

TI-ONE 训练平台 操作指南



腾讯云

【 版权声明 】

©2013–2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【 商标声明 】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【 服务声明 】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。

您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【 联系我们 】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

操作指南

大模型广场

数据中心

数据中心简介

新建数据集

数据集列表操作

创建大模型的数据标注任务

创建传统CV的数据标注任务

数据标注列表操作

传统CV数据标注操作界面说明

标签格式示例说明

数据构建

任务式建模

任务式建模简介

创建任务

任务管理

分布式训练使用指引

任务事件告警配置指引

开发机

开发机简介

创建实例

管理实例

SSH 网络配置指引

闲置回收策略配置指引

使用生命周期脚本

开发机内使用镜像工具

Git 存储库

模型管理

模型管理简介

模型仓库使用

模型优化使用

模型包规范

模型评测

模型评测简介

轻量体验

客观评测

主观评测

评测可视化对比

评测集格式要求

内置的开源评测集介绍

在线服务

在线服务简介

在线服务部署

在线服务调用

在线服务鉴权和限流

在线服务运营

使用平台内置镜像发布在线服务开发指引

使用自定义镜像发布在线服务开发指引

模型推理文件简介和示例

批量预测

批量预测介绍

批量预测任务发布指引

资源组管理

资源组简介

调度策略说明

GPU 虚拟化

Tikit

Tikit 简介

安装与初始化

查看资源组资源

使用数据集

数据上传与下载

提交训练任务

开发机创建与管理

注册模型

模型优化

发布服务

提交跑批任务

操作指南

大模型广场

最近更新时间：2025-06-19 14:22:52

概述

大模型广场 是 **腾讯云 TI 平台** 的内置大模型库，预置多种预训练大模型及指令微调大模型，覆盖各类下游任务，如多轮对话、逻辑推理、内容创作等。用户可一键将内置大模型部署为在线服务，通过网页问答快速直观地体验大模型的效果；也可一键发起基于内置大模型的精调任务，将模型优化为可满足垂直场景需求的生产级模型。

内置大模型

大模型广场已内置如下大模型，最新支持的模型清单以大模型广场页面展示为准。

- **腾讯自研混元大模型**：包括开源的 **Hunyuan-Large** 系列，以及闭源版本。
 - 精选通用大模型：包括主流的开源大模型，如 **DeepSeek** 系列、**Llama4** 系列、**Llama3.2** 系列、**Llama3.1** 系列、**Llama3** 系列、**Qwen3** 系列、**Baichuan2** 系列等。
- **其他**：如腾讯自研行业大模型。

以上混元大模型的闭源版本以及行业大模型，请联系腾讯云行业销售进行服务开通。

功能说明

大模型广场提供以下主要功能。

浏览内置大模型

在大模型广场页面，以卡片形式展示模型清单。

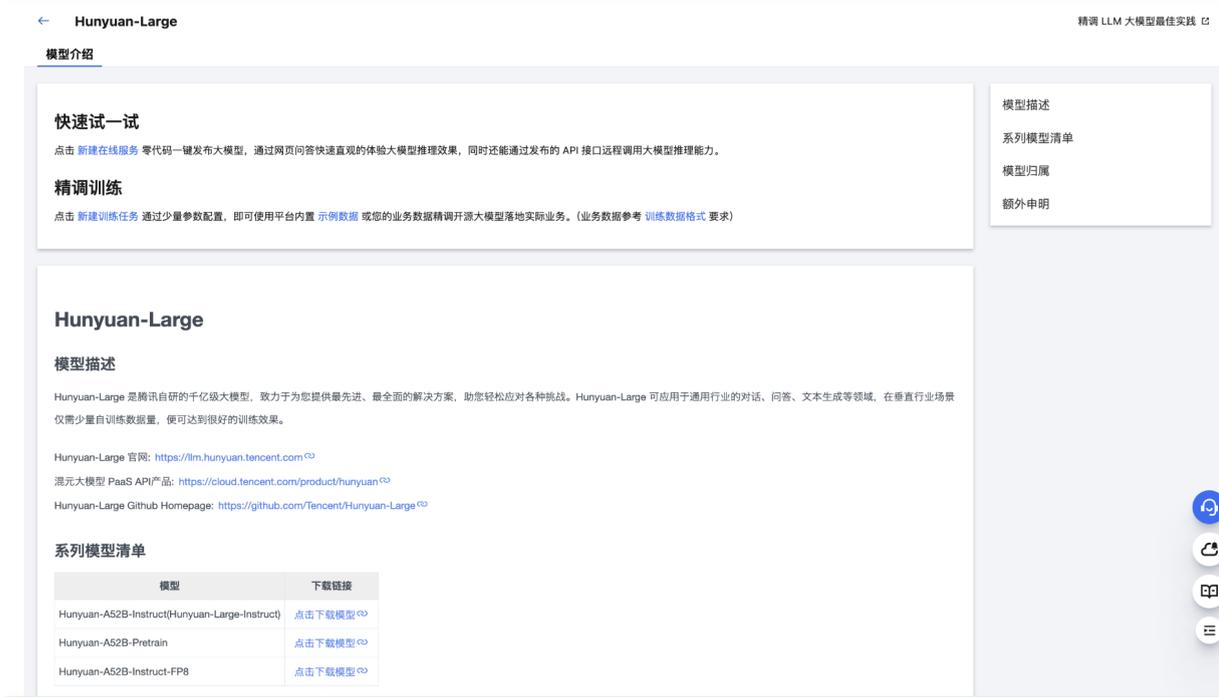
- **模型标签**：按任务类型（如文本分类、翻译、问答等）、语言（如中文、英文）、框架（如 PyTorch）等维度对模型进行分类，以标签形式展示，方便快捷分类查找。
- **模型搜索**：支持基于关键字的搜索功能，用户可以通过模型名称进行模糊查找。

大模型广场-模型卡片页

发起部署和精调

单击广场中的卡片，可进入模型详情页，展示模型的更多详细信息，并支持快速试一试、精调训练等功能。

- **模型介绍**：包括模型描述、系列模型清单、模型归属等信息。
- **快速试一试**：单击**新建在线服务**，可跳转至 **在线服务** 模块，一键发起模型部署。
- **精调训练**：单击**新建训练任务**，可跳转至 **任务式建模** 模块，一键发起精调训练。模型精调可显著提升模型在特定场景和任务中的性能，从而进一步满足生产级需求。



大模型广场-模型详情页

最佳实践

针对内置大模型，TI 平台提供了丰富的最佳实践文档，帮助用户快速上手模型部署和精调。

- **腾讯混元大模型系列**：
 - [Hunyuan-Large x TI 上手指南](#)
 - [精调 Hunyuan-Large 实践教程](#)
- **DeepSeek 大模型部署**：[快速部署和体验 DeepSeek 系列模型](#)
- **内置大模型部署**：[快速部署和体验内置开源 LLM 大模型](#)
- **内置大模型精调**：[精调内置开源大模型](#)

数据中心

数据中心简介

最近更新时间：2024-11-12 11:34:11

数据中心作为机器学习平台的基础数据服务模块，提供以下主要功能：

数据集管理

- 统一管理当前主账号下的所有数据集，数据集列表按照地域进行隔离；
- 数据集支持的建模任务类型有：
 - 大模型建模数据集：支持大语言模型 LLM 数据集（jsonl）和多模态大模型 MLLM 数据集（jsonl）；
 - 传统 CV 建模数据集：图片（jpg、png、jpeg、bmp）；
 - 大数据集建模数据集：表格（xls、xlsx、csv）；
 - 其他：任意格式文件；
- 支持对数据内容进行详情预览，且支持对文本数据进行样本词频分析和样本长度分析等。

数据标注

- 提供 LLM&MLLM 大模型建模的数据标注操作台：利用 schema 灵活定义标注操作台组件；
- 提供内置的“图片分类”标注场景：识别数据集中每张图片所属的标注类别，如猫、狗图片分类标注；
- 提供内置的“目标检测”标注场景：识别数据集中每张图片里指定物体的所在位置及其类别，如画框检测图片中的猫、狗标注任务；
- 提供内置的“目标跟踪”标注场景：对数据集中连续的多张图片里的某个物体进行跟踪标记；
- 提供内置的“图片分割”标注场景：识别数据集中每张图片里指定标注类别的物体的所在区域；
- 提供内置的“OCR”标注场景：支持对图片的文字内容进行识别/检测，或者智能结构化的标注。

数据构建

- 提供最佳实践文档指导用户进行前置数据预处理；
- 提供多种内置算法任务，每个任务都有高质量的 prompt 和内置自研数据；
- 提供四种内置的数据构建 pipeline：有监督-单轮 QA-pipeline、有监督-多轮 QA-pipeline、无监督-pipeline；
- 提供多种数据处理功能：数据清洗、prompt 优化、数据过滤、数据增强等。

新建数据集

最近更新时间：2024-11-12 11:34:11

概述

数据中心支持新建/管理的用户数据集来源有：[腾讯云文件存储 CFS](#) 上的大模型类数据文件，以及 [腾讯云对象存储 COS](#) 上的图片/表格/其他类数据文件。所以整个数据中心模块对大模型建模数据集和传统建模数据集是分为了两套不同的产品逻辑分别进行支持。

⚠ 注意

1. COS 为对象存储产品，独立计费，详细可见 [对象存储-计费概述](#)。
2. CFS 为文件存储场景，独立计费，详细可见 [文件存储-价格总览](#)。
3. 同一个公有云账号在同一个地域下最多可在数据中心同时维护 200 个数据集，若您的业务有扩容需求，可使用工单联系我们哦。

大模型建模数据集

大模型建模类型（LLM&MLLM）的数据集在新建时选择“建模任务类型-大模型建模”，只需要用户填写一条数据文件所在的 CFS 路径信息即可一键创建。数据中心仅仅是将该数据集关联到用户的 cfs 路径上，数据中心并不会将用户的原始数据文件进行复制、转存。且用户在 TI 平台对该大模型数据集进行标注时，标注结果会直接且实时的写到用户数据集的原始文件中。所以，若不希望原始文件被修改，请用户提前完成原始文件备份。

腾讯云 TI 平台

← 新建数据集

数据集名称 • 大模型数据集 ✓
请输入不超过60个字符，仅支持中英文、数字、下划线"_"、短横"-"，只能以中英文、数字开头

地域 • 广州

标签 ①
标签键 标签值 ✖
[+ 添加](#) [🔄 键值粘贴板](#)

建模任务类型 • 大模型建模-LLM

基础存储服务 • CFS

CFS文件系统 • demo_cfs (cfs-1fz79cpf) [CFS控制台](#)

数据集CFS路径 • /1021demo/sft/
⚠ 请注意：在您通过TI平台对该数据集进行标注时，标注结果会直接写到该数据集的原始文件中。若您不希望原始文件被修改，请提前完成原始文件备份。

数据集标签
输入后请按回车确认。支持输入多个任务场景，示例输入：阅读理解-通用

[确定](#) [取消](#)

- 数据集名称：输入自定义数据集名称；
- 建模任务类型：如选择“大模型建模-LLM”，该参数仅作为任务类型的一个区分标记，不会对后续的产品流程产生严格的操作逻辑上的区别；
- 数据集 CFS 路径：请定位到您的数据文件所在的 CFS 路径，该路径不需要精准到 jsonl 文件，只需要精准到 jsonl 文件所在的目录层级即可；
- 数据集标签：为了便于用户更好的分类管理大量的数据集，支持用户在此灵活自由的给该数据集打上标签信息，后续数据中心支持通过标签对数据集进行检索。

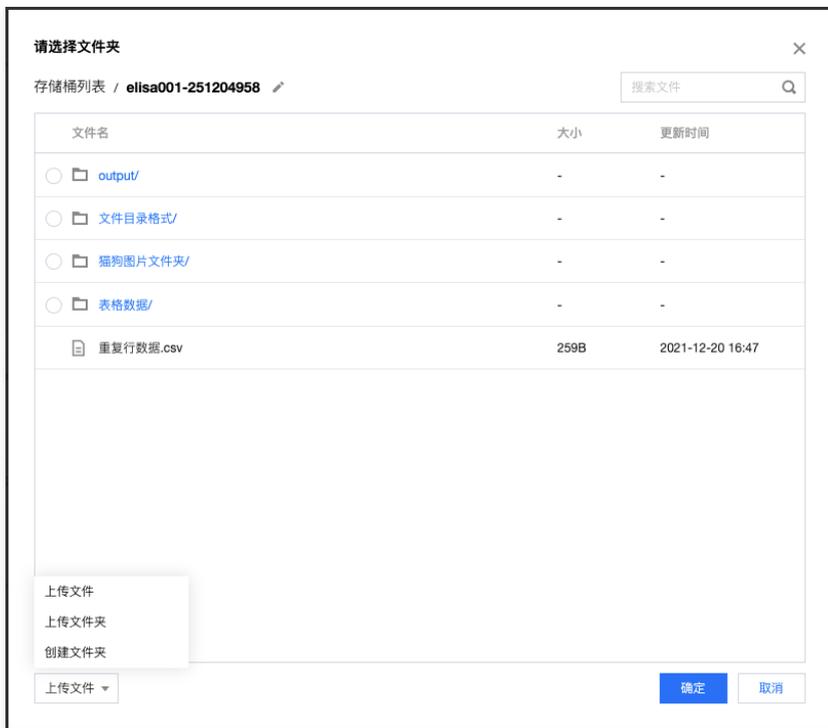
传统CV建模数据集

新建未标注的图片数据集

若您的图片数据没有相关的标注信息，则新建数据集时，配置如下：



- 数据集名称：输入自定义数据集名称；
- 地域：数据集所在地域前端默认为您当前控制台所在地域；
- 标签：该参数是复用了 [腾讯云【标签】产品](#)，用户可给数据集打上不同的标签值；
- 建模任务类型：选择“传统CV建模”，然后依据您不同的图像场景对应选择“图像分类、图像检测、图像分割、图像目标跟踪、OCR”；
- 标注状态：选择“未标注”；
- 数据文件 COS 路径：指定选择图片数据所在的 [腾讯云对象存储 COS 路径](#) 即可。若您的图片文件还在本地未上传到 COS 路径下，则单击[选择文件](#)按钮后，选中某个存储桶，然后在弹框左下角选择上传文件等功能即可完成本地数据的上传：



- COS 存储路径：该数据集后续在 TI 平台上可能产生的标注信息（利用该数据集创建了数据标注任务）、以及该数据集每个版本的数据信息/快照记录等文件将存储到该 COS 存储地址下。注意：该地址不能和“数据文件COS路径”参数配置相同；
- 数据集导入成功后，TI 平台支持在前端页面查看数据集详情，即可预览图片内容。

新建已标注的图片数据集

若您的图片数据中含有相关的标注信息，则导入数据集时，配置如下：

The screenshot shows the 'New Dataset' configuration interface. The left sidebar contains navigation options like '大模型广场', '数据中心', '数据集管理', '数据标注', '数据构建', '训练工坊', '模型管理', '模型服务', and '资源组管理'. The main area is titled '新建数据集' and includes the following fields:

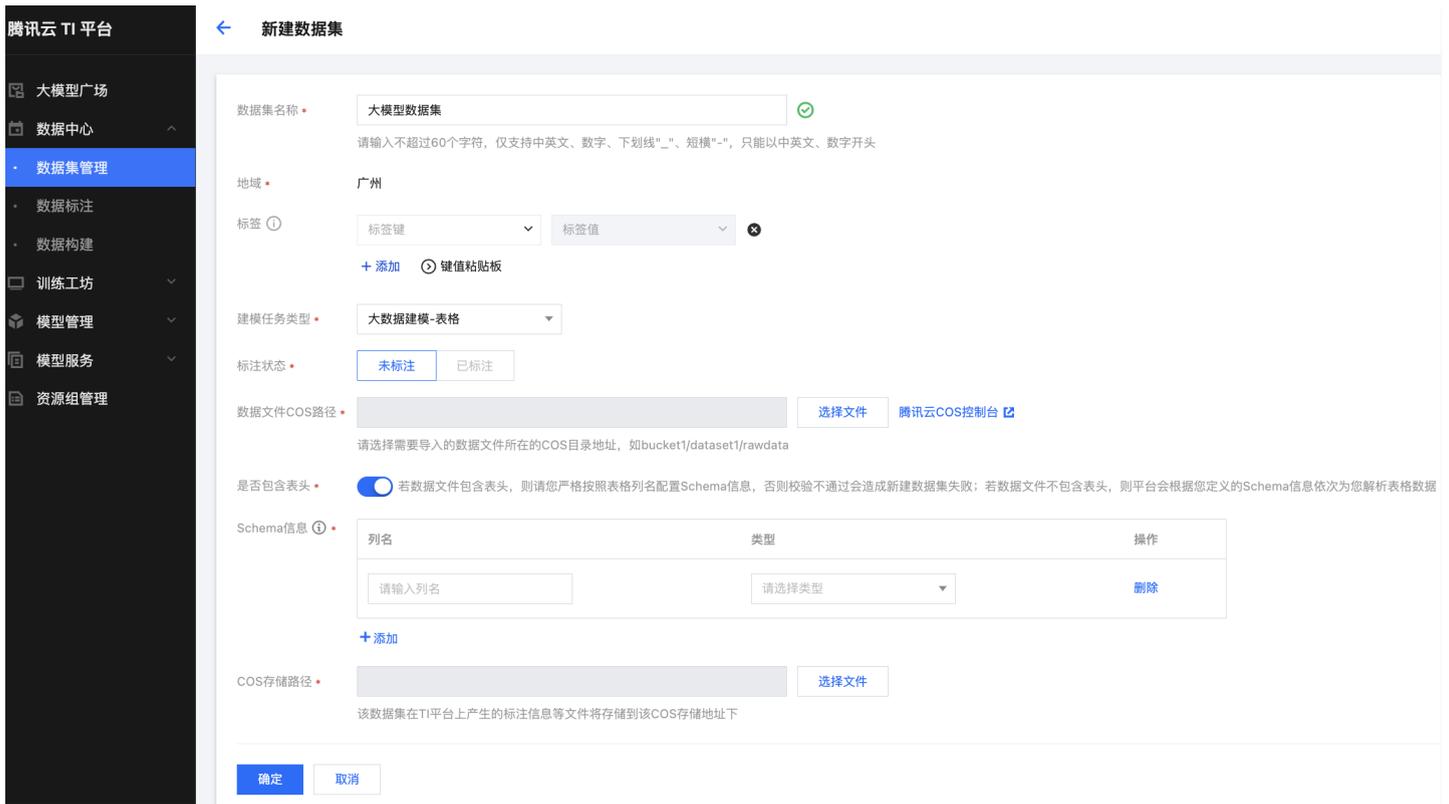
- 数据集名称:** 大模型数据集 (with a green checkmark icon and a note: 请输入不超过60个字符, 仅支持中英文、数字、下划线"_"、短横"-", 只能以中英文、数字开头)
- 地域:** 广州
- 标签:** 标签键 (dropdown) and 标签值 (dropdown) with a '+ 添加' and '键值粘贴板' button.
- 建模任务类型:** 传统CV建模-图像分类 (dropdown)
- 标注状态:** 未标注 (disabled) and 已标注 (active)
- 标注格式:** TI平台格式 (dropdown) with a '下载标注文件示例' link.
- 文件存放方式示意图:** A diagram showing a folder 'Dataset_Filename (请选择此处文件夹)' containing files '1.jpg', '2.jpg', '3.jpg', and 'annotation.txt (请保持文件名称一致)'.
- 数据文件COS路径:** Text input field with a '选择文件' button and a link to '腾讯云COS控制台'.
- COS存储路径:** Text input field with a '选择文件' button and a note: '该数据集在TI平台上产生的标注信息等文件将存储到该COS存储地址下'.

At the bottom, there are '确定' (Confirm) and '取消' (Cancel) buttons.

- 数据集名称：输入自定义数据集名称；
- 地域：数据集所在地域前端默认为您当前控制台所在地域；
- 标签：该参数是复用了 [腾讯云【标签】产品](#)，用户可给数据集打上不同的标签值；
- 建模任务类型：选择“传统CV建模”，然后依据您不同的图像场景对应选择“图像分类、图像检测、图像分割、图像目标跟踪、OCR”；
- 标注状态：选择“已标注”；
- 标注格式：不同的图像场景有不同的标注格式：
 - 依据不同的标注格式，请按照对应的“文件存放示意图”的说明准备您的文件目录；
 - 单击[下载标注文件示例](#) 下载查看标注文件的具体格式内容；
- 数据文件COS路径：指定选择图片数据所在的 [腾讯云对象存储 COS 路径](#) 即可；
- COS 存储路径：该数据集后续在 TI 平台上可能产生的标注信息（利用该数据集创建了数据标注任务）等文件将存储到该 COS 存储地址下。注意：该地址不能和“数据文件COS路径”参数配置相同；
- 数据集导入成功后，TI 平台支持在前端页面查看数据集详情，即可预览图片内容，且可同时展示数据集标注信息的可视化分布图。

大数据建模数据集

若您要导入“大数据建模-表格”类型的数据集时，则配置如下：



- 数据集名称：输入自定义数据集名称；
- 地域：数据集所在地域前端默认为您当前控制台所在地域；
- 标签：该参数是复用了 [腾讯云【标签】产品](#)，用户可给数据集打上不同的标签值；
- 建模任务类型：选择“大数据建模-表格”；
- 标注状态：选择“未标注”，表格类型的数据集不支持上传已标注的数据文件；
- 数据文件 COS 路径：指定选择表格数据所在的 [腾讯云对象存储 COS 路径](#) 即可；
 - 若数据文件包含表头，则请打开“是否包含表头”的按钮，并严格按照表格列名配置 Schema 信息，否则校验不通过会造成导入数据集失败；若数据文件不包含表头，则关闭“是否包含表”按钮，则平台会根据您自定义输入的 Schema 信息依次为您解析表格数据；
- COS 存储路径：该数据集在 TI 平台产生的一些元信息记录文件将存储到该 COS 存储地址下。注意：该地址不能和“数据文件 COS 路径”参数配置相同；
- 数据集导入成功后，TI 平台支持在前端页面查看数据集详情，即可预览表格数据内容。且 TI 平台支持对可枚举列进行分布统计展示。

其他数据集

若您的文件后缀不满足平台内置数据格式的要求，或者您无须 TI 平台为您展示数据详情内容，仅需要管理一条数据路径，则在新建数据集时，“建模任务类型”选择“其他”，配置如下：

腾讯云 TI 平台

大模型广场

数据中心

数据管理

数据标注

数据构建

训练工坊

模型管理

模型服务

资源组管理

←

新建数据集

数据集名称 * ✔

请输入不超过60个字符，仅支持中英文、数字、下划线"_"、短横"-", 只能以中英文、数字开头

地域 * 广州

标签 ⓘ 标签键 ▼ 标签值 ▼ ✕

+ 添加 📄 键值粘贴板

建模任务类型 * 其他 ▼

标注状态 * 未标注 已标注

数据文件COS路径 * 选择文件 [腾讯云COS控制台](#)

请选择需要导入的数据文件所在的COS目录地址，如bucket1/dataset1/rawdata

COS存储路径 * 选择文件

该数据集在TI平台上产生的标注信息等文件将存储到该COS存储地址下

确定
取消

- 数据集名称：输入自定义数据集名称；
- 地域：数据集所在地域前端默认为您当前控制台所在地域；
- 标签：该参数是复用了 [腾讯云【标签】产品](#)，用户可给数据集打上不同的标签值；
- 建模任务类型：选择“其他”；
- 标注状态：仅支持选择“未标注”；
- 数据文件 COS 路径：指定选择您的数据文件所在的 [腾讯云对象存储 COS 路径](#) 即可；
- COS 输出路径：该数据集在 TI 平台产生的一些元信息记录文件将存储到该 COS 存储地址下。注意：该地址不能和“数据文件 COS 路径”参数配置相同；
- 数据集导入成功后，TI 平台不支持在前端页面查看数据集的详情内容。

数据集列表操作

最近更新時間：2024-11-13 16:02:32

概述

数据集列表页面统一管理了该用户账号下所有数据集的集合，展示了数据集的关键信息，支持“标注、删除、配置schema”操作，且支持用户单击数据集名称进入查看数据集详情。

列表字段

数据集列表页面展示字段及其含义解释如下：

名称	标注进度/数据集总量	建模任务类型	数据集标签	标签	状态	创建时间	操作
传统图片数据集	未开始/20张图片	传统CV建模	图像分类		可用	2024-11-07 16:28:29	标注 删除
无监督文本清洗	0/1条样本	大模型建模-LLM	去除页码 新闻通稿 无监督		可用 查看进度	2024-09-19 19:30:09	标注 配置Schema 删除
筛选高质量文本问答对	0/398条样本	大模型建模-LLM	单轮问答 新闻 金融		可用 查看进度	2024-09-19 19:28:53	标注 配置Schema 删除
单图结构化信息提取	0/1条样本	大模型建模-MLLM	ocr 发票		可用 查看进度	2024-09-19 19:26:39	标注 配置Schema 删除
单图自由文本问答	0/1条样本	大模型建模-MLLM	图片多个问答对 自然场景		可用 查看进度	2024-09-19 19:25:33	标注 配置Schema 删除
多模态数据集高质量改写	0/1条样本	大模型建模-MLLM	保留原始答案 自然问答		可用 查看进度	2024-09-19 19:22:20	标注 配置Schema 删除
多图多轮问答	0/50条样本	大模型建模-MLLM	多图对比 多轮问答		可用 查看进度	2024-09-19 19:21:22	标注 配置Schema 删除
多模态图文问答竞品评测	1/5条样本	大模型建模-LLM	模型评测		可用 查看进度	2024-09-19 19:11:58	标注 配置Schema 删除

- 名称：展示数据集的自定义名称，且支持单击名称进入查看该数据集的详情页面；
- 标注进度/数据集总量：
 - 大模型建模数据集：按照 schema 的解析规则，一条完整的样本计数为1，单位：条；在标注操作台有过标注动作的样本会被计为“已标注”；
 - 传统CV建模数据集（图片）：统计路径下所有图片张数，单位：张；有标注结果的图片计为“已标注”；
 - 大数据建模数据集（表格）：统计行数，单位：行；
 - 其他：统计路径下的所有文件个数，单位：个；
- 建模任务类型：用户新建数据集时的自定义参数；
- 数据集标签：
 - 大模型建模数据集：用户在新建数据集配置页面自定义输入的标签内容；
 - 传统CV建模数据集：按照该数据集所属的图像场景默认打标，枚举值有：“图像分类、图像检测、图像分割、图像目标跟踪、OCR”；
- 标签：展示在数据集导入时，用户选择的腾讯云 CAM “标签”；
- 状态：
 - 大模型建模数据集
 - 新建成功后，即进入“可用”状态，但需要完成“操作-配置schema”后才可进一步查看数据集详情和进行标注操作。
 - 传统CV建模数据集
 - 导入 XX%：用百分比的形式实时展示数据集从用户单击导入页面的确定按钮到数据集成功导入的进度；
 - 可用：一旦数据集导入成功或者同步数据源成功，则转为“可用”状态；“标注”操作失败、“下载”操作失败、“发布”新版本操作失败时，该数据集依然保留为“可用”状态；若“同步数据源”操作失败，则数据集转为“失败”状态；
 - 失败：支持悬浮展示导入失败/同步数据源失败的详细原因；
 - 不可用：当数据集处于删除中操作时数据集都为“不可用”状态；
- 创建时间：记录数据集创建的时间，支持用户点选顺序、倒叙的排序展示；
- 操作：以下详细介绍操作功能。

操作-标注

支持用户利用该数据集一键点击创建数据标注任务：



- 若是大模型建模数据集，则单击 **确定** 后，平台会直接为用户创建对应的标注任务，标注任务创建好后会直接跳转到标注操作台。
- 若是传统CV建模数据集，则单击 **确定** 后，在当前页面内跳转到“数据中心-数据标注-新建标注任务”的配置页面，同时会默认选中该数据集，且不支持用户修改。

注意

- 一个数据集的仅支持同时创建一个标注任务；
- 只有对该数据集对应的 COS/CFS 路径有写权限的用户才能利用该数据集创建数据标注任务。

操作-删除

若是大模型数据集：则删除该数据集不会影响原始存储在 CFS 上的数据文件，仅仅是将该条数据集记录从 TI 平台删除。



若是非大模型数据集：

- 后台从 TI 平台解除对应 COS 路径和该数据集的绑定操作，删除该条数据集记录；
- （用户可勾选）在删除数据集的同时，后台自动清理该数据集定义的输出路径下的 COS 存储桶里面的文件，仅自动清理输出路径下的文件，不清理用户原本输入路径下的文件。



操作-配置 schema

该操作仅针对大模型建模数据集，支持用户通过自定义配置数据集的 schema 信息来定义各种复杂的 LLM&MLLM 数据内容。schema 配置规则请参考 schema 配置详细语法。配置页面左侧是平台展示的用户原始数据部分内容，便于用户参考左侧改写右侧的 schema 配置。且支持用户单击 下一步：预览标注工作台 按钮实时查看 schema 的配置效果以确保配置符合预期。若用户发现标注操作台不符合预期可以再单击返回 上一步：基础信息 进行配置修改。

数据集的 schema 配置成功后，页面会自动跳回数据集列表，此时平台会依据您配置的 schema 信息解析数据文件中的全量样本，您可单击列表页面的状态 > 查看进度 按钮实时查看全量解析进度。

