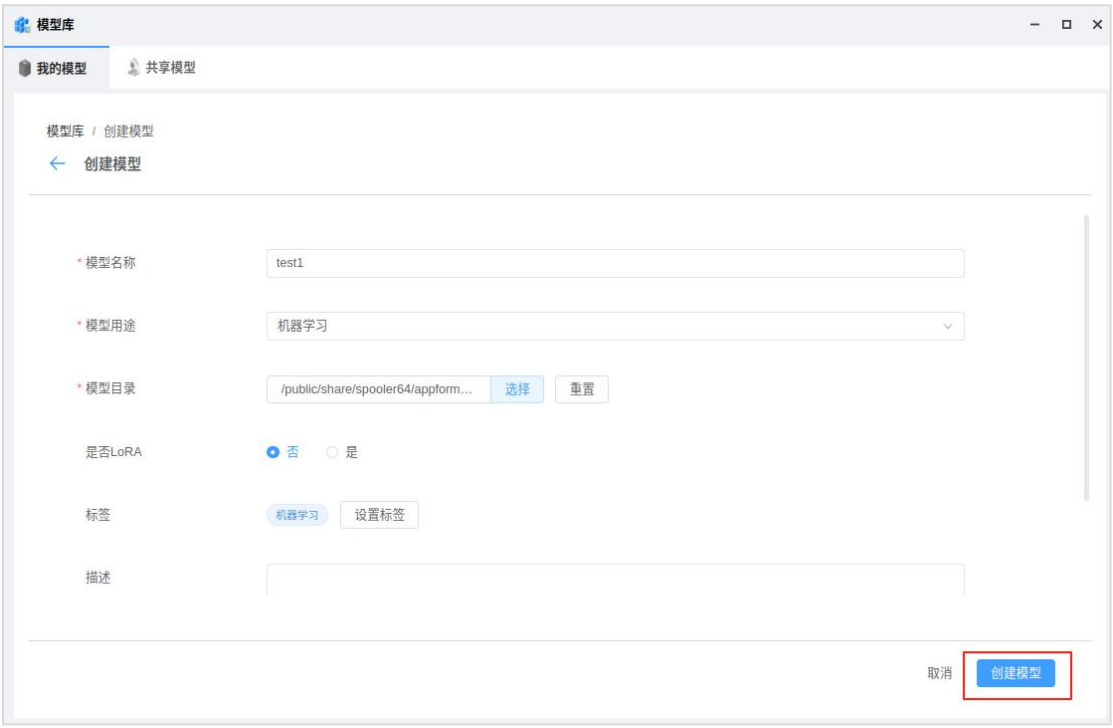
LoRA 选择为“是”的页面

标签：为模型设置标签。选择模型用途的过程中，会默认回填用途标签，也

可以选择内置的所有标签。

描述：用于描述模型结果的相关内容，选填项。

点击“创建模型”按钮后，成功创建模型，并会提示“新建模型成功”，如下图所示：



创建模型按钮

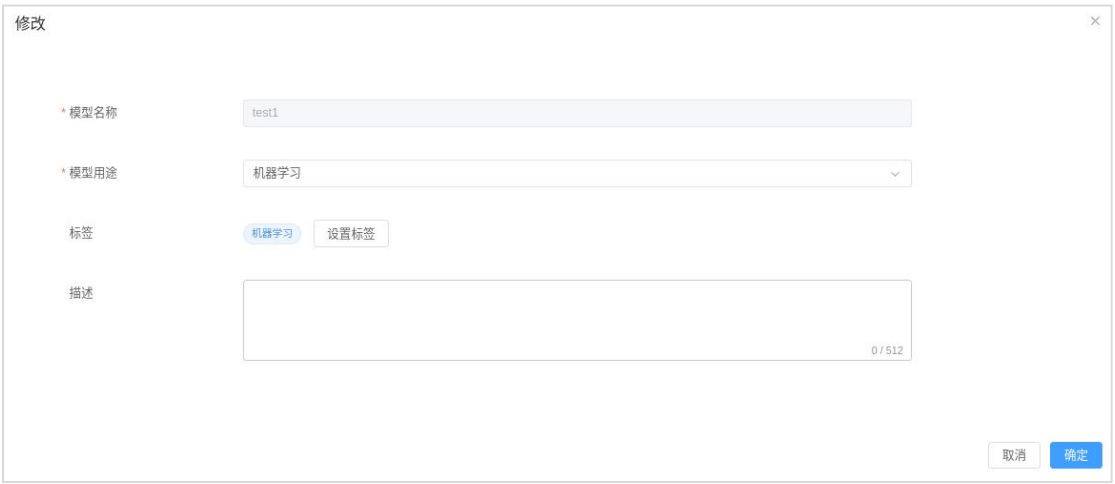
创建模型成功

2.5.3. 模型修改

模型卡片上的“修改”按钮，位于模型卡片的下方，如下图所示：

模型修改

该功能是对整个模型的修改。点击“修改”后，弹出修改模型页面，修改参数如页面所示：



模型修改页面

模型名称：创建模型时设置的名称，该参数不可修改；

模型用途：该参数与创建模型时可选参数一致，可修改；

标签：创建模型时设置的标签，可增加或者删除；

描述：与创建模型时设置的描述信息一致，可修改。

2.5.4. 模型删除

模型卡片上的“删除”按钮，位于模型卡片的下方，如下图所示：

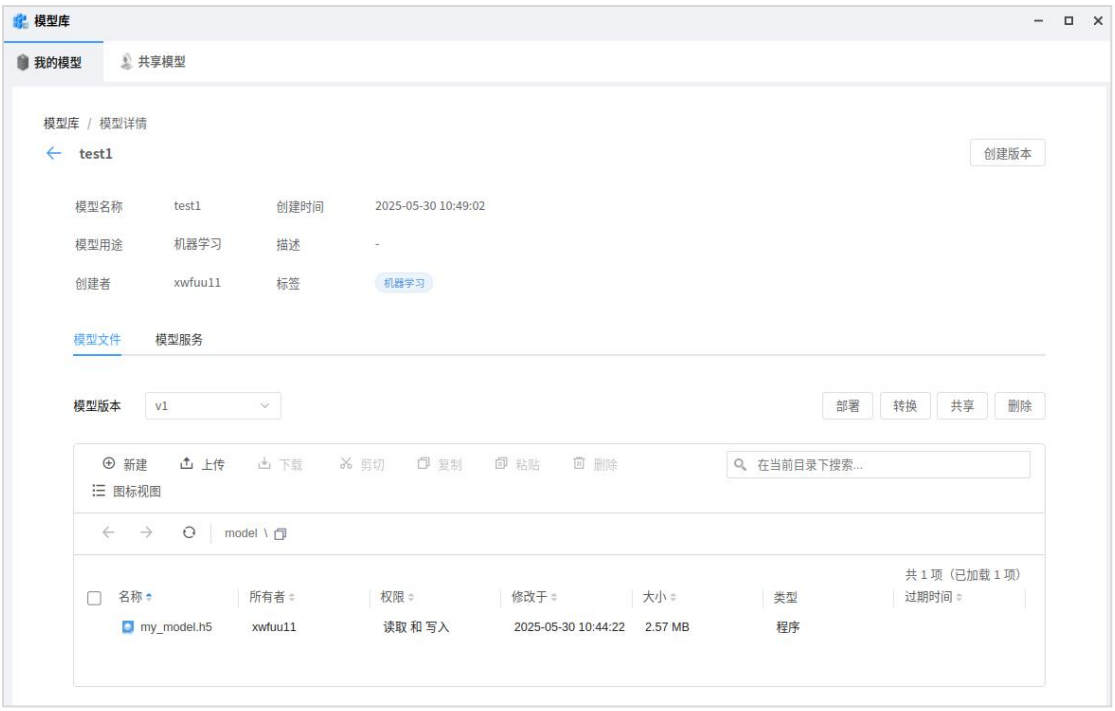
模型删除按钮

当不勾选“删除模型文件”，点击“确定”按钮后，仅删除此模型卡片；当勾选“删除模型文件”，点击“确定”按钮后，不但会删除该模型卡片，还会删除模型文件，请谨慎操作。

模型删除页面

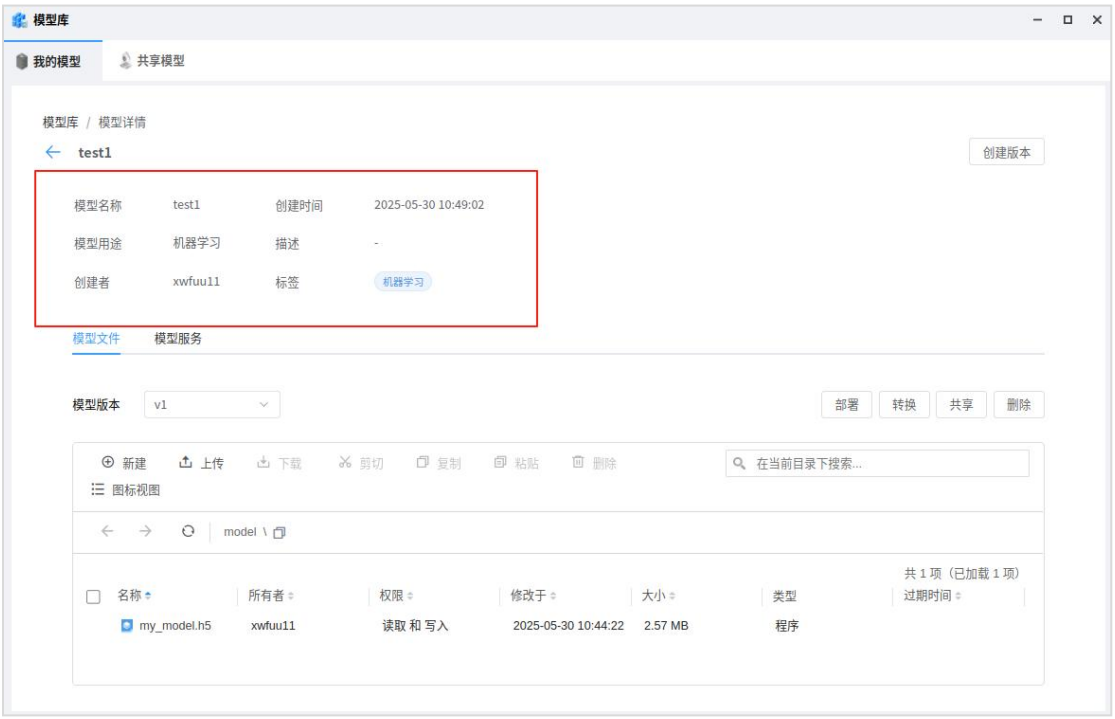
2.5.5. 模型详情

点击具体的模型卡片，如点击“test1”卡片，进入模型详情页面，如下图所示：



模型详情页面

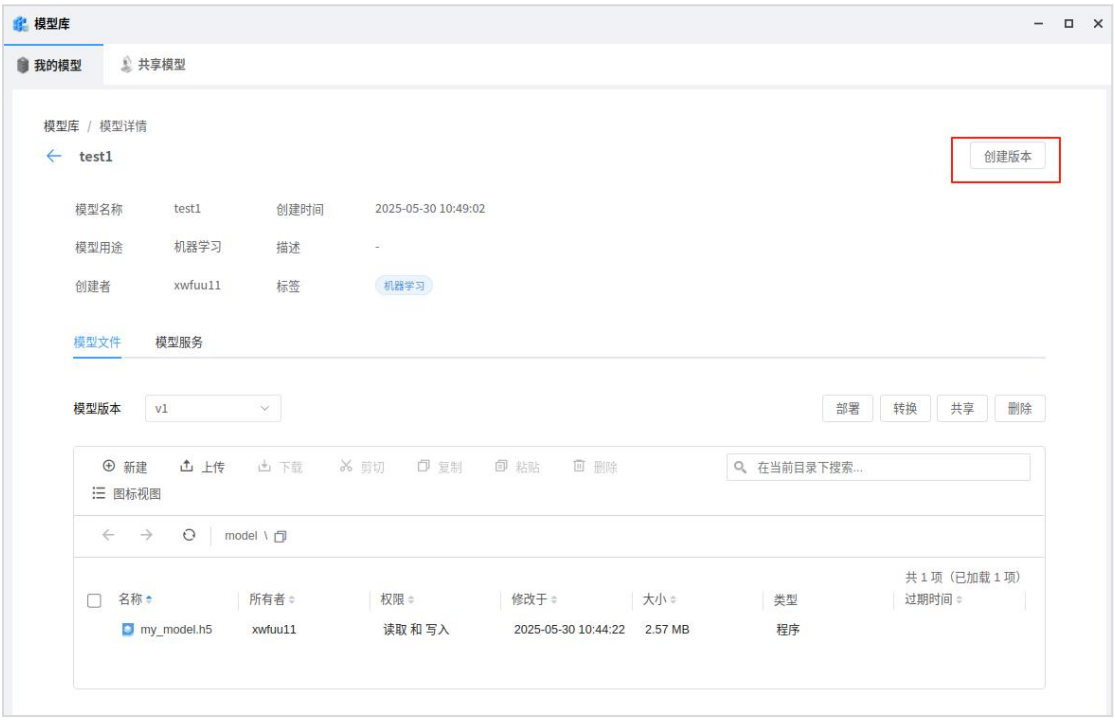
模型详情页面包含了创建模型时设置的所有参数，以及创建版本、模型文件和模型服务模块。



模型详情页面模型创建时设置的参数

2.5.6. 创建版本

进入模型详情页面后，点击页面的右上角“创建版本”，实现模型版本的增加。每次创建的版本都会在已有的版本上进行递增，如现有最高版本为 v2，新建后的模型版本将是 v3。另外，创建新的版本会自动继承模型的标签。



模型详情页面创建版本按钮

点击“创建版本”后，弹出创建版本页面，如下图所示：

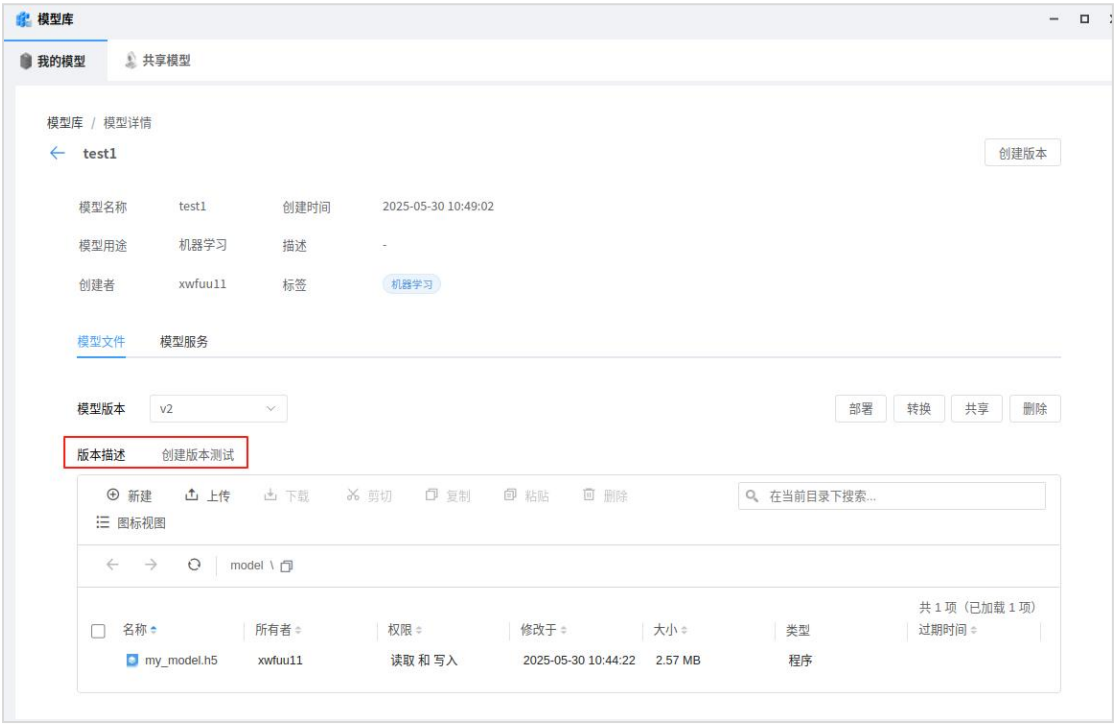
创建版本页面

创建版本页面参数如下：

模型目录：选择模型文件存储的路径。点击目录右侧的“重置”按钮，将置空模型目录；

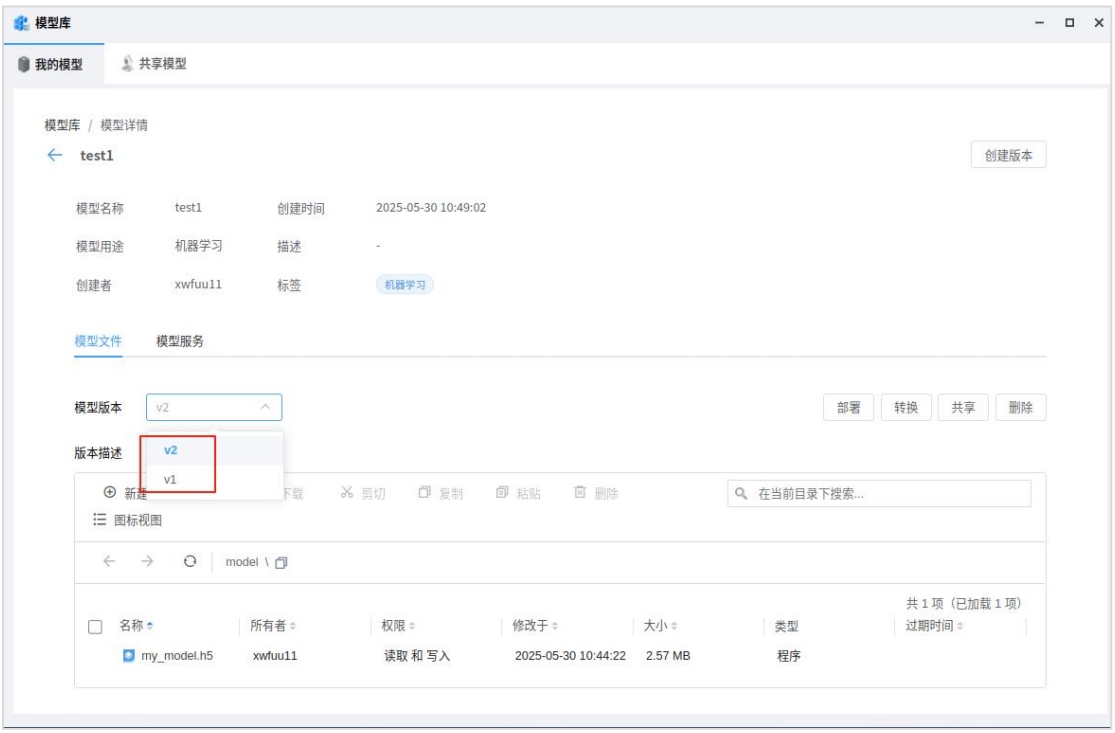
创建模型版本重置按钮

描述：添加对模型的描述，会在模型版本下展示该信息。如下图所示：



版本描述信息

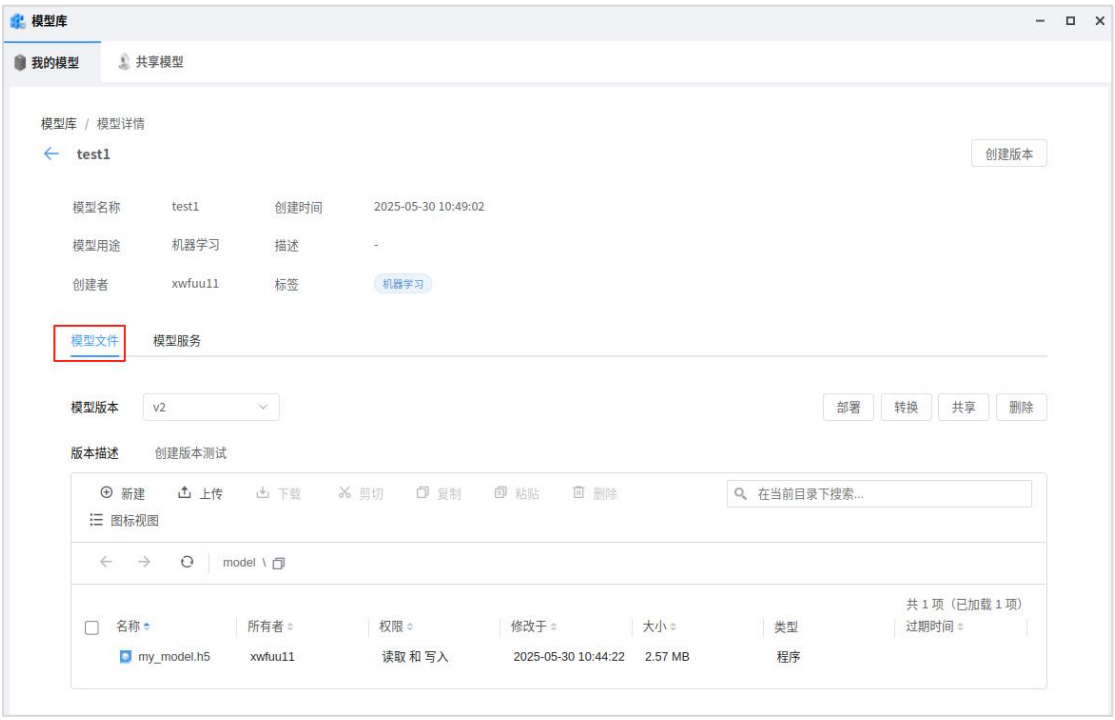
创建版本完成后，会在模型版本处增加一个版本，版本号会自动叠加。查看所有版本号，点击模型版本后的版本按钮即可，如下图所示：



模型版本展示

2.5.7. 模型文件

在模型详情页面的下侧有“模型文件”模块，如下图所示：

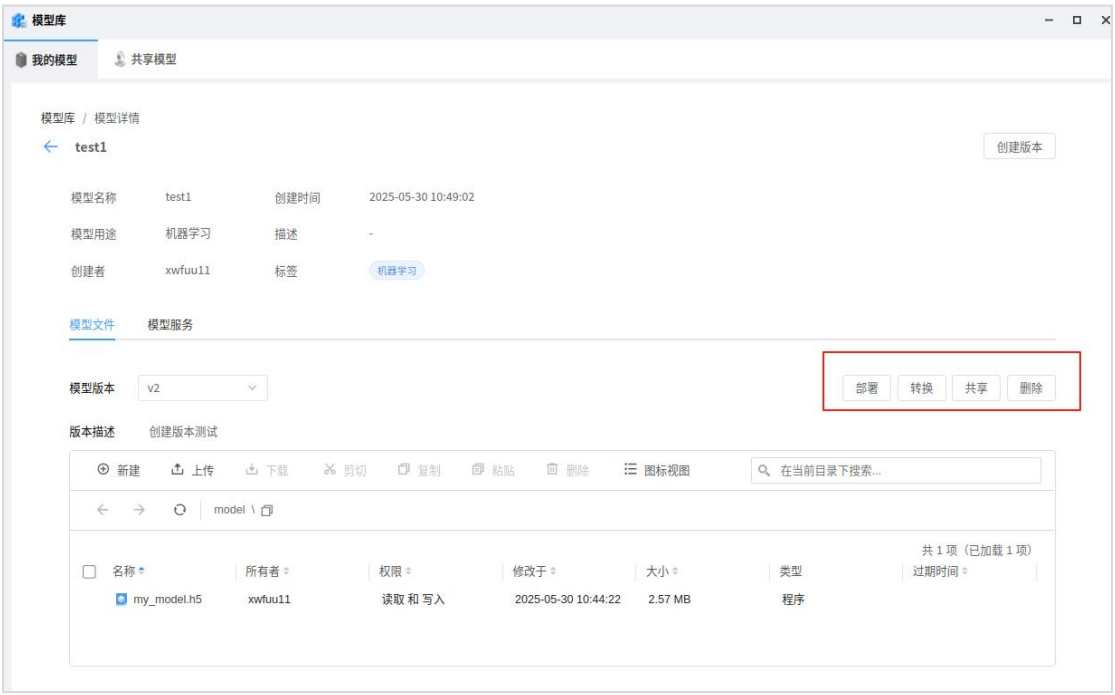


模型文件页面

模型文件页面模型文件默认展示最新版本模型的，内容随着选中的模型版本的改变而改变。模型文件目录支持的操作有新建、上传、下载、剪切、复制、粘贴、删除、图标视图、搜索等操作，如下图所示：