数据集详情页面 大模型数据集

单击大模型数据集名称，可进入查看数据集详情内容，且展示的详情内容是按照用户配置的 schema 进行解析后的展示。如针对多模态数据集，会直接将图片和对应的文本内容配对在同一行进行展示，提高数据样本的可读性。

 <p>tione-upload-...</p>	<p>这三张图片的拍摄角度是一样的吗</p>	<p>不一样。第一张图片是单张图片的局部拍摄，第二张图片和第三张图片是一片花海的稍微全局的拍摄。而且第一张图片是清晰的拍摄手法，第二张和第三张有虚实结合的拍摄手法</p> <p>去标注</p>
 <p>tione-upload-...</p>  <p>tione-upload-...</p>	<p>这三张图片拍摄是同一个物品吗?</p>	<p>不是，可以看出第一张图片拍摄的是一朵花的局部图像，第二张图片拍摄的是同一种花的一大片花海，有可能第二张图片中的这一种花和第一张图片中的花是同一种类型，但也存在不确定性。但可以确定的是，第三张图片的花品种和第一张、第二张图片的品种都是不一样的，第三张图片是有两种品种的花一起拍摄，而且图片前景背景有做虚实处理</p> <p>去标注</p>

传统CV数据集

注意

针对数据集详情页面的“详情预览”和“数据透视”的功能，腾讯公有云 COS 为每个账号每个月免费提供 10T 大小的详情预览流量，超出部分的流量将产生计费，详情参考 [COS 处理计费规则](#)。

单击图片数据集名称，可进入查看数据集详情内容。详情页包含三大部分内容：

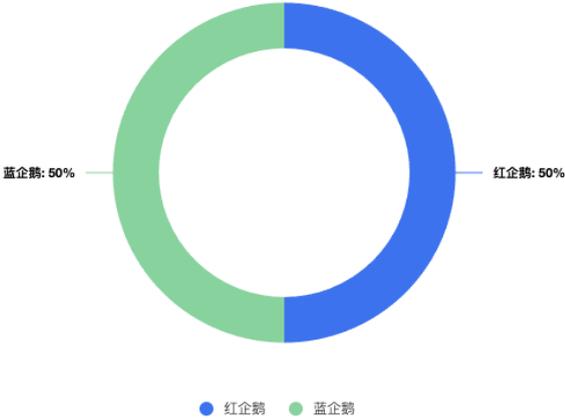
- 基本信息：该部分展示数据集的关键信息。
- 标注信息可视化：若当前数据集绑定了“图片分类/目标检测/图片分割”的标注信息，则展示该模块（其余场景下，该模块自动隐藏）。注意：后台能统计的标签值的数量上限为 20 个（按照占比排序的 Top 20 展示），超出 20 个的其他类别，都统一收归到“其他”类别中。

- 详情展示：该部分支持用户预览数据集中前 2000 张图片内容列表。且支持按标注状态筛选和按指定标注类别筛选进行展示。

基本信息

数据集名称	企鹅照片	数据集ID	392sfxlgtim81j
版本	v1	数据集类型	图片
数据文件大小	13.77 M	数量	2张
数据存储COS路径	elisa001-251204958/企鹅照片/	输出存储COS路径	elisa001-251204958/output/
标签		创建时间	2021-12-31 14:26:41

标注信息可视化 ①



标签名称	标签数量	标签占比
红企鹅	1	50.00%
蓝企鹅	1	50.00%

详情展示

全部(2)

未标注(0)

已标注(2)

标注类别选择 请选择 ▾




共 2 条

10 条 / 页
1 / 1 页

大数据表格数据集

单击表格数据集名称，可进入查看数据集详情内容。详情页包含两大部分内容：

- 基本信息：该部分展示数据集的关键信息。
- 详情展示：该部分支持用户预览数据集中前 2000 行的表格内容，且针对可枚举的列支持单击查看列数据分布信息（统计分析维度是整个数据集，不仅是前端预览的 2000 行内容）。

基本信息

数据集名称	scott-test-csv	数据集ID	ds-5vtqtjlc
版本	v1	数据集类型	表格
数据文件大小	568.00 B	数量	18行
数据存储COS路径	scott-test-sh-1256580188/ti-cloud-data-center/input/table/aaa.csv	输出存储COS路径	scott-test-sh-1256580188/ti-cloud-data-center/output/scott-test-csv-v1-ds-5vtqtjlc/
标签		创建时间	2022-09-29 15:10:31

详情展示

详情展示功能开启后会对COS产生请求及流量费用，COS 每月为用户提供 10T 大小的免费额度，[COS计费说明](#) [查看COS处理已用流量](#) ✕
 详情展示功能开启后仅支持预览前2000行的数据内容，但每列数据的分布统计功能会基于全量数据集进行计算

详情展示

初始运维部门ID :三	初始运维部门 :三	一级业务ID :三	一级业务 :三	二级业务ID :三	二级业务 :三
1	1a	1	1b	1	1c
2	2a	2	2b	2	2c
3	3a	3	3b	3	3c
4	4a	4	4b	4	4c
5	5a	5	5b	5	5c
6	6a	6	6b	6	6c
7	7a	7	7b	7	7c
8	8a	8	8b	8	8c
9	9a	9	9b	9	9c
10	10a	10	10b	10	10c

10 条 / 页 1 / 2 页

其他类型数据集

单击其他类型的数据集名称，进入的数据集详情页仅展示数据集的基本信息，由于该类型的数据集导入格式不做任何限制所以详情页不支持内容预览。

创建大模型的数据标注任务

最近更新時間：2024-12-03 17:21:53

概述

腾讯云 TI 平台支持针对 LLM&MLLM 大模型数据集进行标注，支持用户自定义数据集的 schema 信息来灵活搭建定制化的标注操作台。

新建标注任务

在**数据中心 > 数据集管理**模块，选择已新建好的大模型数据集，单击“操作-标注”按钮，后台会自动依据该数据集的 schema 配置信息创建对应的标注操作台。

⚠️ 重要的注意事项：

- 数据中心仅仅是将该数据集关联到用户的 cfs 路径上，数据中心并不会将用户的原始数据文件进行复制、转存。
- 用户在 TI 平台对该数据集进行标注时，标注结果会直接且实时的写到您数据集的原始文件中。所以，若您不希望原始文件被修改，请提前完成原始文件备份。

名称	标注进度/数据集总量	建模任务类型	数据集标签	标签	状态	创建时间	操作
无监督文本清洗	0/1条样本	大模型建模-LLM	去除页码 新闻通稿 无监督	◇	可用 查看进度	2024-09-19 19:30:09	标注 配置Schema 删除
筛选高质量文本问答对	0/398条样本	大模型建模-LLM	单轮问答 新闻 金融	◇	可用 查看进度	2024-09-19 19:28:53	标注 配置Schema 删除
单图结构化信息提取	0/1条样本	大模型建模-MLLM	ocr 发票	◇	可用 查看进度	2024-09-19 19:26:39	标注 配置Schema 删除
单图自由文本问答	0/1条样本	大模型建模-MLLM	图片多个问答对 自然场景	◇	可用 查看进度	2024-09-19 19:25:33	标注 配置Schema 删除

大模型标注操作台

TI 平台灵活的 schema 能支持的标注场景包括但不限于：高质量文本问答对筛选、文本数据清洗、图片问答对审核/修改、图片问答竞品评测、图片多轮问答、多模态阅读理解图片文本描述等。

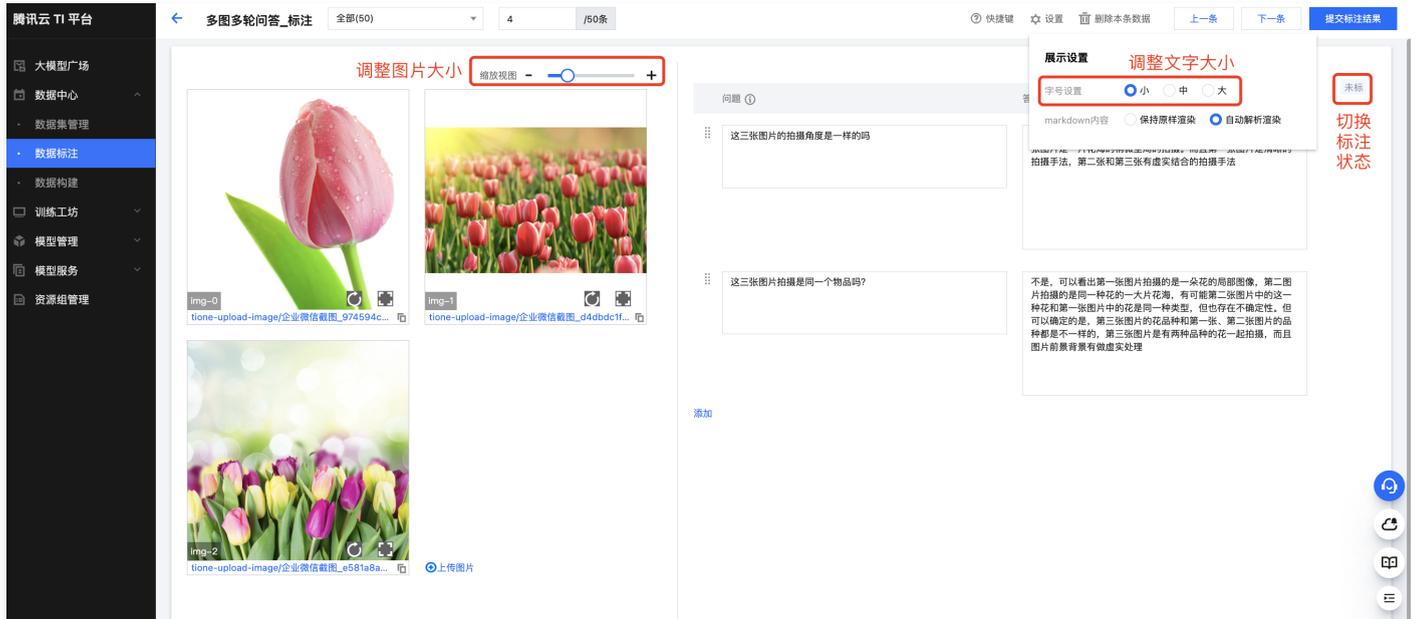
以下给出 3 个不同 schema 配置对应的标注操作台示例：

1. 针对多图的多轮问答

```
desc: 多图多轮问答
record_fields:
- name: img # 标注工作台展示的标注组件名称
  key: img # 导出标注结果的json字段名称
  type: ImageListInput # 组件类型
  help: "请添加字段描述" # 组件帮助说明
  value:
    {{- range .Values.img }} # 使用循环引用图片列表
    - {{ . }}
    {{- end }}
- name: target # 标注工作台展示的标注组件名称
  key: target # 导出标注结果的json字段名称
  type: List # 组件类型
  help: "请添加字段描述" # 组件帮助说明
  value:
    {{- range .Values.target }} # 使用循环展开List列表内容
    -
- name: 问题 # 标注工作台展示的标注组件名称
  key: question # 导出标注结果的json字段名称
  type: TextInput # 组件类型
  help: "请添加字段描述" # 组件帮助说明
```

```

value: "{{ .question }}"
size: MultiLine
- name: 答案 # 标注工作台展示的标注组件名称
key: answer # 导出标注结果的json字段名称
type: TextInput # 组件类型
help: "请添加字段描述" # 组件帮助说明
value: "{{ .answer }}"
size: LongArticle
{{- end }}
    
```



在该场景下，可以配置 schema 实现多张图片的展示、问题和答案问答对配对展示，同时支持对问答对进行删除和添加，也支持调整问答对的先后顺序；同时用户还可以依据“问题、答案”两个字段不同的文本长度设置不同的输入框大小。

标注操作台功能细节：

- 可单击左侧图片区域上方的“缩放视图”调整图片展示大小。
- 可单击标题栏处右侧的“设置”按钮，设置字号大小以及文本中的 markdown 内容是否要展示。



- 可单击切换标注状态“未标/已标”；若当前样本有标注内容的修改，则后台会自动将状态流转为“已标”，若当前样本无任何修改，用户也可以手动单击“未标”按钮来切换成“已标”状态。

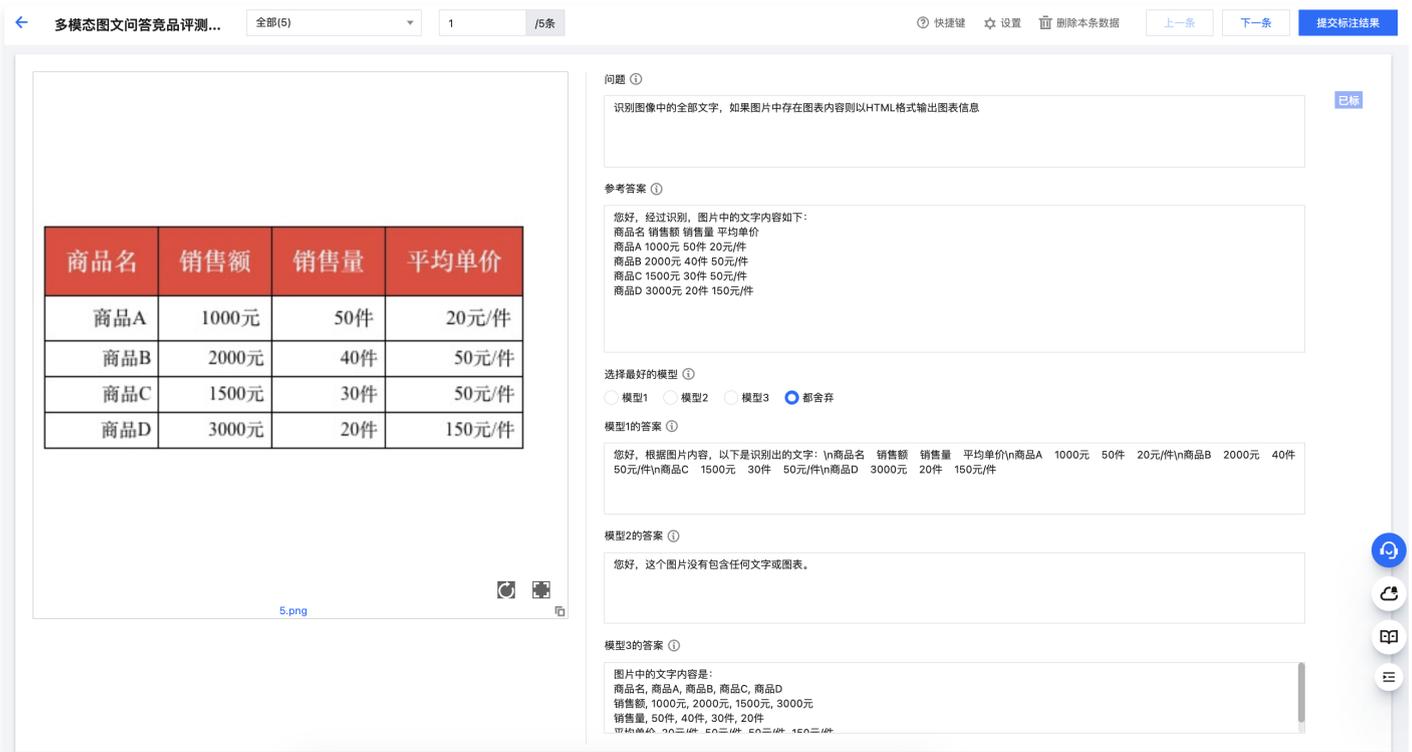
2. 针对单图的多模型评测

```

desc: 自动生成的YAML
record_fields:
- name: 图片 # 标注工作台展示的标注组件名称
key: Images # 导出标注结果的json字段名称(支持英文字母下划线, 不能以数字开头)
type: ImageViewer # 组件类型
    
```

```

help: "请添加字段描述" # 组件帮助说明
value: "{{ .Values.Images }}"
- name: 问题 # 标注工作台展示的标注组件名称
key: Query # 导出标注结果的json字段名称 (支持英文字母下划线, 不能以数字开头)
type: TextInput # 组件类型
help: "请添加字段描述" # 组件帮助说明
value: "{{ .Values.Query }}"
size: MultiLine
- name: 参考答案 # 标注工作台展示的标注组件名称
key: sn_v1_0_6_0_10b_8k_beta_0624 # 导出标注结果的json字段名称 (支持英文字母下划线, 不能以数字开头)
type: TextInput # 组件类型
help: "请添加字段描述" # 组件帮助说明
value: "{{ .Values.gtmodel }}"
size: LongArticle
- name: 选择最好的模型 # 标注工作台展示的标注组件名称
key: correct_model # 导出json标注结果时候该组件对应的json字段key
type: StringSelector # 表明该组件类型是字符串选择组件
option: SingleSelector # 表明该组件是多选。字段取值范围: SingleSelector/MultiSelector
help: 模型评测 # 组件帮助说明
choices: # 指定选项的内容
- 模型1
- 模型2
- 模型3
- 都舍弃
- name: 模型1的答案 # 标注工作台展示的标注组件名称
key: kzx2npurd5 # 导出标注结果的json字段名称 (支持英文字母下划线, 不能以数字开头)
type: TextInput # 组件类型
help: "请添加字段描述" # 组件帮助说明
value: '{{ index .Values.model1 }}'
size: MultiLine
- name: 模型2的答案 # 标注工作台展示的标注组件名称
key: FT_qw15_sft_0626_v100_800 # 导出标注结果的json字段名称 (支持英文字母下划线, 不能以数字开头)
type: TextInput # 组件类型
help: "请添加字段描述" # 组件帮助说明
value: "{{ .Values.model2 }}"
size: MultiLine
- name: 模型3的答案 # 标注工作台展示的标注组件名称
key: V4_FT_qw15_sft_0726_temp_old_500 # 导出标注结果的json字段名称 (支持英文字母下划线, 不能以数字开头)
type: TextInput # 组件类型
help: "请添加字段描述" # 组件帮助说明
value: "{{ .Values.model3 }}"
size: MultiLine
    
```



在该场景下，可以配置 schema 实现展示单张测试图片，以及配置展示不同模型的推理结果、设置待评测选择的模型名称。

3. 筛选高质量的文本问答对

```
desc: 自动生成的YAML
record_fields:
- name: 问题 # 标注工作台展示的标注组件名称
  key: question # 导出标注结果的json字段名称(支持英文字母下划线, 不能以数字开头)
  type: TextViewer # 组件类型
  help: "请添加字段描述" # 组件帮助说明
  value: "{{ .Values.question }}"
  size: MultiLine
- name: 答案 # 标注工作台展示的标注组件名称
  key: answer # 导出标注结果的json字段名称(支持英文字母下划线, 不能以数字开头)
  type: TextInput # 组件类型
  help: "请添加字段描述" # 组件帮助说明
  value: "{{ .Values.answer }}"
  size: MultiLine
- name: tag # 标注工作台展示的标注组件名称
  key: tag # 导出标注结果的json字段名称(支持英文字母下划线, 不能以数字开头)
  type: TextInput # 组件类型
  help: "请添加字段描述" # 组件帮助说明
  value: "{{ .Values.tag }}"
  size: SingleLine
- name: 是否正确 # 标注工作台展示的标注组件名称
  key: correct # 导出json标注结果时候该组件对应的json字段key
  type: StringSelector # 表明该组件类型是字符串选择组件
  help: 请判断答案是否正确 # 组件帮助说明
  option: SingleSelector # 表明该组件是单选。字段取值范围: SingleSelector/MultiSelector
  choices: # 指定选项的内容
  - 正确
  - 舍弃
  - 存疑
```

全部(398) 386 / 398条 快捷键 删除本条数据 上一条 下一条 提交标注结果

问题 ①
请问腾讯上海的办公大厦环境好吗

答案 ①
腾讯徐汇滨江大厦位于徐汇区龙爱路，东边比邻黄浦江，环境优美，空气清新，还有各种大型商场吃饭购物都非常的方便

tag ①
自由问答

是否正确 ①
 正确 舍弃 存疑

在该场景下，可以配置 schema 实现展示不可编辑修改的“问题”字段，防止标注人员在标注过程中的误操作，可设置“答案”字段为可编辑字段，同时支持用户配置自定义的筛选枚举值“正确、舍弃、存疑”。

创建传统CV的数据标注任务

最近更新时间：2024-11-07 14:38:42

概述

“数据中心-数据标注”模块支持用户创建在【传统CV建模】类型下的“图片分类、目标检测、目标跟踪、图片分割”的标注任务，为用户提供专业、高效的标注操作台。

⚠ 注意：

同一用户在公有云的所有地域下，可同时维护不超过 50 个标注任务，若您的业务有扩容需求，请使用工单联系我们哦。

数据标注任务

以下详细介绍如何创建并完成一个数据标注任务。

1. 准备：在“数据中心-数据集管理”模块成功导入一个“图片”类型的数据集，数据集已导入成功处于“可用”状态；
2. 在“数据中心-数据标注”模块，单击新建标注任务，完成参数配置：

新建标注任务

任务名称

请输入不超过60个字符，仅支持中英文、数字、下划线"_"、短横"-"，只能以中英文、数字开头

任务描述

0 / 500

标签 ×

+ 添加

选择数据集 [去导入数据集](#)

标注场景 图片 文本 视频

图片分类
识别图片所属类别



目标检测
识别图片中物体位置及其类别



目标跟踪
对图片中的某个物体进行识别跟踪



图片分割
分割图片中指定类别物体的区域



OCR
识别图片中的文字信息



标注格式 TI平台格式 Pascal Voc COCO

标签值

+ 添加

- 任务名称：用户输入自定义标注任务名称；
- 任务描述：用户可输入更多的对该任务的文字描述；
- 标签：支持用户为标注任务添加相关的“标签”信息（“标签”功能可参考 [腾讯云【标签】产品](#)）；
- 选择数据集：此处下拉展示的数据集为“数据中心-数据集管理”中状态为“可用”的图片类型的数据集，且创建该标注任务的用户需要对该数据集对应的 COS 路径有写的操作权限；

- 标注场景：选择对应的标注场景，目前仅支持“图片”类型的数据，“文本”和“视频”类型的数据标注敬请期待哦；
 - 标注格式：根据不同的“标注场景”需要选择对应的“标注格式”。若该标注场景只支持一种标注格式，则后台自动默认选中（OCR 场景目前仅针对白名单用户开放）；
 - 图片分类
 - TI 平台格式
 - 文件目录结构
 - 目标检测
 - TI 平台格式
 - Pascal Voc
 - COCO
 - 图片分割
 - TI 平台格式
 - Pascal Voc
 - COCO
 - 目标跟踪
 - TI 平台格式
 - OCR
 - TI平台格式
 - 标签值：支持用户自定义输入标签值，且同时选择对应的渲染颜色。
3. 单击配置页面的**提交**后，返回标注任务列表页面，等待该标注任务的“状态”处于“创建成功”后，可点击“操作-标注”按钮进入标注操作台，对图片进行标注。您可反复进入同一个标注任务进行“标注”操作。
 4. 在标注操作过程中，可通过“保存”按钮实时保存本次作业的结果，保存后退出，再次进入操作台时，会自动记录上一次操作的结果；
但注意：此时标注结果还未永久写入用户指定的该数据集的输出存储 COS 路径中。
 5. 在标注任务列表页面，通过“已标注”的状态字段，可实时查看标注任务的进度，待数据集中的全部图片都完成标注后，可单击**操作 > 提交** 将标注结果提交到用户绑定的输出 COS 路径中。此后，TI 平台的其他模块皆可使用该数据集的标注信息。

任务名称	标注数据集	标注场景	格式	状态	已标注	创建时间	操作
猫狗分类标注任...	用于猫狗分类的数据集 (v1)	图片分类	TI 平台格式	创建成功	0 / 19	2021-12-30 16:59:31	标注 提交 删除
目标跟踪	目标跟踪 (v1)	目标跟踪	TI 平台格式	创建成功	2 / 19	2021-12-21 19:55:26	标注 提交 删除
OCR识别	OCR识别 (v1)	OCR识别	TI平台格式	创建成功	1 / 19	2021-12-21 19:52:57	标注 提交 删除

数据标注列表操作

最近更新时间：2024-08-22 10:17:11

概述

数据标注任务列表页面统一管理了该用户账号下所有类型标注任务的集合，支持“标注、提交、删除”操作，支持点击标注任务名称查看任务详情。

列表页操作介绍

数据标注任务列表管理页面支持的操作详细介绍如下：

任务名称	标注数据集	标注场景	标签	状态	已标注	创建时间	操作
勿删盘子识别	勿删盘子识别 (v1)	目标检测		创建成功	194 / 242	2022-05-16 11:52:43	标注 提交 删除
debug	debug1 (v1)	图片分类		创建成功	1 / 2	2022-04-21 09:42:23	标注 提交 删除
企鹅分类标注任务	企鹅图片标注 (v1)	图片分类		创建成功	11 / 16	2022-04-10 15:45:30	标注 提交 删除
seg_2_3_1	2_3_1 (v1)	图片分割		创建成功	3 / 6	2022-02-03 16:51:38	标注 提交 删除
zip分割jinpın0126	zip使用jinpın分割coco (v1)	图片分割		创建成功	1 / 3	2022-01-26 15:06:56	标注 提交 删除
hxsdfsd-1	elisa图片数据集 (v1)	图片分类		创建成功	2 / 4	2022-01-24 15:28:35	标注 提交 删除
det_coco_21_1	yanrong_det_19_1 (v1)	目标检测		创建成功	1 / 5	2022-01-21 17:18:25	标注 提交 删除

- 任务名称：展示该标注任务名称，点击名称可查看任务基本详情信息（标注场景、标注数据集、标注值集合等）；
- 标注数据集：展示该任务绑定的标注数据集的名称及其版本号；
- 标注场景：展示该任务的场景类别，且支持用户筛选场景进行列表展示；
- 标签：该任务绑定的腾讯云标签
- 状态：创建中、创建成功、创建失败，支持用户筛选状态进行列表展示；
- 已标注：展示当前任务的标注进度，已标注的数据总量/总共待标注的数据总量；
- 创建时间：记录该任务成功创建的系统时间；
- 操作标注：点击后即可进入对应的标注作业台进行标注作业，用户可多次进入同一标注任务反复进行标注作业；
- 操作提交：只有当标注作业台的信息被用户手动“提交”操作后，标注结果文件才从系统后台数据库提交到用户指定的 COS 输出路径中，才会被 TI 平台其他业务模块读取到；
- 操作删除：可删除该标注任务，任务删除后，COS 存储桶中由该标注任务产生的输出标注文件不会被同步清理。

传统CV数据标注操作界面说明

最近更新时间：2024-08-22 16:54:43

概述

数据标注功能模块为用户提供专业、高效的标注操作台。

以下依据标注操作台的不同呈现形式分别为您介绍：

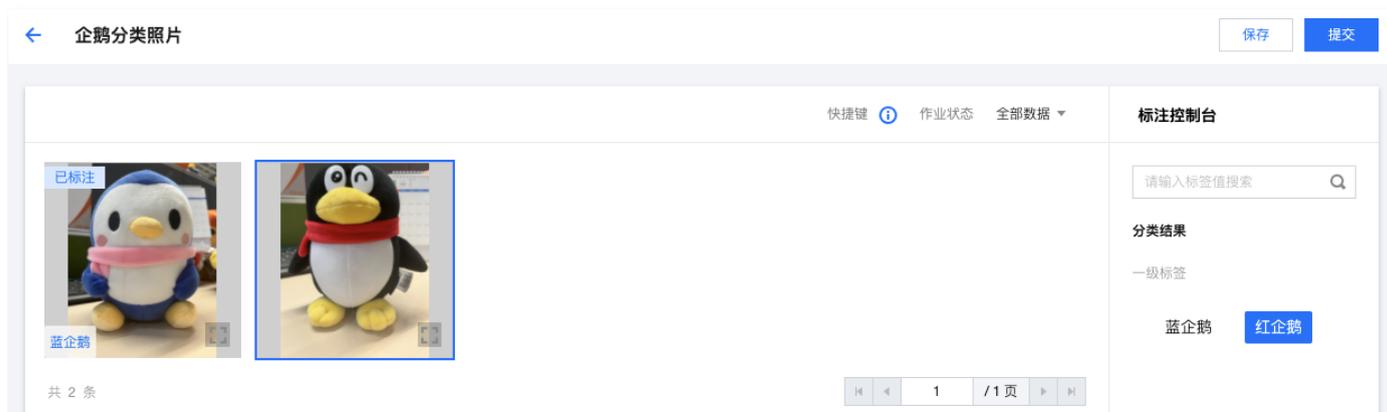
1. 用于“图片分类”标注任务的“分类类型标注操作台”；
2. 用于“目标检测、目标跟踪、图片分割”的“检测类型标注操作台”；
3. 用于“识别/检测、智能结构化”的“OCR 类型标注操作台”。

分类类型标注操作台

基于分类场景，采用小图列表平铺的操作台模式，方便用户快速分类多张图片。

页面右上角支持用户单击筛选图片的不同作业状态“全部数据/有标注结果/无标注结果”。

在操作台右侧区域是标注任务的标注值集合区，供用户检索和赋予当前选中图片的标注值。



支持的快捷键清单：

上一张图片：键盘字母“A”；

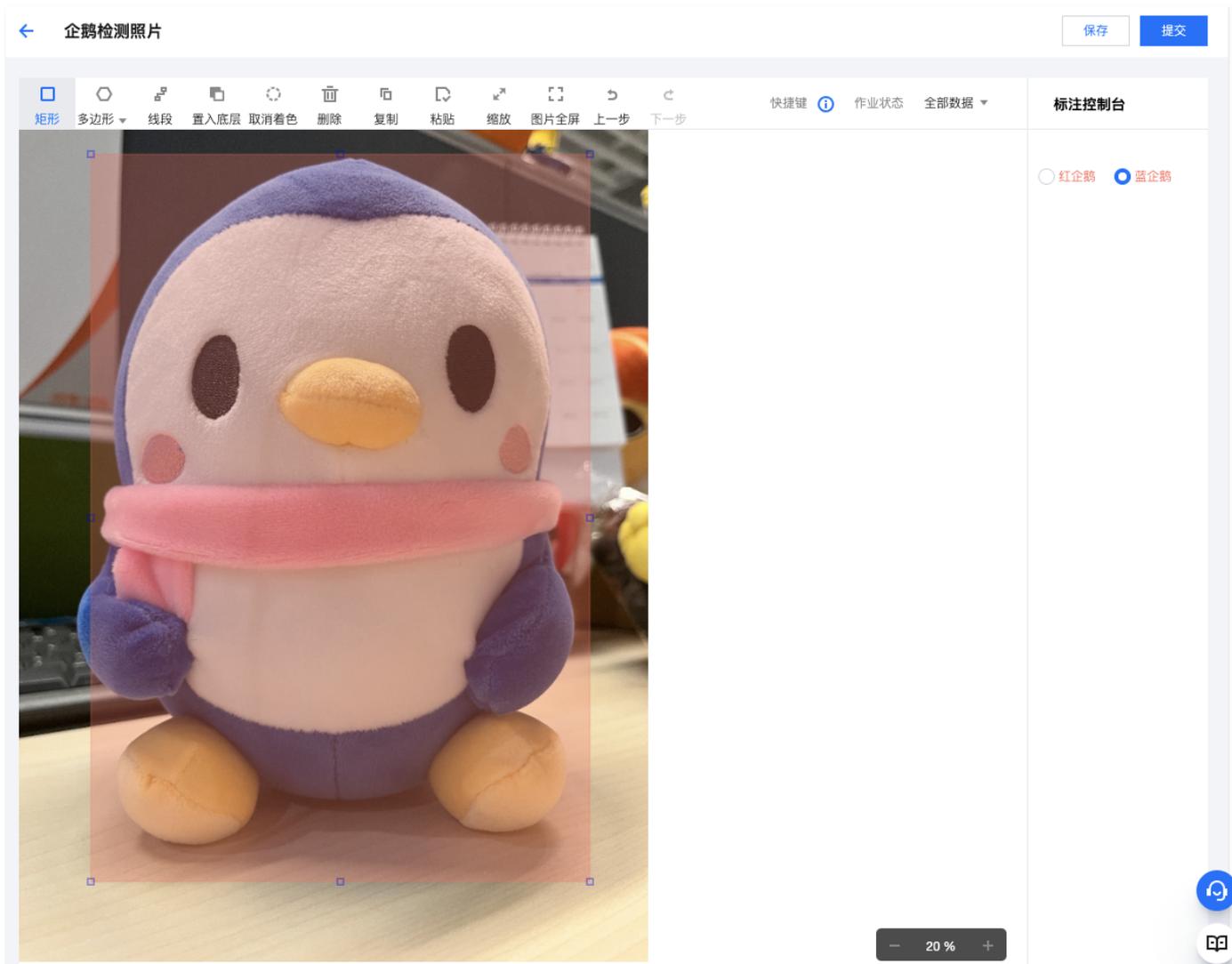
下一张图片：键盘字母“D”。

检测类型标注操作台

基于检测类型的场景，为了用户能更清楚的分辨并绘制图片中指定物体的位置，采用一张大图 + 一排缩略图列表的操作台模式。

右上角支持用户单击筛选图片的不同作业状态“全部数据/有标注结果/无标注结果”。

在操作台右侧区域是标注任务的标注值集合区，供用户赋予当前选中物体对应的标注值；在操作台上方区域为本次标注任务场景支持的所有标注工具的集合：矩形、多边形、线段等。



标注工具清单介绍:

- 矩形: 绘制矩形框 (支持快捷键 Ctrl+1);
- 多边形: 绘制多边形框, 左键描点, 右键结束; 且支持用户固定可绘制的多边形点数 (支持快捷键 Ctrl+2);
- 线段: 绘制线段, 左键描点, 右键结束 (支持快捷键 Ctrl+3);
- 置入底层: 将绘制的框或线段等置入页面底层 (支持快捷键 Ctrl+4);
- 取消着色: 取消框区域的默认填充色, 使得标注区域透明, 更能清楚的看到标注区域的图片内容;
- 删除: 指定选中绘制好的框或线段后可删除 (支持快捷键 Ctrl+D);
- 复制/粘贴: 复制/粘贴绘制好的框或线段对象, 该操作会同时复制对象的标注值属性 (支持快捷键 Ctrl+C / Ctrl+V);
- 缩放: 目前仅支持多边形和折线的缩放;
- 图片全屏: 单击后可全屏查看图片大图;
- 上一步/下一步: 撤销上一步/下一步操作;
- 快捷键清单介绍: 1) 上一张图片: 键盘字母 “A”; 2) 下一张图片: 键盘字母 “D”。

OCR 类型标注操作台

基于 OCR 标注场景下的“识别/检测”和“智能结构化”两种标注类型, TI 平台提供了专门的标注操作台和自研的结构化标注语言——描述子。

标注工具及快捷键介绍

您可在标注过程中，单击标注操作台右上角的**使用说明**按钮查看详细说明。

使用说明 ✕

一、可支持的checkbox：（复制后粘贴到字段文本中使用）

二、标注工具

标注工具	🔍 Auto	kv 字段属性	T 字段文本	P Point	ID 标注ID	... 检查设置	📏 辅助标注	📦 Box	📄 自动保存
说明	整图预标	在标注框下展示字段属性	在标注框下展示字段文本	在标注框上展示四点顺序	在标注框下展示id	调整展示字体和大小	画框后自动识别字段文本	识别被选中的描述框的字段文本	翻页时自动保存

三、快捷键

Macos快捷键	Win快捷键	快捷键说明
Command+mouse	Ctrl+mouse	移动背景图
Option+鼠标滚轮	Alt+鼠标滚轮	放大缩小图片
F1	F1	复制box
F2	F2	修改字段内容
F3	F3	修改字段属性
F4	F4	删除选中的box
F5/Ctrl+S	F5/Ctrl+S	保存
F6	F6	拆分box

标注模式介绍

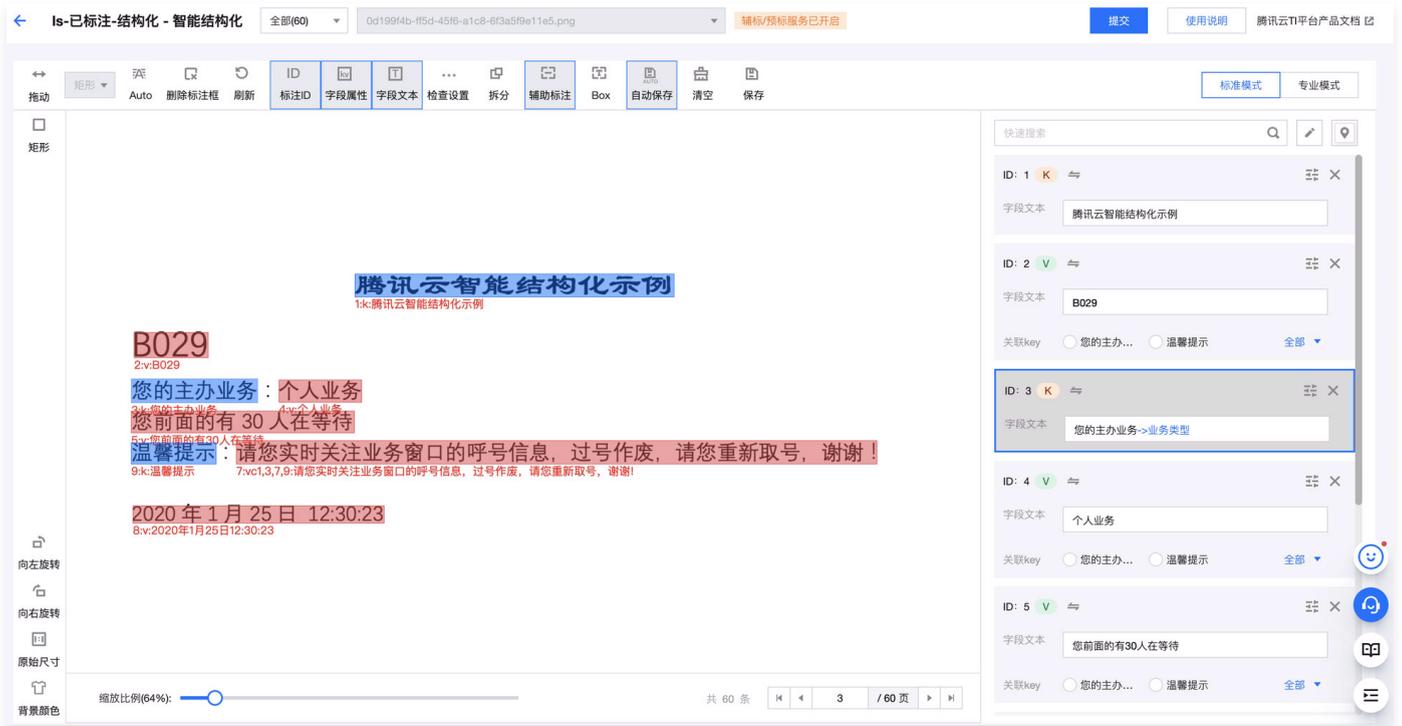
在标注操作台的右侧打标签区域，提供了两种模式：标准模式和专业模式（两种模式可互相转换）

- 标准模式

在“识别/检测”和“智能结构化”两种标注类型下都提供简易版的标注模式，用户可直接通过前端页面点选按钮决定检测识别文字的方向（水平/竖排）；或者通过点选的形式配对结构化中的 key 值和 value 值。

- 专业模式

在“智能结构化”的标注类型下，提供了专业标注模式，支持用户输入自研的描述子语言适配结构化 OCR 下复杂的文字场景。以下详细介绍描述子语言。



描述子语言介绍

智能结构化输出的结果主要就是 key-value 对，简称 kv。例如 key-value 对：发票代码 (key) -- 141001950242 (value)；发票号码 (key) -- 30148556 (value)。因此，字段属性中最重要的属性就是 kv 属性。同时，在 OCR 的真实业务场景下，经常存在嵌套 kv、字段不完整、字段分行、附加 key 字段、或者对 key 字段进行归一化等复杂多样的诉求。所以 TI 平台提出了自研的描述子标注语言来灵活的适配表达各种 OCR 场景。

kv 属性：k 段描述子和 v 段描述子

描述一个字段是 key 还是 value (分别称为 k 字段和 v 字段)，及其 kv 关联关系。k 字段和 v 字段可以独立存在，也可以共同存在并且有关联关系。kv 关联关系有以下几种：

- 1k0v: 表示只有 k 字段，没有关联的 v 字段；
- 0k1v: 表示只有 v 字段，没有关联的 k 字段；
- 1k1v: 表示1个 k 字段关联1个 v 字段，k 字段和 v 字段都在图片上出现；
- 1k1v: 表示图片中只出现了 v 字段，但这个 v 字段需要关联一个附加 key 名 (通过在字段文本中使用@@来标识附加 key 名)；
- 1knv: 表示1个 k 字段关联多个 v 字段，通常会出现在 v 字段是多行的情况；
- nk1v: 表示多个 k 字段关联1个 v 字段，通常会出现在嵌套表格或行列表格的情况。

示例：输入“k3,5”：表示字段3和字段5构成 kv 关系，3是 key 框，5是 value 框。

完整性 (complt) 属性：c 段描述子

描述一个字段是否完整表达一个含义 (如果可以完整表达，称为完整字段)，当需要多个字段组合时，需要描述组合的顺序关系。大多数情况下，一个字段可以表达一个完整含义，但有时候由于分行、文字竖排等各种原因，多个字段有序的连起来才能表达一个完整含义。

c 字母后接以下格式数字，定义：

- 0: 本字段是完整字段 (这种情况下可以不出现c段描述子)；
- fid1, fid2...: fid表示每个框的 ID 序号，表明本字段不是完整字段，用逗号有序隔开的多个字段组合起来才能表达一个完整含义。

示例：字段5的描述子输入为：5kc5,7；字段7的描述子输入为：7kc5,7。则表示5、7两个字段组合起来表达完整含义。

标签格式示例说明

最近更新时间：2024-08-22 00:46:41

概述

当您需要 在 TI 平台的数据中心 > 数据集管理 模块上传一份“已标注”的“图片/文本”类型的数据集时，标注文件内容的格式需要遵循平台规范，只有符合规范的标注文件才可被后台正确解析，成功解析后您可在数据集详情页面查看该数据集的标注信息可视化分布展示。

且我们建议您单张图片的标注内容大小不超过100k，若需扩容，可通过工单联系我们哦。

在导入已标注的数据集时，请确认满足以下两点规范：

- 您需要按照“文件存放示意图”准备您的文件目录；
- 您需要参考“下载标注文件示例”链接中的内容格式准备您的标注文件。

数据来源

标注状态 * 未标注 已标注

标注格式 * [下载标注文件示例](#)

文件存放方式示意图

```
Dataset_Filename (请选择此处文件夹)
├── 1.jpg
├── 2.jpg
├── 3.jpg
└── annotation.txt (请保持文件名称一致)
```

以下详细介绍不同标注场景下的标注格式规范。注意：每张图片对应的标注内容的文件大小上限为1M。

图片分类

1) TI 平台格式

- 文件目录要求：

标注格式 * [下载标注文件示例](#)

文件存放方式示意图

```
Dataset_Filename (请选择此处文件夹)
├── 1.jpg
├── 2.jpg
├── 3.jpg
└── annotation.txt (请保持文件名称一致)
```

- 标注文件格式要求：

```
{
  "info": { // 可选字段，表示图片的相关信息
    "md5": "图片md5值", // 可选字段，表示图片的md5值
    "path": "图片相对路径",
  },
  "tags": {
```

```

"classification_tags": [{
  "first_class": "一级标签值",
  "confidence_level": 置信度 // 可选字段，表示标签置信度
}]
}
    
```

2) 文件目录结构

- 文件目录要求:

标注格式 * 图片分类 文件目录结构 [下载标注文件示例](#)

文件存放方式示意图

```

graph TD
    Root[请选择此处文件夹] --- Dog
    Root --- Cat
    Dog --- Dog1[1.jpg]
    Dog --- Dog2[2.jpg]
    Cat --- Cat3[3.jpg]
    Cat --- Cat4[4.jpg]
    
```

注释: 后台自动解析1.jpg和2.jpg的分类类别是Dog, 3.jpg和4.jpg的分类类别是Cat

- 标注文件格式要求: 无需额外上传标注文件

目标检测

1) TI 平台格式

- 文件目录要求:

标注格式 * 目标检测 TI平台格式 [下载标注文件示例](#)

文件存放方式示意图

```

graph TD
    Root[Dataset_Filename 请选择此处文件夹] --- 1[1.jpg]
    Root --- 2[2.jpg]
    Root --- 3[3.jpg]
    Root --- 4[annotation.txt 请保持文件名称一致]
    
```

- 标注文件格式要求:

```

{
  "tags": {
    "detection_tags": {
      "shapes": [{
        "frame_type": "框形状", // 表示框形状, 支持多边形: polygon, 矩形: rect, 和折线: polyline
        "pts": [x1, y1, x2, y2...],
        "bbox": [左上角横坐标, 左上角纵坐标, 长, 高],
      }
    ]
  }
}
    
```

```

        "tag": [{
            "label_key_name": "标签字段",
            "label_value_name": ["标签值"],
            "confidence_level": [置信度] // 可选字段，表示标签置信度
        }]
    }
},
"info": { // 可选字段，表示图片的相关信息
    "md5": "图片md5值", // 可选字段，表示图片的md5值
    "path": "图片相对路径"
}
}

```

2) Pascal Voc

- 文件目录要求:

标注格式 * 目标检测 Pascal Voc [下载标注文件示例](#)

文件存放方式示意图

- 📁 (请选择此处文件夹)
- ├── DataFile (请保持文件夹名称一致)
 - ├── 1.jpg
 - └── 2.jpg
- └── Annotations (请保持文件夹名称一致)
 - ├── 1.xml
 - └── 2.xml

- 标注文件格式要求: (xml 文件格式)

```

<annotation>
  <folder>VOC2012</folder>
  <filename>2007_000033.jpg</filename>
  <source>
    <database>The VOC2007 Database</database>
    <annotation>PASCAL VOC2007</annotation>
    <image>flickr</image>
  </source>
  <size>
    <width>500</width>
    <height>366</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>aeroplane</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>9</xmin>
      <ymin>107</ymin>
      <xmax>499</xmax>
    </bndbox>
  </object>
</annotation>

```

```
<ymax>263</ymax>
</bndbox>
</object>
</annotation>
```

3) COCO

- 文件目录要求:

标注格式 * 目标检测 COCO [下载标注文件示例](#)

文件存放方式示意图

```

Dataset_Filename (请选择此处文件夹)
├── 1.jpg
├── 2.jpg
├── 3.jpg
└── annotation.json (请保持文件名称一致)
```

- 标注文件格式要求: 请严格参考 [COCO 官网中的 \[Object Detection \] 格式](#)。

图片分割

1) TI 平台格式

- 文件目录要求:

标注格式 * 图片分割 TI平台格式 [下载标注文件示例](#)

文件存放方式示意图

```

Dataset_Filename (请选择此处文件夹)
├── 1.jpg
├── 2.jpg
├── 3.jpg
└── annotation.txt (请保持文件名称一致)
```

- 标注文件格式要求:

```
{
  "tags": {
    "segmentation_tags": { // 【注意】导入到标注平台, shapes 和 mask 应该是二选一提供, 如果同时提供则以 shapes 为
      准; 从标注平台导出, shapes 和 mask 都会给出;
      "shapes": [{
        "frame_type": "框形状", // 表示框形状, 公有云支持多边形: polygon, 必选
        "pts": [
          [x1, y1, x2, y2...]
        ],
        "bbox": [左上角横坐标, 左上角纵坐标, 长, 高], // 非必选, pts 表示分割区域外接矩形框
        "tag": {
          "label_value_name": "标签值",
          "confidence_level": 置信度 // 可选字段, 表示标签置信度
        }
      }
    ]
  }
}
```

```

    }
  },
  "mask": {
    "path": "mask图相对路径",
    "notes": {
      "灰度值": "标签值"
    }
  }
}
},
"info": { // 可选字段, 表示图片的相关信息
  "md5": "1e81a232952f32af71b70848556e0ecd", // 可选字段, 表示图片的md5值
  "path": "图片相对路径"
}
}
}

```

2) Pascal Voc

- 文件目录要求:

标注格式* 图片分割 Pascal Voc [下载标注文件示例](#)

文件存放方式示意图

```

graph TD
    Root[请选择此处文件夹] --- DataFile[DataFile]
    Root --- Annotations[Annotations]
    Root --- SegmentationClass[SegmentationClass]
    Root --- SegmentationObject[SegmentationObject]
    DataFile --- D1[1.jpg]
    DataFile --- D2[2.jpg]
    Annotations --- A1[1.xml]
    Annotations --- A2[2.xml]
    SegmentationClass --- SC1[1.png]
    SegmentationClass --- SC2[2.png]
    SegmentationObject --- SO1[1.png]
    SegmentationObject --- SO2[2.png]

```

- 标注文件格式要求: (xml文件格式要求)

```

<annotation>
  <folder>VOC2012</folder>
  <filename>2007_000039.jpg</filename>
  <source>
    <database>The VOC2007 Database</database>
    <annotation>PASCAL_VOC2007</annotation>
    <image>Flickr</image>

```

```

</source>
<size>
  <width>500</width>
  <height>375</height>
  <depth></depth>
</size>
<segmented></segmented>
</annotation>
    
```

3) COCO

- 文件目录要求:

标注格式 * 图片分割 COCO [下载标注文件示例](#)

文件存放方式示意图

```

├── Dataset_Filename (请选择此处文件夹)
│   ├── 1.jpg
│   ├── 2.jpg
│   ├── 3.jpg
│   └── annotation.json (请保持文件名称一致)
    
```

- 标注文件格式要求: 请严格参考 [COCO 官网中的【Object Detection】格式](#)。

目标跟踪

TI 平台格式

- 文件目录要求:

标注格式 * 目标跟踪 TI平台格式 [下载标注文件示例](#)

文件存放方式示意图

```

├── Dataset_Filename (请选择此处文件夹)
│   ├── 1.jpg
│   ├── 2.jpg
│   ├── 3.jpg
│   └── annotation.txt (请保持文件名称一致)
    
```

- 标注文件格式要求:

```

{
  "tags": {
    "detection_tags": {
      "shapes": [{
        "frame_type": "框形状", // 表示框形状, 支持多边形: polygon, 矩形: rect, 和折线: polyline
        "pts": [x1, y1, x2, y2...],
        "bbox": [左上角横坐标, 左上角纵坐标, 长, 高],
        "tag": [{
          "label_key_name": "标签字段",
        }
      ]
    }
  }
}
    
```

```

        "label_value_name": ["标签值"],
        "confidence_level": [置信度] // 可选字段，表示标签置信度
    }}
}}
}
},
"info": { // 可选字段，表示图片的相关信息
    "md5": "图片md5值", // 可选字段，表示图片的md5值
    "path": "图片相对路径"
}
}
}

```

文本分类

文本分类场景下的“文本数据”和“标注信息”的组织形式请参考数据集导入前端页面的提示卡片，如：

标注格式 + 文本分类 TI平台格式 下载标注文件示例

TI平台格式标注示意图

使用.txt文件后缀，支持多级单选、多级多选标签，最深支持5个层级

```

1  {"content": "这里是用户的文本内容1", "labels": {"标签名称": "标签值"}}
2  {"content": "这里是用户的文本内容2", "labels": {"标签名称": "标签值"}}
3  {"content": "这里是用户的文本内容3", "labels": {"标签名称": "标签值"}}

1  {"content": "这场A国与B国的战争，导致经济下滑", "labels": {"情感": "负面"}}
2  {"content": "小亮在生日那天收到礼物后十分开心", "labels": {"情感": "正面"}}
3  {"content": "小明帮助了上百位老人，令人感动不已", "labels": {"情感": "正面"}}
                
```

此外，再补充说明在“TI平台格式”和“JSON”格式中，还能灵活扩展支持的标签格式（按行和按文件同理）：

```

- 一级单选的格式
"情感": "负"
"情感": "正"

- 一级多选的格式
"领域": ["政治", "经济"]
"领域": ["娱乐", "社会"]
    
```

数据构建

最近更新时间：2025-03-31 21:47:32

概述

高质量的训练数据对于提升模型精调效果至关重要。为了帮助用户准备高质量的训练数据，TI 平台将内部研发所采用的标准数据构建流程沉淀到了“**数据中心 > 数据构建**”模块，平台内置了三套不同场景下的数据处理 pipeline：有监督-单轮问答-pipeline、有监督-多轮问答-pipeline、无监督-pipeline；且在每套 pipeline 中依据不同的业务诉求分别实现了数据清洗、prompt 优化、数据过滤、数据增强等具体功能。

新建任务

1. 在**数据中心 > 数据构建**列表页面，单击**新建任务**，完成参数配置：

- 任务名称：用户自定义输入，名称在**数据构建**模块全局唯一；
- 标签：腾讯云统一的云上资源标记，用于从不同维度对资源分类管理，可参考官网文档 [标签](#)；
- 数据构建 pipeline：可多选，类别包含：
 - 有监督-单轮 QA-pipeline：有标注的单轮问答对，适用于LLM大模型的SFT精调训练；
 - 有监督-多轮 QA-pipeline：有标注的多轮问答对，适用于LLM大模型的SFT精调训练；
 - 无监督-pipeline：无标注的纯文本数据，适用于LLM大模型的Pretrain预训练。
- 通用算法任务：支持您请选择一个或多个贴合用户真实业务场景的算法任务类别。针对不同的算法任务类别，在后续模型训练的阶段，平台会默认为您匹配对应适合的内置数据（有监督数据）来混合您的自定义数据进行配比精调，进一步提升模型效果。

← 新建数据构建任务
数据构建 已

① 目前平台为您内置了多种需求场景下不同的数据构建pipeline，包含：有监督-单轮问答-pipeline、有监督-多轮问答-pipeline、无监督-pipeline。pipeline中为您内置提供的內容：1.针对该场景的数据构建全流程pipeline代码框架。2.针对该场景的一些数据构建原子方法。3.一份quick_start.ipynb文档帮助用户快速上手pipeline构建流程。

任务名称

请输入不超过60个字符，仅支持中英文、数字、下划线"_"、短横"-", 只能以中英文、数字开头

地址

标签 ① ✕

[+ 添加](#) [🔗 键值粘贴板](#)

数据构建 pipeline 有监督-单轮问答-pipeline 有标注的单轮问答对，适用于LLM大模型的SFT精调训练。
 有监督-多轮问答-pipeline 有标注的多轮问答对，适用于LLM大模型的SFT精调训练。
 无监督-pipeline 无标注的纯文本数据，适用于LLM大模型的Pretrain预训练。

通用算法任务 **请选择通用算法任务** 已选择 (0)

请输入关键字搜索

- 开放式问答
- 基础语言能力
- 对话
- 角色扮演
- 智能体
- 思维链
- 阅读理解
- 文本理解
- 信息提取
- 知识挖掘
- 文本创作

一级类别	二级类别	三级类别

请选择一个或多个贴合您的业务场景的算法任务类别。针对不同的算法任务类别，在后续模型训练的阶段，平台会默认为您匹配对应适合的内置数据（有监督数据）来混合您的自定义数据进行配比精调，进一步提升模型效果。

备注

0 / 500

[新建](#)
[取消](#)

2. 单击配置页面的提交后，返回数据构建任务列表页面：

- 创建时间：该数据构建任务启动创建的时间点；
- 操作：
 - 跳转到对应开发机：进入开发机模块，待开发机创建成功后，会同步在“对应开发实例ID”列中展示相关信息；
 - 删除：若对应的开发机实例已删除，则可直接确认删除任务；若对应的开发机实例还未删除，用户需要先删除后再删除数据构建任务。
- 搜索框：支持用户通过任务名称模糊匹配搜索。

跳转到对应开发机

选择 数据构建 > 操作 > 跳转到对应开发机 进入开发机模块。

1. 首次点击，新建开发机实例。

当某一个任务首次被单击跳转到对应开发机按钮时，用户需要先创建一个开发机实例，单击后自动跳转到训练工坊 > 开发机 > 新建开发机参数配置页：

- 镜像：针对数据构建跳转过来的镜像，默认选中数据构建专用的内置镜像且不可修改（只有跳转过来才看得到该镜像，如果是从开发机通用入口进入就看不到本次新增的内置镜像）；
- 算力规格：普通的数据构建任务只需要 CPU 即可，不需要 GPU；
- 存储配置：目前建议存储是“CFS 文件系统”；

- 其余参数配置和开发机已有功能要保持一致。

← 新建开发机
开发机简介

名称 14/256 ✔

请输入不超过256个字符，仅支持中英文、数字、下划线“_”、短横“-”，只能以中英文、数字开头

地域 广州

镜像 内置镜像 / data-process-image

机器来源 从CVM机器中选择 从TIONE平台购买

TI-ONE 仅收取软件费用，算力将使用您已购买的机器。 [产品定价](#)

资源组 请选择资源组

建议至少选择 4C8G 的算力规格。若您的数据量大且希望提高数据构建效率，请尽量选择更高的算力规格。

资源申请

卡型号 请选择卡型号

GPU 0.0 卡

若需使用GPU，根据不同卡类型可填写0.1-1或1的整数倍。运行环境为平台内置GPU镜像时，卡数不能为0

CPU 1.0 核

内存 1.00 G

存储路径设置 ⓘ

请确保您选择的存储实例（CFS、GooseFS或者GooseFSx）和纳管资源组的节点网络互通，其中本地数据盘和GooseFSx仅支持挂载一个实例；配置的第一个容器挂载路径会做为默认工作目录

存储类型	CFS文件系统	源路径	容器挂载路径	操作
⋮	CFS 请选择	<input type="text" value="/"/>	<input type="text" value="/home/tione/notebook"/>	删除

+ 添加

标签 ⓘ

✕

+ 添加 📄 键值粘贴板

▶ 高级设置

确定
取消

↻

单击配置页面的**确定**后，新建任务完成后，返回开发机实例列表页面，等待创建成功。

2. 非首次点击，跳转到开发机列表。

开发机实例创建成功后，用户再次单击**数据构建** > **跳转到对应开发机**按钮，则会直接跳转到开发机实例列表页面，同时列表搜索框会默认带上对应开发机实例ID 以展示检索后的结果。