模型文件目录可操作内容

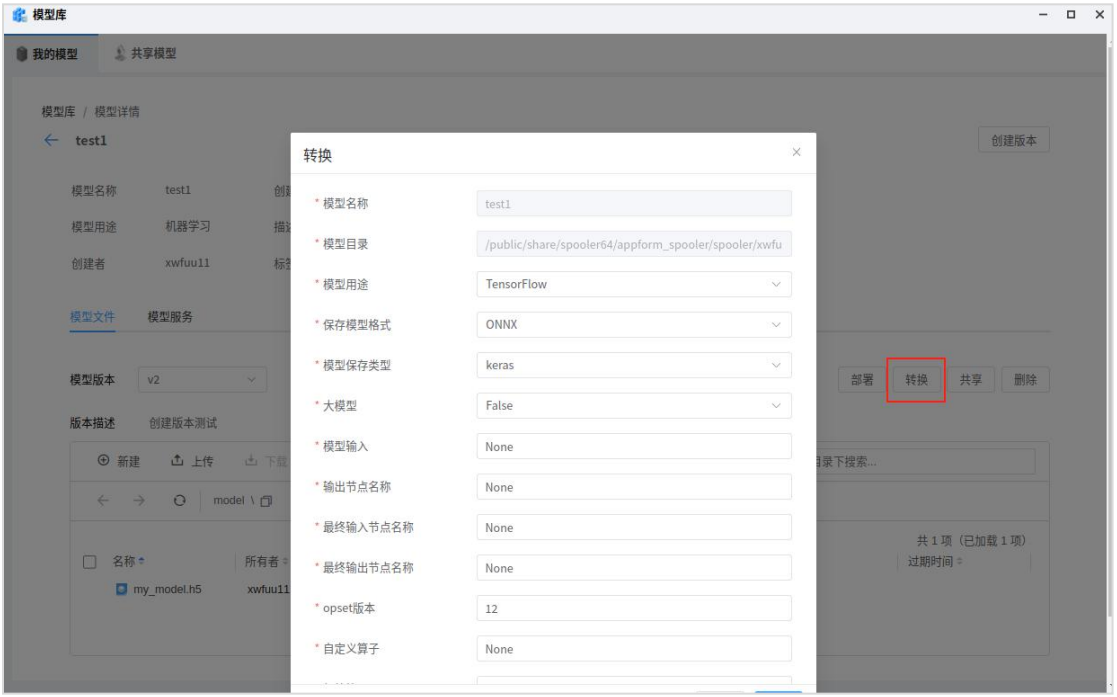
模型版本右侧有“部署”“转换”“共享”“删除”按钮，可以对模型进行部署、转换、共享、删除操作，如下图所示：



模型操作

2.5.7.1. 转换模型

转换模型将训练好的模型统一转换为 onnx 模型，以便在模型部署时选择 Triton Inference Server 类型的服务供外部程序调用实现推理功能。点击“转换”按钮，弹出“转换”页面，如下图所示：

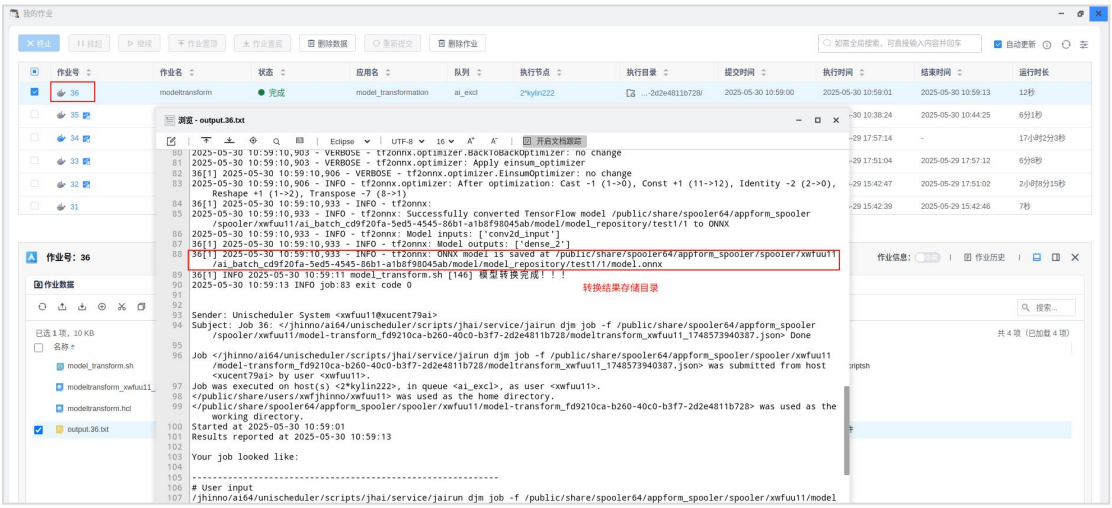


转换模型

选择模型类型、根据模型填写相关参数。点击确定按钮，提交转换模型的作业。

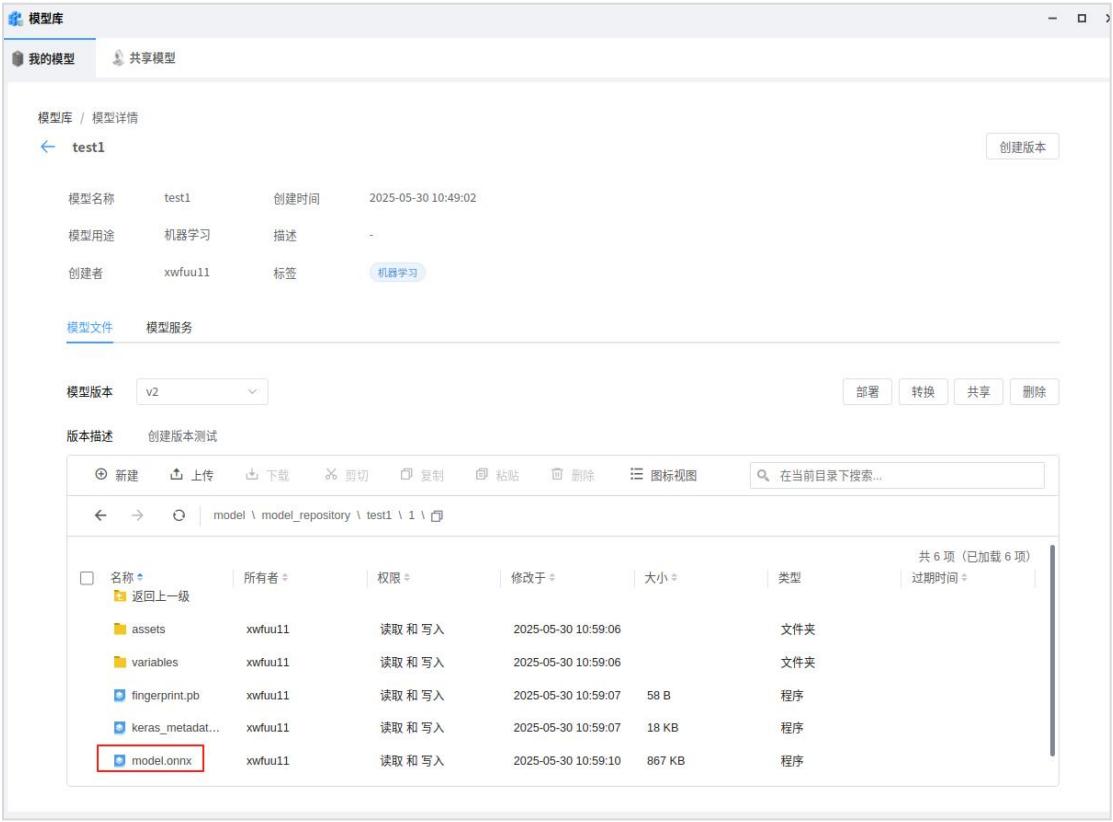
转换模型作业提交

在我的作业中可以看到提交的转换模型作业，如下图：



转换模型作业查看

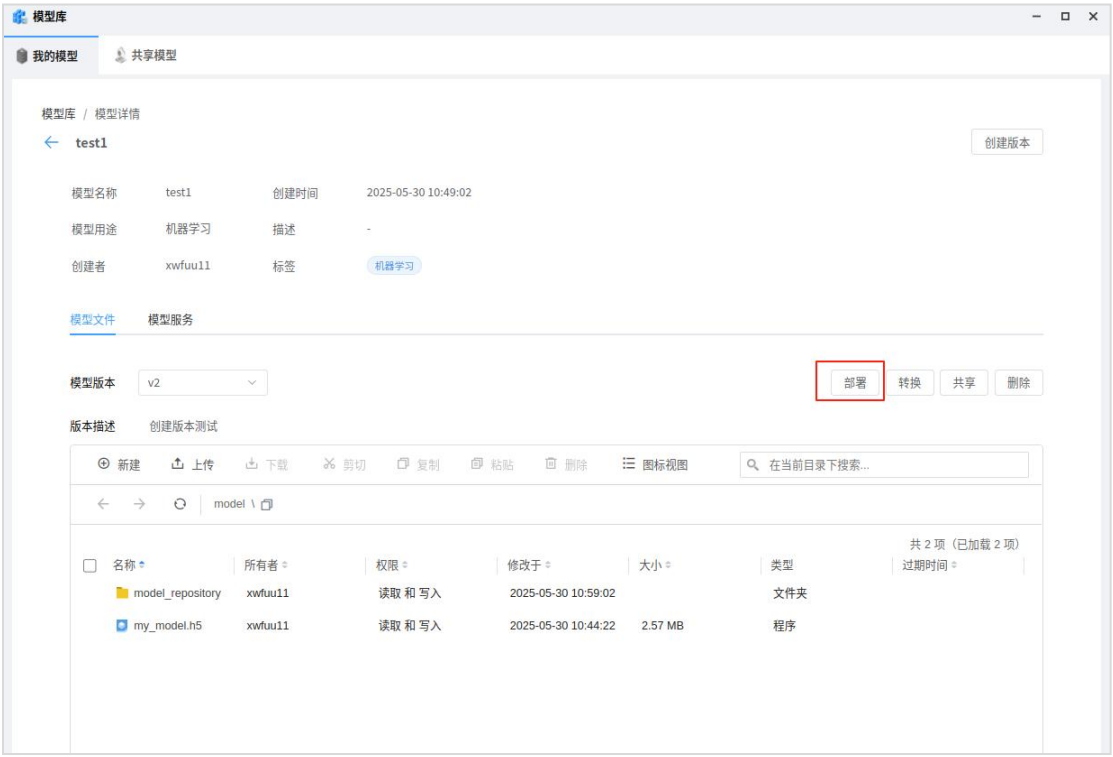
模型转换成功后，在模型结果中能够看到转换后的模型，如下图：



转换模型结果

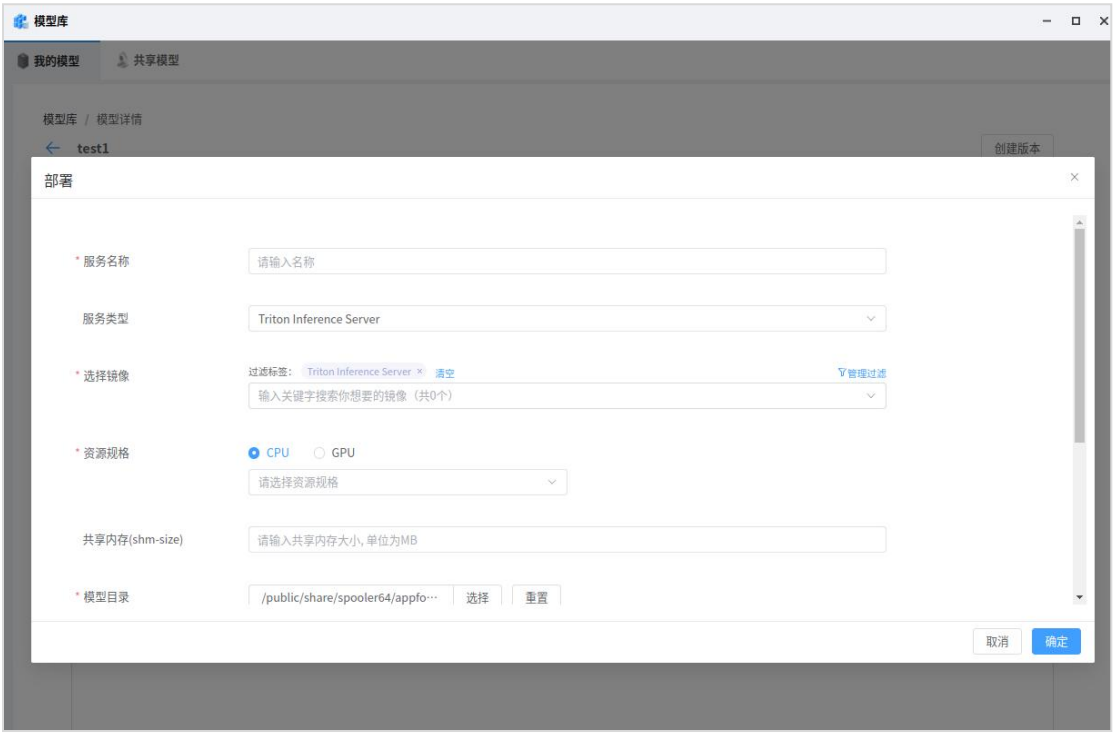
2.5.7.2. 部署模型

部署模型是将训练好的模型部署成推理服务，可以供外部程序调用实现推理功能。



从模型文件页面进行模型部署

点击“部署”按钮后，跳转至模型部署页面，系统会根据模型用途自动推荐“服务类型”。模型部署页面如下所示：



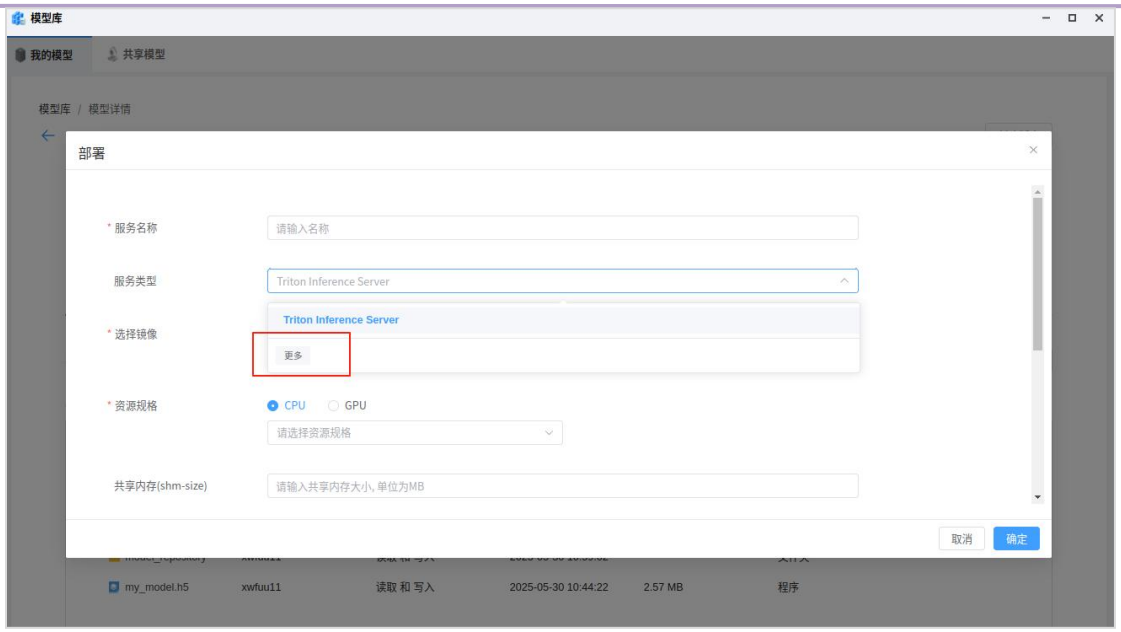
模型部署页面

模型用途和推荐的服务类型对应关系如下：

模型用途	推荐的部署服务框架
机器学习	Triton Inference Server
深度学习	TensorFlow Serving PyTorch Serving Triton Inference Server
强化学习	所有框架

模型用途与服务类型对应表

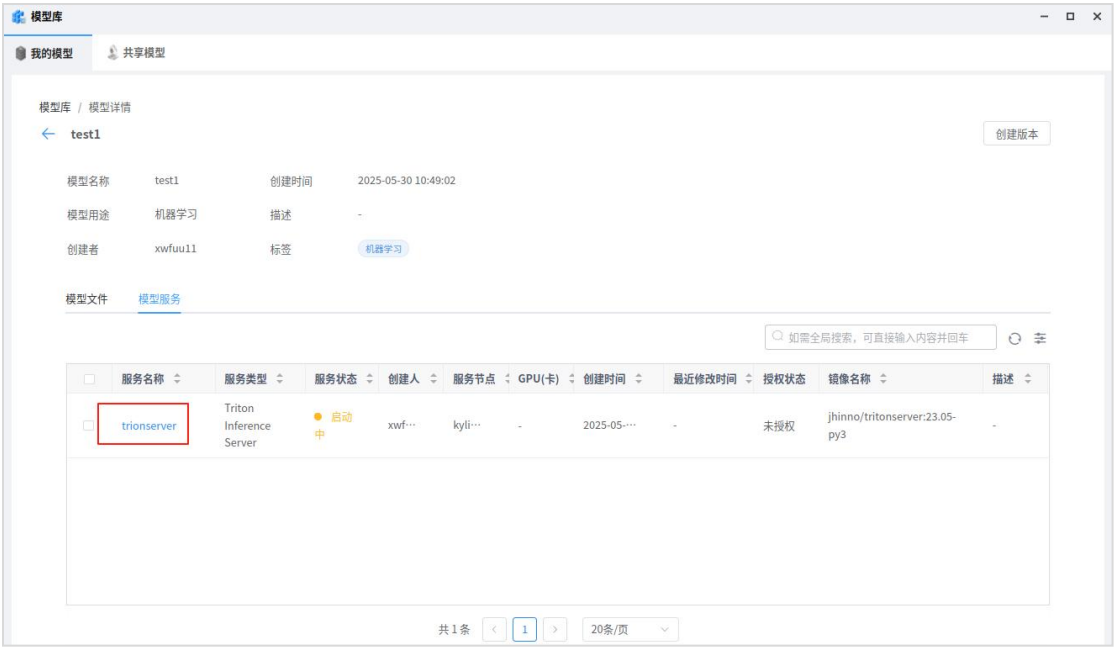
自动推荐服务除上述规则外，还会结合模型文件共同决定推荐的服务，例如：模型文件下是.h5 后缀的文件，此时点击“部署”按钮，服务类型默认推荐的是“TensorFlow Serving”。推荐服务仅作为参考建议，旨在简化用户操作流程。若系统推荐的服务不符合您的使用场景，您可以点击“更多”按钮选择其他服务选项。



服务类型更多按钮

当确定服务类型后，页面服务参数会随着服务类型的改变而改变，“模型目录”会自动回填，也可以点击“选择”按钮重新选择目录。页面的其他的部署参数详见“我的服务”模块。

模型部署完成后，部署的服务可以在模型服务找到。如下图所示：



模型服务列表

2.5.7.3. 删除

点击“删除”按钮，弹出删除对话框，当不勾选“删除模型文件”，点击“确定”按钮后，仅删除此模型卡片；当勾选“删除模型文件”，点击“确定”按钮后，不但会删除该模型卡片，还会删除模型文件，请谨慎操作。如下图所示：

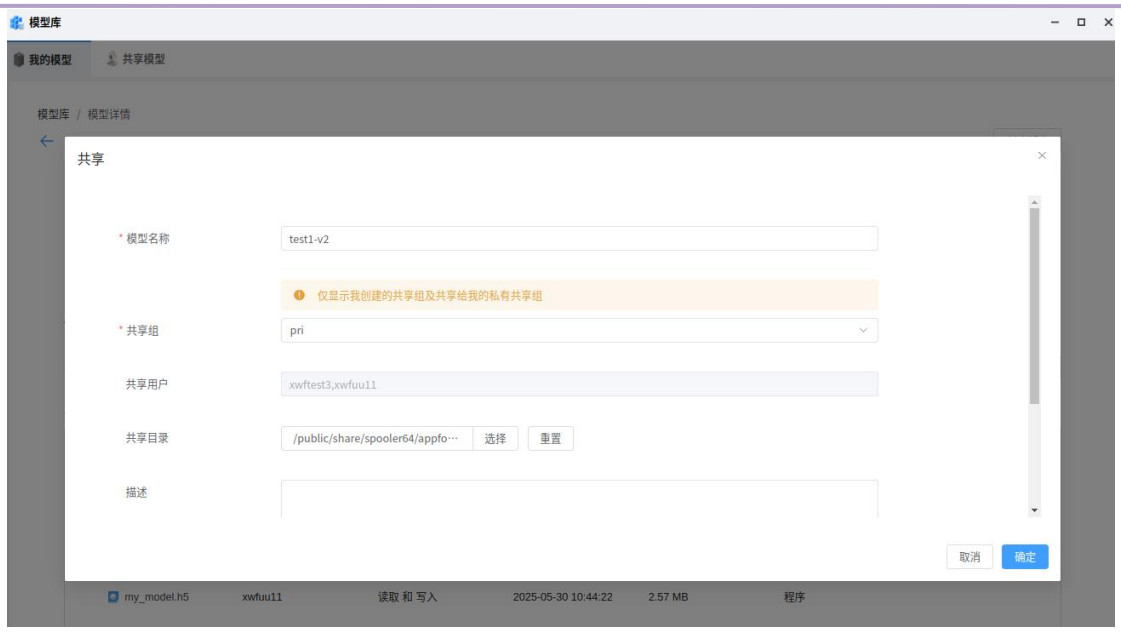
删除模型

2.5.7.4. 共享模型

共享模型是将模型共享给指定共享组，共享组成员可以在自己的共享模型内查看到的模型。

注意：添加共享模型之前需要先创建共享组。

点击“共享”按钮，弹出共享模型页面，如下图所示：



共享模型页面

图中每个参数的具体含义如下：

模型名称：共享后的模型名称，输入任意符合命名规则的名称即可，可修改。

共享组：选择可见的共享组，选项从我的数据->共享数据区获取。

共享用户：选择共享组之后显示，显示为选择的共享组成员。

共享目录：选择共享组之后显示，只能在所选共享组的共享数据区选择文件夹，点击“选择”按钮，然后在弹出的“预览”窗口中选择文件夹即可，用作存放源数据文件目录，只允许选择一个文件夹。

描述：共享后的数据集描述信息。

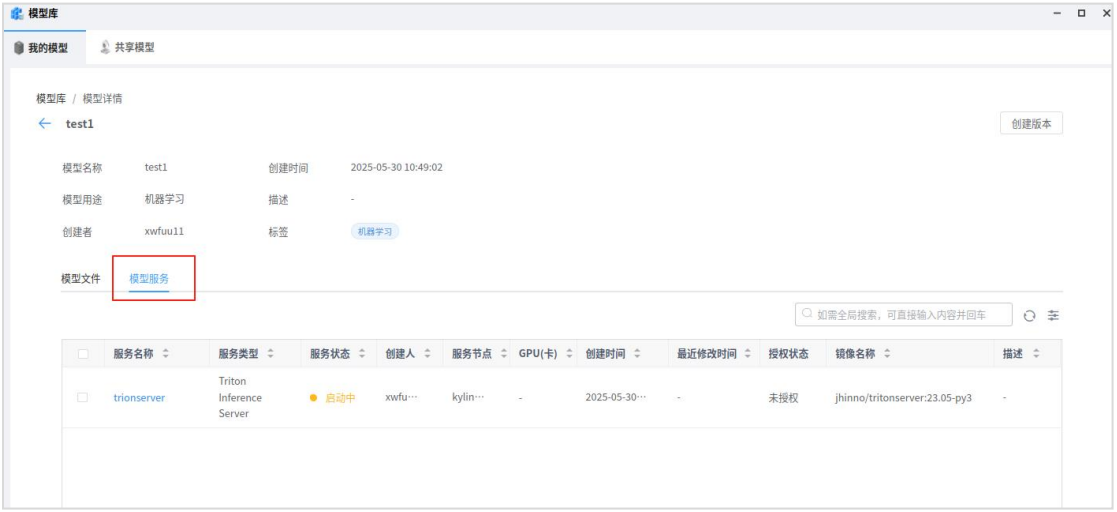
模型后台拷贝：勾选后采用后台形式拷贝源模型文件至共享目录，当前共享页面退出；不勾选则采用前台拷贝方式，当前共享页面不退出。