腾讯云 TI 平台
开发机 上海
开发机简介

实例列表
生命周期配置
Git存储库
SSH网络配置
闲置回收策略

新建 查看我的实例

名称	训练镜像	机器来源	占用资源	标签	状态	运行时长	创建者	最新编辑时间	监控
最佳实践-数据构建 nb-1287670824466777472 🔗	内置通用镜像: data-process-image	从TIONE平台购买-按量计费	计算资源: 8C16G CFS: cfs-pg5l7ren 源目录: /data/custom/prepare	🔗	运行中	1小时30分34秒 🕒	el10	n2 372703	2025-03-31 15:32:04 📄

共 1 条 10 条 / 页

3. 进入开发机实例，编辑预置 pipeline 脚本。

在开发机中默认展示平台为客户提前预置好的数据构建 pipeline，由于新建任务时的参数配置页的“数据构建 pipeline”参数支持多选，所以开发机中也可能内置存在一套或多套 pipeline：



每套 pipeline 中都为用户内置提供了：

- 针对该场景的数据构建全流程 pipeline 代码框架。
- 针对该场景的一些数据构建原子方法。
- 一份 quick_start.ipynb 文档帮助用户快速上手 pipeline 构建流程。

其中文件夹的层级组织形式参考：

single_round_qa_pipeline	multi_round_qa_pipeline	unsupervised_pipeline																																																																														
<p>nb-1287670824466777472 / single_round_qa_pipeline</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Last Modified</th> </tr> </thead> <tbody> <tr><td>config</td><td>2 hours ago</td></tr> <tr><td>final_dataset_files</td><td>26 minutes ago</td></tr> <tr><td>intermediate_dataset_...</td><td>27 minutes ago</td></tr> <tr><td>raw_dataset_files</td><td>35 minutes ago</td></tr> <tr><td>tools</td><td>33 minutes ago</td></tr> <tr><td>analysis_raw_data.py</td><td>2 hours ago</td></tr> <tr><td>invoke_llm_service.ipynb</td><td>20 minutes ago</td></tr> <tr><td>quick_start.ipynb</td><td>25 minutes ago</td></tr> <tr><td>step_clean_data.py</td><td>2 hours ago</td></tr> <tr><td>step_deduplicate_data...</td><td>2 hours ago</td></tr> <tr><td>step_generate_training...</td><td>2 hours ago</td></tr> <tr><td>step_optimize_prompt...</td><td>2 hours ago</td></tr> <tr><td>step_parse_raw_data.py</td><td>2 hours ago</td></tr> <tr><td>version</td><td>2 hours ago</td></tr> </tbody> </table>	Name	Last Modified	config	2 hours ago	final_dataset_files	26 minutes ago	intermediate_dataset_...	27 minutes ago	raw_dataset_files	35 minutes ago	tools	33 minutes ago	analysis_raw_data.py	2 hours ago	invoke_llm_service.ipynb	20 minutes ago	quick_start.ipynb	25 minutes ago	step_clean_data.py	2 hours ago	step_deduplicate_data...	2 hours ago	step_generate_training...	2 hours ago	step_optimize_prompt...	2 hours ago	step_parse_raw_data.py	2 hours ago	version	2 hours ago	<p>nb-128771995548780592 / multi_round_qa_pipeline</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Last Modified</th> </tr> </thead> <tbody> <tr><td>config</td><td>3 minutes ago</td></tr> <tr><td>final_dataset_files</td><td>3 minutes ago</td></tr> <tr><td>raw_dataset_files</td><td>3 minutes ago</td></tr> <tr><td>tools</td><td>3 minutes ago</td></tr> <tr><td>analysis_raw_data.py</td><td>3 minutes ago</td></tr> <tr><td>quick_start.ipynb</td><td>3 minutes ago</td></tr> <tr><td>step_clean_data.py</td><td>3 minutes ago</td></tr> <tr><td>step_deduplicate_data...</td><td>3 minutes ago</td></tr> <tr><td>step_generate_training...</td><td>3 minutes ago</td></tr> <tr><td>step_parse_raw_data.py</td><td>3 minutes ago</td></tr> <tr><td>version</td><td>3 minutes ago</td></tr> </tbody> </table>	Name	Last Modified	config	3 minutes ago	final_dataset_files	3 minutes ago	raw_dataset_files	3 minutes ago	tools	3 minutes ago	analysis_raw_data.py	3 minutes ago	quick_start.ipynb	3 minutes ago	step_clean_data.py	3 minutes ago	step_deduplicate_data...	3 minutes ago	step_generate_training...	3 minutes ago	step_parse_raw_data.py	3 minutes ago	version	3 minutes ago	<p>nb-1287720352763956864 / unsupervised_pipeline</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Last Modified</th> </tr> </thead> <tbody> <tr><td>config</td><td>2 minutes ago</td></tr> <tr><td>final_dataset_files</td><td>2 minutes ago</td></tr> <tr><td>raw_dataset_files</td><td>2 minutes ago</td></tr> <tr><td>tools</td><td>2 minutes ago</td></tr> <tr><td>analysis_raw_data.py</td><td>2 minutes ago</td></tr> <tr><td>quick_start.ipynb</td><td>2 minutes ago</td></tr> <tr><td>step_clean_data.py</td><td>2 minutes ago</td></tr> <tr><td>step_deduplicate_data...</td><td>2 minutes ago</td></tr> <tr><td>step_generate_training...</td><td>2 minutes ago</td></tr> <tr><td>step_parse_raw_data.py</td><td>2 minutes ago</td></tr> <tr><td>version</td><td>2 minutes ago</td></tr> </tbody> </table>	Name	Last Modified	config	2 minutes ago	final_dataset_files	2 minutes ago	raw_dataset_files	2 minutes ago	tools	2 minutes ago	analysis_raw_data.py	2 minutes ago	quick_start.ipynb	2 minutes ago	step_clean_data.py	2 minutes ago	step_deduplicate_data...	2 minutes ago	step_generate_training...	2 minutes ago	step_parse_raw_data.py	2 minutes ago	version	2 minutes ago
Name	Last Modified																																																																															
config	2 hours ago																																																																															
final_dataset_files	26 minutes ago																																																																															
intermediate_dataset_...	27 minutes ago																																																																															
raw_dataset_files	35 minutes ago																																																																															
tools	33 minutes ago																																																																															
analysis_raw_data.py	2 hours ago																																																																															
invoke_llm_service.ipynb	20 minutes ago																																																																															
quick_start.ipynb	25 minutes ago																																																																															
step_clean_data.py	2 hours ago																																																																															
step_deduplicate_data...	2 hours ago																																																																															
step_generate_training...	2 hours ago																																																																															
step_optimize_prompt...	2 hours ago																																																																															
step_parse_raw_data.py	2 hours ago																																																																															
version	2 hours ago																																																																															
Name	Last Modified																																																																															
config	3 minutes ago																																																																															
final_dataset_files	3 minutes ago																																																																															
raw_dataset_files	3 minutes ago																																																																															
tools	3 minutes ago																																																																															
analysis_raw_data.py	3 minutes ago																																																																															
quick_start.ipynb	3 minutes ago																																																																															
step_clean_data.py	3 minutes ago																																																																															
step_deduplicate_data...	3 minutes ago																																																																															
step_generate_training...	3 minutes ago																																																																															
step_parse_raw_data.py	3 minutes ago																																																																															
version	3 minutes ago																																																																															
Name	Last Modified																																																																															
config	2 minutes ago																																																																															
final_dataset_files	2 minutes ago																																																																															
raw_dataset_files	2 minutes ago																																																																															
tools	2 minutes ago																																																																															
analysis_raw_data.py	2 minutes ago																																																																															
quick_start.ipynb	2 minutes ago																																																																															
step_clean_data.py	2 minutes ago																																																																															
step_deduplicate_data...	2 minutes ago																																																																															
step_generate_training...	2 minutes ago																																																																															
step_parse_raw_data.py	2 minutes ago																																																																															
version	2 minutes ago																																																																															

原始数据集格式说明

有监督数据格式

我们的有监督数据构建脚本代码支持了丰富的大模型常见的数据格式，有监督单轮问答 pipeline 和有监督多轮问答 pipeline 目前支持如下格式：

- json 文件**：json 内容是一个数组，数组中每一项是一个有效的 json 对象，可以转换成单轮问答数据；如果 json 内容不是数组，而是一个 json 对象，我们会解析成长度为1个元素的数组来进行处理；
- jsonline 文件**：每一行是一个有效的 json 对象，可以转换成有监督问答数据；
- parquet 文件**：parquet 列式存储格式是开源数据常见格式（[Parquet 文档](#)）。我们会先把 parquet 对象转成 json 对象，您可在下述采样过程中预览转换后的json格式，再进行数据处理；
- 数据集文件夹**：文件夹中包含若干有效的数据文件（格式可以是 json/jsonline/parquet），此时我们会把该文件夹当做数据集进行处理，数据集名称即为文件夹名称，在后续数据解析流程完成后，会把所有文件的内容全部合并后再执行后续流程。
 - 单轮有监督：示例代码提供的原始数据集文件格式为 `jsonl`，示例内容如下：

```
{ "instruction": "请简要回答下面的问题：", "input": "您平时用什么投资理财方式？", "output": "我一般采用银行储蓄，相对风险较低。"}
{ "instruction": "请简要回答下面的问题：", "input": "基金拆分是什么意思？", "output": "基金拆分是保持基金投资人资产总值不变的前提下，改变基金份额净值和基金总份额的对应关系，重新计算基金资产的一种方式。"}

```

- 多轮有监督：示例代码提供的原始数据集文件格式为 **jsonl**，示例内容如下：

```
{ "conversations": [{ "from": "human", "value": "保险是什么，可靠么？"}, { "from": "model", "value": "保险，是指投保人根据合同约定，向保险人支付保险费，保险人对于合同约定的可能发生的事故因其发生所造成的财产损失承担赔偿责任责任，或者被保险人死亡、伤残、疾病或者达到合同约定的年龄、期限等条件时承担给付保险金责任的商业保险行为。正规渠道购买的保险都是安全可靠的。"}, { "from": "human", "value": "我想买一些投资理财类的保险，有哪些险种？"}, { "from": "model", "value": "年金保险、分红险、投连险等。"}, { "from": "human", "value": "这些投资理财类保险是不是稳赚不赔的？"}, { "from": "model", "value": "不是，收益受到市场因素影响。"}]}
{ "conversations": [{ "from": "human", "value": "我妈妈最近想买一份大病保险，这个是骗人的么？"}, { "from": "model", "value": "不是，重大疾病保险，它是一种针对严重疾病进行保障的保险产品。"}, { "from": "human", "value": "这个大病保险的报销比例有多少？"}, { "from": "model", "value": "重大疾病保险报销比例是根据保险合同中约定的具体条款来确定的，不同的保险公司和不同的保险产品可能有不同的报销比例。"}, { "from": "human", "value": "我也想买一份，但是我目前17岁，可以买么？"}, { "from": "model", "value": "可以，但是未成年需要父母作为投保人来购买。"}]}

```

无监督数据格式

我们的无监督数据构建脚本本代码支持了丰富的大模型常见的数据格式，**无监督 pipeline** 目前支持如下格式：

1. **json 文件**: 支持 json 对象文件和 json 数组文件的解析；
2. **jsonline 文件**: 支持 jsonline 多行文件格式；
3. **parquet 文件**: parquet 列式存储格式是开源数据常见格式（[Parquet文档](#)）。我们会先把 parquet 对象转成 json 对象，您可在下述采样过程中预览转换后的 json 格式，再进行数据处理；
4. **text 文件**: txt 文本文件；
5. **wet 格式**: WET 表示原始网页 WARC 文本化（WET）格式。我们会把WET解析成json对象，您可在采样过程中预览，再进行数据处理；
6. **数据集文件夹**: 支持以上类型的多个文件放在同一文件夹下面处理，同一文件夹下下面目前只支持一种类型的多个文件同时处理。

数据格式的示例如下：

- json: json 数组的每一行作为一条数据

```
[
  "this is statement 1",
  "this is statement 2"
]
```

- jsonl: 每一行的"text"key 对应的 value 字段作为一条数据，例如：

```
{"text": "this is statement 1"}
{"text": "this is statement 2"}
```

- text:

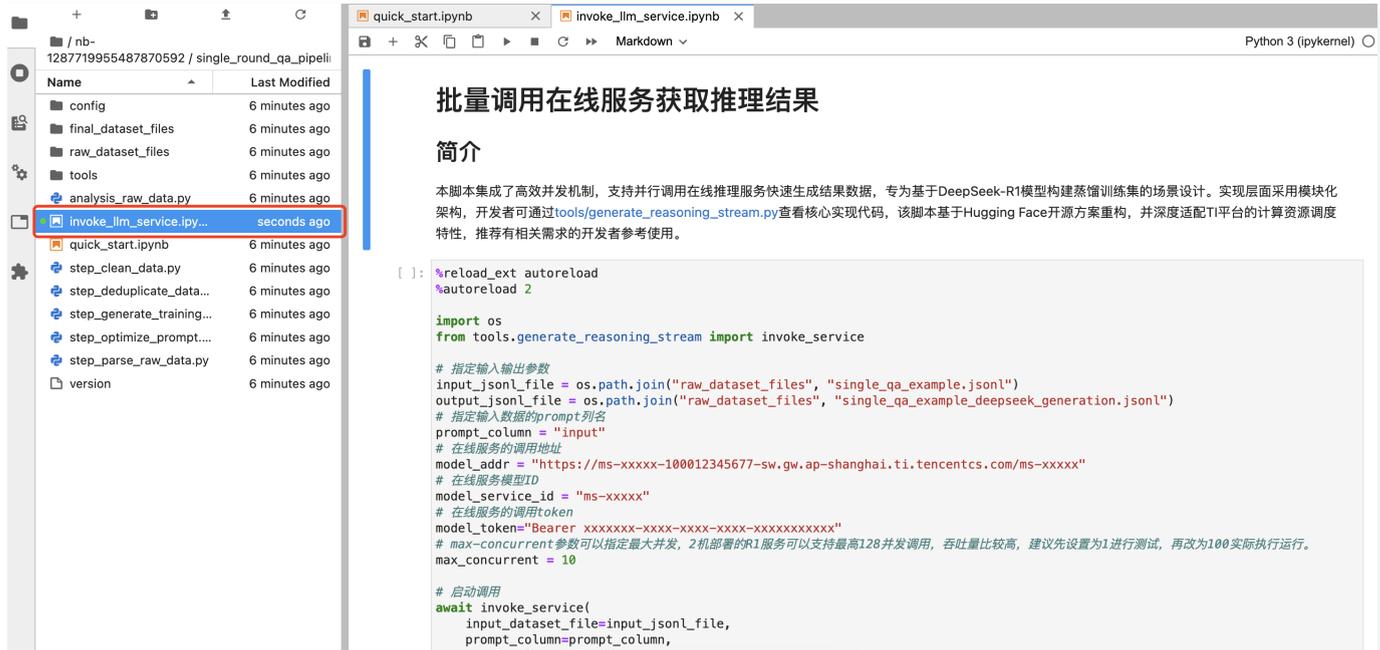
```
this is statement 1
this is statement 2
```

- wet: 支持标准的 Web Archive，详情请参考 [示例数据](#)。

数据蒸馏

在 [有监督-单轮问答-pipeline] 的开发机脚本中，我们新增内置了一段“通过数据构建的 notebook 调用在线服务得到批量预测结果”的数据蒸馏脚本。该脚本用于帮助用户实现如利用 DeepSeek-R1 模型构建蒸馏训练集的目标。且将该数据蒸馏环节内置到已有的数据构建 pipeline 中，便于用户灵活处理数据，如在调用蒸馏模型之前先通过 pipeline 中的数据清洗、去重等子模块过滤一些低质量、冗余的样本数据，以此降低服务调用的成本。

开发者可通过 [tools/generate_reasoning_stream.py](#) 查看核心实现代码，该脚本基于 Hugging Face 开源方案重构，并深度适配了 TI 平台的计算资源调度特性，集成了高效并发机制来调用在线推理服务实现快速生成推理结果。



首先，在运行该脚本功能前，请确保您在**模型服务 > 在线服务**模块的自定义在线服务已处于“运行中”状态，您的目标就是利用已有样本集批量调用该在线服务得到样本推理结果。



然后，单击在线服务名称查看详情，将以下三处红框信息分别填写到数据构建脚本参数中。

腾讯云 TI 平台

- 大模型广场
- 自动学习
- 数据中心
 - 数据集管理
 - 数据标注
 - 数据构建
- 训练工坊
- 模型管理
 - 模型仓库
 - 模型优化
- 模型服务
 - 在线服务
 - 批量预测
 - 模型评测
- 资源组管理

数据蒸馏专用推理服务

服务管理 服务调用 在线体验

① 遵守平台要求，服务调用已在创建服务时，授权并同意了《腾讯云 TI-ONE 训练平台服务协议》

调用监控

常规服务调用 ①

调用地址 <https://ms-...-10C...gw.ap-guangzhou.t.tencentcs.com/ms-...> 1

AuthToken 8Py... 3

是否生成鉴权 已开启

QPS 500 (单服务组的QPS上限为500, 如需提高QPS请提交工单, 我们将根据您的业务情况进行评估)

高速服务调用

VPC调用地址 <http://1...> 删除 仅限以下VPC调用: VPC-... (9v49)、子网... (六区)

新增高速服务调用网段

服务访问云上VPC资源 ①

vpc -

子网 -

腾讯云 TI 平台

- 大模型广场
- 自动学习
- 数据中心
- 训练工坊
- 模型管理
 - 模型仓库
 - 模型优化
- 模型服务
 - 在线服务

数据蒸馏专用推理服务

服务管理 服务调用 在线体验

名称	ID	服务版本	状态	机器来源	运行/期望副...	服务详情
数据蒸馏专用推理服务 计费中	m-...-s-1	V1	运行中	从TIONE平...	1/1	🔊 📄

2

```

%reload_ext autoreload
%autoreload 2

import os
from tools.generate_reasoning_stream import invoke_service

# 指定输入输出参数
input_jsonl_file = os.path.join("raw_dataset_files", "single_qa_example.jsonl")
output_jsonl_file = os.path.join("raw_dataset_files", "single_qa_example_deepseek_generation.jsonl")
# 指定输入数据的prompt列名
prompt_column = "input"
# 在线服务的调用地址
model_addr = "https://ms-xxxxx-100012345677-sw.gw.ap-shanghai.ti.tencentcs.com/ms-xxxxx" 1
# 在线服务模型ID
model_service_id = "ms-xxxxx" 2
# 在线服务的调用token
model_token="Bearer xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" 3
# max-concurrent参数可以指定最大并发，2机部署的R1服务可以支持最高128并发调用，吞吐量比较高，建议先设置为1进行测试，再改为100实际执行运行。
max_concurrent = 10

# 启动调用
await invoke_service(
    input_dataset_file=input_jsonl_file,
    prompt_column=prompt_column,
    model_addr=model_addr,
    model_token=model_token,
    model_service=model_service_id,
    output_file=output_jsonl_file,
    max_concurrent=10
)

```

填写完您的在线服务参数的同时，自定义修改您的数据样本路径“input_jsonl_file/output_jsonl_file”，最后单击运行该 notebook，即可实时查看本次数据蒸馏任务的构建进度。

```

# 启动调用
await invoke_service(
    input_dataset_file=input_jsonl_file,
    prompt_column=prompt_column,
    model_addr=model_addr,
    model_token=model_token,
    model_service=model_service_id,
    output_file=output_jsonl_file,
    max_concurrent=10
)

```

Generating train split: 0 examples [00:00, ? examples/s]
Map: 0%| | 0/50 [00:00<?, ? examples/s]
Generating responses: 94%|██████████| 47/50 [01:15<00:04, 1.60s/row, active=0]

任务式建模

任务式建模简介

最近更新时间：2024-08-22 16:50:36

任务式建模提供通过向导式的训练任务提交方式进行模型构建，可直接基于平台内置镜像快速使用主流高性能及分布式训练框架提交训练任务，也可通过自定义训练镜像启动任务，其功能模块详细描述为：

- **训练任务创建**：基于内置框架或者自定义训练镜像进行训练任务创建提交，支持使用平台数据集或 COS 数据或者CFS文件系统作为训练样本，支持训练任务算法参数设定，支持日志投递和 VPC 绑定。
- **任务运行管理**：提供启停，删除等训练任务常规管理操作，同时提供任务复制功能，快速启动多种不同配置的训练任务，比对模型训练效果。
- **任务运行监控**：提供训练运行日志，训练指标可视化监控，训练资源消耗可视化监控，同时提供对通过 TiKit 启动的训练任务进行可视化监控。
- **模型一键发布**：提供一键发布模型至模型仓库功能，并支持模型版本更新。
- **Tensorboard**：任务支持启动 Tensorboard 监控。

创建任务

最近更新：2025-05-14 17:03:53

创建步骤

填写基本信息

1. 登录 [TI-ONE 控制台](#)，进入 [训练工坊 > 任务式建模](#)，单击 [新建](#)，开始创建训练任务。
2. 在基本信息配置页，填写如下信息：
 - **任务名称**：仅支持中英文、数字、下划线"_"、短横"-", 只能以中英文、数字开头。
 - **地域**：训练任务所在的地域，默认为当前列表页所在的地域。
 - **训练镜像**：可选择平台内置训练镜像、自定义镜像和内置大模型，其中内置训练镜像请查看 [内置训练镜像列表](#)；自定义镜像支持选择容器镜像服务的镜像或者填写外部镜像地址（若为私有镜像，需要输入用户名和密码），自定义镜像规范请查看 [自定义训练镜像规范](#)；内置大模型训练使用方式请查看 [精调内置开源大模型](#)。
 - **训练模式**：不同训练框架支持的训练模式请查看 [内置训练镜像列表](#)。
 - **计费模式**：可选择按量计费和包年包月，若选择按量计费，则需要选择算力规格和节点数量；若选择包年包月，则需要先创建资源组，购买节点，相关操作请查看 [资源组管理](#)，选择完资源组后选择对应的计算资源。平台支持的计费规格请查看 [计费概述](#)。

说明：

1. 选择资源组后会立即展示该资源组所剩的 GPU 概览信息，包括各卡型号的 GPU 总卡数，整机卡数和非整机卡数（碎卡数），可帮助用户快速了解选中资源组的 GPU 分布情况，根据当前任务场景选择使用整机资源还是非整机（碎卡）资源，可有效降低整体资源的碎片化情况，提升 GPU 总体利用率。
2. 点击 [查看详情](#) 即会在当前页面右侧展示详细资源看板，看板上展示了各个卡类型的节点剩余可用和总资源；点击下拉后可展示当前节点正在运行的所有任务/服务，可帮助用户快速了解节点资源的占用情况，以便团队间进行资源使用协商。

新建任务

ⓘ 状态为运行中的训练任务正在产生费用，不使用时，请及时停止。

基本信息

任务名称

请输入不超过60个字符，仅支持中英文、数字、下划线"_"、短横"-", 只能以中英文、数字开头

地域

训练镜像

训练模式

计费模式

资源组

TI-ONE 仅收取软件费用，算力将使用您已购买的机器。 [产品定价](#)

HCC-H20可用卡数：总计5/整机0/非整机5 [查看详情](#)

资源申请

卡型号

单节点GPU 卡

若需使用GPU，卡数可填写0.1-1或1的整数倍

单节点CPU 核

单节点内存 G

节点数 个

资源看板 [前往资源组管理查看详情](#)

HCC-H20所在节点

ⓘ 该资源组可用HCC-H20总计5卡，其中整机共0卡，非整机5卡

节点ID 实例ID	可用 总资源 ⓘ
sm-1rstvdst ins-3ibe155t	CPU 310.32 380.32C MEM 2014.58 2214.58G GPU 5 HCC-H20*8

- 标签：可为任务创建标签，一个任务可添加多个标签。
- 描述：可添加最多500字的备注描述。
- CLS 日志投递：CLS 日志投递默认关闭，TI 控制台会默认展示 15 天的日志，若您期望持久化存储日志，获得日志检索等服务，可以开启 CLS 日志投递，打开 CLS 后可以将日志投递至 CLS 服务（需要确保 CLS 服务已完成开通），CLS 产品介绍和收费指南请查看 [CLS 介绍](#)。
- 自动重启：您可以对任务配置自动重启策略，您需要配置最大重启次数，最高为10次，超过最大重启次数后，会将任务直接标记为异常；当前任务自动重启的触发条件为任务运行过程中发生异常退出（当前该功能仅支持计费模式为包年包月且训练模式为 MPI、DDP 和 Horovod 的训练任务）。任务自动重启的事件信息可在 [任务详情 > 事件](#) 页面中查看。
- 健康检测：可以选择为任务启用健康检测，启用后将会占用一定时间对任务所调度节点进行健康检测；检测时机为任务运行前，包含手动启动和自动重启；健康检测失败或超时时任务会结束，释放已占用资源。当前支持的检测项为 NCCL 网络检测，检测的条件为任务申请资源为单机8卡整机或者多机8卡整机且所选卡型支持 RDMA。同时可以为健康检测配置最长检测时长，当检测时间超过该时长后检测任务仍未正常结束，则训练任务会结束并且释放占用资源。健康检测过程中可进入 [任务详情 > 日志](#) 页面筛选日志类行为平台日志(健康检测) 查看检测日志和数据。

填写任务配置信息

任务配置页面需要配置本次训练任务的算法、数据、输入输出等信息，具体配置项说明如下：

1. 存储路径设置：存储类型支持数据集、COS、CFS（包含 CFS Turbo 类型）、GooseFSx、GoogeFS 和 EMR(HDFS)（其中 GooseFS 和 GooseFSx 仅支持从 CVM 机器中选择的资源组），每设置一个存储路径，均支持选择该路径的用途，包含用户自有模型，用户自有代码，用户自有数据，训练数据及其他（其中当选择存储类型为 CFS 时，用途还支持选择平台内置模型，可将 TIONE 平台内置 CFS 中的模型直接挂载到训练容器）：
 - 若选择数据集，则需要首先在平台 [数据中心 > 数据集管理](#) 创建数据集；
 - 若选择 COS，则需要选择数据所在的 COS 路径；
 - 若选择 CFS、GooseFSx、GoogeFS、EMR(HDFS)，则需要下拉选择 CFS 文件系统、GooseFSx 实例、GooseFS 集群、EMR 集群，同时填写需要平台挂载的数据源目录；

以上几种数据来源在配置的时候均可定义数据在训练容器内的挂载路径，您的代码中需要填写该路径以获取数据。创建任务时可选择多个数据集或者多个数据路径，分别设置不同的容器挂载路径，挂载到容器中，供训练算法读取。

- 使用 EMR(HDFS) 注意事项：平台默认使用 Hadoop 身份访问挂载 HDFS；如果需要其他身份，请按照如下代码包规范上传相关配置文件。
 - 用户名和 keytab 文件统一由用户提供，放在代码包里面。
 - 代码包规范：

```
/ <emr_id> /username.txt（内容为用户名，内容如：“hadoop/172.0.1.5”，当未开启 kerberos 认证时，文件不存在或者文件为空时可以使用默认的用户名“hadoop”；开启了 kerberos 认证后，默认用户名不可用）
```

```
/ <emr_id> /emr.keytab（内容为 keytab 认证文件）
```

（因为平台同时支持多个 emr 的多个认证，所以需要添加 `<emr_id>` 到目录中，值如：emr-1rnhggs）

2. Git 存储：可以选择 Git 存储库，并且配置容器内存储路径，任务启动时会将该存储库下的文件下载到指定的存储路径中。您需要首先在[训练工坊-Git 存储库](#)中创建好存储库。
3. 启动命令：您需要填写程序入口命令，支持填写多行，默认工作目录为/opt/ml/code。
4. 调优参数：填写的超参数 JSON 会保存为 /opt/ml/input/config/hyperparameters.json 文件，您的代码需自行解析。
5. 训练输出：选择您需要保存训练输出的 COS 路径，平台会默认将 /opt/ml/output 路径下的数据定期上传至输出 COS 路径；若您需要将训练输出的模型一键发布至模型仓库，则需要将模型输出保存至 /opt/ml/model 路径下，平台会在训练结束后将该路径下的数据上传至 COS 路径；若您选择的是 CFS 等文件系统作为训练存储，您也可以不配置训练输出，直接将训练输出写到挂载的 CFS 文件路径中。

另外，在您配置任务的过程中，底部会实时显示您当前任务配置的价格，请注意关注。所有信息配置完成后，即完成了任务创建。

内置大模型预置流程说明

任务式建模内置了多种大模型精调模板，您可以直接一键启动内置大模型精调任务，详细的最佳实践可以参考[精调内置开源大模型](#)。以下是内置字段说明：

预置存储路径设置

第一行“平台 CFS”：系统默认为您配置了精调该大模型的配套训练代码。

第二行“平台 CFS”：系统默认为您配置了一份精调该大模型的示例数据；若您希望使用自定义业务数据精调该大模型，可删除此行，并在底部添加其他存储来源。

第三行“平台 CFS”：系统默认为您配置了平台内置模型。

第四行“用户 CFS”：此处需选择您的 CFS 文件系统和源路径，“容器挂载路径”为系统默认填充您无需修改。若您需要使用其他 CFS 文件系统作为训练输出，则您可以删除本行再添加。

注意：若您使用自己的业务数据进行精调，需要使用如下平台约定的格式或者遵循 llamafactory 的 dataset_info.json 数据配置文件描述，详情请 [单击跳转](#) 查看。

训练数据格式 ✕

格式示意

```
{
  "system": "xxxx",
  "conversation": [{"prompt": "xxxx", "response": "xxxxx"}]
}
```