在共享窗口中填写完成后，点击“确定”按钮，可以保存此次共享的模型。并且在共享模型页面看到，如下图所示：

共享模型列表

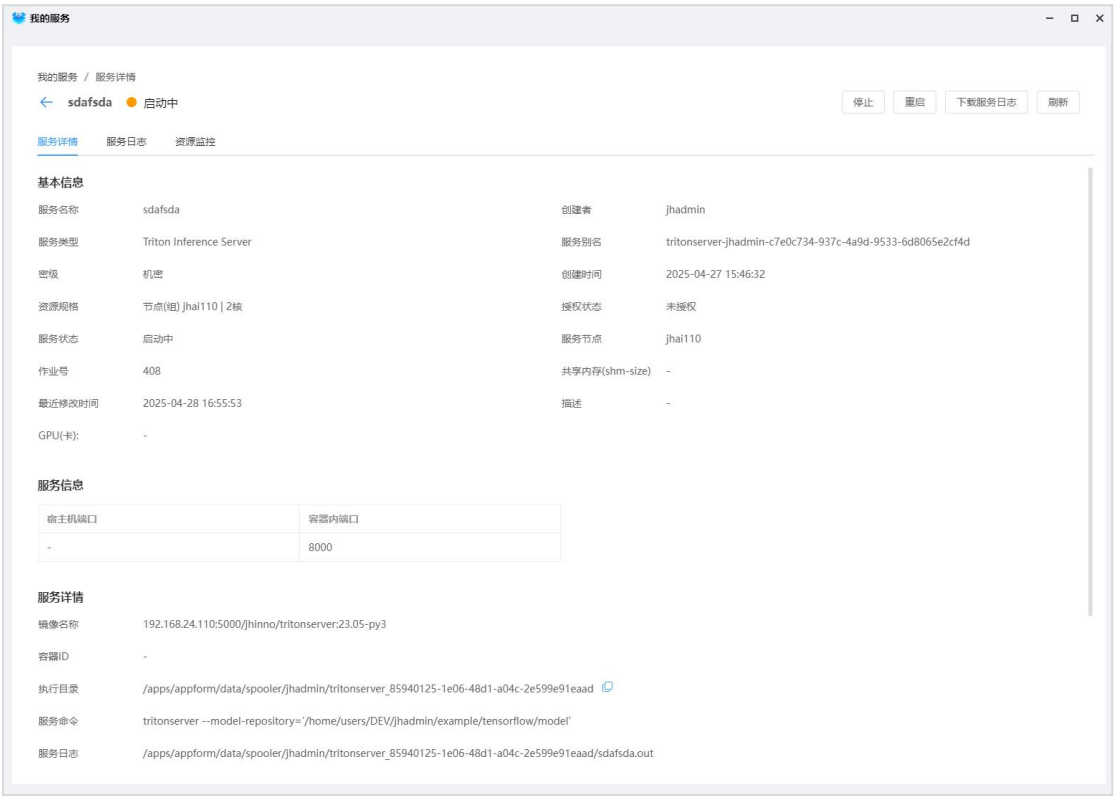
2.5.8. 模型服务

模型服务位于模型文件的右侧，包含了在模型库部署该模型的所有服务，如下图所示：



模型服务列表

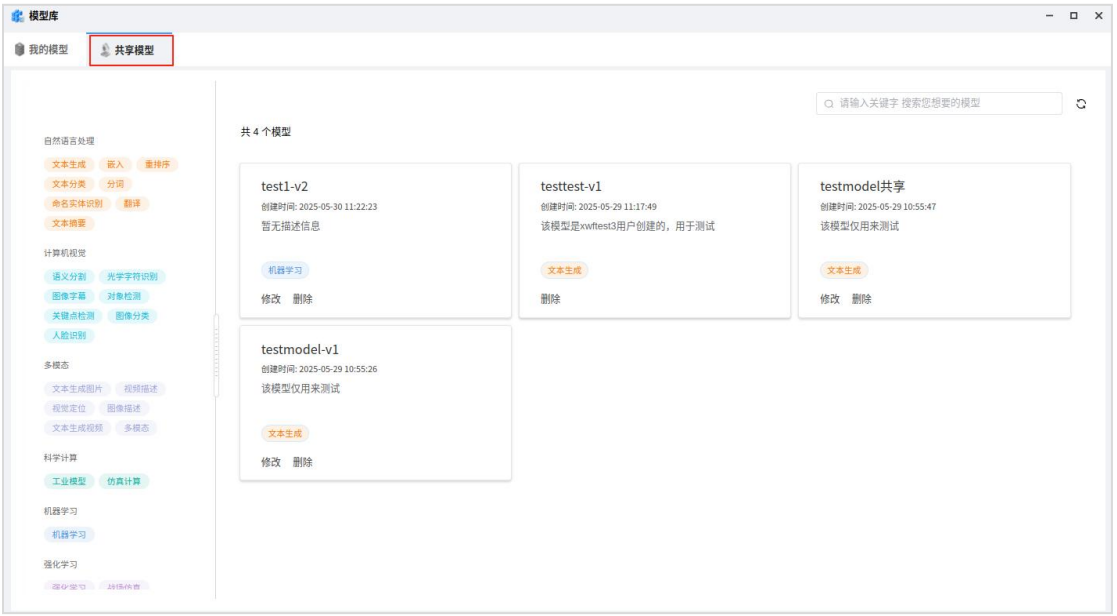
点击具体服务名称，会弹出该服务的详情页面。



服务详情页面

2.5.9. 共享模型

在“模型库”中点击“共享模型”按钮，可以展示共享模型页面，界面如下图所示：



共享模型

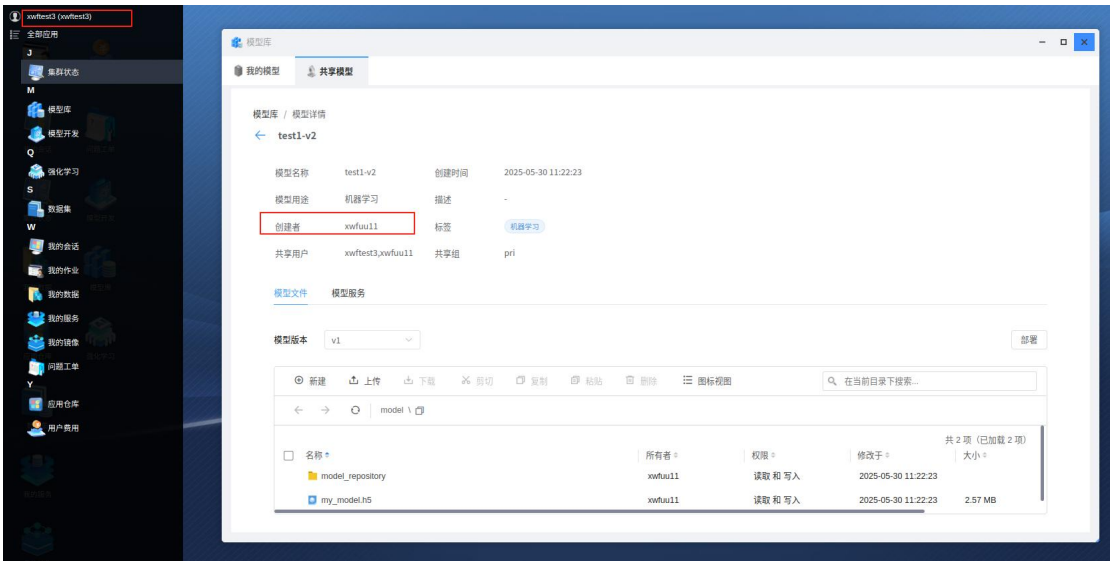
共享模型下存放用户自己共享的模型以及其他用户共享的模型，卡片信息默认展示模型名称、创建时间、描述、标签信息，部分卡片还会展示“修改”“删除”按钮。

用户自己共享的模型：模型卡片上显示修改、删除按钮，用户可以继续修改和删除模型。

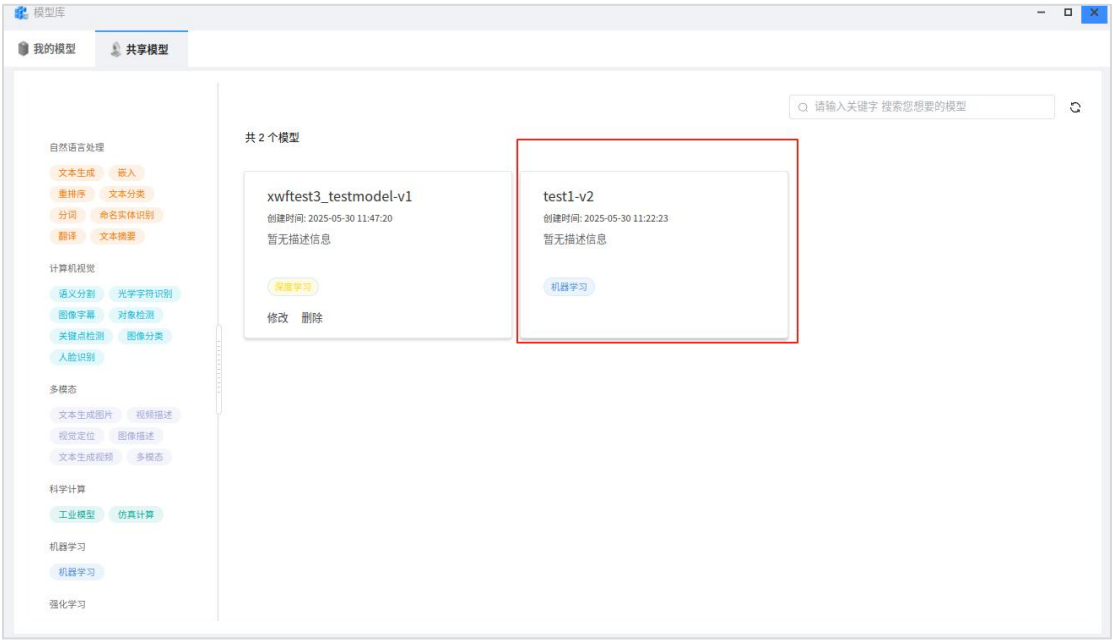
用户自己共享的模型

用户自己共享的模型显示修改、删除按钮

其他用户共享的模型：其他用户共享给我的模型，模型卡片上不显示修改、删除按钮。但管理员可以删除其他用户共享给自己的模型。



其他用户共享的模型

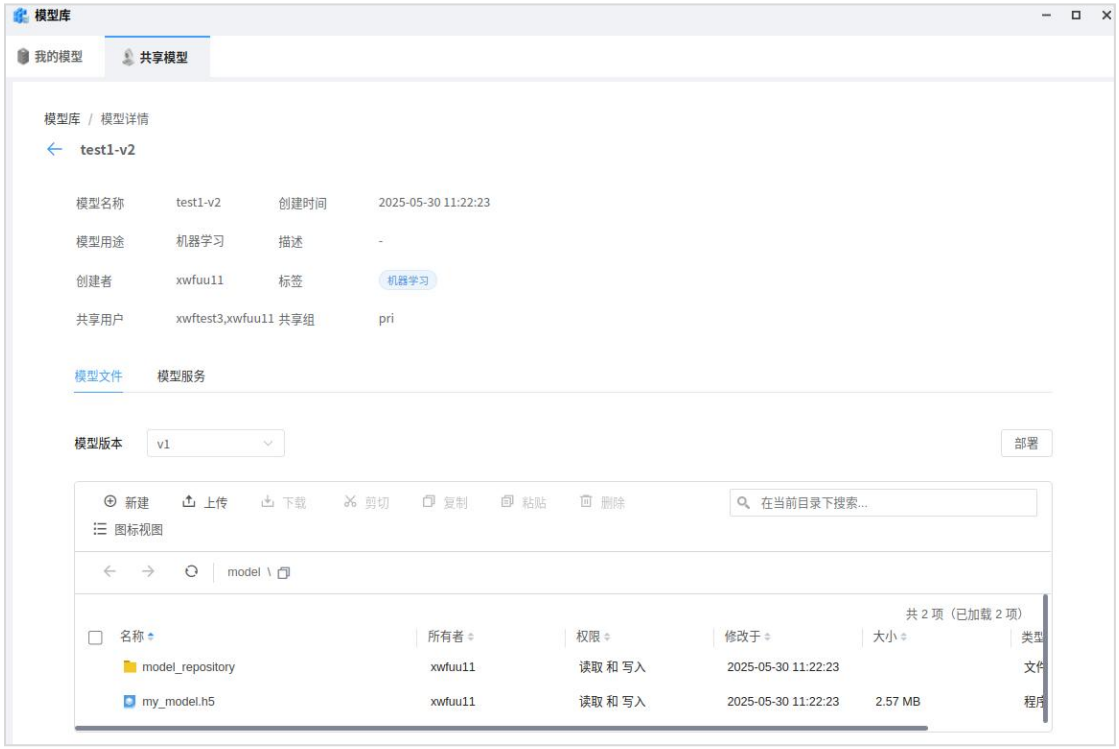


普通用户查看其他用户共享的模型无修改、删除按钮

管理员查看其他用户共享给自己的模型有删除按钮

模型卡片详情

点击具体的模型卡片，如点击“test1-v2”卡片，进入模型详情页面，模型详情页面包含了共享模型的所有信息参数，以及创建版本、模型文件和模型服务模块，如下图所示：



模型详情页面

部署模型：

共享模型部署行为和“我的模型”中一致，详见“我的模型”-“模型文件”-“部署模型”章节。

2.6. 我的服务

我的服务包含创建服务和服务管理功能，支持“Tensorboard”“Tensorflow Serving”“Pytorch Serving”“Triton Inference Server”“VisualDL”“MindInsight”“Docker Compose”“自定义服务”类型的服务。如下所示：

- **Tensorboard**：用于可视化查看模型数据。
- **Tensorflow Serving**：将训练的 Tensorflow 模型发布为 Web 服务。
- **Pytorch Serving**：将训练的 Pytorch 模型发布为 Web 服务。
- **Triton Inference Server**：将 onnx 模型发布为 Web 服务。
- **VisualDL**：可视化展示 PaddlePaddle 训练过程数据。
- **MindInsight**：可视化展示 MindSpore 训练过程数据。
- **Docker Compose**：用于定义和运行多容器应用程序。
- **自定义服务**：通过可配置参数自定义服务。

各个角色对服务实例的操作权限：

- **添加者**：对服务具有新建、启动、停止、重启、修改、授权、删除、使用的权限；
- **管理员**：对其他人添加的服务实例仅有启动、停止、重启的权限。

我的服务主页面，如下图所示：

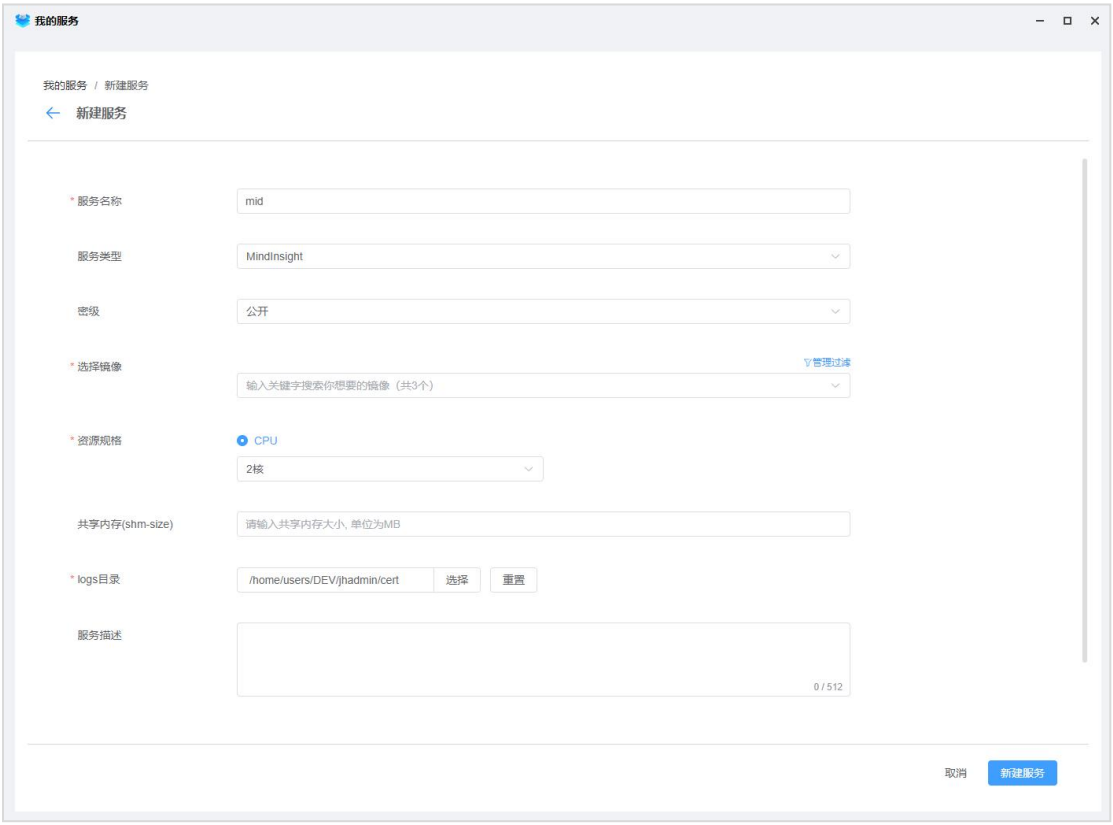
我的服务主界面

2.6.1. 新建服务

2.6.1.1. “MindInsight”类型的服务

点击“新建服务”按钮，弹出“新建服务”窗口，选择 MindInsight 服务类型并填写相关参数，如下图所示：

新建服务功能入口



添加“MindInsight”类型的服务配置

图中每个参数的具体含义如下：

服务名称：用户自定义。

服务类型：选择“MindInsight”。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

镜像名称：选择 mindinsight 镜像。

资源规格：选择启动服务所需要的资源配置，默认只能选择 CPU 资源。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点 /etc/docker/daemon.json 中配置 default-shm-size 参数大小。

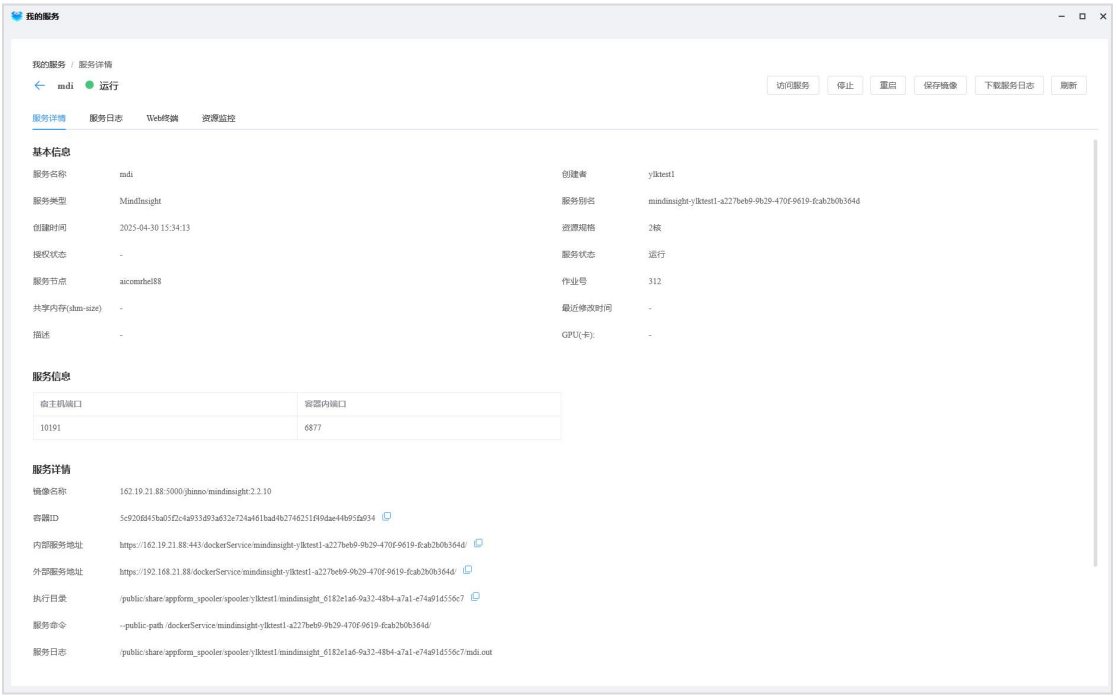
Logs 地址：保存的 Log 文件路径。

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务成功。

创建完成的 Mindinsight 服务

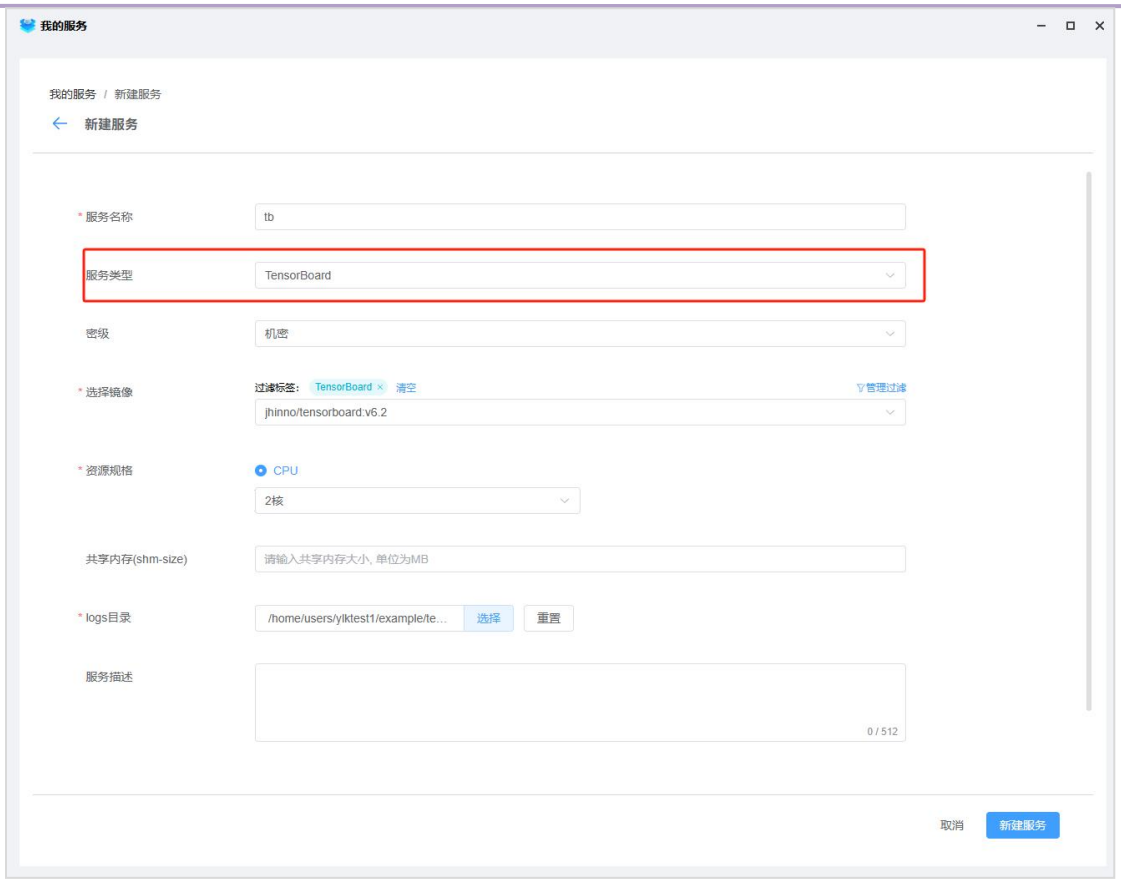
点击创建好的服务名称，如创建的“mdi”，会跳转到服务详情页面，如下图所示：



MindInsight 服务详情

2.6.1.2. “Tensorboard”类型的服务

点击“新建服务”按钮，弹出“创建服务”窗口，选择 Tensorboard 服务类型，并填写相关参数，如下图所示：



添加“Tensorboard”类型的服务配置

图中每个参数的具体含义如下：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“Tensorboard”。

镜像名称：选择 Tensorboard 镜像。

资源规格：选择启动服务所需要的资源配置，默认只能选择 CPU 资源。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点 /etc/docker/daemon.json 中配置 default-shm-size 参数大小。

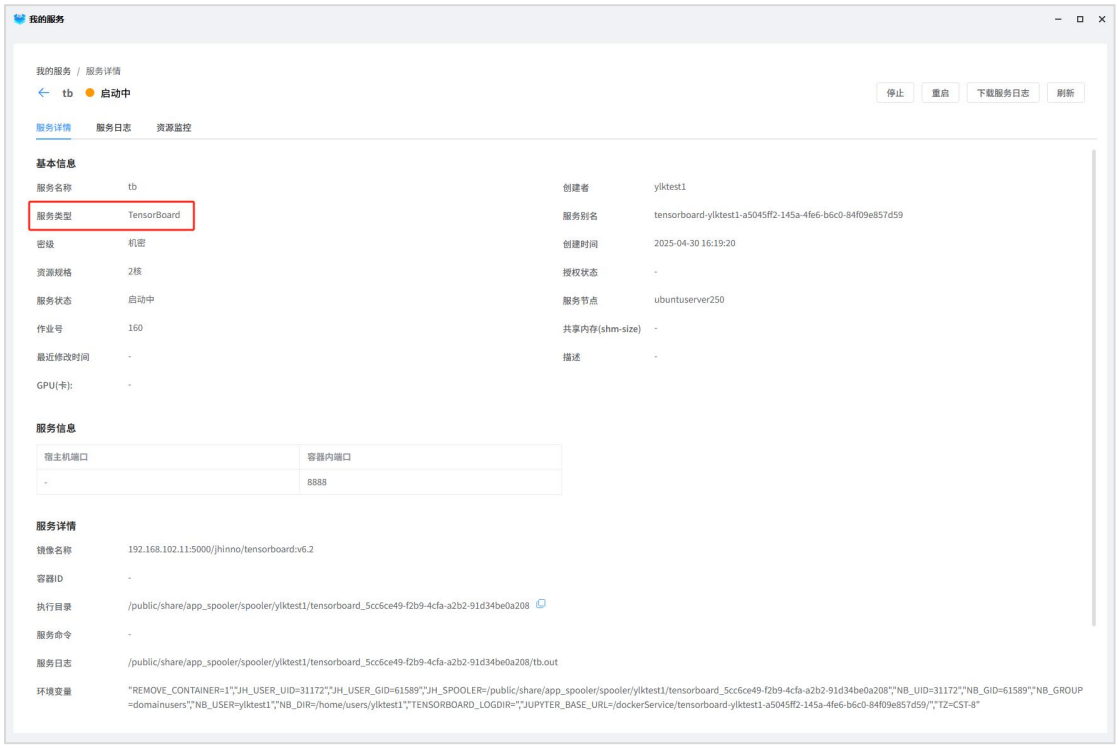
Logs 地址：保存 Log 日志的路径。

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务成功。

创建完成的 tensorboard 服务

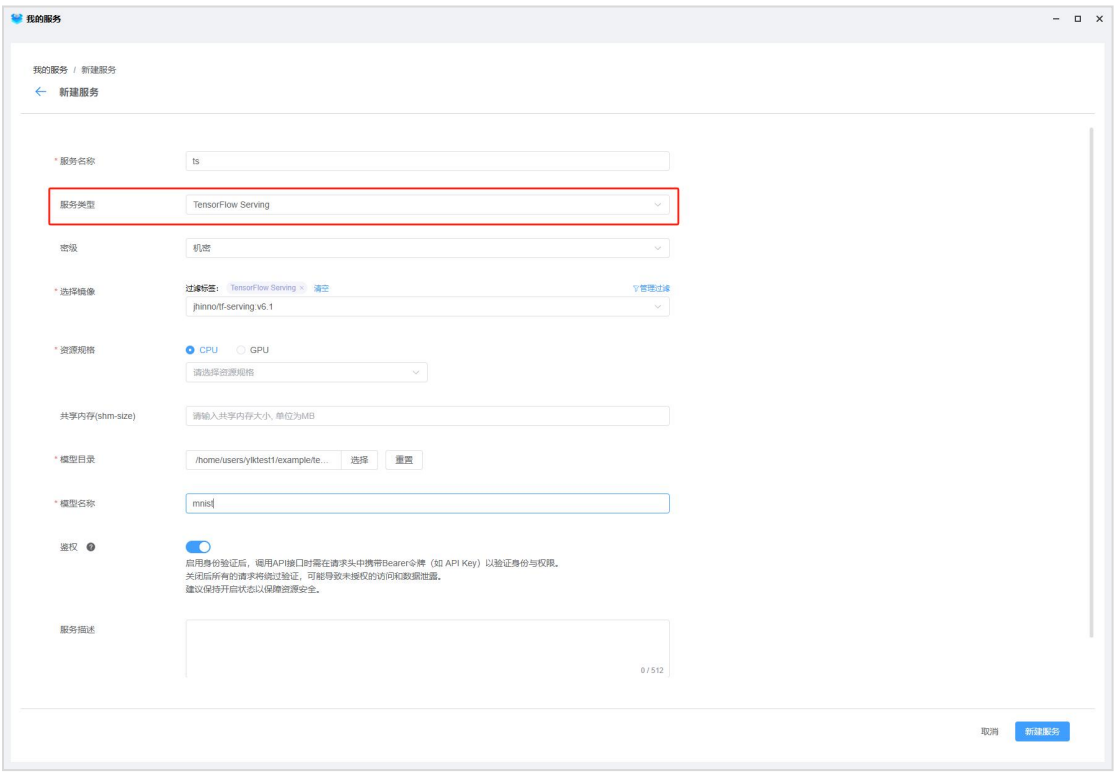
点击创建好的服务名称，如创建的“tb”，会跳转到服务详情页面，如下图所示：



Tensorboard 服务详情页面

2.6.1.3. “Tensorflow Serving”类型的服务

点击“新建服务”按钮，弹出“新建服务”窗口，选择 Tensorflow Serving 服务类型和填写相关参数，如下图所示：



“Tensorflow Serving”服务

图中每个参数的具体含义如下：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“Tensorflow Serving”。

镜像名称：选择 tf serving 镜像。

资源规格：选择启动服务所需要的资源配置。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点/etc/docker/daemon.json 中配置 default-shm-size 参数大小。

模型名称：设置启动服务时，使用的模型名称。

模型目录：选择模型文件所在的目录。

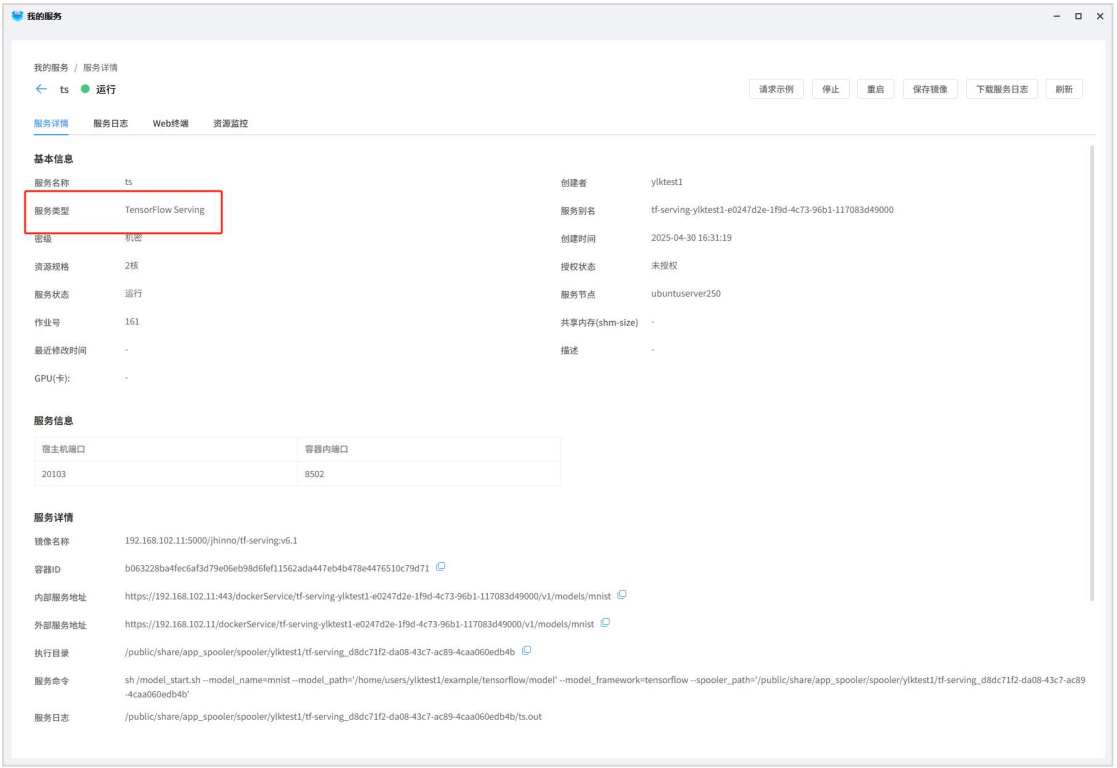
鉴权：选择是否启动 Bearer 令牌（如 API Key）确认身份与权限，开启鉴权后，在调用服务的时候，需要添加 API Key。

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务成功。

创建完成的 tensorflow serving 服务

点击创建好的服务名称，如创建的“ts”，会跳转到服务详情页面，如下图所示：



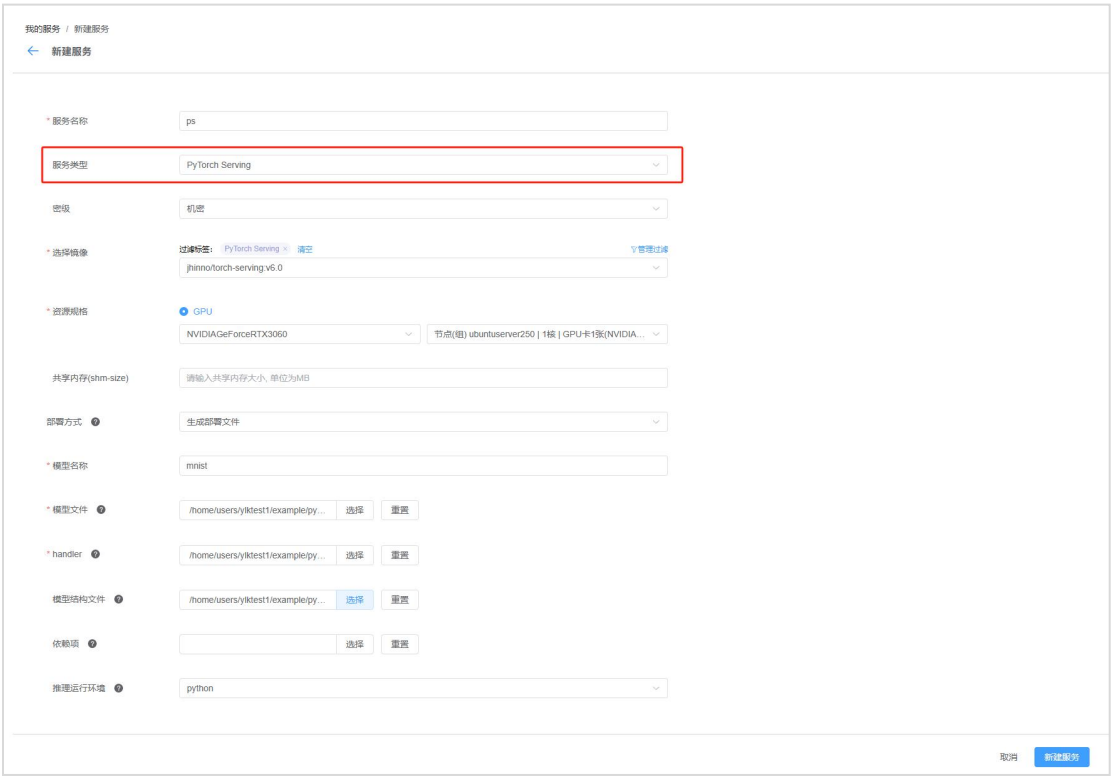
Tensorflow serving 服务详情页面

在服务详情页面，与普通的服务不同的是，包含了请求示例，也就是如何调用服务的 url。由于开启了鉴权，在调用示例中包含了${YOUR_API_KEY}，使用示例的时候，使用 API Key 替换${YOUR_API_KEY}就可以正常调用服务。tensorflow serving 的请求示例内容如下：

Tensorflow serving 请求示例

2.6.1.4. “Pytorch Serving”类型的服务

点击“新建服务”按钮，弹出“新建服务”窗口，选择 Pytorch Serving 服务类型和填写相关参数，如下图所示：



添加“Pytorch Serving”类型的服务配置

图中每个参数的具体含义如下：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“Pytorch Serving”。

镜像名称：选择 torch serving 镜像。

资源规格：选择启动服务所需要的资源配置，默认只能选择 GPU 资源。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点 /etc/docker/daemon.json 中配置 default-shm-size 参数大小。

部署方式：可以选择模型文件部署，也可以使用直接可以部署的 mar 文件进行部署。当选择“可部署文件”时，参数仅有“mar 文件”和配置文件，选择“生成部署文件”时，参数除“mar 文件”之外。

模型名称：用户自定义。

模型文件：选择*.pt 模型文件。

handler：选择*.py 文件。处理器文件，选择输入输出处理脚本，一般为 Python 脚本，例如 mnist_handler.py。

模型结构文件：选择*.py 模型结构文件，模型结构文件，选择包含模型结构的 Python 脚本，例如 model.py。

依赖项：选择依赖项。选择加载模型所需的标签等文件，文件类型不限，可以多选，例如 index.json。

推理运行环境：使用哪种语言进行推理，选择 python 或者 python3。

yaml 文件：包含模型配置的 yaml 文件。

配置文件：config.properties 文件中除端口之外的参数输入，多个参数使用。

mar 文件：mar 文件所在路径。

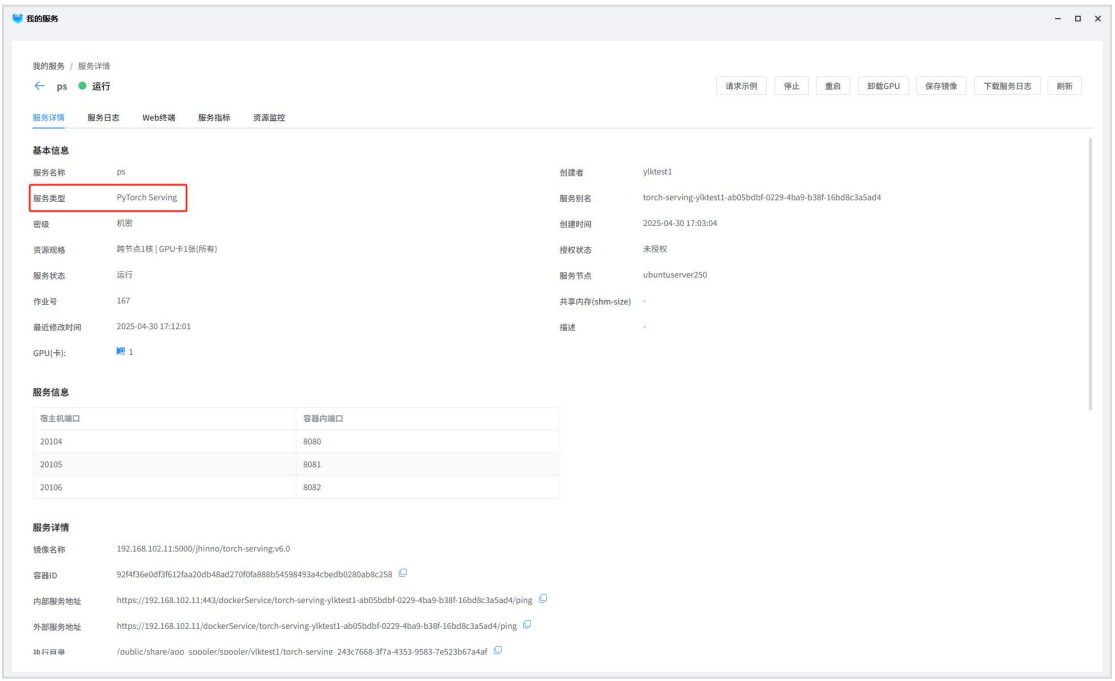
鉴权：选择是否启动 Bearer 令牌（如 API Key）确认身份与权限，开启鉴权后，在调用服务的时候，需要添加 API Key。

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务成功。

创建完成的 Pytorch serving 服务

点击创建好的服务名称，如创建的“ps”，会跳转到服务详情页面，如下图所示：



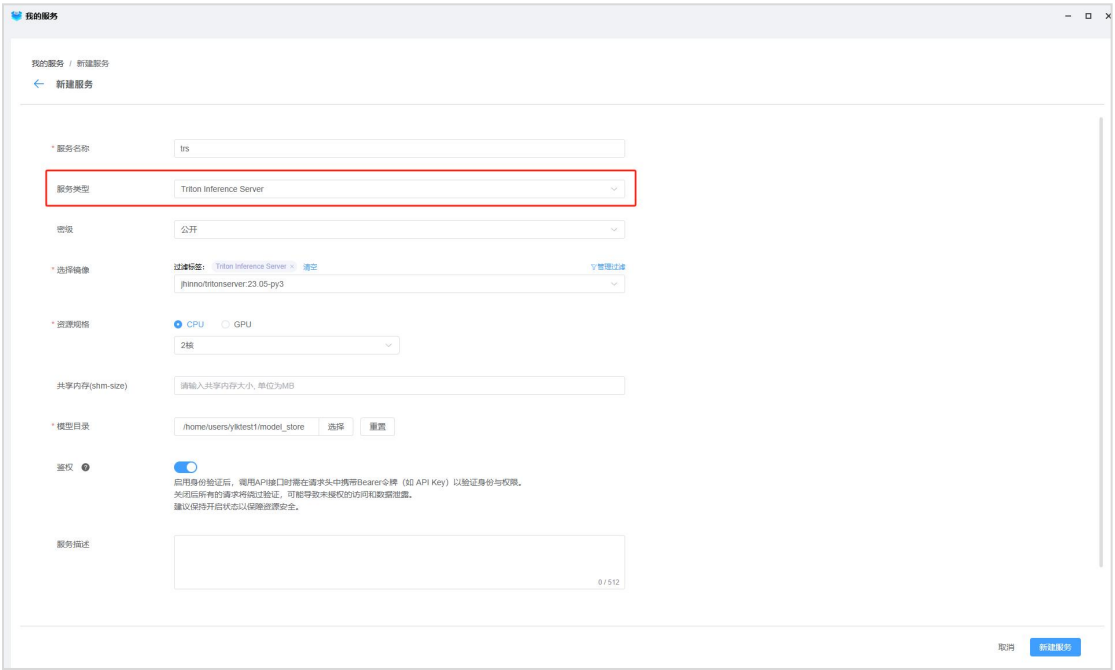
Pytorch serving 服务详情页面

在服务详情页面，与普通的服务不同的是，包含了请求示例，也就是如何调用服务的 url。由于开启了鉴权，在调用示例中包含了${YOUR_API_KEY}，使用示例的时候，使用 API Key 替换${YOUR_API_KEY}就可以正常调用服务。Pytorch serving 的请求示例内容如下：

Pytorch serving 请求示例

2.6.1.5. “Triton Inference Server”类型的服务

点击“新建服务”按钮，弹出“新建服务”窗口，选择 Triton Inference Server 服务类型和填写相关参数，如下图所示：



添加“Triton Inference Server”类型的服务配置

图中每个参数的具体含义如下：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“Triton Inference Server”。

镜像名称：选择 triton server 镜像。

资源规格：选择启动服务所需要的资源配置。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点/etc/docker/daemon.json 中配置 default-shm-size 参数大小。

鉴权：选择是否启动 Bearer 令牌（如 API Key）确认身份与权限，开启鉴权后，在调用服务的时候，需要添加 API Key。

模型目录：用户自定义。

注意：

每个模型目录下都至少包含一个模型版本目录和一个模型配置文件。

模型版本目录：包含模型文件，且必须以数字命名，作为模型版本号，数字越大版本越新。

模型配置文件：用于提供模型的基础信息，通常命名为 config.pbtxt。

假设模型存储目录在/examplebucket/models/triton/路径下，模型存储目录的格式如下：

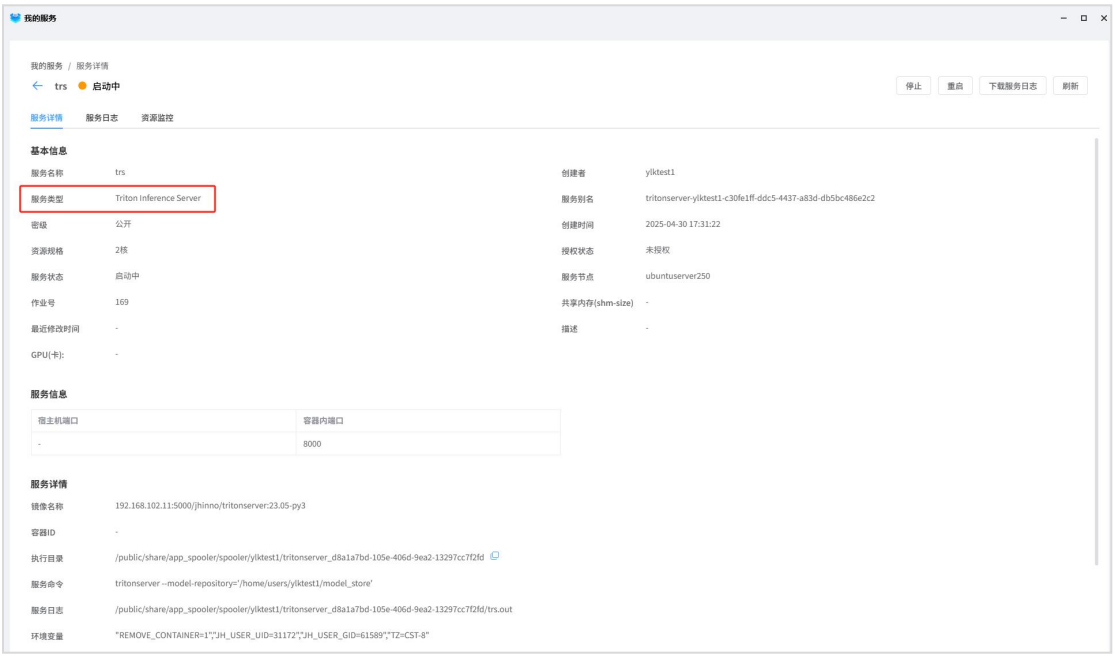
```
triton
└──resnet50_pt
    ├── 1
    │   └── model.pt
    ├── 2
    │   └── model.pt
    ├── 3
    │   └── model.pt
    └── config.pbtxt
```

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务成功。

创建完成的 Triton Inference Server 服务

点击创建好的服务名称，如创建的“trs”，会跳转到服务详情页面，如下图所示：



Triton Inference Server 服务详情页面

在服务详情页面，与普通的服务不同的是，包含了请求示例，也就是如何调用服务的 url。由于开启了鉴权，在调用示例中包含了${YOUR_API_KEY}，使用示例的时候，使用 API Key 替换${YOUR_API_KEY}就可以正常调用服务。Pytorch serving 的请求示例内容如下：