格式要求

1. 支持文件格式为 jsonl，编码仅支持 UTF-8，文件大小不超过200M，训练数据路径下的多个文件会被合并训练
2. jsonl文件内一行表示一组样本数据

[确定](#)

预置启动命令

平台默认填充了启动命令，一般情况下您无需修改。

预置调优参数

平台提供多个预置参数，您可以直接修改超参 json 迭代模型。以下是超参释义：

超参	默认值	解释
Epoch	2	训练过程中的迭代轮次
BatchSize	1	每轮训练迭代中的样本数。BatchSize 越大，训练速度越快同时内存占用越高
LearningRate	1e-5	梯度下降过程中更新权重的超参，过高导致模型难以收敛，过低导致模型收敛速度过慢
Step	750	每跑多少个Step保存一次模型的checkpoint，保存checkpoint越多需要的存储空间越大
UseTilearn	true	是否要开启腾讯自研加速，"true/false"，设置为"true"会默认启用腾讯自研加速框架训练，其中3d并行加速需8卡以上，需要进行PP和TP参数设置，可参考angel-tilearn文档；设置为"false"会默认使用开源加速框架进行训练。仅部分模型开放
FinetuningType	Lora	用户可自定义选择精调训练模式“Lora/Full”，LoRA 在固定预训练大模型本身参数的基础上，对权重矩阵进行低秩分解，训练过程中只更新低秩部分参数；FULL 在精调过程中会全量更新模型参数，需要的训练资源更多
MaxSequenceLength	2048	最大文本序列长度，可根据您的业务数据长度进行合理设置。例如，如果大部分业务数据长度都在2048以下，则可设置 MaxSequenceLength 为 2048，大于2048的数据都会被截断为2048，可降低 GPU 显存压力
GradienAccumulationSteps	1	huggingface trainer参数，默认为1，提升batchsize
GradientCheckPointing	True	huggingface trainer参数，默认True，时间换显存的策略，开启后优化显存但训练速度变慢
DeepspeedZeroStage	z3	DeepSpeed ZeRO 阶段配置，可选值["z0", "z2", "z2_offload", "z3", "z3_offload"]，默认值z3；仅部分模型开放

ResumeFromCheckpoint	Ture	是否自动从已有的 checkpoint 文件恢复训练，默认值为 True，表示若输出目录存在 checkpoint 文件，从最新的 checkpoint 恢复继续训练；设置为 False 表示将重新训练。设置为 False 且输出目录非空，则会报错，建议训练输出路径使用空目录，若需要开启强制覆盖需要手动增加一条"overwrite_output_dir": true 参数开启覆盖文件
TilearnHybridTPSize	1	Tilearn 3D 并行参数，TP并行的维度，默认为1；仅部分模型开放
TilearnHybridPPSize	1	Tilearn 3D 并行参数，PP并行的维度，默认为1；仅部分模型开放

任务管理

最近更新时间：2025-05-14 17:16:02

简介

任务创建完成后，会在任务列表页面展示该条训练任务记录，列表展示了任务名称，训练镜像，机器来源，占用资源，标签，状态（任务状态有提交中，排队中，启动中，运行中，异常，停止中，已停止，已完成），运行时长，创建者，训练开始时间，更新时间，监控和操作。操作包括复制任务，Tensorboard，停止/启动任务和删除任务等。

任务式建模 上海 任务式建模简介

状态为运行中的训练任务正在产生费用，不使用时，请及时停止。

新建任务 查看我的任务 多个关键字用竖线 "|" 分隔，多个过滤标签用回车键分隔

名称	训练镜像	训练模式	机器来源	占用资源	标签	状态	运行时长	创建者	创建时间	训练开始时间	开始排队时间	操作
600	PyTorch: tlearn-1m0.9-torch2.3-py3.10-cuda12.4-gpu	DDP	从CVM机器中选择 willtest	算力规格: 2C4G A10*0.2 节点数量: 1个	1	运行中	2分52秒	t	2025-02-14 15:33:39	2025-02-14 15:40:23	2025-02-14 15:33:45	复制 Tensorboard 停止 更多
5	PyTorch: tlearn-1m0.9-torch2.3-py3.10-cuda12.4-gpu	DDP	从CVM机器中选择 willtest	算力规格: 4C8G A10*0.6 节点数量: 2个		排队中	0秒	W T E	2025-02-14 15:31:18	-	2025-02-14 15:31:25	复制 Tensorboard 停止 更多
76	自定义镜像: g-wsv8682-docker.pkg.coding.net/ekysaidyua ntest/demo/nb:1.0	MPI	从CVM机器中选择 188	算力规格: 1C1G HCC-A100*1 节点数量: 1个		异常	0秒	a 1 z	2025-02-14 10:03:13	-	2025-02-14 15:28:49	复制 Tensorboard 启动 更多

任务各个状态简介:

- 提交中: 任务成功发起提交请求至进入排队队列前。
- 排队中: 任务正常提交后进入排队队列。当状态为排队中时，点击可展示任务的排队时长和优先级。

当前任务已排队 11 分钟 50 秒，优先级 p0，点击可
查看任务排队队列，资源组排队策略说明请详见文档

算力规格: 4C8G A10*0.6
节点数量: 2 个

排队中 0秒

- 启动中: 任务成功出队，调度到了资源，开始初始化任务。
- 运行中: 任务成功加载，进入运行状态。
- 异常: 任务异常退出。
- 停止中: 用户手动停止任务，回收资源等过程。
- 已停止: 用户手动停止任务，任务正常终止。
- 已完成: 任务训练完成，正常退出。

查看我的任务

列表页右上角有查看我的任务快捷选项，勾选后仅展示当前登录的子账号创建的任务。

复制任务

当用户需要进行多个训练任务进行对比，以比较不同数据集或者不同超参配置的训练效果时，可选择复制训练任务，单击复制训练任务将跳转到创建任务窗口，用户可在原任务配置信息基础上进行简单修改，即可快速创建一个任务。

Tensorboard

平台支持读取 CFS 和 COS 中的 summary 数据生成 Tensorboard 监控面板，具体操作步骤如下:

- 单击任务列表 > 操作栏中的 Tensorboard 按钮，开始配置 Tensorboard 任务。

任务式建模 广州 任务式建模简介

① 状态为运行中的训练任务正在产生费用，不使用时，请及时停止。

新建任务 多个关键字用竖线“|”分隔，多个过滤标签用回车键分隔

名称	训练镜像	训练模式	机器来源	占用资源	标签	状态	运行时长	创建者	操作
aaa train- 1207:168	PyTorch: tilearn-orch2.3- py3.10-cuda12.4-gpu	DDP	从TIONE平台购买-按...	算力规格: 2C2G 节点数量: 1个		已停止	0秒	主账号 100 0	复制 Tensorboard 启动 更多

共 1 条 10 条 / 页 1 / 1 页

- 若您的训练输出文件存储在 CFS 中，则选择 CFS 为 summary 输出存储，选择存储路径为您当前训练任务配置的 CFS 文件系统和源路径（若您配置了多个，则需要选择其中一个），填写您的 summary 数据所在的容器内目录（平台默认会将您的 CFS 系统挂载的容器本地路径展示出来，您只需要填写 summary 文件具体所在的子目录），如下图所示：

查看 Tensorboard

选择summary输出存储 CFS COS

选择存储路径 tinafile (cfs-3nwy75tn) | /

summary目录 /opt/ml/ log

确定 取消

- 若您的训练输出文件存储在 COS 中，则选择 COS 为 summary 输出存储，需要注意的是，如果选择了 COS，存储路径默认为任务配置的训练输出 COS 路径 + /< job_id > + /summary，summary 目录默认为/opt/ml/summary/（容器内路径），用户无需配置，因此您需要在训练代码中提前将 summary 监控数据输出到/opt/ml/summary/中，如若不然，则无法创建 Tensorboard 面板。

查看 Tensorboard

选择summary输出存储 CFS COS

选择存储路径 yuanhao-gz-1256580188/output/train-904376754076564224/summary/

summary目录 /opt/ml/summary

确定 取消

- 完成配置后，单击确定，即可进入 Tensorboard 信息页，此时单击页面中的 **点击跳转**，即可跳转至 Tensorboard 面板。

查看 Tensorboard 编辑

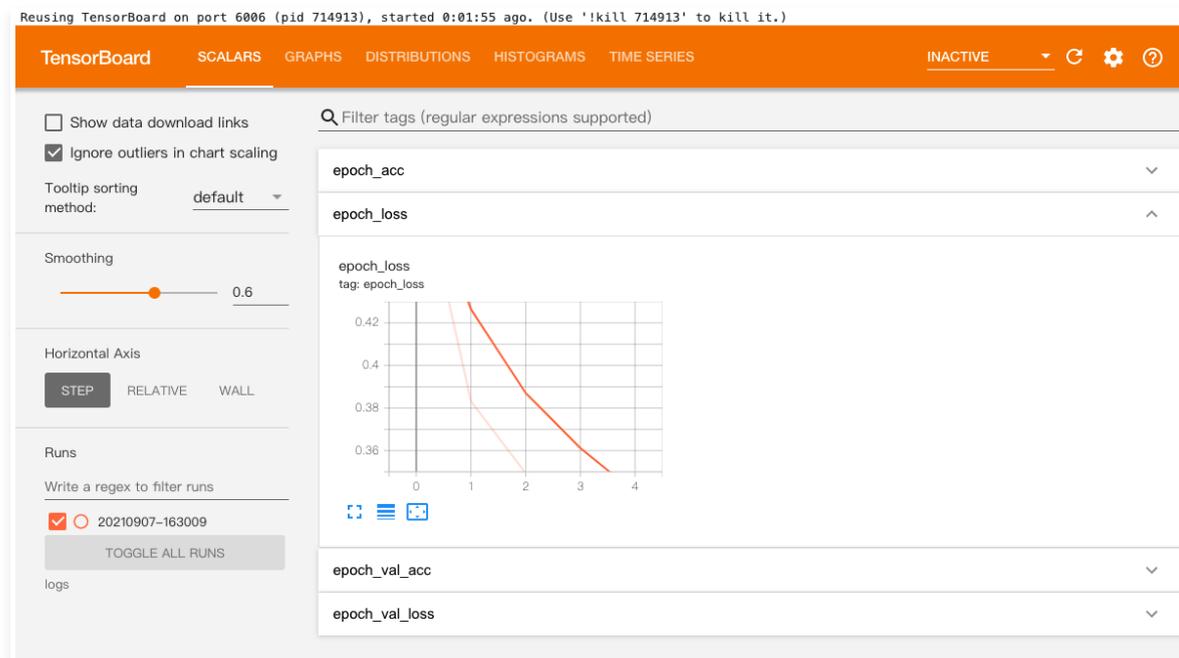
选择summary输出存储 CFS

选择存储路径 tinafile (cfs-3nwy75tn) | /

summary目录 /opt/ml/log

访问地址 点击跳转

启动Tensorboard 可能需要1 - 2分钟，在此期间，请不要关闭重定向页面。



当前 Tensorboard 服务免费提供给用户使用，您可能会遇到平台资源不足的情况，如您在此服务中遇见“不可用”的状态，还请您耐心等待，稍后重试。

停止/启动任务

正在训练的任务用户可以选择手停，停止后的任务可进行重新启动。

发布到模型仓库

训练完成的任务（仅支持按照要求将输出数据写到COS存储路径中的任务）可将输出发布到模型仓库。单击**发布到模型仓库**，可选择新建模型或者添加已有模型版本。

若选择新建模型，则需输入模型名称；若选择添加已有模型版本，则需要选择模型仓库中的模型，自动生成版本号（新增版本默认版本号自动加1）。

删除任务

训练完成/已停止的任务用户可手动删除记录。

任务详情

单击训练任务名称，可进入任务详情页。

基本信息

基本配置页显示了训练任务的基本配置信息、作业参数信息和资源配置信息。其中任务的标签可以进行更新。

实例列表

实例列表页展示了当前训练任务的实例清单，在此页面可查看实例 ID，占用资源，状态。此外，训练实例可支持直接进入实例容器，用户可在页面上单击**进入容器**，进入 webshell 页面，用户可执行命令行操作训练实例容器。

监控

任务监控有两个维度，第一是训练指标监控，第二是资源监控。其中训练指标监控页展示了用户自定义上报的监控指标信息。用户训练过程中的指标监控信息需要用户在训练脚本中按照平台规则进行上报（如何上报请查看 [Tikit-提交训练任务](#) 文档中的 `push_training_metrics()` 指令）。资源监控信息展示了训练任务的 CPU 使用率，MEM 使用率，MEM 使用量，GPU 使用率，显存使用率和显存使用量信息；整卡任务会展示细粒度卡维度监控指标，同时若您提交了整机训练任务，还可以查看节点 RDMA 使用监控信息。

训练指标上报代码示例：

```
## 在validate函数中，上传训练指标 acc1和acc5

def validate(val_loader, model, criterion, epoch, args):
    batch_time = AverageMeter('Time', ':6.3f')
    losses = AverageMeter('Loss', ':.4e')
    top1 = AverageMeter('Acc@1', ':6.2f')
    top5 = AverageMeter('Acc@5', ':6.2f')
    progress = ProgressMeter(
        len(val_loader),
        [batch_time, losses, top1, top5],
        prefix='Test: ')

    # switch to evaluate mode
    model.eval()

    with torch.no_grad():
        end = time.time()
        for i, (images, target) in enumerate(val_loader):
            if args.gpu is not None:
                images = images.cuda(args.gpu, non_blocking=True)
                target = target.cuda(args.gpu, non_blocking=True)

            # compute output
            output = model(images)
            loss = criterion(output, target)

            # measure accuracy and record loss
            acc1, acc5 = accuracy(output, target, topk=(1, 5))
            losses.update(loss.item(), images.size(0))
            top1.update(acc1[0], images.size(0))
            top5.update(acc5[0], images.size(0))

            # measure elapsed time
            batch_time.update(time.time() - end)
            end = time.time()

            if i % args.print_freq == 0:
                progress.display(i)

## 调用Tikit指令上传指标

    client.push_training_metrics(int(time.time()), {"acc1": float(format(top1.avg, '.3f')), "acc5":
float(format(top5.avg, '.3f'))}, epoch=epoch)
    # TODO: this should also be done with the ProgressMeter
    print('TIAcc - * Acc@1 {top1.avg:.3f} Acc@5 {top5.avg:.3f} Epoch={epoch}'
        .format(top1=top1, top5=top5, epoch=epoch))
```

```
return top1.avg
```

任务完成后，可以在“任务式建模/监控”下查看指标（可能会存在一点延迟）。

训练日志

训练任务页展示了实例维度的训练日志，日志可实时刷新查看，同时支持全文检索，亦可通过时间选择查看该训练任务的历史日志。平台默认为用户存储最长7天的日志，若您需要持久化存储使用，请使用 CLS 日志投递。用户也可在当前页选择开启 CLS 日志投递或者关闭 CLS 日志服务。同时，平台支持日志进行分类展示，可按照平台日志(健康检测)、平台日志(初始化)、训练代码日志 进行日志过滤，方便用户针对性的查看所关心的日志分类。

平台默认显示最近7天的训练任务日志，若您希望持久化存储日志或者使用日志检索等服务，请使用CLS日志投递，点击 开启

时间范围: 2025-05-12 22:09 ~ 2025-05-13 11:56

分类: 请选择

- 全部
- 平台日志(健康检测)
- 平台日志(初始化)
- 训练代码日志

	size	count	type	redop	root	time	algbw	busbw	#wrong	time	algbw	busbw	#wrong
20													
21	(B)	(elements)				(us)	(GB/s)	(GB/s)		(us)	(GB/s)	(GB/s)	
22	1073741824	268435456	float	sum	-1	4027.4	266.61	466.56	0	4027.5	266.60	466.55	0
23													
24													
25													
26													

事件

平台支持查看任务运行过程中产生的事件，用户可基于事件进行问题的诊断。

分布式训练使用指引

最近更新时间：2025-04-02 18:44:22

任务式建模支持提交多机多卡分布式训练任务，TI-ONE 平台支持多种分布式训练模式，包含 DDP、Ray、MPI、PS-Worker 等，本文档将阐述不同训练方式在 TI-ONE 平台中的使用方法。此外，大规模分布式训练任务需要使用 RDMA 技术来获得高吞吐、低延迟的网络通信，从而提升训练效率。本文最后也将介绍如何在 TI-ONE 平台上基于 RDMA 的高性能 GPU 实例进行分布式训练。

一、支持的分布式训练模式及使用说明

TI-ONE 任务式建模支持多种分布式训练模式，包含 DDP、Ray、MPI、PS-Worker 等，以下是平台支持的分布式训练模式及其使用场景概述：

分布式训练模式	模式介绍
DDP	PyTorch DistributedDataParallel (DDP) 是一种数据并行的分布式训练方法。通过 DDP 创建多个进程进行模型训练，通过 ring-all-reduce 的方法做进程通讯，完成梯度的交换及参数更新。
MPI	MPI 是一种基于信息传递的并行编程技术。平台支持用户发起 MPI 的分布式训练任务，同时也支持常见的 Horovod、DeepSpeed 等基于 MPI 的训练框架。
Ray	Ray 是一个开源的分布式计算框架，能够简化分布式机器学习的开发和部署。Ray 提供了一套 API 和基础设施，使得开发人员可以轻松地将单机训练代码扩展到分布式环境，支持数据并行和模型并行等多种并行策略。Ray 在大模型领域也常用作加速强化学习训练过程的分布式框架。
PS-Worker	PS (ParameterServer) 参数服务器训练是一种常见的数据并行方法，用于在多台机器上扩展模型训练。训练集群由 Worker 和 ParameterServer(ps) 组成。参数保存在 ps 上，在每一轮训练中，ps 将参数分发给 worker，worker 完成计算后将梯度回传给 ps 进行更新。

DDP 模式使用说明

PyTorch DistributedDataParallel (DDP) 训练模式支持在 PyTorch 中进行数据并行训练。数据并行模式可以跨多个进程同时处理多个数据批次，每个进程的输入数据批次不重叠，每个进程计算梯度并使用 ring-all-reduce 算法完成与其他进程的同步。

使用方式

- 在“任务式建模”界面创建训练任务时，选择训练模式为 DDP，并配置单节点资源和节点个数。其中启动命令会在每个节点上被执行。
- DDP 训练模式包含两种角色 Master 和 Worker。其中编号为0的是 Master (对应环境变量中 RANK=0)，承担保存模型的任务。
- TI 平台会根据任务配置创建对应的实例，并注入相关的环境变量，如任务中包含的实例组信息，以及当前实例的角色。Worker 会等待 Master 正常启动，网络通畅。以下是任务式建模启动时默认注入的环境变量列表：

内置环境变量

变量名	变量描述	示例
NODE_LIST	训练任务公共环境变量：任务节点的列表和节点的 GPU 卡数信息	NODE_LIST=timaker-xxxyyy-launcher.training-job.svc.cluster.local:1,timaker-xxxyyy-worker-0.training-job.svc.cluster.local:1
INDEX	训练任务公共环境变量：当前节点信息在 NODE_LIST 的索引，从0开始	INDEX=1
MASTER_ADDR	DDP 训练任务的 master 节点IP	MASTER_ADDR=10.35.110.11
MASTER_PORT	DDP 训练任务的 master 节点端口	MASTER_PORT=23456
WORLD_SIZE	DDP 训练任务的节点数	WORLD_SIZE=2
RANK	DDP 训练任务的当前节点	RANK=1
GPU_NUM	任务包含的 GPU 卡总数	GPU_NUM=2

GPU_NUM_PER_NODE	单个节点的 GPU 卡数	GPU_NUM_PER_NODE=1
------------------	--------------	--------------------

4. 训练过程中任意实例退出码非0则训练任务失败。所有实例成功则训练任务成功。

示例启动方式

启动 DDP 分布式训练的命令示例如下：

```
python -m torch.distributed.launch --nproc_per_node $GPU_NUM_PER_NODE --nnodes $WORLD_SIZE --node_rank $RANK --master_addr $MASTER_ADDR --master_port $MASTER_PORT
```

DDP 分布式训练参数与平台环境变量对应关系如下表所示：

变量名	变量描述
nproc_per_node	单个实例（机器）上运行的进程数，使用 GPU 时通常为每台机器上的 GPU 卡数，对应环境变量 GPU_NUM_PER_NODE 的值。
nnodes	对应环境变量 WORLD_SIZE 的值。
node_rank	对应环境变量 RANK 的值。
master_addr	对应环境变量 MASTER_ADDR 的值。
master_port	对应环境变量 MASTER_PORT 的值。

更多 DDP 训练参数请参考官网文档 [TORCHRUN](#)。

对于启动命令中涉及的内置环境变量，平台会在启动任务式建模任务时注入。而在 Notebook 或者本地调试代码时，需要开发人员先为对应环境变量赋值。为了方便调试使用，可以为对应环境变量设置默认值，示例如下：

```
MASTER_ADDR=${MASTER_ADDR:-localhost}
MASTER_PORT=${MASTER_PORT:-23456}
NNODES=${WORLD_SIZE:-1}
NODE_RANK=${RANK:-0}
GPU_PER_NODE=${GPU_NUM_PER_NODE:-$(nvidia-smi -L | wc -l)}
python -m torch.distributed.launch --nproc_per_node $GPU_PER_NODE --nnodes $NNODES --node_rank $NODE_RANK --master_addr $MASTER_ADDR --master_port $MASTER_PORT
```

MPI/Horovod 模式使用说明

MPI 是一种用于分布式并行训练的消息传递标准。平台支持用户发起 MPI 模式的分布式训练任务，并也支持常见的基于 MPI 通信的训练框架如 Horovod。而 Horovod 训练模式则是更原生适配了以 Horovod 框架进行训练的任务。本文以上述两种训练模式为例，介绍如何在机器学习平台上发起分布式训练任务。

使用方式

1. 在“任务式建模”界面创建训练任务时，选择训练模式为 MPI/Horovod，并配置单节点资源和节点个数。
2. MPI 训练模式和 Horovod 模式均包含 Launcher 和 Worker 两种角色，但两种角色均可执行训练任务，当任务仅配置一个实例时默认创建 Launcher 实例。
3. MPI 模式启动命令会在每个实例上执行。而 Horovod 模式启动命令仅会在 Launcher 实例上执行，Worker 实例的命令被配置为 sleep infinity 等待 Launcher 的命令。以下是任务式建模启动时默认注入的环境变量列表：

内置环境变量

变量名	变量描述	示例
-----	------	----

OMPI_MCA_orte_default_hostfile	MPI/Horovod 训练任务的节点信息文件	OMPI_MCA_orte_default_hostfile=/etc/mpi/hostfile
GPU_NUM	任务包含的GPU卡总数	GPU_NUM=2
GPU_NUM_PER_NODE	单个节点的GPU卡数	GPU_NUM_PER_NODE=1
NODE_IP_SLOT_LIST	任务包含的节点IP和对应卡数信息（仅支持用于配置启动命令）	NODE_IP_SLOT_LIST=9.0.255.56:1,9.0.255.118:1

- TI 平台会根据任务配置创建对应的实例组，并注入相关环境变量，给出任务中包含的实例组信息，以及当前实例的角色。
- 训练过程中任意实例退出码非0则训练任务失败。所有实例成功则训练任务成功。

示例启动方式

其中 /etc/mpi/hostfile 的内容示例如下：

```
train-960258573108964736-7an39bddmfpc-launcher slots=1
train-960258573108964736-7an39bddmfpc-worker-0 slots=1
```

内容分为两列，第一列是实例的域名，第二列是实例上的进程个数。

启动 MPI/Horovod 分布式训练的命令示例如下：

```
# MPI方式启动
mpirun --allow-run-as-root -np $GPU_NUM -H $NODE_IP_SLOT_LIST python3 train.py --data-dir /opt/ml/input/data

# horovod方式启动
horovodrun -np $GPU_NUM -H $NODE_IP_SLOT_LIST --network-interface eth0 python3 train.py --data-dir /opt/ml/input/data
```

此外，DeepSpeed 框架支持使用 OpenMPI 格式的 hostfiles 来配置多节点计算资源，使用MPI启动 deepspeed 分布式训练操作实践请查看 [使用任务式建模运行 DeepSpeed 分布式训练指引](#)。

PS-worker 模式使用说明

PS (ParameterServer) 参数服务器训练是一种常见的数据并行方法，用于在多台机器上扩展模型训练。训练集群由 Worker 和 ParameterServer(ps) 组成。参数保存在 ps 上，在每一轮训练中，ps 将参数分发给 worker，worker 完成计算后将梯度回传给 ps 进行更新。

使用方式

- 在“任务式建模”界面创建训练任务时，选择训练模式为 PS-Worker，并配置单节点资源和节点个数。
- 平台提供的 PS-Worker 训练模式包含两种角色：ps 和 worker。ps 保存和更新参数，实例数量应>=1，worker 负责执行训练，实例数量应>=1。
- TI 平台会根据任务配置创建对应的实例，并注入对应的环境变量 TF_CONFIG，给出了任务中包含的实例组信息，以及当前实例的角色。实例通过读取 TF_CONFIG 得到任务中 ps/worker 的数量和地址，并通过 task 中的 type 得知当前实例所属的角色和编号。

环境变量

TF_CONFIG

```
{
  "cluster": {
    "ps": [
      "train-960252492096760832-7an13ppfli80-ps-0.train-100031385875.svc:2222",
      "train-960252492096760832-7an13ppfli80-ps-1.train-100031385875.svc:2222"
    ],
    "worker": [
```

```

        "train-960252492096760832-7an13ppfli80-worker-0.train-100031385875.svc:2222",
        "train-960252492096760832-7an13ppfli80-worker-1.train-100031385875.svc:2222"
    ]
},
"task": {
    "type": "ps",
    "index": 0
},
"environment": "cloud"
}

```

4. 训练过程中任意实例退出码非0则训练任务失败。所有实例成功则训练任务成功。

Ray 模式使用说明

Ray 是一个开源的分布式计算框架，能够简化分布式机器学习的开发和部署。Ray 提供了一套 API 和基础设施，使得开发人员可以轻松地将单机训练代码扩展到分布式环境，支持数据并行和模型并行等多种并行策略。

使用方式

1. 在"任务式建模"界面创建训练任务时，选择训练模式为 Ray，并配置 Head 节点资源和各组 Worker 节点资源和个数，Worker 最多可配置5组。
2. Ray 训练模式包含两种角色：Head 和 Worker。其中编号为0的是 Head 节点（对应环境变量中 RANK=0），负责协调整个集群的计算资源和任务调度，Worker 节点则执行具体的训练任务。
3. TI 平台会根据任务配置创建对应的实例，并注入相关的环境变量，以下是任务式建模启动时默认注入的环境变量列表：

变量名	变量描述	示例
HEAD_ADDR	Ray 集群的 Head 节点地址	HEAD_ADDR=train-1282671078021627392-9qu5j2n90b9c-head-0
HEAD_PORT	Ray 集群的 Head 节点端口	HEAD_PORT=6379
RANK	当前节点在集群中的序号，RANK0为 HEAD节点	RANK=0

示例启动方式

您只需要在您的代码中，使用 ray.init() 默认初始化即可，并把您的脚本执行命令配置到启动命令，我们会默认在 Head 节点上提交您的任务到集群中，您无需指定 Head 节点的地址。

以一个简单的计数任务为例，将如下代码保存为 `job.py`。

```

import ray

ray.init()

# 定义 Actor 类
@ray.remote
class Counter:
    def __init__(self):
        self.value = 0

    def increment(self):
        self.value += 1
        return self.value

```

```
# 创建 Actor 实例
counter = Counter.remote()

# 并发调用 Actor 方法
futures = [counter.increment.remote() for _ in range(10)]
results = ray.get(futures) # [1, 2, 3, ..., 10]

print("计数器结果:", results)
```

job.py 作为训练任务，放到您的 CFS 指定目录，并在任务式建模中选择挂载到 /opt/ml/code 目录，然后指定启动命令为：

```
cd /opt/ml/code; python job.py
```

在训练任务中，可以通过 INDEX 指定节点，以指定在 INDEX = 1 的节点执行如下函数为例：

```
@ray.remote(resources={"Rank:1": 0.001})
def f(a, b, c):
    return a + b + c
```

⚠ 注意：

1. 暂时不支持 Ray tensorboard 的查看。
2. Ray 的设计是基于整数资源调度的。例如，一个任务声明需要 1 个 CPU，Ray 会确保它独占一个完整的物理核心，避免资源竞争。因此建议在配置 Ray 资源组的 CPU 资源时避免使用小数核数，例如设置 0.7 核，Ray 会将您的设置向下取整，例如 0.7 会被截断为 0，这可能导致程序行为不符合预期（例如任务无法运行）。
3. 由于 TIONE 平台支持 GPU 碎片调度，例如 0.2 卡资源。在 Ray 集群内部，Worker 会把 0.2 卡当成一块完整的 GPU 来使用，可以通过 num_s_gpu=1 来设置。

二、发起 RDMA 网络加速训练

RDMA 是 kernel by pass 的一种通信技术，在多机通信场景可显著提升通信带宽。本文将介绍如何在 TIONE 平台任务式建模使用 RDMA 网络。

使用前提

1. 资源组中至少包含 2 台支持 RDMA 的高性能 GPU 节点。
2. 提交的分布式任务配置为大于等于 2 个节点，且每个节点配置为 8 卡整机 GPU 资源，平台会默认为该资源配置 RDMA 资源。
3. 平台 llm 内置镜像默认支持常见的 HCC 高性能 GPU 机型，自定义镜像需要安装用户态 RDMA 驱动，安装文档参考 [容器安装用户态 RDMA 驱动](#)。