Triton Inference Server 请求示例

注意：

（1）在部署模型时，模型算法名称应和模型文件所在目录保持一致，否则在加载模型时会出错。如模型算法名称为 maskrcnn，那么目录结构应该如下：

```
triton
└——maskrcnn
    ├—— 1
    |    └—— model.onnx
    └—— config.pbtxt
```

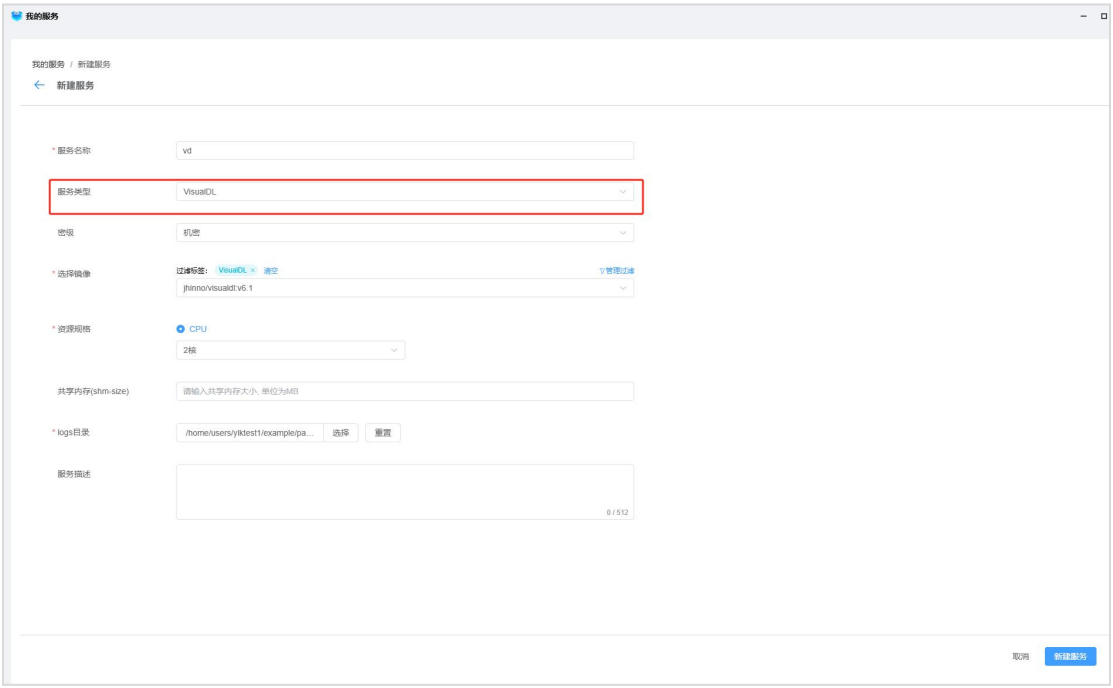
（2）部署模型时，会出现算子不支持情况，请使用 tensorflow serving 或者 torch serving 部署。

（3）如果需要部署多个版本的模型，需要在 config.pbtxt 文件中添加 version_policy 参数如：

```
version_policy: { all: {}} #所有版本
version_policy: { latest: { num_versions: 2}} #最近的两个版本
version_policy: { specific: { versions: [1,3]}} #指定版本 1 和 3
```

2.6.1.6. “VisualDL”类型的服务

点击“新建服务”按钮，弹出“新建服务”窗口，选择 VisualDL 服务类型和填写相关参数，如下图所示：



添加“VisualDL”类型的服务配置

图中每个参数的具体含义如下：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“VisualDL”。

镜像名称：选择 visualdl 镜像。

资源规格：选择启动服务所需要的资源配置，默认只能选择 CPU 资源。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点 /etc/docker/daemon.json 中配置 default-shm-size 参数大小。

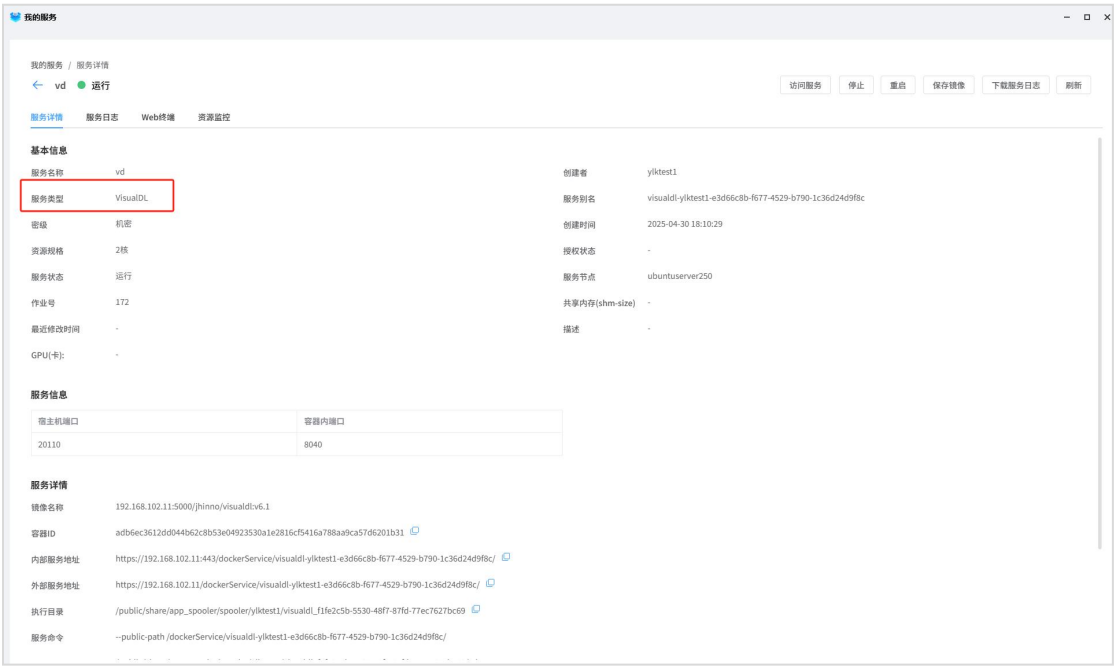
logs 地址：保存的模型 Log 路径。

服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务成功。

创建完成的 VisualDL 服务

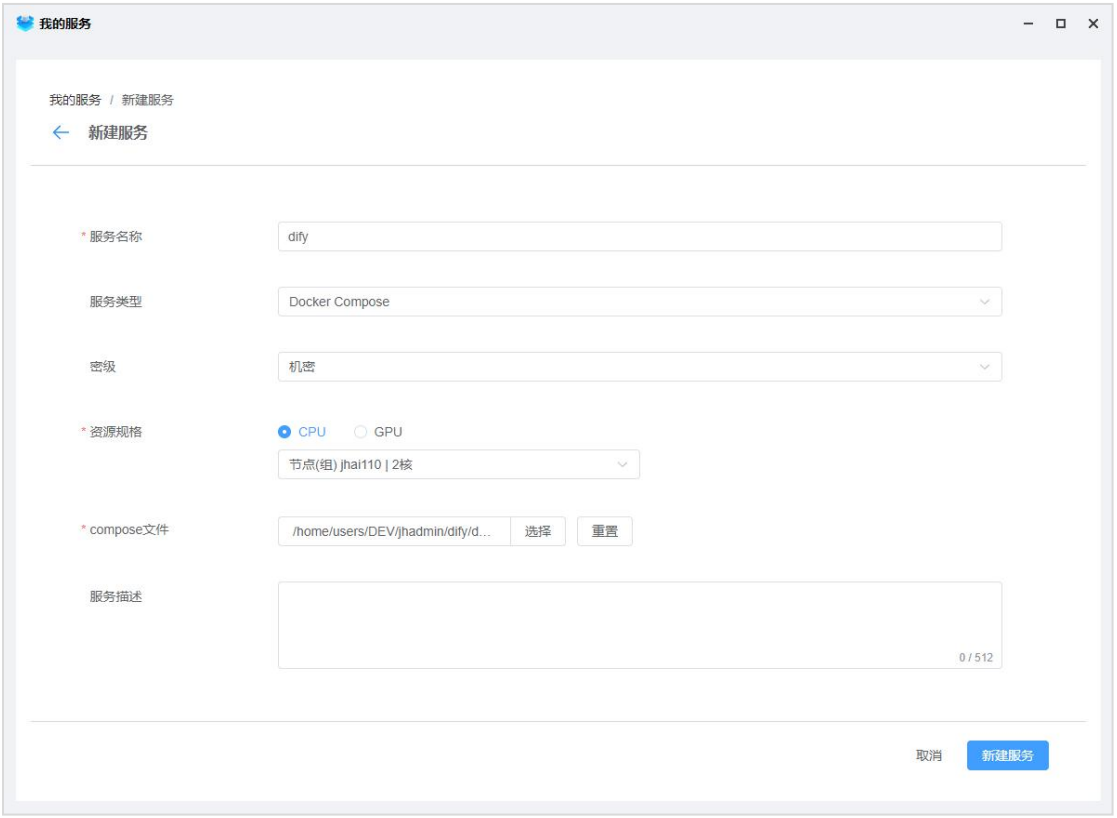
点击创建好的服务名称，如创建的“vd”，会跳转到服务详情页面，如下图所示：



VisualDL 服务详情页面

2.6.1.7. “Docker Compose”类型服务

点击“新建服务”按钮，弹出“新建服务”窗口，选择 Docker Compose 服务类型和填写相关参数，如下图所示：



添加“Docker Compose”类型的服务配置

服务中参数的含义：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“Docker Compose”。

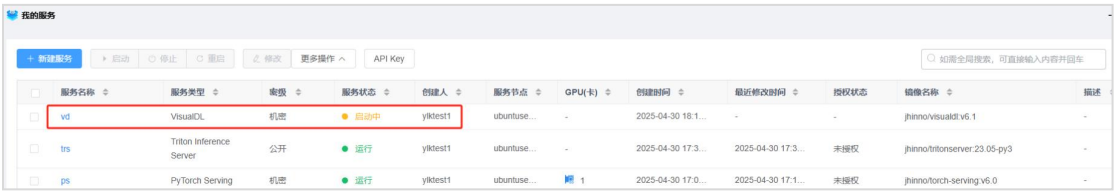
资源规格：选择启动服务所需要的资源配置，默认只能选择 GPU 资源。

compose 文件：选择启动 docker compose 服务需要的 yaml 配置文件。

自定义参数：手动添加 LLaMa Box 中的更多参数，异常参数或重复参数可能会导致服务启动失败。

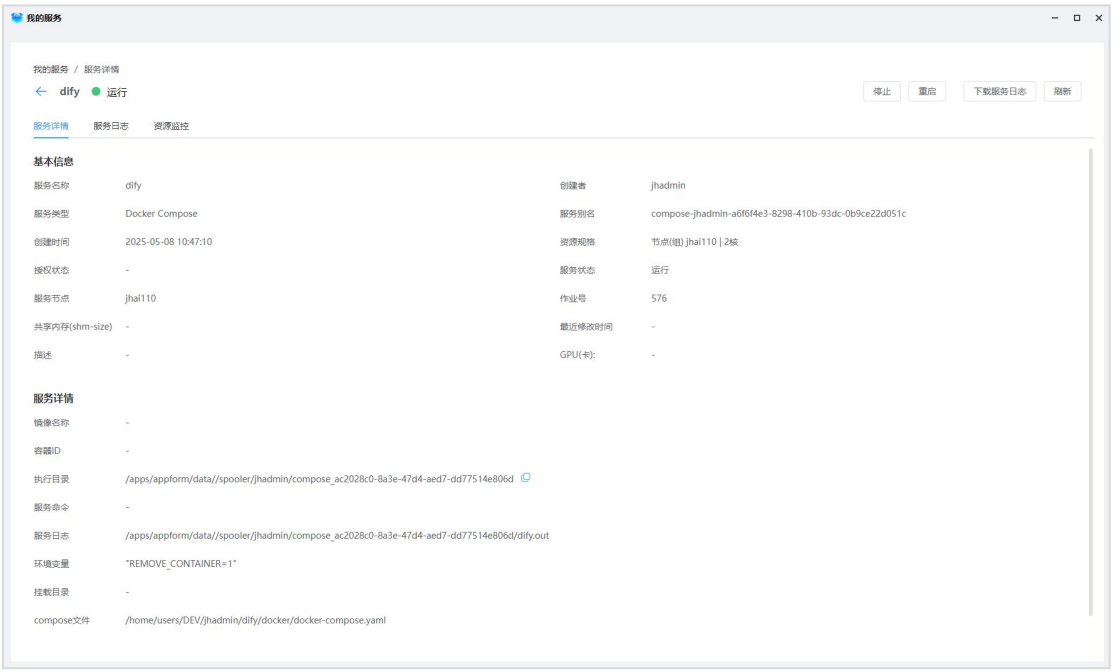
服务描述：填写启动服务的相关信息，选填。

填写完成后，点击“新建服务”按钮，添加服务实例，在我的服务列表可以看到新建的 docker compose 服务，如下图所示：



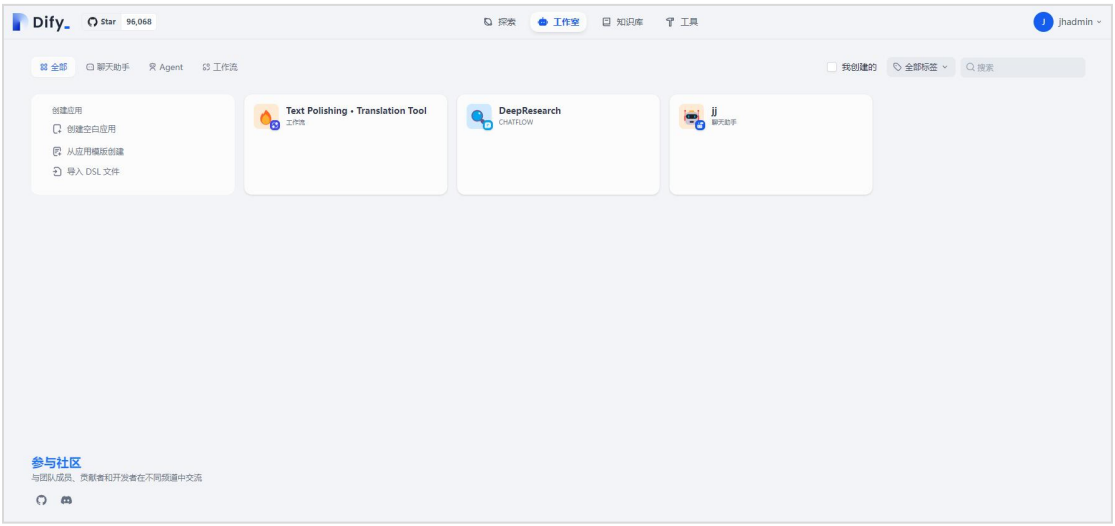
docker compose 服务

点击我的服务列表中的 docker compose 服务名称，跳转至 docker compose 服务详情页面，如下图所示：



docker compose 服务详情页面

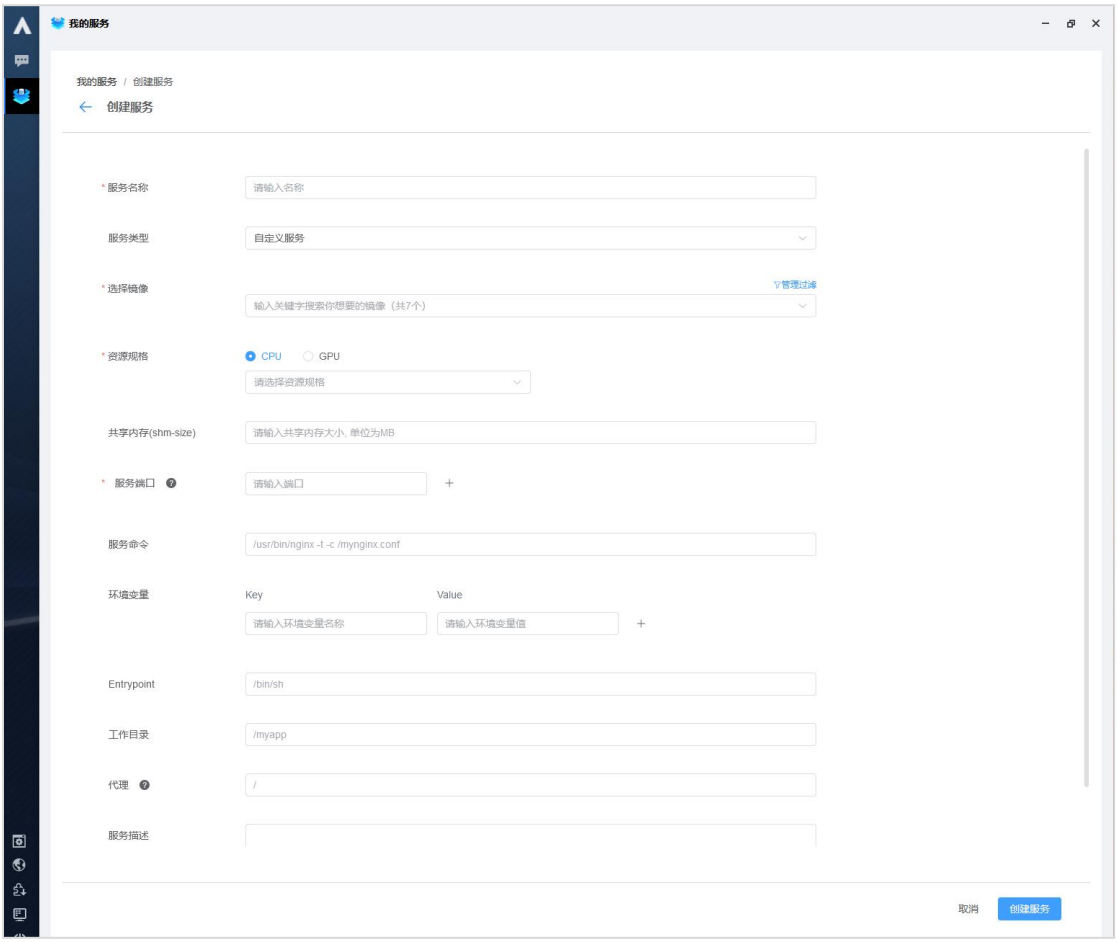
示例中启动“dify”，按照 compose 文件中的设置，打开 dify 界面，如下图所示：



dify 界面

2.6.1.8. “自定义服务”类型的服务

点击“新建服务”按钮，弹出“创建服务”窗口，选择自定义服务服务类型和填写相关参数，如下图所示：



添加“自定义服务”类型的服务配置

服务中参数的含义：

服务名称：用户自定义。

密级：管理员开启密级功能后，显示此选项，默认为用户密级，用于数据安全、保密。

服务类型：选择“自定义服务”。

镜像名称：选择自定义服务镜像。

资源规格：选择启动服务所需要的资源配置，默认只能选择 GPU 资源。

共享内存(shm-size)：设置作业运行容器的共享内存，默认运行所在节点 /etc/docker/daemon.json 中配置 default-shm-size 参数大小。

服务端口：容器内服务使用的端口。

服务命令：容器内服务执行的命令。

Entrypoint：设置容器启动时执行的主程序，并允许向该程序传递参数。

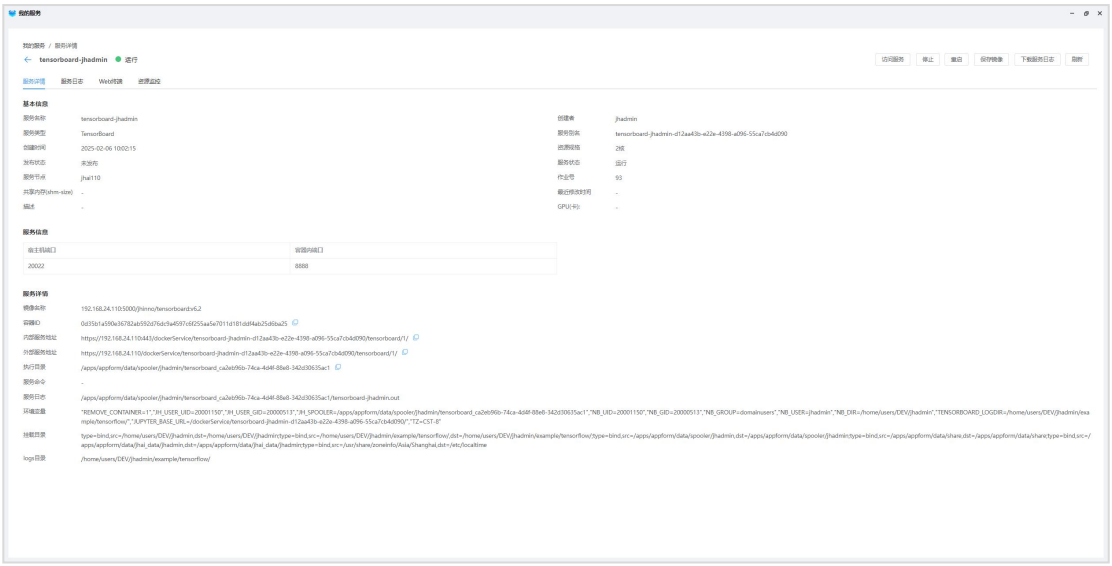
工作目录：容器内部的一个目录。

代理：代理用于设置容器内 Web 服务的访问路径，仅支持 HTTP 服务。例如，若容器内 Web 服务路径为 http://ip:port/demo，可以将代理设置为 /demo，启动后通过服务地址访问该服务。

服务描述：输入服务描述信息，可选。

2.6.2. 访问服务

点击某些类型服务（“MindInsight”“Tensorboard”“VisualDL”和“自定义服务”）名称，弹出“服务详情”页面，右上角点击“访问服务”按钮，会打开服务相关页面，如下图所示：