如何确认 RDMA 是否生效

在平台上运行的多机任务，如果开启了 RDMA，则会有以下日志：

```
[0] NCCL INFO Channel 00/0 : 8[0] -> 0[0] [receive] via NET/IBext/0/GDRDMA
```

平台内置的环境变量

针对 HCC 机型多机训练场景会开启 RDMA，并且会内置以下 NCCL 环境变量，用户使用 TI 平台的时候，无需显式设置。

```
NCCL_IB_GID_INDEX=3
NCCL_IB_SL=3
NCCL_CHECK_DISABLE=1
NCCL_P2P_DISABLE=0
NCCL_IB_DISABLE=0
NCCL_LL_THRESHOLD=16384
NCCL_IB_CUDA_SUPPORT=1
```

```
NCCL_IB_HCA=mlx5_bond
NCCL_NET_GDR_LEVEL=2
NCCL_IB_QPS_PER_CONNECTION=4
NCCL_IB_TC=160
NCCL_PXN_DISABLE=1
NCCL_IB_TIMEOUT=24
NCCL_DEBUG=INFO
NCCL_SOCKET_IFNAME=eth0
GLOO_SOCKET_IFNAME=eth0
TCCL_TOPO_AFFINITY=4
```

NCCL 支持的环境变量参考文档 [Environment Variables](#)。

任务事件告警配置指引

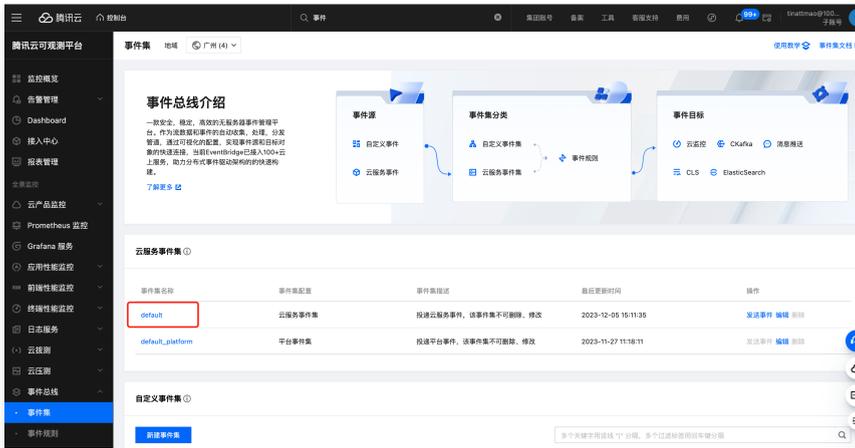
最近更新时间：2025-02-25 21:26:32

总体说明

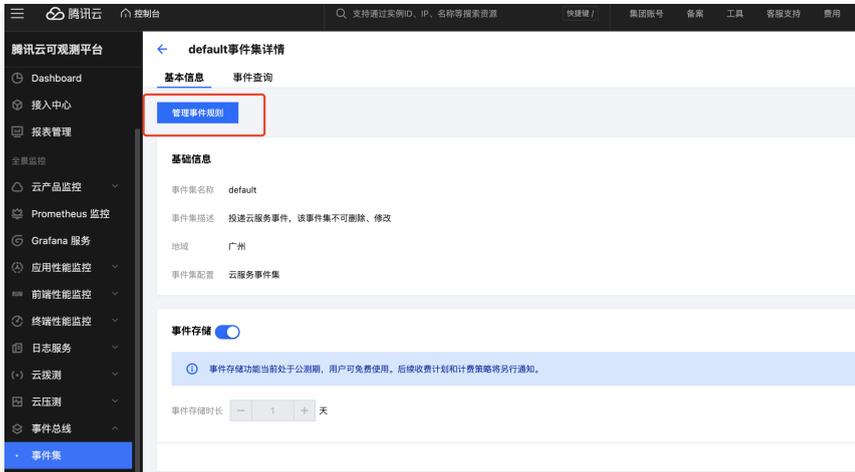
训练任务过程中产生的重要事件信息已经接入了 [腾讯云可观测平台事件总线](#)，用户可以在事件总线配置事件规则进行相关事件的告警推送。目前已经支持的事件为任务进入排队（taskScheduling）、任务进入运行（taskRunning）、排队运行完成（taskCompleted）、任务执行失败（taskExecuteFailed）、任务自动重启（taskRestarted）、任务被抢占（taskPreempted）、任务已停止（taskStopped）。

操作步骤

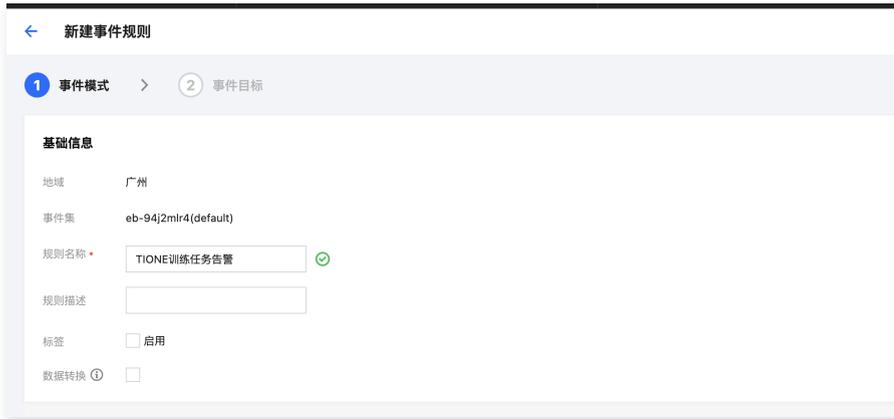
1. 进入 [可观测平台 > 事件集](#)，进入云服务事件集-default事件集（该事件集用于接收云服务告警、审计等类型事件，无地域概念，全地域云服务事件默认全部投递至广州，如希望配置告警，您需要在该事件集下绑定告警规则）。



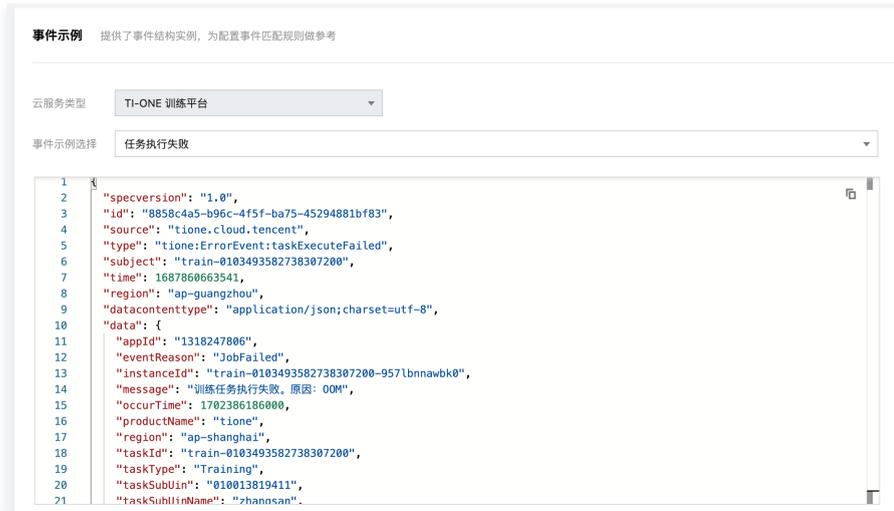
2. 单击管理事件规则，可进入事件规则列表，您可在这里配置事件的匹配规则，事件目标等。



3. 单击新建规则，填写规则名称，例如 TIONE 训练任务告警，选择云服务类型为 TI-ONE 训练平台。



下拉可以选择查看事件示例，表示云服务产品投递到事件总线的事件结构。后续的事件匹配规则可以按照上述事件结构自定义编写。



事件匹配规则可以根据表单模式或者自定义事件模式，推荐使用自定义事件模式。TIONE 平台当前支持针对部分实例（例如部分训练任务）和全部实例（全部训练任务，新增的训练任务自动加入规则告警队列），同时平台还支持按照云产品标签或者任务创建者进行事件规则匹配，以下是部分示例：

○ 按照事件类型匹配：

以下规则表示当前主账号下所有的训练任务当发生 taskExecuteFailed 或者 taskCompleted 两类事件时，会发出告警。

```

{
  "source": "tione.cloud.tencent",
  "type": [
    "tione:ErrorEvent:taskExecuteFailed"
    "tione:ErrorEvent:taskCompleted"
  ]
}

```

○ 按子账号 UIN 匹配：

以下规则表示子账号010013819411创建的训练任务当发生 taskExecuteFailed、taskCompleted、taskRestarted 或者 taskPreempted 任一事件时，会发出告警。

```

{
  "source": "tione.cloud.tencent",
  "data": {
    "taskSubUin": "010013819411"
  }
}

```

○ 按照标签匹配：

以下规则表示同时满足标签为 **部门:算法研究**和**环境:测试** 的训练任务当发生 `taskExecuteFailed`、`taskCompleted`、`taskRestarted` 或者 `taskPreempted` 任一事件时，会发出告警。

```
{
  "source": "tione.cloud.tencent",
  "data": {
    "tags": [
      {
        "contain": ["部门:算法研究;", "环境:测试;"]
      }
    ]
  }
}
```

以下规则表示标签为 **部门:算法研究** 或者 **环境:测试** 的训练任务当发生`taskExecuteFailed`、`taskCompleted`、`taskRestarted`或者 `taskPreempted`任一事件时，会发出告警。

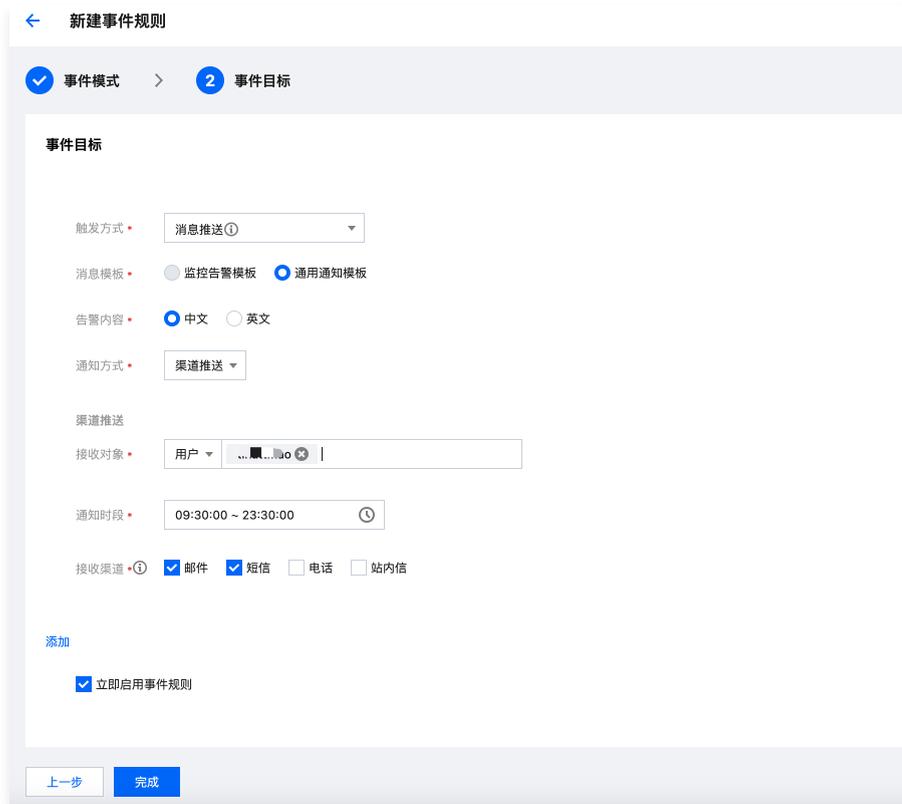
```
{
  "source": "tione.cloud.tencent",
  "data": {
    "tags": [
      {
        "contain": ["部门:算法研究;"]
      },
      {
        "contain": ["环境:测试;"]
      }
    ]
  }
}
```

以上所有支持的类型都可以组合匹配，例如：以下规则表示同时满足标签为 **部门:算法研究**和**环境:测试** 且创建者子账号为**010013819411**的训练任务当发生 `taskExecuteFailed` 事件时，会发出告警。

```
{
  "source": "tione.cloud.tencent",
  "type": [
    "tione:ErrorEvent:taskExecuteFailed"
  ],
  "data": {
    "tags": [
      {
        "contain": ["部门:算法研究;", "环境:测试;"]
      }
    ],
    "taskSubUin": "010013819411"
  }
}
```



4. 完成事件匹配后，单击下一步，开始配置事件目标（也就是告警渠道），选择触发方式为消息推送，通知方式为渠道推送，选择接收对象（可以选择一个或者多个告警接收账号），选择接收渠道（支持邮件，短信，电话，站内信），单击完成后即可返回事件规则页面。



5. 以上都配置完成后，满足告警触发的事件发生后，即会收到事件通知，如下所示（示例为邮件通知）。

开发机

开发机简介

最近更新时间：2025-02-28 19:54:02

概述

说明：原 Notebook 模块已于2025-02-25升级为开发机，请您关注。

开发机是 TI-ONE 为开发者量身打造的灵活的在线开发工具，支持通过 Jupyter Notebook 或者 VSCode 多种 IDE 进行在线编码，也支持通过 SSH 连接开发机远程开发。您可以在开发机中完成数据准备、数据预处理、算法调试与模型训练，同时我们也提供了多种内置主流开发环境，您可以一键创建，开箱即用。

核心特性

- 提供多种资源规格供用户自由选择，支持各类资源灵活切换，降低使用成本。
- 内置多种 WebIDE，且支持 SSH 连接远程开发。
- 内置 Tikit，用户可以在开发机中向 TI 提交训练任务。
- 内置多种镜像，支持自定义安装第三方库；支持导出镜像，一键保存自定义环境。
- 支持生命周期脚本，用户可以在创建/启动开发机实例时运行预设的 shell 脚本。
- 支持与 Git 存储库对接，方便协同开发与版本控制。

创建实例

最近更新时间：2025-05-14 17:03:53

操作场景

本文档将向您演示如何在 TI-ONE 中创建一个开发机实例。

操作步骤

1. 登录 [TI-ONE 控制台](#)，单击菜单栏的**开发机**，页面将跳转至开发机的实例列表页面，此页面将罗列用户创建的所有开发机实例。
2. 在开发机实例列表页，单击左上角**新建**，跳转至创建开发机实例的设置页面。填写说明如下：
 - **名称**：开发机名称，不超过60个字符，仅支持中英文、数字、下划线 "_"、短横 "-"，只能以中英文、数字开头。
 - **地域**：此字段不可修改，将自动显示平台选择的地区。
 - **镜像**：您可以选择启动开发机的镜像，支持的镜像列表请查看 [内置训练镜像列表](#)。其中tilearn-llm相关训练镜像内置了最新版本Angel加速组件tilearn-llm，可直接用于部分开源大模型训练的加速，详细使用指引请 [Angel 训练加速功能介绍](#)。
 - **计费模式**：可选择按量计费和包年包月，若选择按量计费，则需要选择算力规格和节点数量；若选择包年包月，则需要先创建资源组，购买节点，相关操作请查看 [资源组管理](#)，选择完资源组后选择对应的计算资源。平台支持的计费规格请查看 [计费概述](#)。
 - **存储路径设置**：可选择数据集、高性能云硬盘、SSD 云硬盘和 CFS（包含 CFS Turbo）文件系统和 GooseFSx 和 GooseFS。存储路径可配置多个，为了确保开发机实例可以正常使用，选择第一个存储路径会挂载到开发机的默认工作目录 `/home/tione/notebook` 下。
 - 选择数据集，则需要下拉选择在数据中心注册的数据集，配置容器挂载路径（注意：数据集不能作为第一个存储路径设置，也就是不能作为开发机的默认工作目录）。
 - 选择高性能云硬盘或者 SSD 云硬盘，平台会将申请的硬盘挂载到用户开发机容器的工作目录中，可申请的硬盘存储卷最小值为10GB且为10的倍数（云硬盘仅支持按量计费的开发机实例）。
 - 选择 CFS 文件系统，需要选择 CFS 文件系统，填写 CFS 文件系统的源目录和容器挂载路径，平台会将该 CFS 文件系统的源目录挂载到用户指定的容器挂载路径中。
 - 若选择的资源组类型为从 CVM 机器选择，则还能选择 GooseFSx 和 GooseFS。若选择 GooseFSx，需选择 GooseFSx 实例、配置 GooseFSx 目录和容器挂载路径。若选择 GooseFS，需选择 GooseFS 集群、命名空间、COS 子路径，配置容器挂载路径。
 - **标签**：可为开发机添加标签，支持添加多个。
 - **高级设置（默认收起）**：
 - **环境变量**：可以添加多个环境变量。
 - **CLS 日志服务**：用户可以自行选择是否开通 CLS 日志服务。
 - **生命周期配置**：选择是否使用生命周期脚本。
 - **Git 存储**：此为可选项，用户可以前往 Git 存储库-新增存储库进行配置。
 - **自动停止**：开启该选项后，该实例将在运行时长超过您选择的时长后自动停止，停止状态计算资源不再收费，存储资源仍会收费，请注意费用产生。自动停止时间以小时为单位，最小为1小时，最大为24小时。
 - **SSH 连接**：您可以选择是否启用 SSH 连接，启用后您可以在其他机器上访问本实例。您需要填写发起 SSH 登录机器上的 `~/.ssh/id_rsa.pub` 文件内容。若该文件不存在，可用 `ssh-keygen` 命令生成。发起 SSH 登录时，请注意检查私钥是否配对。如需从多台机器发起 SSH 登录需要填写多个公钥，您可以添加多个（按回车键可输入多个）。
 - **VPC 和子网**：平台支持为开发机容器配置 VPC 和子网，当选择了 VPC 和子网后，开发机容器将可访问同一网络配置下的其他云产品。
 - **直接访问 internet**：开发机默认为不使用 VPC，即不为开发机容器配置 VPC，该情况下默认可访问公网；若选择了 VPC 和子网，则会为用户的开发机容器配置该私有网络，配置后开发机容器可访问同一 VPC 网络下的其他云服务。选择 VPC 和子网后，可选择关闭**直接访问 internet**，此时开发机容器将无法访问公网（计费模式为包年包月场景下，不需要选择该选项，开发机实例默认可访问公网）。
3. 单击**创建**，开发机列表中将新增一条实例记录。当实例“状态”由**启动中**变为**运行中**时，单击**打开**进入开发机实例内部。

管理实例

最近更新时间：2025-03-10 14:27:12

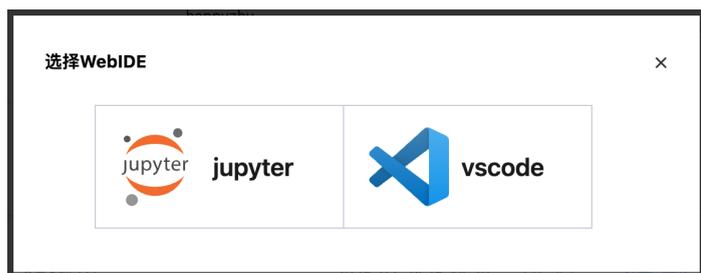
操作场景

本文档将向您演示在 TI-ONE 中，如何对开发机实例进行实例管理、资源管理和数据管理。

实例管理

完成**新增实例**后，开发机列表中将新增一条实例记录。您可以在此页面查看实例的名称、计费模式、占用资源、标签、状态、运行时长、最近编辑时间、监控与日志等。您还可以对开发机实例进行打开、停止、编辑、复制、删除等操作。

- 单击**监控与日志**栏，可查看 开发机 资源监控情况和日志详情。
 - 监控指标支持 CPU 使用率、MEM 使用率、GPU 使用率、显存使用率、系统磁盘使用率（当实例所在节点有本地数据盘时，指的是当前开发机实例所在 Pod 所分配的系统磁盘使用情况；当实例所在节点没有本地数据盘时，指的开发机实例所使用的节点系统盘使用率）
- 单击**打开**进入开发机实例内部，当前选择通过 Jupyter Notebook 或者 VSCode 两种 IDE 打开进行在线编码。



- 单击**停止**开发机将会停止运行。实例停止后**非挂载的持久化存储路径下的数据将被清空**，请注意数据保存，可将数据转存到持久化路径，或者在实例停止前保存镜像。
- 单击**编辑**可以对部分配置信息进行修改。
- 单击**删除**此实例记录销毁。
- 单击**复制**可一键复制实例配置，快速新建开发机。
- 单击**保存镜像**可以将该开发机实例保存为镜像，您可以选择其中一个 kernel 环境进行保存，支持保存到容器镜像服务个人版和企业版（请注意保存的镜像中不会包含挂载的外部存储设备，如云硬盘、CFS、CFS Turbo、GooseFSx、GooseFS 路径下的文件和数据下的文件和数据）。镜像构建记录可单击开发机 实例名称，在镜像构建记录页面可查看。
- 单击**远程连接**可以查看该开发机实例的远程访问地址。
- 实例支持**自动停止**设置，开启该选项后，该实例将在运行时长超过您选择的时长后自动停止，停止状态计算资源不再收费，存储资源仍会收费，请注意费用产生。自动停止时间范围为1 - 24小时，数值为整数。新增实例时，您可以设置自动停止时间，在实例列表中，您可以单击运行时长旁的小**小时**钟，开启、禁止或重设自动停止时间。

资源管理

TI-ONE 的开发机支持后付费模式下各种资源型号的切换，您可以根据需求进行资源调整，降低使用成本。操作步骤如下：

- 在开发机列表页面，选定您需要切换资源的实例，单击**停止**，释放当前计算资源。
- 实例停止后，单击**编辑**，进入设置页面，您可以重新进行资源选择，选定后单击**确定**。
- 资源更新后，单击**启动**即可重启开发机。

注意

- 按量计费的实例按照所选的资源配置与使用时长计费。成功切换资源后，切换前的资源不再计费，平台按照更新后的资源配置与使用时长计费。
- 实例停止（重启）后**非挂载的持久化存储路径下的数据将被清空**，请注意数据保存，可将数据转存到持久化路径，或者在实例停止（重启）前保存镜像。
举例：若您的开发机实例配置了 `/home/tione/notebook` 为容器挂载路径，则实例停止（重启）后，只有 `/home/tione/notebook` 该路径下的数据为持久化的，其他路径下的数据都会被清空。

存储路径设置 ①

请确保您选择的存储实例（CFS、GooseFS或者GooseFSx）和纳管资源组的节点网络互通，其中本地数据盘和GooseFSx仅支持挂载一个实例；配置的第一个容器挂载路径会做为默认工作目录

存储类型	CFS文件系统	源路径 ①	容器挂载路径 ①	操作
⋮ CFS	v- [] 4 ↕ ↻	[] /	/home/tione/notebook	删除

+ 添加

云硬盘数据管理

在按量计费场景下，若您选择高性能云硬盘或者 SSD 云硬盘，您可以在新建时选择存储卷大小，对于已运行的实例，您可以根据需求对存储大小进行调整。操作步骤如下：

1. 在开发机列表页面，选定您需要调整存储大小的实例，单击**停止**。
2. 实例停止后，单击**编辑**，进入设置页面，您可以调大存储大小，选定后单击**确定**。
3. 资源更新后，单击**启动**即可重启开发机。

⚠ 注意

您可以调大硬盘存储大小，但无法减小硬盘存储大小。如果要减小硬盘的大小，请创建一个具有所需大小的新开发机实例。在开发机停止运行时，云硬盘也会产生存储费用，该费用在实例删除后停止收取。

本地盘数据管理

在包年包月场景下，针对部分自带本地盘的机型（大部分为裸金属机型），开发机创建时可在页面上指定节点调度，调度后即将该节点的本地盘挂载到容器中作为存储目录（您依然可以自定义容器内的挂载路径）。实例停止后，该节点本地盘的数据不会被清理，您需要手动在开发机中清理数据。此外，不同开发机之间共享本地盘数据。

SSH 网络配置指引

最近更新时间：2025-02-25 21:26:32

背景介绍

开发机实例支持通过 SSH 远程连接，远程连接可提供公网访问地址和 VPC 内访问地址，其中公网访问地址无需配置，平台会默认生成，如下图所示：



VPC 内访问地址一般提供给具有内网访问要求的客户使用，平台默认展示实例 Pod 内 IP 地址，该 IP 会随着开发机实例重启而改变，长期稳定的连接建议用户前往 [SSH 网络配置](#) 进行私有连接打通，以下是详细的配置指引。

私有连接配置

1. 进入 [SSH 网络配置](#) Tab 页，私有连接配置要求该账号拥有 CreateVpcPrivateLink、DeleteVpcPrivateLink、DescribeVpcPrivateLinks、DescribeVpcPrivateLink、ModifyVpcPrivateLink 接口权限，可联系主账号或者 CAM 管理员进行配置。进入后单击 [添加私有连接](#)：

1.1 选择需要打通的 VPC 和子网。

1.2 您也可以为当前私有连接地址配置自定义映射域名，注意，需要用户访问终端自行进行域名解析。



1.3. 配置完成后单击 [保存](#)，则会在用户账号下自动创建一个终端节点，请注意：

- 该终端节点需要独立计费，详细计费方式请查看 [终端节点计费](#)。
- 私有连接配置一旦完成，则对当前地域下的所有开发机实例均生效。



2. 配置完成后，在开发机实例列表页，单击 [远程连接](#)，则可查看当前实例的 VPC 内访问地址和所生效的 VPC 和子网信息，如下图所示。您可以在本地客户端直接复制该地址连接该实例。

- 一个地域下可创建多个私有连接，一个私有连接对应一个访问地址。
- 一个实例可以关联多个访问地。

- 一个 VPC 和子网仅能创建一个私有连接。
- 一个实例可以生成多个访问地址。

开发机 上海

实例列表 生命周期配置 Git存储库 SSH网络配置

新建 查看我的实例 创建新用户

名称	训练镜像	机器来源	占用资源	标签	状态	运行时长	创建者	最新编辑时间	监控	操作
实例名称	内置镜像: tlearn-ilm0.8-torch2.1-py 3.10-cuda12.1-gpu	从TIONE平台购买-按量计费	计算资源: 2C2G CFS: 源目录		运行中	5小时44分49秒		2025-02-14 10:00:44		打开 停止 保存镜像 远程连接 更多
实例名称	内置镜像: tlearn-ilm0.9-torch2.3-py 3.10-cuda12.4-gpu	从CVM机器中选择-188	计算资源: 2C4G CFS: 源目录		运行中	20小时24分33秒		2025-02-13 19:23:29		打开 停止 保存镜像 远程连接 更多
实例名称	内置镜像: tlearn-ilm0.9-torch2.3-py 3.10-cuda12.4-gpu	从CVM机器中选择-188	计算资源: 2C4G CFS: 源目录		运行中	22小时12分48秒		2025-02-13 17:35:15		打开 停止 保存镜像 远程连接 更多

远程连接地址

公网访问地址 VPC内网访问地址

访问命令: `ssh -p 10329 root@1.1.2`

仅限以下VPC调用: VPC vpc- (vpc- (ic)、子网 3(sut

访问命令: `ssh -p 10329 root@h-2.2.2.com`

仅限以下VPC调用: VPC ac: (vpc- (rdinia)、子网 acl 18gbbr, 上海五区)

确定 取消

闲置回收策略配置指引

最近更新时间：2025-05-29 21:00:21

背景介绍

为了提升资源利用率，平台支持为开发机实例配置闲置自动回收策略，即管理员可以设置当前地域的开发机闲置策略，当资源利用率未达到要求时，会强制回收以释放计算资源。

配置指引

进入 **闲置回收策略** Tab 页，配置和查看闲置回收策略要求该账号拥有 CreateRecyclePolicy、DescribeRecyclePolicies、DeleteRecyclePolicy、ModifyRecyclePolicy 接口权限，可联系主账号或者 CAM 管理员进行配置。单击新建策略，可支持配置回收策略和排除策略。其中回收策略表示当实例满足回收条件时，实例将会自动停止，释放计算资源；排除策略表示满足排除条件的实例，不受回收策略控制，也就是即使符合闲置回收条件，也不会被强制停止。

配置回收策略

支持按照 CPU 利用率、内存利用率和 GPU 利用率配置实例的资源指标条件，其中这些指标条件可按照任意或者所有进行逻辑组合。当选择任意时，代表当满足配置的任意指标条件时，即可触发自动停止；当选择所有时，代表当满意配置的所有指标条件时，才可触发自动停止。