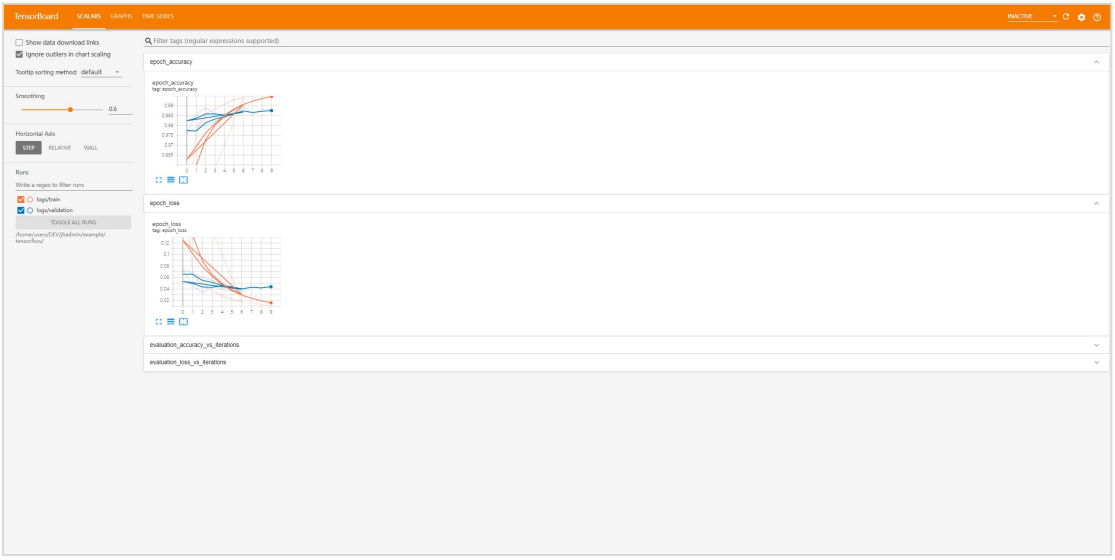


访问服务入口

注意：

(1) 访问服务功能仅在服务实例的状态为“运行”状态时可用。

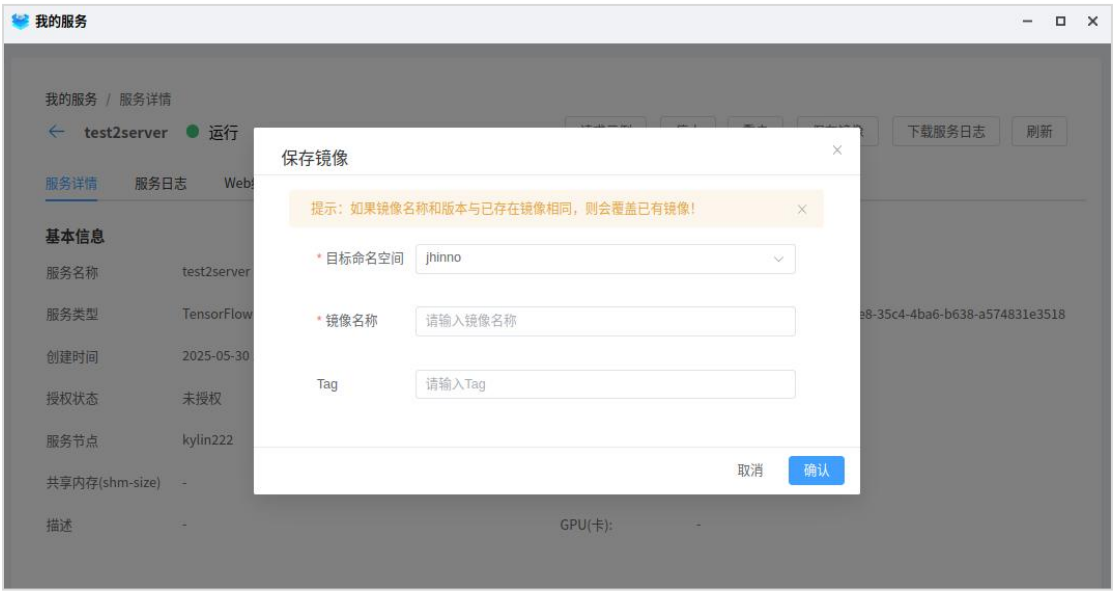
(2) 访问服务功能仅限用户自己通过服务管理发布的服务或训练模型打开的服务，开发中心创建的服务不存在该功能。



服务内容

2.6.3. 保存镜像

保存镜像可以将服务当前容器保存为新的容器镜像。点击“保存镜像”按钮，弹出“保存镜像”窗口，如下图所示：



保存镜像

图中每个参数的具体含义如下：

目标命名空间：选择镜像保存的命名空间。

镜像名称：输入保存后的镜像名称。

Tag：输入镜像版本，默认为 latest。

点击“确定”按钮，将运行服务的容器保存为镜像，如下图所示：

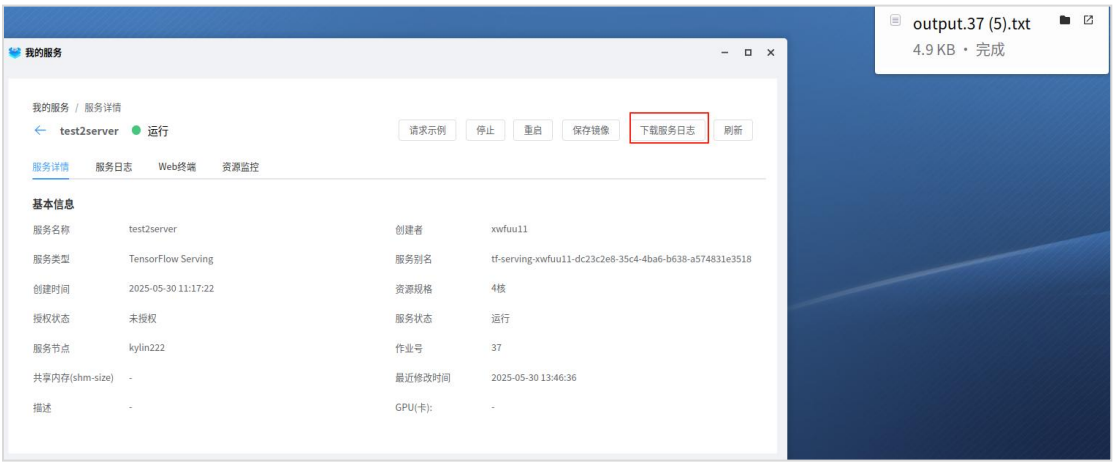
保存镜像

2.6.4. 下载服务日志

下载服务日志会将运行服务过程中的日志保存到本地。点击“下载服务日志”按钮，自动下载日志到本地，如下图所示：

下载服务日志

保存的日志命名规则为 output.+作业号+.txt，日志保存在浏览器默认下载目录。



下载日志命名详情

2.6.5. 刷新

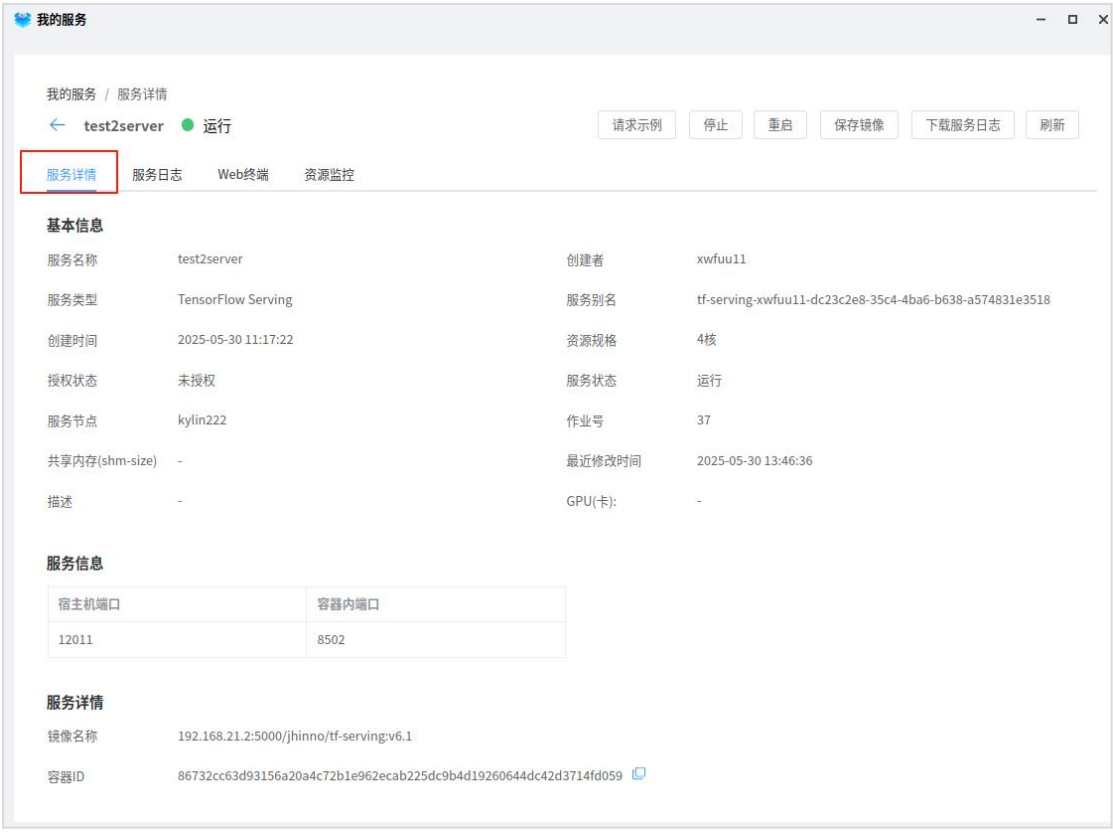
刷新服务会刷新整个服务内容，服务状态不会自动更新，只有在点击刷新的情况下更新，如下图所示：

刷新服务

2.6.6. 服务详情

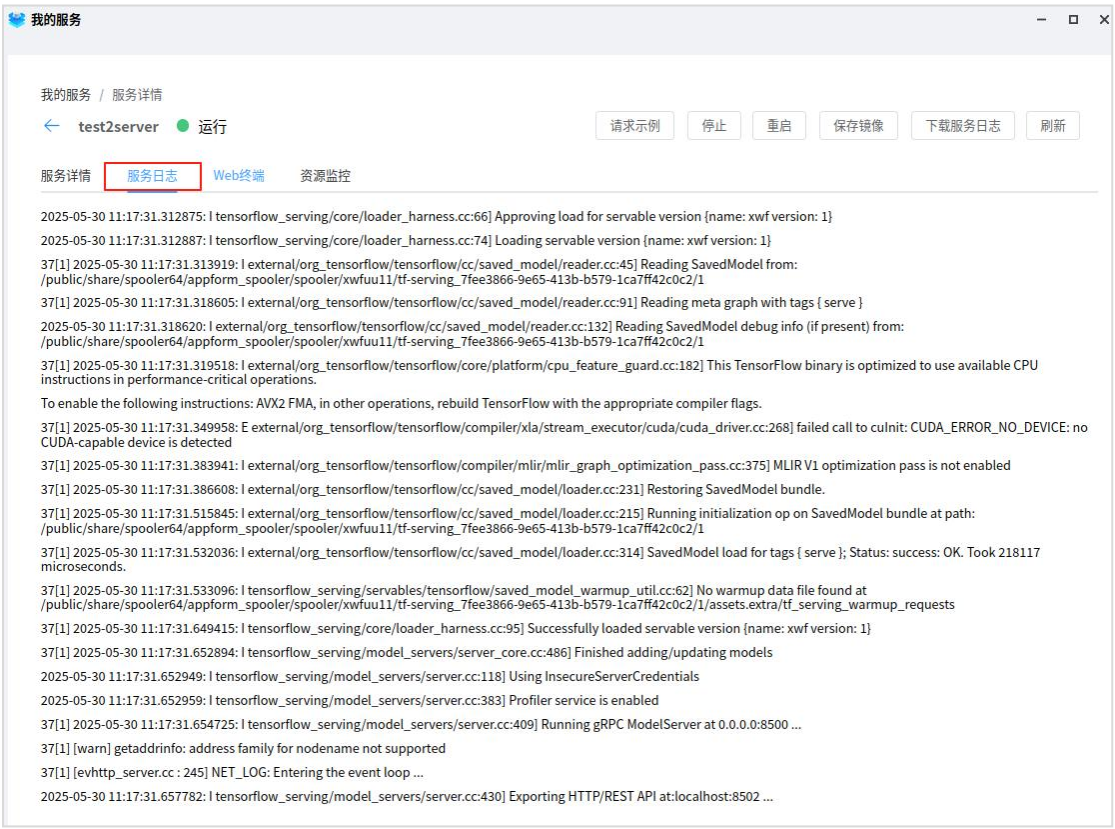
点击服务列表中的服务实例的服务名称，会打开服务详情页面。

服务详情：展示服务的基本信息，服务信息和服务详情，如下图所示：



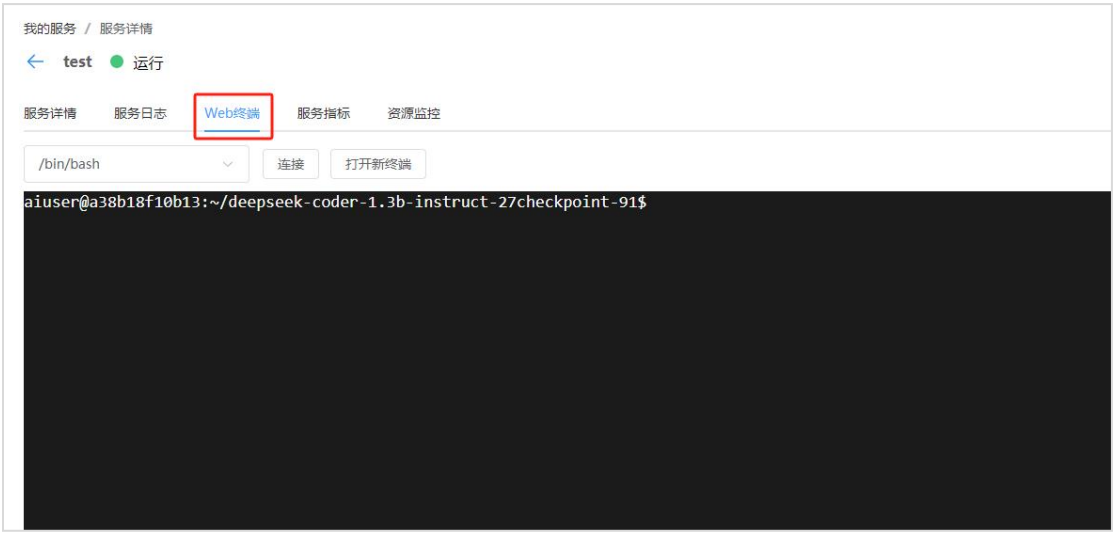
服务详情

服务日志：点击“服务日志”，切换至服务日志页面，查看服务日志。具体详情如下图所示：



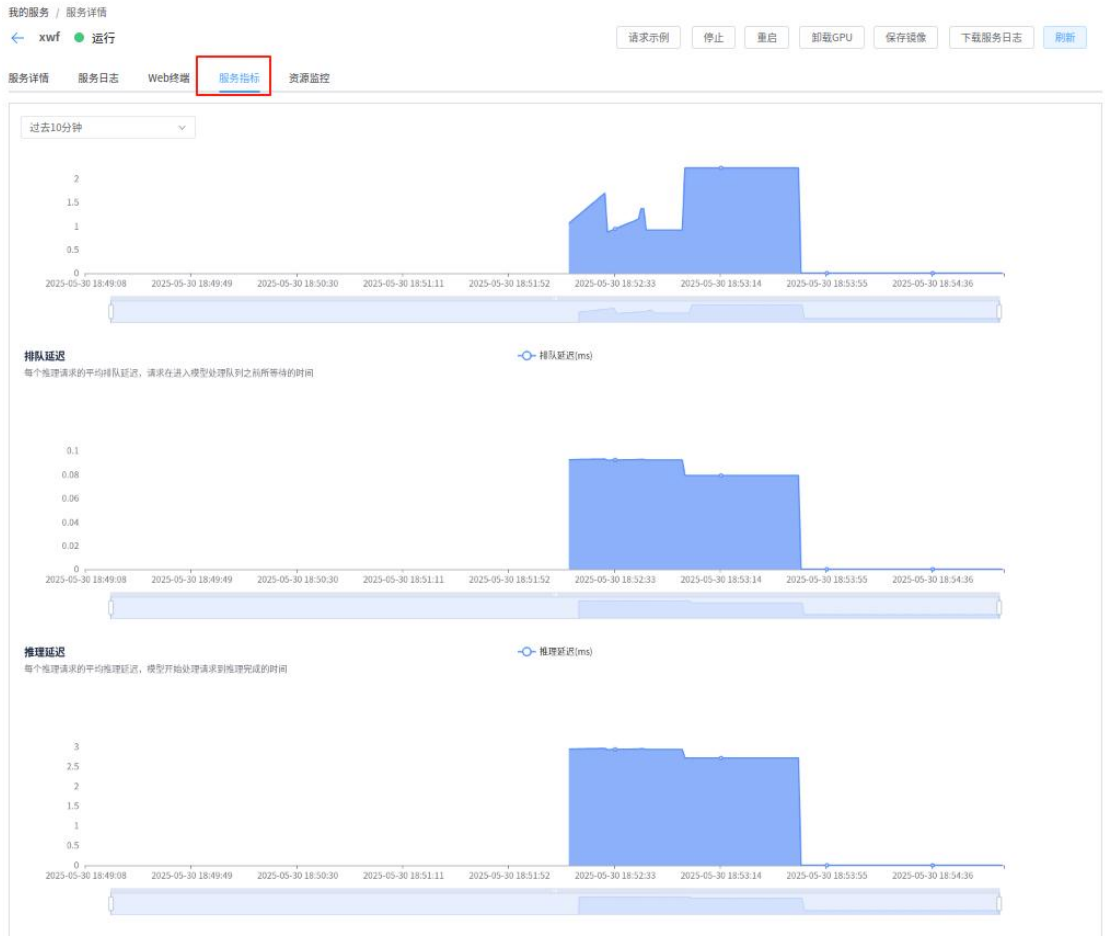
服务日志

Web 终端：点击“Web 终端”，切换至 Web 终端页面，可以在页面模拟终端操作服务容器，如下图所示：



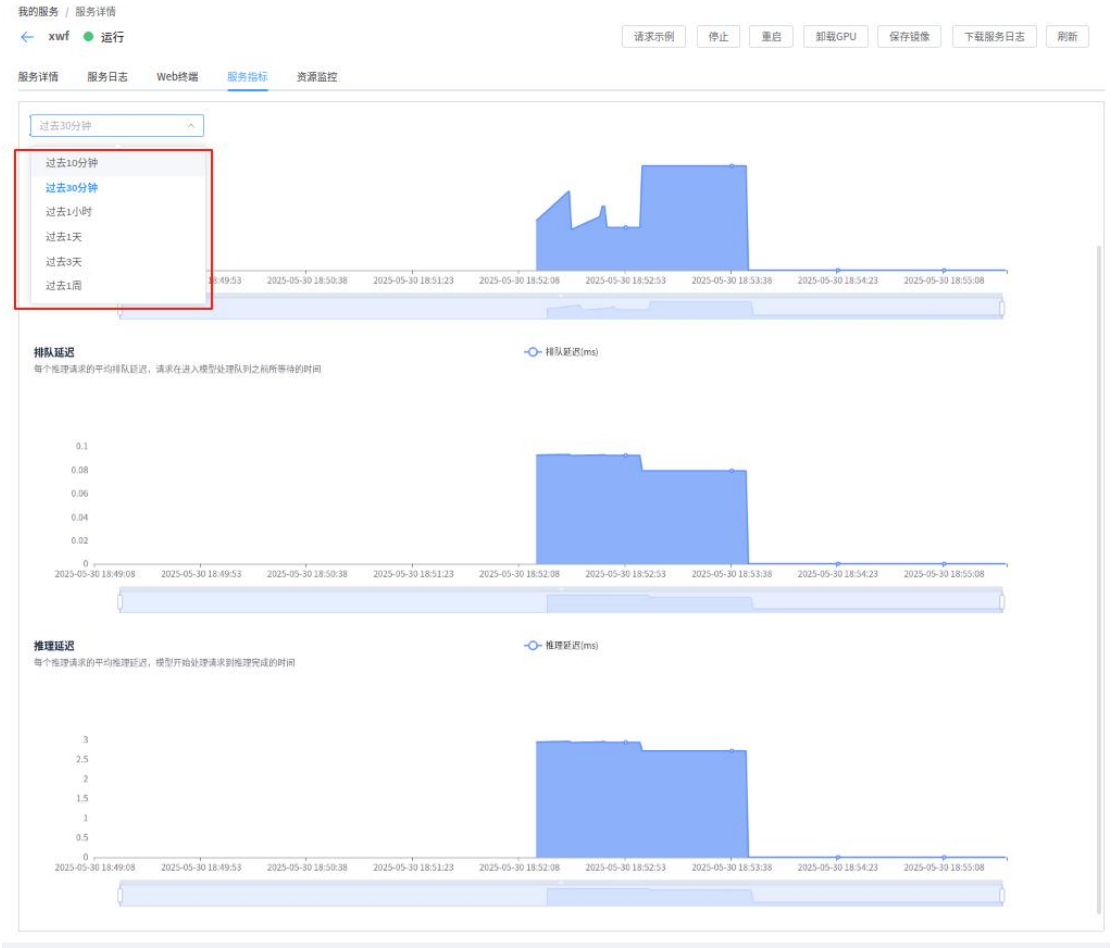
Web 终端

服务指标：点击“服务指标”，切换到服务指标页面，查看推理服务的服务指标的变化。如下图所示：



服务指标内容

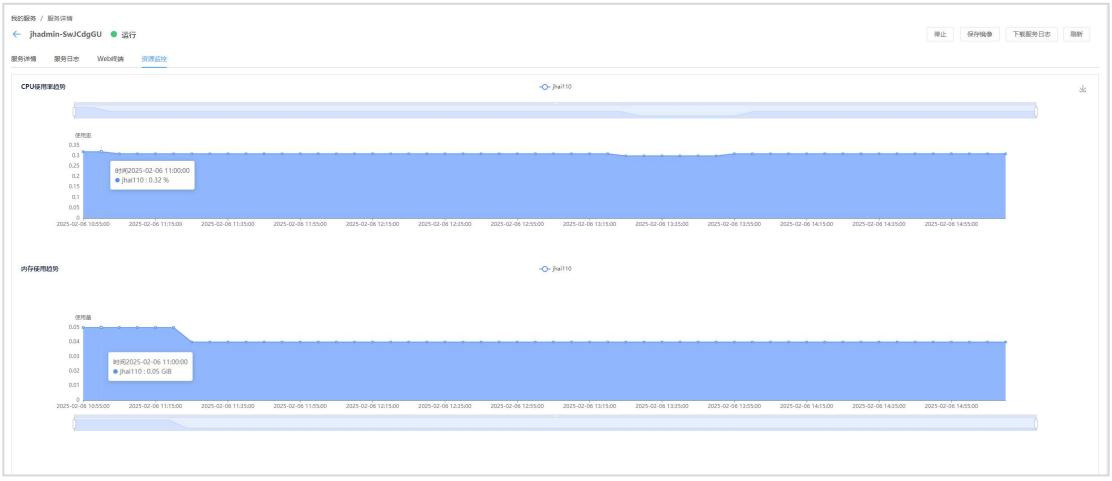
这些指标可以按照时间段筛选，时间段包含过去 10 分钟、过去 30 分钟、过去 1 小时、过去 1 天、过去 3 天和过去 1 周，如下图所示：



按照时间段查看指标

注意：只有运行状态的 Pytorch Serving 服务有“服务指标”按钮。

资源监控：点击“资源监控”，切换到资源监控页面，看到服务使用的资源情况。启动服务只选择 cpu 资源，资源监控将显示 cpu 和内存的使用情况；如果启动服务选择 gpu 资源，则在资源监控页面会多出 gpu 的监控信息。如下图所示：

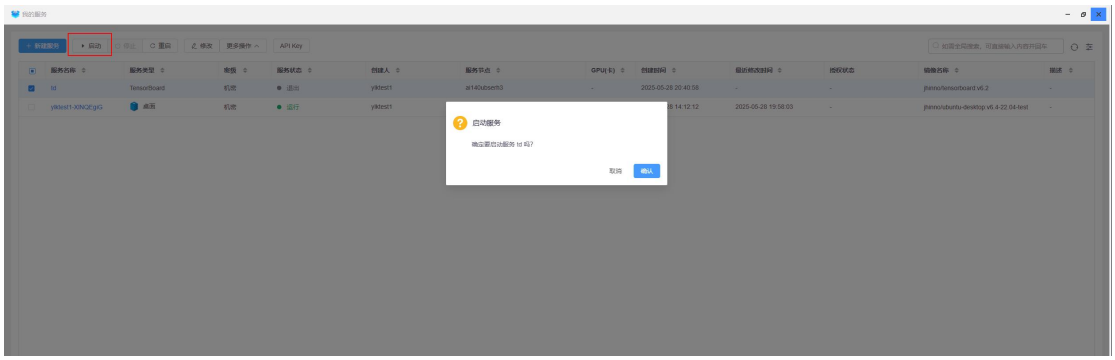


未选择 GPU 的资源监控

选择 GPU 的资源监控

2.6.7. 启动服务

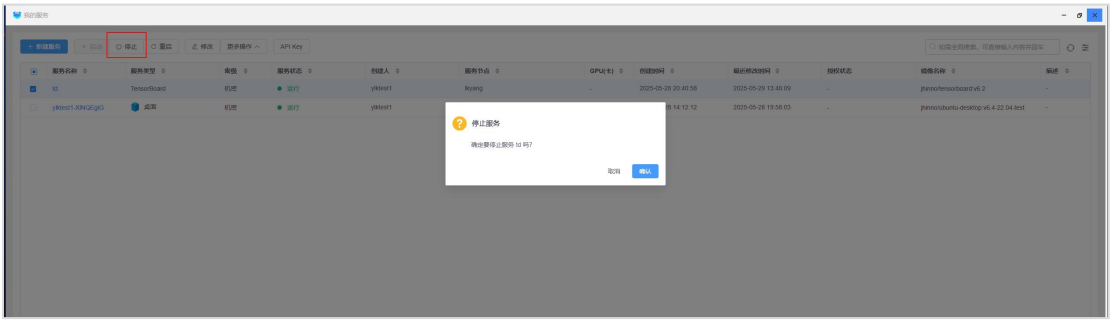
选中服务列表中某一条在服务管理启动后处于非启动状态的服务实例后，点击“启动”按钮，弹出“是否启动此服务”提示窗口，点击“确定”按钮，即可启动该条服务实例。