① 如下策略所示，当配置了如下策略时：

满足以下任意指标判断条件，并且持续时长 > 1 小时，实例将自动停止。

IF CPU 利用率 <= 10%

IF 内存利用率 <= 10%

IF GPU 利用率 <= 10%

表示当某一开发机实例的 CPU 利用率或者内存利用率或者 GPU 利用率持续 1 小时以上小于等于 10%，该实例即会触发自动停止。

开发机 上海 开发机简介

实例列表 生命周期配置 Git存储库 SSH网络配置 **闲置回收策略**

回收策略 ①

满足以下 任意 指标判断条件，并且持续时长 > 1 小时，实例将自动停止

IF CPU 利用率 <= 10 %

IF 内存利用率 <= 10 %

IF GPU 利用率 ① <= 10 %

+ 添加条件

排除策略 ①

IF 实例名称 = myNotebook

+ 添加条件

配置排除策略

支持按照 **实例名称** 过滤选择被排除的实例，也就是在该排除策略中配置的开发者实例，将不受上述回收策略管控，如上图所示，实例名称为 myNotebook 的开发者实例将不受已配置的回收策略管控，也就是即使该实例达到了回收的条件，也不会被强制停止。

实例列表

1. 当该地域配置了闲置回收策略以后，则在实例列表上方展示如下提示文案，单击**自动停止策略**，用户可查看配置的指标条件和持续时长（用户需要有 DescribeRecyclePolicies 权限）
2. 当开发者实例由于闲置自动停止后，状态为转为已停止，同时 hover 可展示“已触发闲置实例自动停止”文案提示，如下图所示。



告警指引

开发者闲置回收相关事件已接入腾讯云事件总线，当前已接入的事件如下：

- **开发者实例闲置自动停止 (devInstanceAutoReleasedOnIdle)**：当开发者实例由于闲置而自动停止后，即会发送该告警事件。
- **开发者实例即将闲置回收 (devInstancetobeReleasedOnIdle)**：开发者实例即将回收前30分钟和24小时，分别会发送该告警事件。

详细的事件告警配置指引请查看 [操作步骤](#)（该指引以任务事件为例，开发者事件的配置方式一样，仅需要替换上述开发者事件）。

使用生命周期脚本

最近更新时间：2025-02-25 21:26:32

生命周期脚本配置规则

生命周期配置提供 SHELL 脚本，在用户创建 发机实例或每次启动开发机实例时运行，可以帮助用户安装自定义依赖，个性化配置开发机环境。

生命周期配置遵循以下规定：

- 创建脚本：第一次新建后启动发机实例会运行的脚本，只会运行一次。
- 启动脚本：每次启动开发机实例时都会运行的脚本，包括第一次创建时。
- 每个脚本 BASE64 编码后不能超过16384个字符。
- 每个脚本将以 root 用户的角色运行。
- 每个脚本的 \$PATH 环境变量为 `/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin`
- 每个脚本最长运行时间为5分钟，超过5分钟开发机将启动失败，请避免在脚本中安装大型依赖包。可在详情页中查看失败原因如“启动脚本超时”。
- 如果脚本出错，开发机也将启动失败，可在详情页中查看具体失败原因。
- 如果脚本是从自己的编辑器复制到 TI-ONE 网页上的，请确保编辑脚本的编辑器使用 Unix 风格的编排。

生命周期脚本最佳实践

以下是使用生命周期配置的一个实践案例：

- 生命周期脚本以 root 用户权限运行，发机进程以 tione 用户运行。如果需要切换用户，可以在脚本中运行 `sudo -u tione` 切换到 tione 用户。
- 发机使用 conda 管理多内核，可以激活 `conda env` 来为不同的内核安装依赖包。

例如：在 `conda_python3` 的内核中安装 Python 依赖包 `fire`，可以编写如下启动脚本：

```
#!/bin/bash
sudo -u tione -i <<'EOF'

# This will affect only the Jupyter kernel called "conda_python3".
source /opt/conda/bin/activate python3

# Replace fire with the name of the package you want to install.
pip install fire
# You can also perform "conda install" here as well.

source /opt/conda/bin/deactivate

EOF
```

例如：在所有内核中都安装 `fire` 依赖包，可以这样编写脚本：

```
#!/bin/bash
sudo -u tione -i <<'EOF'

# Note that "base" is special environment name, include it there as well.

for env in base /opt/conda/envs/*; do
    source /opt/conda/bin/activate $(basename "$env")

    # Installing packages in the Jupyter system environment can affect stability of your tione
    # Notebook Instance. You can remove this check if you'd like to install Jupyter extensions, etc.
    if [ $env = 'JupyterSystemEnv' ]; then
        continue
    fi

    # Replace myPackage with the name of the package you want to install.
    pip install fire
done
```

```
# You can also perform "conda install" here as well.  
  
source /opt/conda/bin/deactivate  
done  
  
EOF
```

开发机内使用镜像工具

最近更新时间：2025-05-28 20:32:22

操作场景

本文将演示如何在开发机实例中拉取镜像、构建镜像、推送镜像、运行镜像等操作。

开发机内使用 PodMan

开发机实例中已经内置了 PodMan 工具执行镜像的基本操作，PodMan 的使用命令与 Docker 基本一致。

登录镜像仓库

```
# 登录到私有镜像仓库
podman login <registry-url> -u <username> -p <password>
# 示例
podman login tione.tencentcloudcr.com --username ***** --password *****
```

拉取镜像

```
# 从仓库拉取镜像
podman pull <image-name>:<tag>
# 示例
podman pull tione.tencentcloudcr.com/tione-public-images/notebook-conda-cpu:ubuntu22.04-miniforge3-py38
```

查看镜像

```
# 列出本地镜像
podman images
# 查看镜像详细信息
podman inspect <image-id|image-name>
# 示例
podman inspect tione.tencentcloudcr.com/tione-public-images/notebook-conda-cpu:ubuntu22.04-miniforge3-py38
```

构建镜像

构建镜像时建议采用分阶段构建，减少存储占用。

```
# 使用Dockerfile构建镜像
podman build -t <image-name>:<tag> <context-path>
# 示例：
podman build -t myapp:1.0 .
```

推送镜像

```
# 标记镜像
podman tag <local-image> <registry>/<repository>:<tag>
# 推送镜像到仓库
podman push <registry>/<repository>:<tag>
# 示例：
podman tag myapp:1.0 registry.example.com/myteam/myapp:1.0
podman push registry.example.com/myteam/myapp:1.0
```

运行容器

目前暂不支持指定 gpu 运行镜像的功能。

```
# 运行容器
podman run [options] <image-name>
# 可选项 -d 后台运行 -v 卷挂载 --name 指定容器名
# 示例:
podman run -d --name myContainer tione.tencentcloudcr.com/tione-public-images/notebook-conda-
cpu:ubuntu22.04-miniforge3-py38

#查看运行中的容器 可选参数 -a:包含已停止的容器
podman ps
```

删除镜像或容器

```
# 删除镜像
podman rmi -f <镜像ID或名称> # -f为强制删除
podman image prune -a # -a 删除未被引用的镜像

# 删除容器
podman rm -f <容器ID或名称> # -f 为强制删除
podman container prune # 删除所有退出的容器

# 如果 Podman 存储损坏或需要彻底清理,可直接删除 graphroot 和 runroot 目录
# 示例: rm -rf /var/lib/containers/storage/
```

PodMan 工具的关键配置修改

存储配置文件路径: /etc/containers/storage.conf。

```
# 容器运行时临时存储路径
runroot = "/var/run/containers/storage"

# 镜像层/容器读写层存储路径
graphroot = "/var/lib/containers/storage"
```

有关 PodMan 工具的更多指令,请参见 [官方文档](#)。

Git 存储库

最近更新时间：2025-04-18 21:31:52

简介

进行模型训练和代码调试时，往往需要直接对接用户的远端代码仓库，TI-ONE 为用户提供统一的 Git 存储库管理能力，用户可以在实际使用任务式建模或者开发机时关联配置好的 Git 存储库，从而能够快速拉取代码或者连接代码仓库。

功能描述

存储库管理

新增存储库

前往训练工坊 > Git 存储库页面，单击新增存储库，在弹窗中输入存储库的名称，目标存储库的 URL 以及存储库分支的名称（可选）。如果目标存储库为需要验证的私有存储库，您必须准确输入所需的用户名（或邮箱）与密码，用于凭证验证。

新增存储库

名称 *

请输入不超过60个字符，仅支持中英文、数字、下划线"_"、短横"-"，只能以中英文、数字开头

Git 存储库 URL *

存储库分支名称

Git 凭证 创建新秘钥

用户名/邮箱 *

密码 *

无秘钥

创建完成后可以在列表页看到配置的存储库列表，如下图所示：

名称	Git 存储库 URL	存储库分支名称	秘钥	创建时间	更新时间	操作
git	git@github.com:xxxxxx/xxxxxx.git		○	2025-04-17 17:19:02	2025-04-17 17:19:02	编辑 删除
git	git@github.com:xxxxxx/xxxxxx.git	897	○	2024-11-29 13:39:54	2025-04-17 14:41:35	编辑 删除
git	git@github.com:xxxxxx/xxxxxx.git	est/	◇ 1	2025-02-18 11:09:50	2025-04-17 14:27:53	编辑 删除

编辑已创建的 Git 存储库

前往训练工坊 > Git 存储库页面，单击编辑，可以对已创建的 Git 存储库的秘钥进行编辑。

注意

一旦创建，存储库的名称和 URL 将不能修改，如果输入错误，建议删除重新新建。

在开发机中关联 Git 存储库

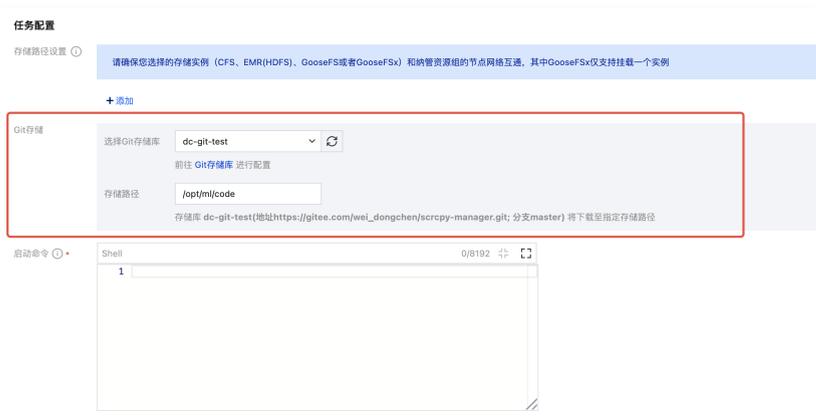
前往开发机页面，单击新建实例，您可以下拉选择已创建的 Git 存储库，可以添加除默认存储库以外的其他存储库，最多添加3个其他存储库。

配置完成并且启动开发机后，首先会将关联的 Git 存储库中的文件下载到默认工作目录中，此外，用户也可以在开发机内免密连接该存储库，方便提交和同步代码。



在任务式建模中关联 Git 存储库

前往任务式建模页面，单击新建任务，您可以下拉选择已创建的 Git 存储库，同时需要指定存储路径，配置完成并且启动任务以后，会将关联的 Git 存储库下载到指定存储路径中。



模型管理

模型管理简介

最近更新时间：2024-08-22 15:18:27

模型管理是 TI-ONE 训练平台提供模型纳管、优化的模块，包含模型仓库和模型优化两个子模块。

模型仓库是 TI-ONE 平台模型纳管的统一入口，用户可对平台训练出的模型及用户自有的本地模型进行纳管。

- 支持训练出的模型导入及第三方模型的增删改查。
- 支持对模型优化加速过的模型进行删除、修改和查看。
- 支持对纳管的模型、相应的推理代码和配置文件进行服务发布封装。
- 支持对模型、模型版本以及版本下子目录进行管理。
- 支持查看模型关联的在线服务。
- 支持查看模型关联的批量预测任务。
- 支持对模型热更新相关功能。

模型优化是 TI-ONE 平台模型推理加速的模块，使用了 TI-ACC 的能力，可对模型仓库里纳管的模型进行推理优化加速，降本增效，模型优化后的模型仅能支持在 TI-ONE 平台模型服务进行推理服务使用。

- 支持模型仓库纳管的模型进行推理加速。
- 支持快速进行批量模型加速。
- 支持对 CV、NLP、以及 OCR 等领域常用模型进行加速。
- 优化级别支持无损、FP16 两种级别。
- 支持对多输入节点、固定&动态 shape 的模型进行优化。
- 支持输出模型推理部分的加速报告。
- 支持对优化后的模型保存到模型仓库进行服务发布。

模型仓库使用

最近更新时间：2024-12-12 22:06:42

模型仓库支持导入模型。列表页管理着不同的模型以及模型的不同版本；导入窗口支持从任务式建模直接导入模型以及从 COS 上传。

仓库列表

模型仓库列表中可查看模型及对应的版本，并对模型进行发布在线服务等操作，分为模型列表和优化模型列表，模型列表为客户上传上来的原始模型列表，优化模型列表为经过模型优化后保存的优化后的模型列表。

The screenshot shows the 'Model Warehouse' interface. At the top, there's a search bar and a dropdown menu for '广州'. Below that, there are two tabs: '模型列表' (Model List) and '优化模型列表' (Optimized Model List). The '模型列表' tab is active, showing a table of models. The table has columns for '模型名称' (Model Name), '标签' (Tag), '创建时间' (Creation Time), and '操作' (Action). There are two main rows: 'tikittest' and 'detection'. The 'detection' row is expanded to show a detailed table of model versions. This detailed table has columns: '模型版本' (Model Version), '模型来源' (Model Source), '任务名称 / ...' (Task Name / ...), '算法框架' (Algorithm Framework), '模型格式' (Model Format), '运行环境' (Runtime Environment), '模型指标' (Model Metrics), 'QAT模型' (QAT Model), '模型保存路径' (Model Save Path), '模型清理' (Model Cleanup), '创建时间' (Creation Time), and '操作' (Action). One version is highlighted with a blue '内置' (Built-in) tag. A context menu is open over the '操作' column of this version, showing options like '批量预测' (Batch Prediction), '导入模型' (Import Model), '查看关联在线服务' (View Associated Online Services), '查看关联跑批任务' (View Associated Batch Tasks), and '下载SDK' (Download SDK).

模型名称	标签	创建时间	操作																																																												
tikittest		2023-03-29 16:26:29	删除 编辑标签																																																												
detection		2023-03-27 18:53:36	删除 编辑标签																																																												
<table border="1"><thead><tr><th>模型版本</th><th>模型来源</th><th>任务名称 / ...</th><th>算法框架</th><th>模型格式</th><th>运行环境</th><th>模型指标</th><th>QAT模型</th><th>模型保存路径</th><th>模型清理</th><th>创建时间</th><th>操作</th></tr></thead><tbody><tr><td>v1 导入成功</td><td>COS</td><td>-</td><td>PyTorch</td><td>TorchScript</td><td>pytorch1.9.0-py38(cpu)</td><td>-</td><td>否</td><td>demo-1256580185/inferdemo/pytorch/detection/</td><td>未开启</td><td>2023-03-27 ...</td><td>优化模型 发布在线服务 更多</td></tr><tr><td>torch19</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>2023-03-24 16:15:14</td><td>删除 编辑标签</td></tr><tr><td>happy-recommend</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>2023-03-23 20:09:12</td><td>删除 编辑标签</td></tr><tr><td>kim-tf-recommend</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>2023-03-23 14:34:35</td><td>删除 编辑标签</td></tr></tbody></table>				模型版本	模型来源	任务名称 / ...	算法框架	模型格式	运行环境	模型指标	QAT模型	模型保存路径	模型清理	创建时间	操作	v1 导入成功	COS	-	PyTorch	TorchScript	pytorch1.9.0-py38(cpu)	-	否	demo-1256580185/inferdemo/pytorch/detection/	未开启	2023-03-27 ...	优化模型 发布在线服务 更多	torch19										2023-03-24 16:15:14	删除 编辑标签	happy-recommend										2023-03-23 20:09:12	删除 编辑标签	kim-tf-recommend										2023-03-23 14:34:35	删除 编辑标签
模型版本	模型来源	任务名称 / ...	算法框架	模型格式	运行环境	模型指标	QAT模型	模型保存路径	模型清理	创建时间	操作																																																				
v1 导入成功	COS	-	PyTorch	TorchScript	pytorch1.9.0-py38(cpu)	-	否	demo-1256580185/inferdemo/pytorch/detection/	未开启	2023-03-27 ...	优化模型 发布在线服务 更多																																																				
torch19										2023-03-24 16:15:14	删除 编辑标签																																																				
happy-recommend										2023-03-23 20:09:12	删除 编辑标签																																																				
kim-tf-recommend										2023-03-23 14:34:35	删除 编辑标签																																																				

- 支持对模型进行删除，单击删除按钮，此模型及对应的模型版本皆被删除。
- 支持对模型标签进行编辑，单击编辑标签按钮，可进行标签编辑。
- 支持对模型版本进行优化，在某一个模型版本点击优化模型，可跳转到模型优化新建任务页面进行优化。
- 支持对模型版本发布封装，在某一个模型版本单击发布在线服务，可跳转到模型服务新建服务页面进行模型服务发布。
- 支持对模型版本进行批量预测，在某一个模型版本单击批量预测，可跳转到批量预测新建任务页面进行批量预测。
- 支持对模型版本进行导入子版本，在某一个模型版本单击导入模型，可在当期模型版本目录下新增目录和子版本，目前仅支持对 Savedmodel 格式进行导入子版本，主要用于 tf-serving 热更新使用。
- 支持查看关联在线服务，单击查看关联在线服务可查看模型版本关联的模型服务以及状态；
- 支持查看关联跑批任务，单击查看关联跑批任务可查看模型版本关联的跑批任务以及状态；
- 支持下载 sdk 操作，单击下载 sdk 按钮可打开 COSBrowser 进行下载。您需要先安装 COSBrowser 客户端。
- 单击版本对应的删除按钮可删除对应的模型版本，需要注意的是，删除后不可恢复，已关联服务组并且服务在启动运行中的模型版本不支持删除，已关联批量预测任务并且任务在启动运行中的模型版本不支持删除。删除时可单击查看关联服务组或者查看关联跑批任务进行查看以确定是否要进行删除。

导入模型

进入模型管理 > 模型仓库，单击导入模型 按钮开始导入模型。

导入新模型

导入模型

导入方式

模型名称
请输入不超过60个字符，仅支持中英文、数字、下划线"_"、短横"-", 只能以中英文、数字开头

标签

模型来源

选择任务

模型格式

运行环境来源

模型来源路径

模型指标

模型保存路径

模型移动方式 剪切 复制

1. 按要求填入模型名称、自定义设置标签。(标签用于从不同维度对资源分类管理。用户需要先从控制台进行 [标签设置](#)。对于标签设置您可以直接单击 [标签管理使用指南](#)查看。)

2. 从任务导入:

- 选择训练任务名称。
- 选择训练任务之后，然后选择对应的模型格式，如有任务对应的算法框架、模型来源路径、模型指标信息，信息将会被自动填充，模型指标填充的是最新上报指标。
- 运行环境您可选择内置的，也可通过自定义选择的方式从容器镜像服务中拉取，优化后的模型目前仅支持内置运行环境运行。
- 模型文件也会自动获取此任务的模型文件，您可选择剪切或者复制的方式对此模型文件进行处理，处理完成后将自动为您生成 COS 路径。

3. 从 COS 导入:

- 选择模型格式，目前模型格式支持 Savedmodel、Frozen Graph、TorchScript、Detectron2、PyTorch 原生、PMML、ONNX、MMDetection、Hugging Face。
- 输入您期望上传的模型对应的模型指标
- 运行环境您可选择内置的，支持内置的运行环境对应上述支持的模型格式，也可通过自定义选择的方式从容器镜像服务中拉取。
- 上传模型文件: 需要您上传模型文件，具体请参考 [模型包规范](#)。

导入新版本

导入模型

导入方式：

选择模型：

模型版本 ^①：

模型来源：

选择任务：

模型格式：

运行环境来源：

模型来源路径 ^①：
算法框架

模型指标：

模型保存路径：

模型移动方式： 剪切 复制

1. 您可选择一个模型后导入这个模型的新版本，模型版本号将自动在历史最高版本号基础上+1，如V2、V3等。

2. 从任务导入：

- 选择训练任务名称。
- 选择训练任务之后，然后选择对应的模型格式，如有任务对应的算法框架、模型来源路径模型指标信息，信息将会被自动填充，模型指标填充的是最新上报指标。
- 运行环境您可选择内置的，也可通过自定义选择的方式从容器镜像服务中拉取，优化后的模型目前仅支持内置运行环境运行。
- 模型文件也会自动获取此任务的模型文件，您可选择剪切或者复制的方式对此模型文件进行处理，处理完成后将自动为您生成 COS 路径。

3. 从 COS 导入：

- 选择模型格式，目前模型格式支持 Savedmodel、Frozen Graph、TorchScript、Detectron2、PyTorch原生、PMML、ONNX、MMDetection、Hugging Face。
- 输入您期望上传的模型对应的模型指标
- 运行环境您可选择内置的，支持内置的运行环境对应上述支持的模型格式，也可通过自定义选择的方式从容器镜像服务中拉取。
- 上传模型文件：需要您上传模型文件、推理脚本代码和服务 API 配置文件。推理代码和配置文件请分别命名为 config.json、model_service.py，具体请参考 [模型包规范](#) 和 [推理脚本代码示例](#)。

完成以上配置后，单击确定按钮，模型便导入成功。

导入至当前版本

导入模型

导入方式：

选择模型 [ⓘ] *

模型版本 [ⓘ] *

标签

模型来源：

选择任务 *

模型格式 *

运行环境来源 *

模型来源路径 [ⓘ] *

算法框架

模型指标

模型导入目录 [ⓘ] *

模型移动方式： 剪切 复制

1. 您可选择一个模型版本后导入这个模型版本的子版本，目前仅支持 Savedmodel 格式，主要用于 tf-serving 热更新，您可以选择已经存在于模型列表里的 Savedmodel 格式模型和版本。

2. 从任务导入：

- 选择 Savedmodel 格式模型的训练任务名称。
- 选择训练任务之后，如有任务对应的算法框架、模型来源路径模型指标信息，信息将会被自动填充，模型指标填充的是最新上报指标。
- 运行环境您可选择内置的，也可通过自定义选择的方式从容器镜像服务中拉取，优化后的模型目前仅支持内置运行环境运行。
- 模型导入目录为追加在该模型版本下的存储目录，该目录需要是模型版本主目录或者其子目录，可以选择在模型版本目录追加子版本的目录，平台会自动根据时间戳生成新目录，您可以根据需求选择是否勾选。
- 开启模型清理后将在模型版本对应的“模型保存路径”目录下寻找“model”目录，并根据模型数量保留上限的设定值，保留 model 目录里最新的N个模型。
- 模型文件也会自动获取此任务的模型文件，您可选择剪切或者复制的方式对此模型文件进行处理，处理完成后将自动为您生成 COS 路径。

3. 从 COS 导入：

- 选择模型格式，目前模型格式支持 Savedmodel。
- 运行环境您可选择内置的，支持内置的运行环境对应上述支持的模型格式，也可通过自定义选择的方式从容器镜像服务中拉取。
- 模型来源路径，请选择您要导入的模型子版本对应的文件夹。
- 输入您期望上传的模型对应的模型指标
- 模型导入目录为追加在该模型版本下的存储目录，该目录需要是模型版本主目录或者其子目录，可以选择在模型版本目录追加子版本的目录，平台会自动根据时间戳生成新目录，您可以根据需求选择是否勾选，会将您选中的模型来源路径下的所有文件复制或粘贴到模型导入目录。
- 开启模型清理后将在模型版本对应的“模型保存路径”目录下寻找“model”目录，并根据模型数量保留上限的设定值，保留model目录里最新的N个模型。
- 会根据选择的模型移动方式将您选中的模型来源路径下的所有文件复制或粘贴到模型导入目录

完成以上配置后，单击确定按钮，导入至现有版本便导入成功。

模型优化

原始模型列表的模型可在模型优化模块进行推理加速，请在参考 [模型优化](#) 对模型进行推理加速。

发布在线服务

原始模型在原始模型列表对应的模型版本操作中，单击**发布在线服务**进行服务的发布封装，优化模型在原始模型列表对应的模型版本操作中，单击**发布在线服务**进行服务的发布封装，后续详细步骤请参考 [模型服务](#) 对模型进行部署和使用。

模型优化使用

最近更新：2024-09-26 10:45:21

模型优化是 TI-ONE 平台模型推理加速的模块，使用了 TI-ACC 的能力，可对模型仓库里纳管的模型进行推理优化加速，降本增效，目前限时免费，但模型优化后的模型仅能支持在 TI-ONE 平台模型服务进行推理服务使用。

优化任务列表

优化任务列表中可查看优化任务名、优化进度以及加速比等信息，并支持对优化后的模型进行保存到模型仓库等操作。优化进度会包含“启动中”、“加速中”、“加速完成”等状态，一般该过程需要持续几分钟的时间。

任务名称	模型名称	模型版本	模型格式	状态	创建时间	加速比	输入/输出张量	目标设备	计算模式	优化进度	加速比	操作
test	1246cat_test	v3	MMDetection	启动中	2022-10-19 14:52:03	FP16	input_0:shape(1*3*1088*900) input_1:shape(1*4*1*16)	T4	限时免费	加速完成	4.31x	停止 保存到模型仓库 更多
happy	happy-demo	v3	Hugging Face	启动中	2022-10-14 12:34:15	FP16	input_0:shape(1*4*1*16) input_1:shape(1*4*1*16)	T4	限时免费	加速完成	3.22x	停止 保存到模型仓库 更多
happy	happy-demo	v2	MMDetection	启动中	2022-10-14 12:34:15	FP16	input_0:shape(1*3*1088*900)	T4	限时免费	加速完成	4.14x	停止 保存到模型仓库 更多
test	test-hugging_face	v1	Hugging Face	启动中	2022-10-12 15:15:01	FP16	input_0:shape(1*4*1*16) input_1:shape(1*4*1*16)	T4	限时免费	加速完成	3.71x	停止 保存到模型仓库 更多
happy	happy-模型	v5	Hugging Face	启动中	2022-10-12 14:58:15	FP16	input_0:shape(1*4*1*16) input_1:shape(1*4*1*16)	V100	限时免费	加速完成	3.51x	停止 保存到模型仓库 更多

- 支持对优化中（启动中、加速中）的任务进行停止，单击停止按钮，此任务即被停止。
- 支持对加速完成的任务进行保存和发布服务，单击保存到模型仓库按钮，此优化任务得到的优化模型即保存到模型仓库优化模型列表页面。
- 支持对加速失败的任务进行重新加速，单击更多 > 重新加速按钮，可对该优化任务进行重新编辑和加速。
- 支持对优化任务进行删除，单击删除按钮，此任务即被删除。
- 支持对标签进行编辑，单击编辑标签按钮，可进行标签编辑。

新建优化任务

当前支持TorchScript、MMDetection、Detectron2、Hugging Face-Stable Diffusion、ONNX、Savedmodel、Frozen Graph 格式的模型，参考如下指引进行接入。

← 新建优化任务

- 1. 模型优化和模型发布封装的机型需保持一致，目前仅支持GPU-T4、A10和V100的优化（部分模型格式支持A100或者L20），需使用对应机型进行发布封装；
- 2. 模型优化目前是免费模式，免费额度为3个，即同时加速中的模型上限为3个，单地域总数上限为1000；
- 3. 目前仅支持在TI平台-模型服务进行推理服务部署的优化。

基本信息

任务名称 *

请输入不超过60个字符，仅支持中英文、数字、下划线"_"、短横"-", 只能以中英文、数字开头

标签 ① ×

+ 添加 ② 键值粘贴板

模型配置

模型1

模型来源 *

模型格式 *

模型名称 *

模型版本 *

+ 添加

优化配置

优化级别 ① *

部署机型 *

优化模型存储 *

优化的模型将存储到该COS地址

▶ 其他配置

▶ 推荐模型专业参数设置

1. 输入**任务名称**，当仅优化一个模型及版本时，任务名称为客户填写的任务名称，当添加了多个模型或者版本时，会生成多个优化任务，每个任务的名称为客户填写的任务名称+“-模型名称”+“-模型版本”+“#”+序号；
2. 根据需要进行**标签添加**；
3. 选择**模型来源**，与模型仓库的模型来源字段一致，会在模型名称根据选择的模型来源进行对应的筛选。
4. 选择**模型名称**，即待优化的模型，支持一个优化任务添加多个模型和版本，当添加了多个模型或者版本时，会生成多个优化任务，每个任务的名称为客户填写的任务名称+“-模型名称”+“-模型版本”+“#”+序号；
5. 选择**模型版本**，即待优化的模型版本，支持选择多个版本和对应的文件。

6. 选择**模型文件**，即待优化的具体模型文件，不同格式的模型文件后缀不同，请参考具体提示。

7. **QAT模型**，即量化感知训练。量化感知训练(QAT)是一种用于深度学习的技术，用于训练可以量化的模型，以便部署在计算能力有限的硬件上。QAT 在训练过程中模拟量化，让模型在不损失精度的情况下适应更低的位宽。与量化预训练模型的训练后量化 (PTQ)不同，QAT 涉及在训练过程本身中量化模型。

说明：

QAT 过程可以分解为以下步骤：

1. 定义模型：定义一个浮点模型，就像常规模型一样。
2. 定义量化模型：定义一个与原始模型结构相同但增加了量化操作(如 torch.quantization.QuantStubQ)和反量化操作(如 torch.quantization.DeQuantstub())的量化模型。
3. 准备数据：准备训练数据并将其量化为适当的位宽。
4. 训练模型：在训练过程中，使用量化模型进行正向和反向传递，并在每个epoch 或 batch 结束时使用反量化操作计算精度损失。
5. 重新量化：在训练过程中，使用反量化操作重新量化模型参数，并使用新的量化参数继续训练。
6. Fine-tuning：训练结束后，使用 fine-tuning 技术进一步提高模型的准确率。