启动服务

2.6.8. 停止服务

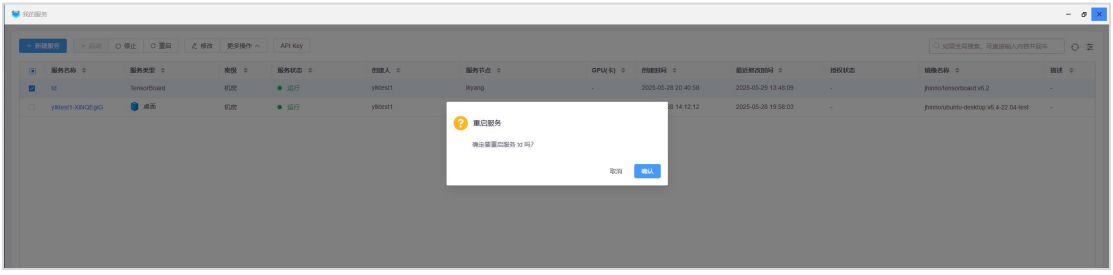
选中“运行”中的服务，点击“停止”按钮，弹出“是否停止此服务”提示窗口，点击“确定”按钮，即可停止该条服务实例。如下图所示：



停止服务

2.6.9. 重启服务

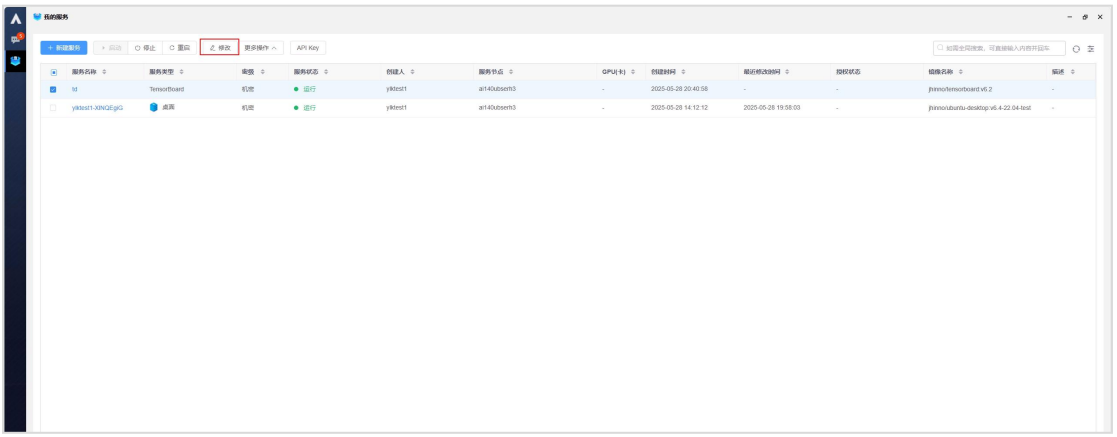
选中“运行”中且“非开发环境”启动的服务，点击“重启”按钮，弹出“是否重启此服务”提示窗口，点击“确定”按钮，即可重启该条服务实例。如下图所示：



重启服务

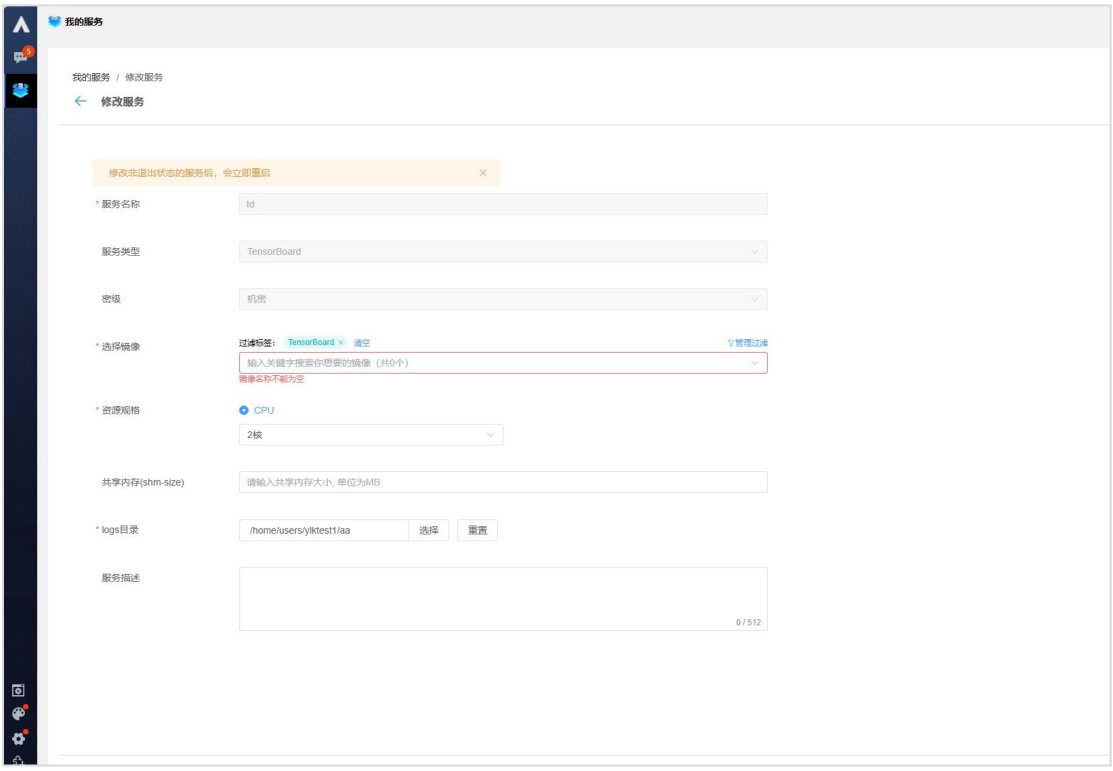
2.6.10. 修改服务

选中在服务管理页面创建的服务（非开发环境启动的服务），点击“修改”按钮，修改服务窗口，重新设置服务参数，点击“确定”按钮，即可修改完成该条服务实例。当服务处于运行状态，修改后服务会按照最新参数重启；其他状态仅会修改服务参数，服务再次启动时，会按照最新参数启动。如下图所示：



修改服务入口

修改服务，除了服务名称、服务类型和密级之外的参数都可以修改，如下图所示：

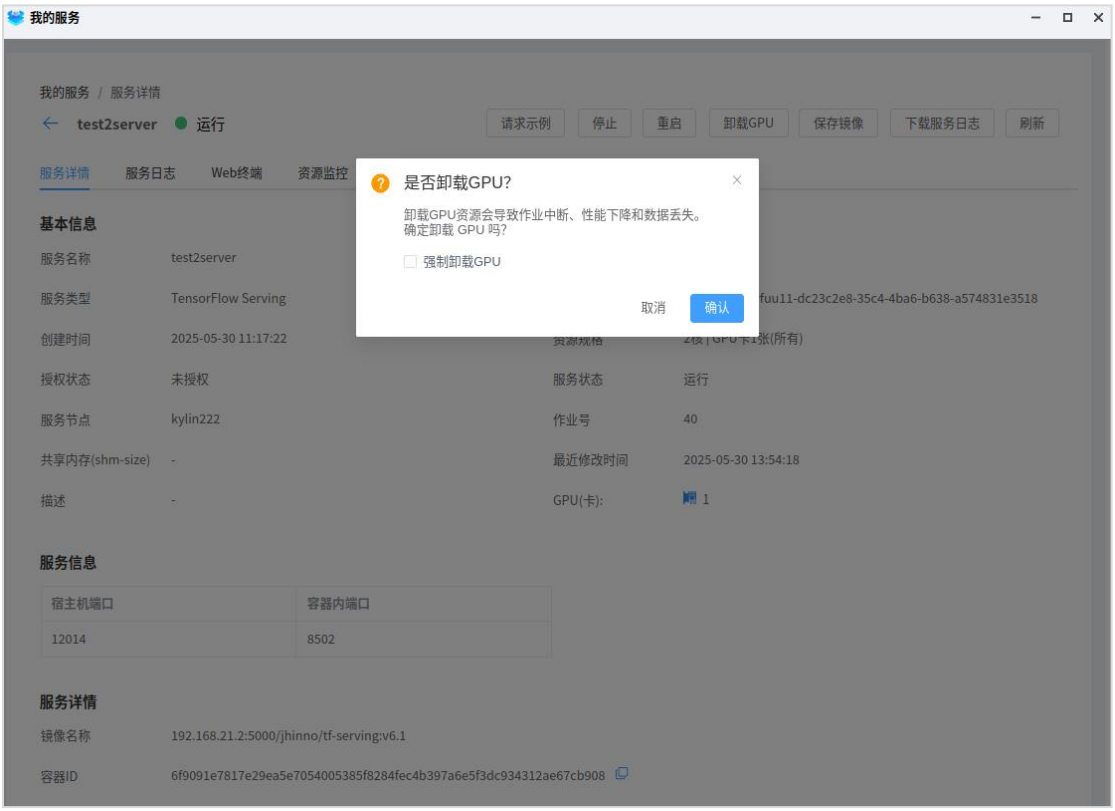


修改服务页面

参数修改完成后，点击“修改服务”按钮，即可完成服务修改。

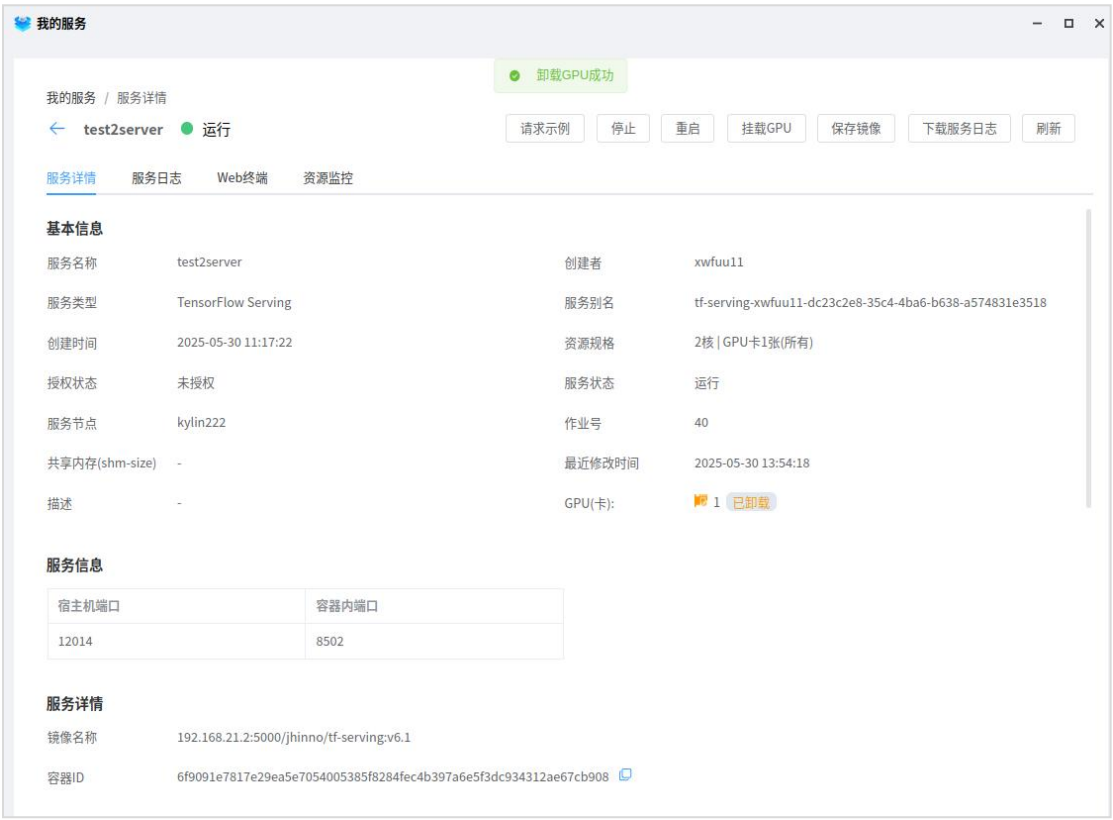
2.6.11. 卸载 GPU

创建服务的过程中，选择 GPU 资源后，在服务列表中会看到 GPU 信息。点击自己创建选择 GPU 卡且运行中的服务，进入服务详情页面，在服务详情页面的右上角有“卸载 GPU”按钮。点击“卸载 GPU”按钮后，弹出“是否卸载 GPU”弹框，点击“确认”后，GPU 卸载成功，在服务列表和服务详情页面可以看到 GPU 处于卸载状态，如下图所示：



是否卸载 GPU

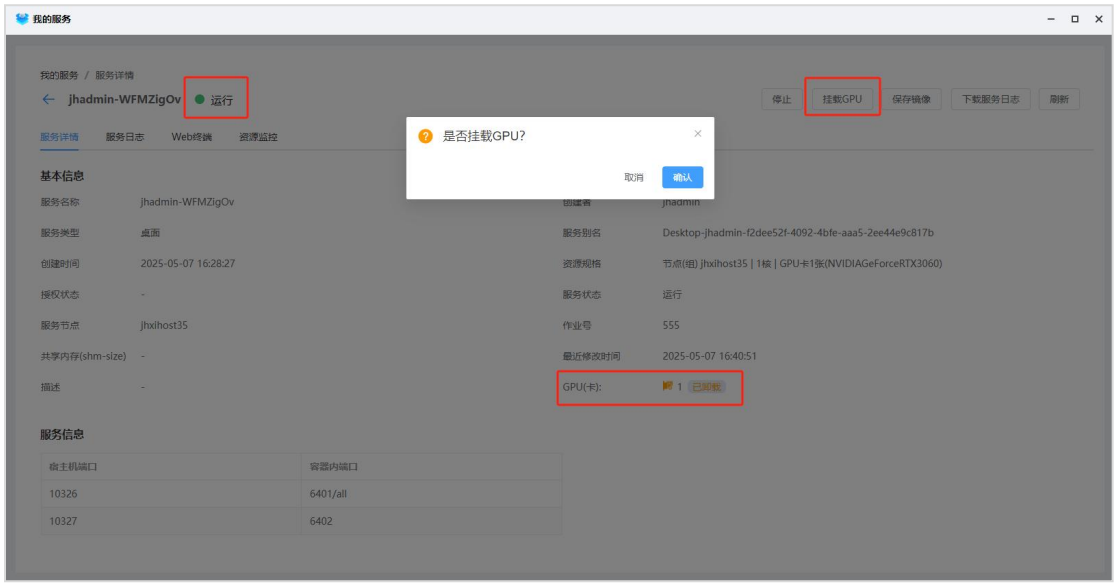
卸载完成后，会提示“卸载 GPU 成功”，并且 GPU（卡）属性区域提示“已卸载”，如下图所示：



卸载 GPU 成功的服务详情页面

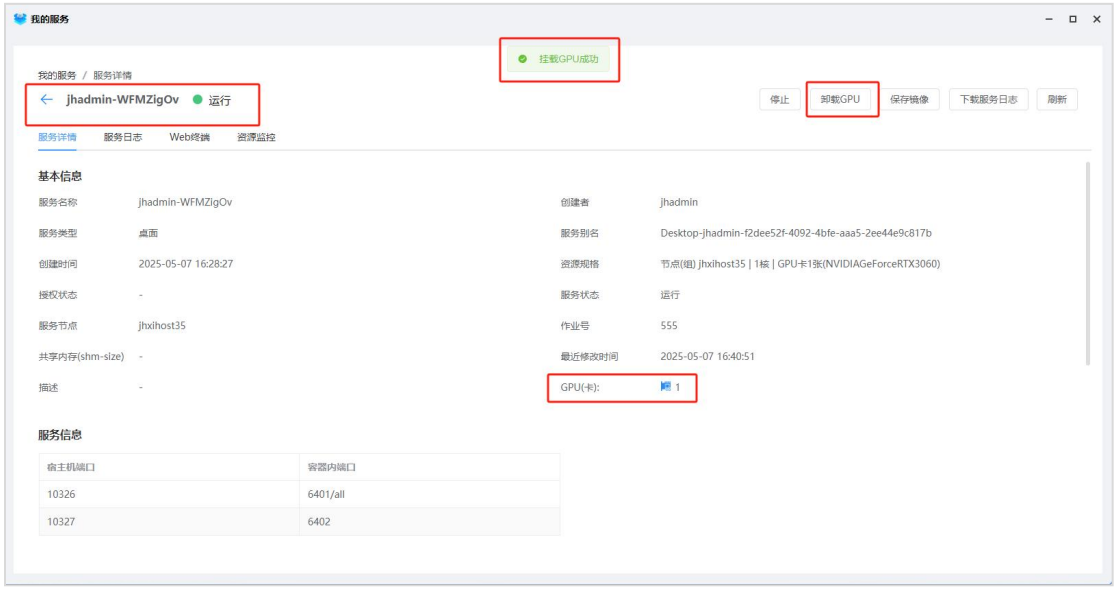
2.6.12. 挂载 GPU

卸载 GPU 后运行的服务，可以选择“挂载 GPU”，为服务添加 GPU 资源。点击“挂载 GPU”，弹出对话框，确认是否挂载 GPU，点击“确认”后，容器服务重新加载到 GPU。



挂载 GPU 页面

挂载 GPU 成功的服务，服务状态为“运行”，“挂载 GPU”按钮变为“卸载 GPU”，GPU（卡）属性不再有“已卸载”关键字，如下图所示：



挂载 GPU 成功的服务详情页面

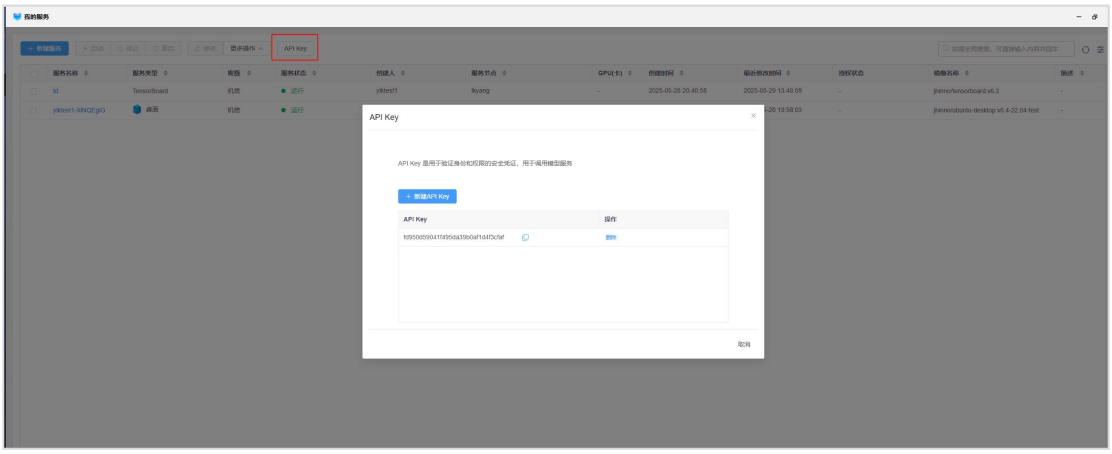
2.6.13. API key

API Key 是用于验证身份和权限的安全凭证，用于调用模型服务。启动指定服务时，如果开启鉴权，则需要在调用服务时，添加此处生成的 API key。

在我的服务页面，API Key 位于服务列表上方，如下图所示：

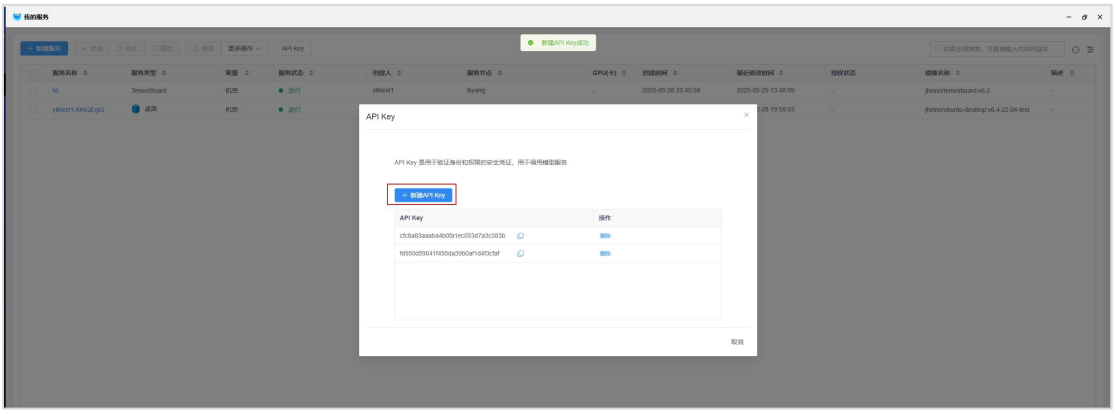
“API Key”入口

点击“API Key”进入 API Key 设置页面，新增或者删除 API Key，如下图所示：



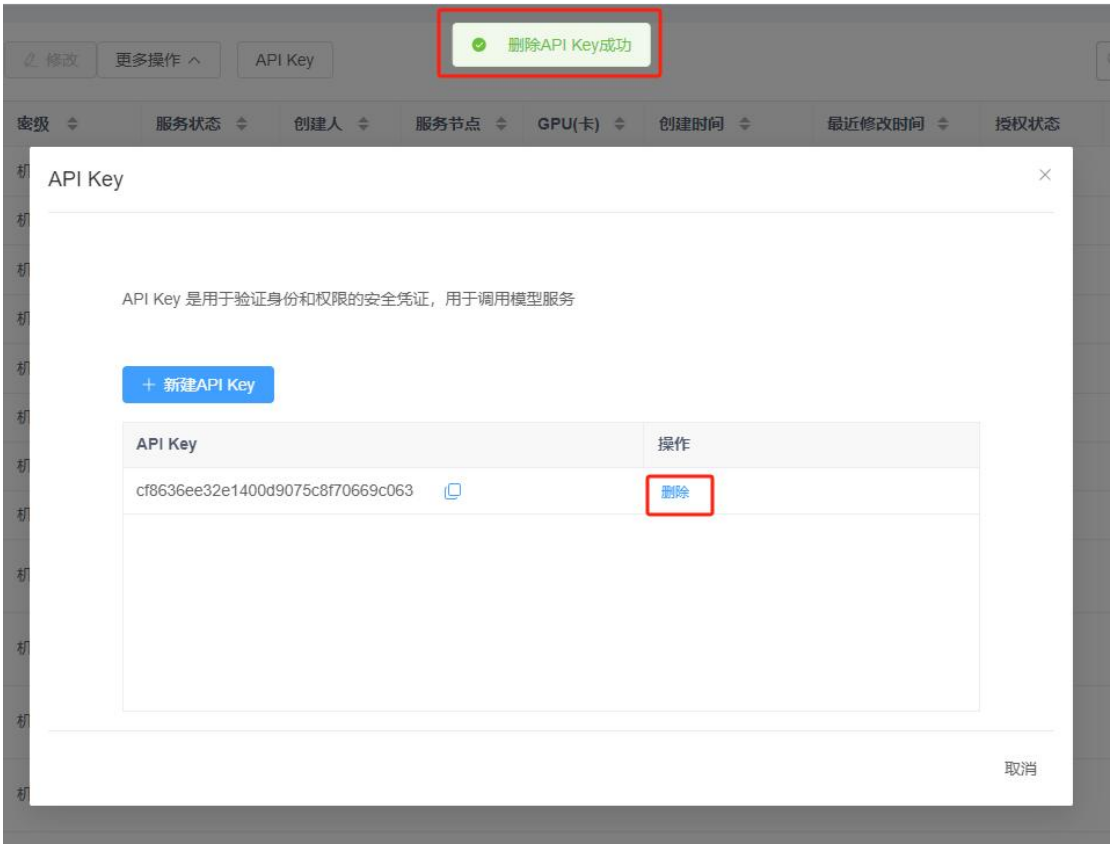
API Key 设置页面

每个用户可以创建 5 个 API Key，在调用服务时，使用任意一个 key 都可以。新增 API Key 只需点击“+新建 API key”即可，如下图所示：



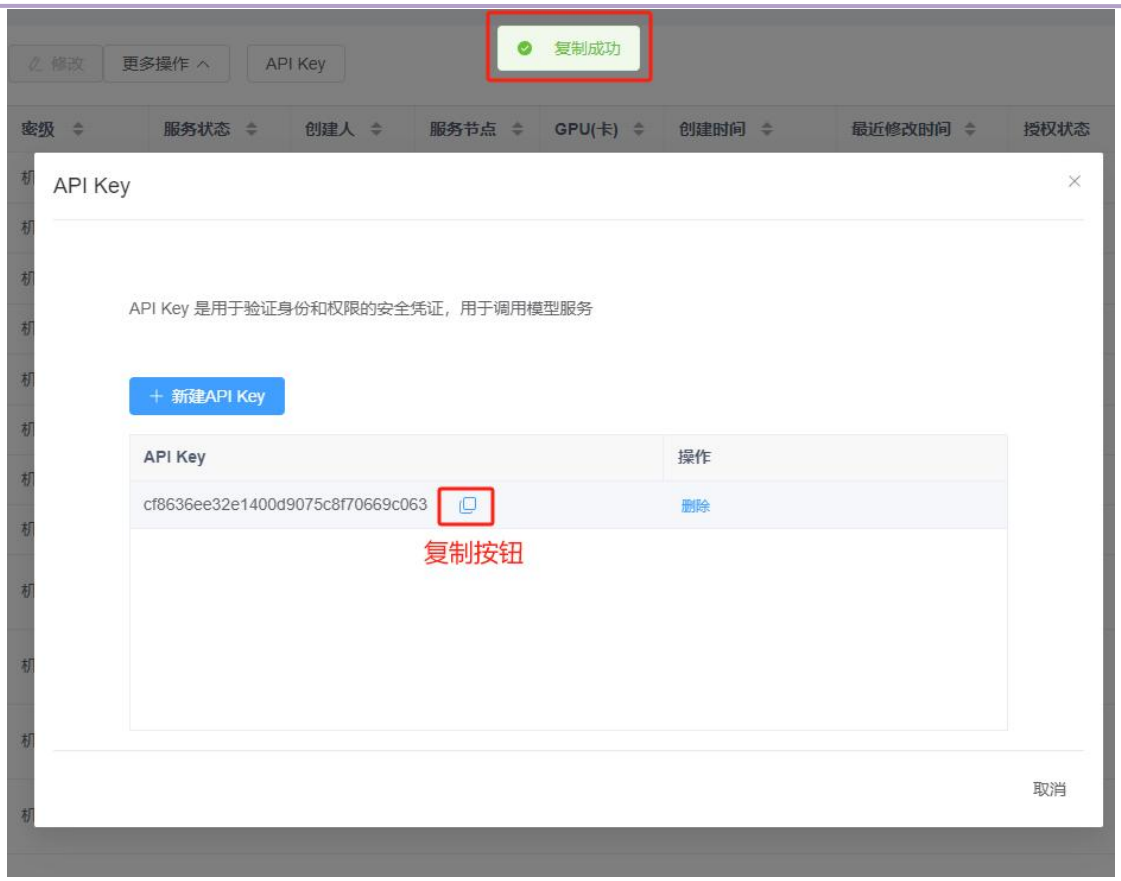
新建 API Key

不需要某个 key 时，可以点击对应 key 后的“删除”按钮，如下图所示：



删除 API Key

复制 key 可以通过点击复制按钮实现，如下图所示：

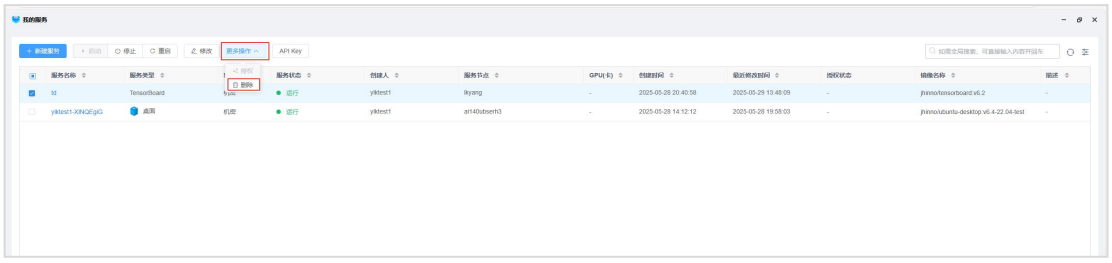


复制 API key

2.6.14. 更多操作

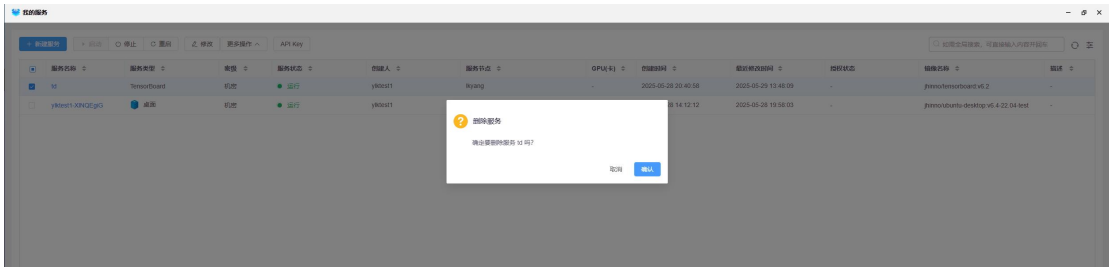
选中在服务管理页面服务，点击“更多操作”，弹出“授权”和“删除”按钮，所有服务均可被删除。

用户自己创建且运行中的“Pytorch Serving”“Tensorflow Serving”“Triton Inference Server”类型的服务实例可以“授权”。



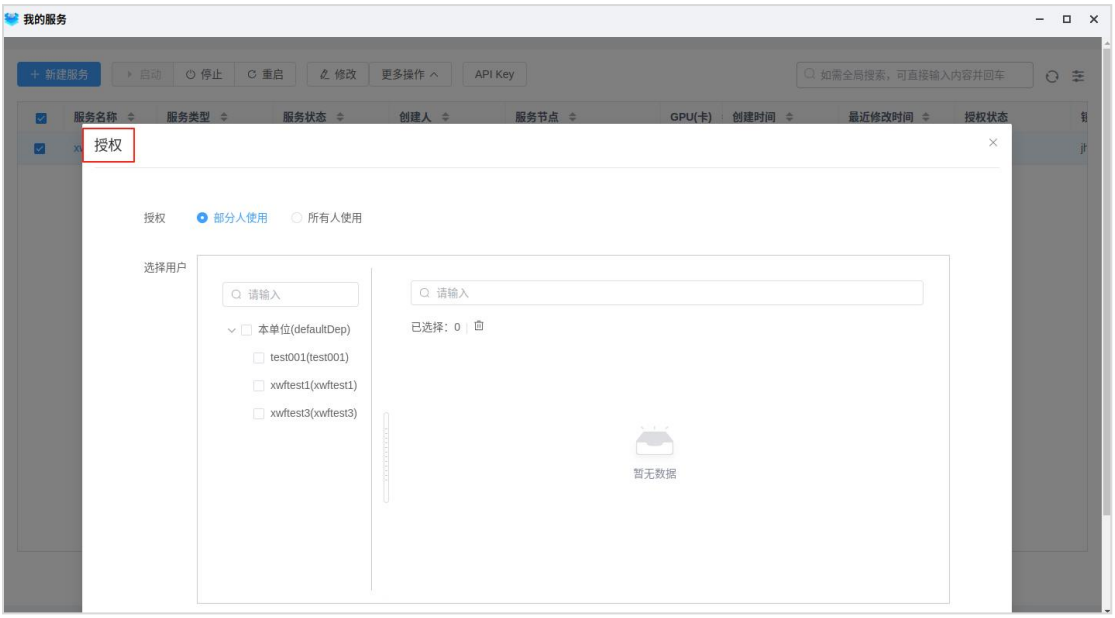
更多操作

- 删除服务



删除服务

- 授权服务



附录：常见问题及解决办法

(1) 如何在容器服务中使用用户自己的数据

解决方案：

容器服务启动时会默认挂载用户家目录、作业数据区、共享数据区，可以在启动的容器服务中直接访问用户家目录、作业数据区、共享数据区下的数据文件，目录绝对路径和宿主机保持一致。

(2) 如何保存在容器服务中创建的文件

解决方案：

容器服务启动时会默认挂载用户家目录、作业数据区和共享数据区，可以将需要保存的文件拷贝到用户家目录、作业数据区或者共享数据区下。

(3) 如何在非 root 启动的镜像中安装依赖包

解决方案：

使用 sudo 执行 yum 或 apt-get 命令。

(4) 模型训练任务异常退出，显示 out of memory

解决方案：

调整训练程序代码或数据大小。

(5) 容器服务启动失败报错，output.txt 中提示：“service start failed,current status is: rejected, reason is: No such image: ahaha/aaa:v_testa1”

原因：

创建容器服务时，选择/输入的镜像名是不存在的，或者是浏览器页面缓存的镜像名实际上已经不存在。

解决方案：

清除浏览器缓存，输入/选择真正存在的镜像。

(6) 挂起 TensorFlow、Pytorch 等 AI 作业。作业状态显示挂起，但后台进程仍然运行

原因：

作业容器启动前无法被挂起。

(7) 数值数据集列名中包含符号.的数据集，预览无法显示数据。

原因：

不支持列名中包含符号.的数据集。

解决方案：

不要在列名中包含符号。

(8) 创建一个新的开发环境，此时开发环境一直启动不了，报错：“The agent was unable to contact any other agent located on a manager node。”

原因：

发现和虚拟机虚拟网卡有关系。

解决方案：

当出现这种情况时关闭网卡的 checksums，使用以下命令执行：

```
ethtool -K <interface> tx off
```

(9) 容器桌面中，执行 mount，报错：“mount: /ubuntuxwf/: mount failed: Operation not permitted.”

原因：

容器桌面安全考虑，没有开放 privileged 权限。

解决方案：

把需要挂载的文件解压后放到容器中使用。

(10) tensorboard 有时打开后为空页面，没有加载出 tensorboard 主页面。

原因：

可能 tensorboard 需要的 json 文件没有加载完。

解决方案：

关闭页面重新打开。

(11) 开发环境容器桌面中启动 vscode 闪退。

原因：

vscode 在容器桌面中启动需要使用非隔离模式。

解决方案：

启动参数增加--no-sandbox，例如：./code --no-sandbox

(12) 使用 libaiflow.log_figure 保存并记录图表 html 时，在实验管理界面中 html 无法显示。

原因：

plotly 生成的图表 html 文件中对于 plotly 库的引用使用的是 cdn 模式，需要能够联网加载 plotly.js 能正常显示图表。

解决方案：

建议保存和记录 plotly 时使用.png, .jpeg 等图片格式替代 html 格式。

(13) 桌面容器在 firefox 中获取剪切板内容失败，导致无法粘贴从本地

复制的内容

原因：

Firefox 存在剪贴板访问安全限制。

解决方案 A：

可使用门户登录页推荐的浏览器。

解决方案 B：

步骤 1、打开火狐浏览器地址栏输入：about:config

步骤 2、点击接收风险并继续，搜索：

dom.events.testing.asyncClipboard、

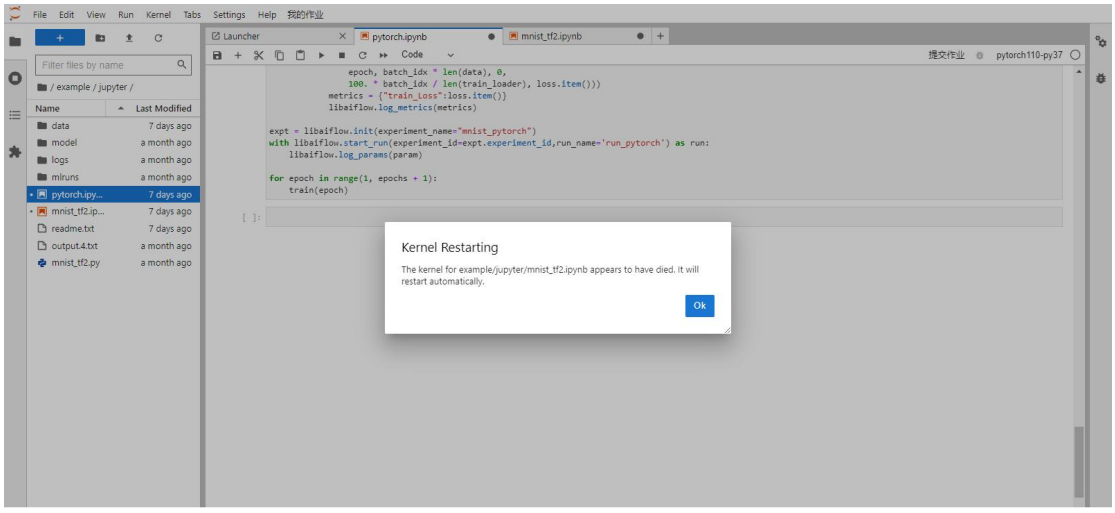
dom.events.asyncClipboard.readText 将其值修改为 true；（注：如果

dom.events.asyncClipboard.readText 不存在可以不用修改）

步骤 3、重启浏览器即可成功。

(14) 开发环境的内存耗尽后可能会引起下面一系列问题

1）Jupyter 中 kernel Restarting



原因：

可能是因为内存占满而导致 Kernel 重启。

解决方案：

可以尝试切换更高版本内存的硬件规格。

2）Jupyter 报错内存已满或者“CUDA error: out of memory”

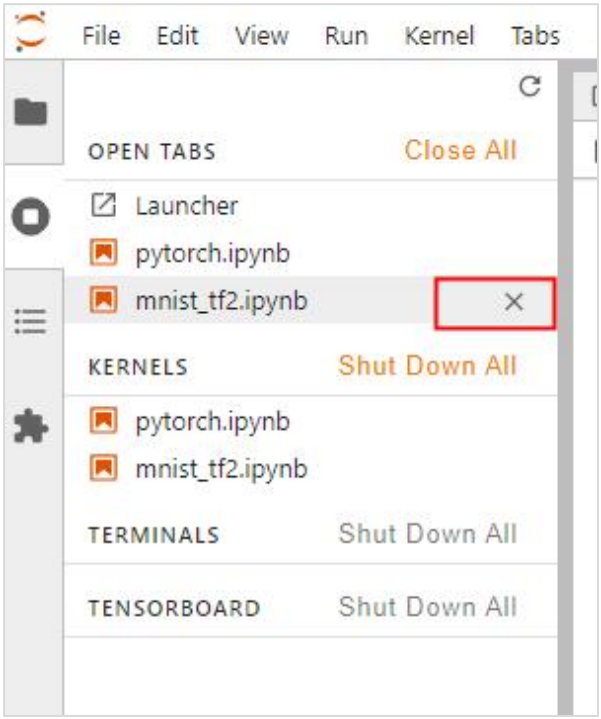
原因：

这类内存或显存满了的错误，原因是正在运行中的案例脚本太多。

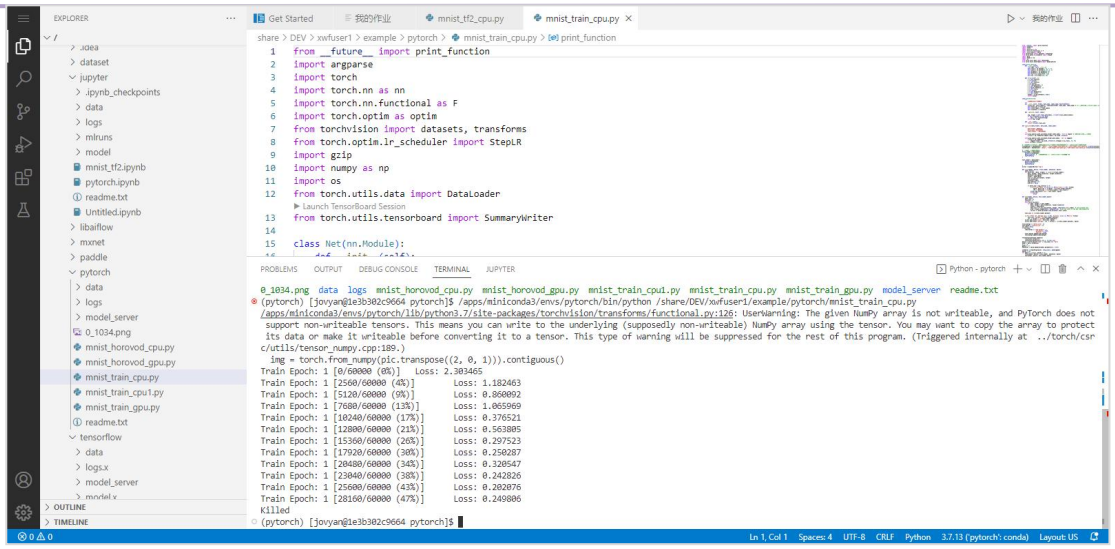
解决方案：

关掉一些正在运行的脚本即可释放内存或显存，操作方法如下：

进入如下页面，检查后台正在运行的脚本，点击“×”关闭不需要运行的脚本。



3）问题：vscode 中运行的程序直接被 Killed。



原因：

可能是内存占满导致。

解决方案：

检查开发环境内存是否用完，释放内存，或者可以尝试切换更高版本内存的硬件规格，或调整程序减少内存使用。

4）**问题：CentOS/Ubuntu 桌面开发环境连不上，提示错误：“会话登录超时！”。**

原因：

可能是内存占满导致。

解决方案：

检查 CentOS/Ubuntu 桌面开发环境内存是否用完，可以等待一段时间，CentOS/Ubuntu 桌面中使用的内存释放后就会恢复正常，或者可以尝试切换更高版本内存的硬件规格。

(15) MindSpore 任务结束自动标密失败

原因：

如果 mindspore 训练代码中使用了 SummaryCollector 记录 summary 信息，保存的 summary 为只读权限，如果使用保存模型 api，保存的模型权限

为只读权限。

解决方案：

自动标密失败时，可以在训练结束后作业数据区手动执行标密操作。

(16) 在服务管理部署的 Triton Inference Server 类型服务，点击“访问服务”，跳转页面并没有显示服务信息，只显示一个空列表。

原因：

模型目录选择有误。

解决方案：

请确保选择的模型目录遵循以下规则：

每个模型目录下都至少包含一个模型版本目录和一个模型配置文件：

- 模型版本目录：包含模型文件，且必须以数字命名，作为模型版本号，数字越大版本越新。
- 模型配置文件：用于提供模型的基础信息，通常命名为 config.pbtxt。

假设模型存储目录在/examplebucket/models/triton/路径下，模型存储目录的格式如下：

```
triton
└──── resnet50_pt
    ├──── 1
    │    └──── model.pt
    ├──── 2
    │    └──── model.pt
    ├──── 3
    │    └──── model.pt
    └──── config.pbtxt
```

参考：

https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/user_guide/model_repository.html

(17) 应用不同型号的 gpu 卡，提交 Pytorch Horovod 并行 gpu 作业，无法正常运行。

问题现象：

提交 Pytorch Horovod 并行 gpu 作业，日志报：“One or more tensors were submitted to be reduced, gathered or broadcasted by subset of ranks and are waiting for remainder of ranks for more than 60 seconds. This may indicate that different ranks are trying to submit different tensors or that only subset of ranks is submitting tensors, which will cause deadlock.”

问题原因：

应用不同型号的 gpu 卡，提交 Pytorch Horovod 并行 gpu 作业。

解决方案：

应用相同型号的 gpu 卡，提交 Pytorch Horovod 并行 gpu 作业，即可正常工作。

(18) 运行单机多卡程序，报：“RuntimeError: NCCL Error 2: unhandled system error (run with NCCL_DEBUG=INFO for details)”

问题原因：

容器中包含多张网卡，可能导致 NCCL 多卡通讯时失败。

解决方案：

运行程序前指定 NCCL_SOCKET_IFNAME 环境变量，指定 NCCL 使用的网卡。

开发环境中可以指定使用 eth1 网卡。

(19) tensorflow 模型转换和 tensorflow serving 模型部署，报 ModuleNotFoundError: No module named 'xxx.export_model'错误

问题原因：

模型目录下存在非模型名称或者数字命名的文件夹，且文件夹下没有 export_model.py 文件，导致加载文件失败。

解决方案：

如果模型为权重文件，则需要在模型目录下创建模型加载文件，文件下包括 export_model.py（格式如下所示），文件加载模型文件并返回模型。如果不需要加载其他额外的文件，建议删除该文件。

```
#export_model.py
def export_model(model_file):
from FaceNet.nets.facenet import facenet
input_shape = [160, 160, 3]
model = facenet(input_shape, backbone='mobilenet', mode="predict")
model.load_weights(model_file, by_name=True)
return model
```

(20) MindSpore 任务输出中有报错信息：STDERR: setfattr: xxxxxx: Permission denied（xxxxxx 为文件路径）

问题原因：

如果 mindspore 训练代码中使用了 SummaryCollector 则会在训练结束后把生成的相关文件修改为只读权限。

解决方案：

训练结束后在作业数据区手动执行标密操作。

(21) 提交 AI 作业（如：mindspore），发现作业正常 pull 镜像时，动态

输出尚未打印日志

问题现象：

提交 AI 作业（如：mindspore），发现作业正常 pull 镜像时，动态输出尚未打印日志。

解决方案：

查看动态日志，通过浏览日志文件开启自动跟踪。

(22) 提交 AI 并行作业（如：提交 tensorflow horovod 并行作业）以及启动 AI 服务，发现动态输出尚未打印日志

问题现象：

提交 AI 并行作业（如：提交 tensorflow horovod 并行作业）以及启动 AI 服务，发现动态输出尚未打印日志。

问题原因：

门户和调度获取日志的方式不同。

解决方案：

查看动态日志，通过浏览日志文件开启自动跟踪。

(23) 将“流体云仿真”模型实例，部署成 triton 服务失败

问题现象：

将“流体云仿真”模型实例，部署成 triton 服务失败

问题原因：

triton 不支持流体云仿真模型中的一些算子。

解决方案：

使用 tensorflow serving 部署该模型。

(24) 节点 GPU 卡缺失，提交的 CPU 作业失败

问题现象：

节点 GPU 人为拔掉，运行在该节点的 CPU 作业退出

问题原因：

某些系统 nvidia docker 不能自动切换为 CPU 模式。

解决方案：

修改/etc/docker/daemon.json 的 default-runtime 配置为 runc。

(25) Docker compose 类型的服务或其他服务作业，启动提示端口冲突

问题现象：

存在 docker compose 服务时，其他服务启动日志提示端口冲突；或者启动 docker compose 服务时日志提示端口冲突。

问题原因：

存在 docker compose 服务时，docker compose 服务的端口未由调度分配，可能导致与调度分配的端口冲突；或者启动 docker compose 服务时，该服务的端口已被其他程序占用。

解决方案：

修改 docker compose 服务的占用端口，使用的端口时不要跟调度配置的端口范围重合，可以避免相互干扰。

(26) GPU 共享模式下，Tensorflow ps 和 Paddlepaddle ps 提交作业运行失败

问题现象：

GPU 共享模式下，Tensorflow ps 和 Paddlepaddle ps 模式提交作业，运行失败，报错：[Hint:'cudaErrorInvalidDevice'. This indicates that

the device ordinal supplied by the user does not correspond to a valid CUD118[1] A device or that the actionrequested is invalid for the specified device.]

问题原因：

Tensorflow 和 paddlepaddle ps 并行运行在 GPU 共享模式下不支持。