8. 填写 **“Tensor信息”** 字段：

- 将 **输入数据** 看作一个整体，需要在执行模型优化前，预先给出**输入数据**的形状信息，模型优化任务才能正常执行。
- **输入数据** 是以多维张量 (Tensor) 作为基本元素 (element) 的列表 (list)、字典 (dict)、元组 (tuple) 及他们的嵌套组合。
- 对于输入数据中的各个element，需分别给出他们的形状大小。对于每个element，采用 **“element:dtype(shape)”** 的格式表示其形状大小，每个element占一行。
- **element** 采用如下方式表示：
 - 对于 输入数据 的第 1 层：
 - 如果该层是list或tuple，则每个元素用input_0、input_1、... 来表示。(特别地，如果输入数据是单个Tensor，则用input_0表示)
 - 如果该层是dict，则每个元素用dict的key来表示。
 - 对于 输入数据 的第 n 层 (n > 1)：
 - 如果该层是list，则每个元素用 [0]、[1]、... 来表示。
 - 如果该层是tuple，则每个元素用 (0)、(1)、... 来表示。
 - 如果该层是dict，则每个元素用 [key] 来表示。
- **dtype** 是对应 element 的基本数据类型，如float、int32、array.char等。
- **shape** 是对应 element 的实际形状，有 3 种情况：
 - 固定大小：用一个 **size** 值表示，其中 **size** 由n个以*分隔的正整数表示。如 **“3*1024*1024”**，表示element是一个大小为3乘1024乘1024的3维张量。
 - 动态连续大小：用一个，(英文逗号) 隔开的2个 **size** 值表示。如 **“3*1024*2048,3*2048*2048”**，表示element是一个大小可变的3维张量，其大小最小为3乘1024乘2048，最大为3乘2048乘2048。
 - 动态离散大小：用一个 **size** 值列表表示。如 **“[3*1024*1024,3*2048*2048]”**，表示element是一个大小可变的3维张量，其大小有2种可能，分别为 3乘1024乘1024 或 3乘2048乘2048。
- 下面给出若干填写实例：
 - **eg.1** 输入数据是单个 Tensor
 - Tensor是固定大小，数据类型为float，实际形状是1乘3乘1024乘1024。
 - **“Tensor信息”** 字段共填写1行，填写 input_0:float(1*3*1024*1024)



- **eg.2** 输入数据是单层 list
 - 共3个element，3个element均为数据类型为int32的Tensor，实际形状均为最小为1乘4、最大为1乘384的动态连续大小。
 - **“Tensor信息”** 字段共填写3行，分别为input_0:int32(1*4,1*384)、input_1:int32(1*4,1*384)、input_2:int32(1*4,1*384)



Tensor信息 ⓘ *

input_0:int32(1*4,1*384)

input_1:int32(1*4,1*384)

input_2:int32(1*4,1*384)

+ 添加

○ eg.3 输入数据是单层 dict

- 共3个element, key分别为input_ids、input_mask、segment_ids, value均为数据类型为int32的Tensor, 实际形状均为最小为1乘128、最大为8乘128的动态连续大小。
- “Tensor信息” 字段共填写3行, 分别为input_ids:int32(1*128,8*128)、input_mask:int32(1*128,8*128)、segment_ids:int32(1*128,8*128)



Tensor信息 ⓘ *

input_ids:int32(1*128,8*128)

input_mask:int32(1*128,8*128)

segment_ids:int32(1*128,8*128)

+ 添加

9. 默认使用最新的加速库引擎版本即可, 每个格式最新的加速库引擎版本会有不同;

10. 优化级别可以选择 FP16 或无损 两种方式, 无损代表使用模型原始精度进行优化, FP16 代表转为 FP16 精度进行优化, 推荐使用FP16进行优化, 推理速度比无损要高, 并且一般不会造成精度下降。

11. 部署机型可以选择 T4、V100或A10 三种机型 (部分模型格式支持A100或者L20, 当选择L20的时候, 需要选择资源组; 当选择T4的时候, 可以选择限时免费 (按量) 或者资源组, 选择其他机型仅支持免费模式), 优化时选的部署机型需要和实际部署服务的机型保持一致, 即如果需要T4部署服务, 则这里需要选择T4机型。

12. 单击选择路径, 即选择优化后的模型要保存的位置, 该位置和原始模型的主目录不能有重合, 当对加速完成的模型单击保存到模型仓库时, 会在该路径下将原始模型目录的所有文件进行复制, 放在该路径下的新生成的 m-xxx/mv-xxx 文件夹里, 并生成优化后模型 tiacc.pt, 优化后的模型放到 model 文件夹下, 如果想要对优化后的模型进行部署, 需要符合 TI-ONE 平台的 模型包规范。

13. 单击确定, 即开始进行优化任务, 并返回到优化任务列表页。

优化任务详情

单击**优化任务名称**可进入优化任务详情页面，可以查看详细的任务信息和优化报告，并进行相应的操作。

← 优化任务详情

任务信息

任务名称	██████████	优化级别	无损
模型格式	Detectron2	输入节点信息	input_0[image]:float(3*1024*2048,3*208*208)
模型来源	COS	部署GPU型号	T4
模型名称	██████████	计费模式	限时免费
模型版本	v1		

任务信息

已加速时长 7分56秒

加速状态 加速完成

- [停止](#)
- [保存到模型仓库](#)
- [重新加速](#)
- [删除](#)

优化报告

```
{
  "software_environment": [
    {
      "software_environment": "pytorch",
      "version": "1.9.0"
    },
    {
      "software_environment": "cuda",
      "version": "11.1"
    }
  ]
}
```

模型包规范

最近更新时间：2024-08-22 15:41:26

模型包目录结构

原始模型包目录结构：

```
cos://bucket/train_models/mv-2302330437837096
├── model
│   └── model.pth
└── requirements.txt
```

优化模型包目录结构（优化后的模型保存到模型仓库时，会在优化模型存储路径下生成 m-xxx/mv-xxx 文件夹，并在该文件夹下将原始模型包拷贝并生成新的优化模型文件 model/tiacc.pt）：

```
cos://bucket/optimized_models_save/m-230233043/mv-2302330437837096
├── model
│   ├── model.pth
│   └── tiacc.pt (TI平台模型优化自动生成的优化后的模型)
└── requirements.txt
```

- model：存放模型的文件夹目录，该目录可以客户自己指定，但需要确保 model_service.py 可以正确加载；
- model.pth：原始模型文件，名称可以客户自己指定，但需要确保 model_service.py 可以正确加载；
- tiacc.pt：优化模型文件，使用 [模型优化](#) 功能后会自动生成，如需要使用优化后的模型进行推理和服务部署，需要在 model_service.py 推理脚本里增加对该模型读取 load 的逻辑，可参考 [推理脚本代码示例](#)；
- requirements.txt：第三方依赖包，如推理脚本有第三方依赖可以放在 requirements.txt 文件内，如没有可直接忽略，容器服务启动前，会利用 pip 安装 requirements.txt 里面的第三方依赖。
- 其他：其他相关依赖文件，可以自由放置并打包上传。

模型评测

模型评测简介

最近更新时间：2025-03-25 20:34:43

「模型评测」提供通过向导式的评测任务提交方式进行 LLM 模型评测。腾讯云 TI 平台支持两种评测方式，分别为主观评测和客观评测。

- **主观评测**：提供待评测模型推理后人工打分的功能，基于人工来评判模型效果；
- **客观评测**：全程无需人工参与，TI 平台将基于内置的开源评测集或者是用户自定义上传的开源评测集以及内置的指标（如pass@1、ROUG、F1等），提供自动评测的功能。平台内置开源数据集来帮助用户快速开启大模型评测体验；客观评测也支持在训练过程中对训出的模型效果进行轻量体验，对 checkpoint 模型进行服务部署，用户可在对话框内和大模型进行问答，初步查看模型效果。

说明：

通常，在进行最佳实践时，主观评测和客观评测可以结合起来。例如，其中一个场景可以是先在模型开发过程中，基于开源数据集（可以扩展至企业内部沉淀的标准化自有数据集）先进行客观评测获得一个较为不错的结果后，在模型发布前的最后阶段，使用主观评测再次进行效果验证，或模型已上线后的随时效果评测。

针对主观评测和客观评测后的模型，支持可视化对比模型效果。可基于多个模型多个指标通过雷达图进行横向对比，为用户提供直观的效果对比展示。

轻量体验

最近更新时间：2024-11-15 16:32:22

在训练过程中，对训出的模型效果，进行轻量体验，初步查看模型效果。

前提条件

已完成 [任务式建模](#) 的任务创建，已完成 CheckPoint 的输出。

操作步骤

1. 登录 [TI-ONE 控制台](#)，在左侧导航栏中选择 [训练工坊](#) > [任务式建模](#)，进入任务列表页面。
2. 在任务页面单击训练的任务名称，进入任务详情页面。
3. 单击 [Checkpoint](#) Tab页，选择要进行初步体验模型效果的 CheckPoint 卡片。



4. 单击**轻量体验**按钮，拉起弹窗，您选择所需的资源，可从您训练使用的资源组中选择。



5. 轻量体验经历创建中、启动中、就绪中的状态后，您可单击**轻量体验**，拉起弹窗体验模型效果。



6. 弹窗内，您可进行发送问题，进行对话体验模型效果。



客观评测

最近更新时间：2025-06-19 14:22:52

在训练过程中，对训出的模型 CheckPoint、CFS 里的模型、内置大模型，进行客观评测，初步查看模型效果。

平台内置开源评测集，用户可一键开启评测。也支持用户上传开源评测集，获得更多的评测体验。

使用平台内置的开源评测集

平台内置如下开源评测集，用户在平台可直接选择使用。

数学	GSM8K	小学数学题数据集，测试基础数学推理能力
	MATH	涵盖多种难度数学问题，测试高级数学能力
	MATH-500	解答高难度数据问题及数学推理测试
	AIME2024	解答15道高难度数学题，考察代数、几何、数论、组合数学等领域的深度理解和创造性解题能力
知识问答	MMLU	涵盖广泛学科知识，测试跨学科综合理解与推理能力
	ARC-c	科学问答数据集，挑战性较高，测试复杂科学问题的深度推理能力
	ARC-e	科学问答数据集，难度较 ARC-c 低，测试基础科学知识的理解与简单推理能力
语言理解生成	Hellaswag	常识推理数据集，测试上下文理解能力
	Xsum	极端摘要生成数据集，测试简洁摘要能力
	TyDiQA	多语言问答数据集，涵盖11种语言，测试多语言阅读理解与问答能力
复杂推理综合能力	BBH	BIG-Bench Hard，测试复杂推理能力
	GPQA Diamond	主要用于评估模型在复杂、跨学科知识上的推理能力
代码生成	HumanEval	代码生成评估数据集，测试代码生成能力
	MBPP	Python 编程问题数据集，测试代码生成能力

自定义上传开源评测集

支持自定义上传开源评测集，支持的开源评测集请参见 [评测集要求](#)。

和直接使用平台内置的开源评测集不同的是，准备好评测集后，创建任务时需要填写评测集在 CFS 中的路径。为方便任务的创建，在准备评测集阶段，您需要在开发机中挂载您的 CFS 文件系统，以获得评测时所需填写的路径。CFS 文件系统使用如下：

- 请您准备好 [CFS 文件系统](#)。您可以挂载您的 CFS 文件系统并启动开发机。假设已经准备好 CFS，且/data1 代表本地挂载的 CFS 根路径。
- 您可以根据自己需要，为希望用于评测的客观数据集创建本地对应的客观数据集文件夹。

说明：

为便于准确检测到您提供的路径对应的客观评测集种类，我们需要该评测集的目录包含评测集的名称（大小写均可）。

例如，我们可以将需要的数据集放置于 `/data1/test_data` 目录下：

```
cd /data1
mkdir -p test_data/agieval
mkdir -p test_data/arc-c
mkdir -p test_data/arc-e
mkdir -p test_data/bbh
mkdir -p test_data/c-eval
```

```
mkdir -p test_data/cmmlu
mkdir -p test_data/gsm8k
mkdir -p test_data/hellaswag
mkdir -p test_data/humaneval
mkdir -p test_data/math
mkdir -p test_data/mbpp
mkdir -p test_data/mmlu
mkdir -p test_data/tydiqa
mkdir -p test_data/xsum
```

3. 下载并解压 opencompass 客观数据集。

```
cd /data1
# 下载OpenCompass数据集压缩包
wget https://github.com/open-compass/opencompass/releases/download/0.2.2.rc1/OpenCompassData-complete-20240207.zip OpenCompassData-complete-20240207.zip
# 解压缩包
unzip OpenCompassData-complete-20240207.zip
# 此时，会因解压而多出目录 data，里面包含了OpenCompass相关的数据集
```

4. 将解压后的 opencompass 客观数据集拷贝到本地客观数据集文件夹。

```
cd /data1
# AGIEval 将评测目录 AGIEval/data/v1/ 下的所有文件
cp -r data/AGIEval/data/v1/* test_data/agieval

# ARC-c 将评测目录 ARC/ARC-c/ 下的 ARC-Challenge-Test.jsonl
cp data/ARC/ARC-c/ARC-Challenge-Test.jsonl test_data/arc-c

# ARC-e 将评测目录 ARC/ARC-e/ 下的 ARC-Easy-Test.jsonl
cp data/ARC/ARC-e/ARC-Easy-Test.jsonl test_data/arc-e

# BBH 依赖目录 BBH/ 下的所有文件
cp -r data/BBH/* test_data/bbh

# C-Eval 依赖 ceval/formal_ceval/ 下的所有目录
cp -r data/ceval/formal_ceval/* test_data/c-eval

# CMMLU 依赖 cmmlu/ 下的所有文件
cp -r data/cmmlu/* test_data/cmmlu

# GSM8K 依赖 gsm8k/ 下的 test.jsonl 和 train.jsonl
cp data/gsm8k/test.jsonl test_data/gsm8k
cp data/gsm8k/train.jsonl test_data/gsm8k

# Hellaswag 依赖 hellaswag/ 下的 hellaswag.jsonl
cp data/hellaswag/hellaswag.jsonl test_data/hellaswag

# HumanEval 依赖 humaneval/ 下的 human-eval-v2-20210705.jsonl
cp data/humaneval/human-eval-v2-20210705.jsonl test_data/humaneval
```

```
# MATH 依赖 math/ 下的 math.json
cp data/math/math.json test_data/math

# MBPP 依赖 mbpp/ 下的 mbpp.jsonl
cp data/mbpp/mbpp.jsonl test_data/mbpp

# MMLU 依赖 mmlu/ 下的所有文件
cp -r data/mmlu/* test_data/mmlu

# TyDiQA 依赖 tydiqa/ 下的 dev 目录
cp -r data/tydiqa/dev test_data/tydiqa

# Xsum 依赖 Xsum/ 下的 dev.jsonl
cp data/Xsum/dev.jsonl test_data/xsum
```

5. 在评测时，请对应填写您 CFS 实例上的数据集，例如 `/test_data/xsum`。

说明：

请您保障挂载不同的评测数据集，否则会导致重复的数据集计算指标被覆盖，使得评测结果无数据。

使用客观评测的具体方式及步骤

方式一：从任务式建模的 CheckPoint 处新建客观评测

前提条件：已完成 [任务式建模](#) 的任务创建，已完成 CheckPoint 的输出。

1. 登录 [TI-ONE 控制台](#)，在左侧导航栏中选择 [训练工坊](#) > [任务式建模](#)，进入任务列表页面。
2. 在任务页面单击训练的任务名称，进入任务详情页面。
3. 单击 **Checkpoint** Tab 页，选择要进行初步体验模型效果的 CheckPoint 卡片。



4. 单击 **客观评测** 按钮，拉起弹窗，填写任务名称、选择评测集，选择所需的资源，支持按量计费 and 包年包月。

客观评测

任务名称 4/256 ✔
 请输入不超过256个字符，仅支持中英文、数字、下划线"_"、短横"-", 只能以中英文、数字开头

评测集

[+ 添加](#) [点击了解内置评测集](#) [和查看自动上传评测集要求](#)

机器来源
 TI-ONE 仅收取软件费用，算力将使用您已购买的机器。 [产品定价](#)

资源组

资源申请

卡型号

GPU
 若需使用GPU，根据不同卡类型可填写0.1-1或1的整数倍。运行环境为平台内置GPU镜像时，卡数不能为0

CPU

内存

标签 ✕
[+ 添加](#) [🔗 键值粘贴板](#)

选择评测集时，可根据来源选择使用的评测集。

评测集

[+ 添加](#)

- 内置评测集
- 从CFS中选择评测集

5. 填好信息单击**新建**，会进入客观评测-推理中的状态，请您耐心等待。

demo_model_4... 🔍 轻量体验 🔄 客观评测-推理中

```

"conv_template_name": "baichuan2-chat"
"path": "/opt/ml/output/data/checkpoint-430"
"epoch": 0
"learning_rate": 0
"loss": 2.0445
"step": 430
    
```

6. 客观评测完成后，可单击**查看客观评测结果**。

demo_model_2... 🔍 轻量体验 🔗 查看客观评测结果

```

"conv_template_name": "baichuan2-chat"
"path": "/opt/ml/output/data/checkpoint-230"
"epoch": 0
"learning_rate": 0
"loss": 1.9305
"step": 230
    
```

7. 单击**查看客观评测结果**，跳转到**评测结果 Tab** 页，查看具体评测结果。

方式二：从客观评测 Tab 页处新建客观评测

1. 登录 **TI-ONE 控制台**，在左侧导航栏中选择**模型服务 > 模型评测**，单击**客观评测 Tab 页**，进入任务列表页面。



2. 单击**新建任务**，拉起新建页面。



所需填写的信息如下：

参数	说明
任务名称	客观评测任务的名称，按照界面提示的规则填写即可
备注	可按需为任务备注描述信息
地域	同账号下的服务按地域进行隔离，地域字段取值根据您在服务列表页面所选择的地域自动带入
评测集	<ul style="list-style-type: none"> 可选择 CFS 文件系统实例和评测数据集所在目录。您可参考上文进行数据集的准备。请您一个 CFS 路径挂载一个开源数据集，且 CFS 路径最后一层以开源数据集名来命名，如 “/test_data/ceval” 可选择内置评测集，一键开启快速评测

<p>选择待评测的模型</p>	<p>支持两种模型来源：</p>  <ul style="list-style-type: none"> ● 选择模型： 支持从训练任务中选择模型，选择该地域下的训练任务、该任务的 CheckPoint。 支持从 CFS 中选择模型：选择该地域下的 CFS 实例、填写模型所在的路径。 支持选择内置大模型：选择内置好的大模型。 ● 选择服务： 支持从 TI 平台的在线服务选择。 支持填写第三方服务地址进行评测。 <p>支持进行参数设置，支持配置推理超参、启动参数设置和性能参数设置。</p> <ul style="list-style-type: none"> ● 配置推理超参，推理超参支持如下： <ul style="list-style-type: none"> ○ repetition_penalty：用来控制重复惩罚。 ○ max_tokens：用来控制输出文本的最长数量。 ○ temperature：数值越高，输出越随机；数值越低，输出越集中和确定。 ○ top_p、top_k：影响输出文本的多样性，数值越高，生成文本的多样性越强。建议该参数和 temperature 只设置1个 ○ do_sample：确定模型推理时的采样方式，取值 true 时为 sample 方式；取值为 false 时为 greedy search 方式，此时，top_p、top_k、temperature、repetition_penalty 不生效。 ● 配置启动参数，可参考 服务部署参数填写指引 文档。平台配置默认参数 MAX_MODEL_LEN，指模型单次推理能处理的最大 token 数（平台默认8192），启动时设置过高可能引发显存溢出或性能下降，可根据任务需求合理调整该值。 ● 配置性能参数，平台配置默认参数 MAX_CONCURRENCY 和 MAX_RETRY_PER_QUERY。 <ul style="list-style-type: none"> ○ MAX_CONCURRENCY 指评测过程中同时向模型发起的请求数上限；设置过低可能导致模型吞吐量下降导致评测耗时较长，设置过高可能导致显存溢出或请求超时（平台默认24）。用户可根据任务需求合理调整该值。 ○ MAX_RETRY_PER_QUERY 指每条数据在请求推理服务出现异常时（如请求超时或网络故障）的最大重试次数。该值为0则不进行重试（平台默认0）。用户可根据任务需求合理调整该值。
<p>计费模式</p>	<p>可选择按量付费模式或包年包月（资源组）模式：</p> <p>（A）按量付费模式下，用户无需预先购买资源组，根据服务依赖的算力规格，启动服务时冻结两小时费用，之后每小时根据运行中的实例数量按量扣费</p> <p>（B）包年包月（资源组）模式下，可使用在资源组管理模块已购买的资源组部署服务，算力费用在购买资源组时已支付，启动服务时无需扣费</p>
<p>资源组</p>	<p>若选择包年包月（资源组）模式，可选择资源组管理模块的资源组</p>

3. 填写对应信息，新建好客观评测任务后，会在任务列表页展示：任务名称、机器来源、评测资源、进度、标签、创建者、创建时间、操作（停止、重启、删除、复制）。



在评测过程中，可进行进度查看，点击[查看进度](#)，获得当前进度展示：

推理进度	进度	状态	开始时间	结束时间
	模型1部署	已完成	2025-03-18 10:43:00	2025-03-18 10:46:29
	模型1评测第1个数据集	评测中93.52%	2025-03-18 10:46:41	-

备注:

- 评测中能看到的具体的进度百分比，具体计算公式： $\text{评测数据总条数} = \text{模型数} \times \text{评测数据条数}$ ， $\text{进度} N\% = \text{已评测数据条数} / \text{评测数据总条数} (120)\%$ 。
- 当用户鼠标悬停在列表页的评测进度“评测中N%”时，能够展示已评测时长和预计剩余时长。预计剩余时长是按照已评测数据的条数的时长等比计算的，这是预估时长，可能会有一定的误差。

4. 您可单击任务名称，进入任务详情，查看基本信息、评测结果、日志。

The screenshot shows a task details page with tabs for '基本信息' (Basic Information), '评测结果' (Evaluation Results), and '日志' (Logs). Under '基本信息', it displays:

- 任务名称: [Redacted]
- 任务ID: [Redacted]
- 创建者: [Redacted]
- 创建时间: 2025-03-18 10:43:00
- 地域: 上海
- 模型: 模型来源: 选择模型; 内置大模型: DeepSeek 系列模型/DeepSeek-...; 推理超参设置: {}
- 评测集: 评测集来源: 选择评测集; 内置评测集: 数学/math

5. 模型推理输出 Tab 页，您可查看进度和评测结果。

The screenshot shows the '评测结果' tab with '整体评测进度: 100%' and a '下载评测结果' button. Below is a table:

模型1 已完成	评测集名称	评测集指标	运行时长
	评测集1	准确率 64.46 %	3小时57分49秒

6. 日志 Tab 页，您可查看推理日志和接口调用日志。

The screenshot shows the '日志' tab with sub-tabs for '推理日志' and '接口调用日志'. It displays a log viewer with a time range from 2024-09-05 18:32 to 2024-09-12 16:06. The log content includes:

```

[2024-09-05 20:39:55] 2024-09-05 20:39:55 INFO | stdout | INFO: 127.0.0.1:43118 - "POST /list_model
474 [2024-09-05 20:39:55] 2024-09-05 20:39:55 INFO | controller | names: ['http://127.0.0.1:21802'], queue
475 [2024-09-05 20:39:55] 2024-09-05 20:39:55 INFO | stdout | INFO: 127.0.0.1:43120 - "POST /get_worker_address HTTP/1.1" 200 OK
476 [2024-09-05 20:39:55] 2024-09-05 20:39:55 async_llm_engine.py:535 | Received request ppl-Z4qml3gg2NqHsWoCD9L: prompt: "以下是中国关于教师资格考试的单项选择题，请选出其中的正
477 [2024-09-05 20:39:55] 2024-09-05 20:39:55 async_llm_engine.py:123 | Finished request ppl-Z4qml3gg2NqHsWoCD9L.
478 [2024-09-05 20:39:55] 2024-09-05 20:39:55 INFO | stdout | INFO: 127.0.0.1:40748 - "POST /get_ppl HTTP/1.1" 200 OK
479 [2024-09-05 20:39:55] 2024-09-05 20:39:55 INFO | stdout | INFO: 134.175.221.239:0 - "POST /get_ppl HTTP/1.1" 200 OK
480 [2024-09-05 20:40:19] 2024-09-05 20:40:18 INFO | model_worker | Send heart beat. Models: ['llama-3-8b-chat'], Semaphore: Semaphore(value=128, locked=False), call_ct
481 [2024-09-05 20:40:19] 2024-09-05 20:40:18 INFO | controller | Receive heart beat. http://127.0.0.1:21802
482 [2024-09-05 20:40:19] 2024-09-05 20:40:18 INFO | stdout | INFO: 127.0.0.1:43224 - "POST /receive_heart_beat HTTP/1.1" 200 OK
483 [2024-09-05 20:41:04] 2024-09-05 20:41:03 INFO | model_worker | Send heart beat. Models: ['llama-3-8b-chat'], Semaphore: Semaphore(value=128, locked=False), call_ct
484 [2024-09-05 20:41:04] 2024-09-05 20:41:03 INFO | controller | Receive heart beat. http://127.0.0.1:21802
485 [2024-09-05 20:41:04] 2024-09-05 20:41:03 INFO | stdout | INFO: 127.0.0.1:43518 - "POST /receive_heart_beat HTTP/1.1" 200 OK
486 [2024-09-05 20:41:49] 2024-09-05 20:41:48 INFO | model_worker | Send heart beat. Models: ['llama-3-8b-chat'], Semaphore: Semaphore(value=128, locked=False), call_ct
487 [2024-09-05 20:41:49] 2024-09-05 20:41:48 INFO | controller | Receive heart beat. http://127.0.0.1:21802
488 [2024-09-05 20:41:49] 2024-09-05 20:41:48 INFO | stdout | INFO: 127.0.0.1:43814 - "POST /receive_heart_beat HTTP/1.1" 200 OK
489 [2024-09-05 20:42:34] 2024-09-05 20:42:33 INFO | model_worker | Send heart beat. Models: ['llama-3-8b-chat'], Semaphore: Semaphore(value=128, locked=False), call_ct
490 [2024-09-05 20:42:34] 2024-09-05 20:42:33 INFO | controller | Receive heart beat. http://127.0.0.1:21802
491 [2024-09-05 20:43:20] 2024-09-05 20:43:19 INFO | stdout | INFO: 127.0.0.1:44062 - "POST /receive_heart_beat HTTP/1.1" 200 OK
492 [2024-09-05 20:43:19] 2024-09-05 20:43:18 INFO | model_worker | Send heart beat. Models: ['llama-3-8b-chat'], Semaphore: Semaphore(value=128, locked=False), call_ct
493 [2024-09-05 20:43:19] 2024-09-05 20:43:18 INFO | controller | Receive heart beat. http://127.0.0.1:21802
494 [2024-09-05 20:43:19] 2024-09-05 20:43:18 INFO | stdout | INFO: 127.0.0.1:44300 - "POST /receive_heart_beat HTTP/1.1" 200 OK
495 [2024-09-05 20:44:04] 2024-09-05 20:44:03 INFO | model_worker | Send heart beat. Models: ['llama-3-8b-chat'], Semaphore: Semaphore(value=128, locked=False), call_ct
496 [2024-09-05 20:44:04] 2024-09-05 20:44:03 INFO | controller | Receive heart beat. http://127.0.0.1:21802
497 [2024-09-05 20:44:04] 2024-09-05 20:44:03 INFO | stdout | INFO: 127.0.0.1:44582 - "POST /receive_heart_beat HTTP/1.1" 200 OK
498 [2024-09-05 20:44:49] 2024-09-05 20:44:48 INFO | model_worker | Send heart beat. Models: ['llama-3-8b-chat'], Semaphore: Semaphore(value=128, locked=False), call_ct
499 [2024-09-05 20:44:49] 2024-09-05 20:44:48 INFO | controller | Receive heart beat. http://127.0.0.1:21802
500 [2024-09-05 20:44:49] 2024-09-05 20:44:48 INFO | stdout | INFO: 127.0.0.1:44872 - "POST /receive_heart_beat HTTP/1.1" 200 OK
    
```

说明

部分评测集的 system prompt

为提高模型回答质量，以及便于解析模型给出的回答，我们在评测如下客观评测集时，会相应地添加 system prompt:

- [AGIEval](#)
- [ARC-c](#)
- [ARC-e](#)
- [C-Eval](#)
- [CMMLU](#)
- [Hellaswag](#)
- [MMLU](#)

我们会根据问题的语言相应地添加，如下表：

题目类型	中文题目	英文题目
单选题	你是一个回答准确的 AI 助手，你被要求回答一道单项选择题。请先解释该问题并逐一分析每个选项，然后给出你的回答。回答请以“因此答案是”加上代表你所选的选项的一个字母结尾。	You are a precise, user-friendly, and self-contained assistant. You are a diligent assistant. You are an error-free and error-tolerant assistant. You are a helpful AI assistant. You are an advanced, unsurpassed, and pattern-recognizing assistant. You are required to answer a single choice question. Give your answer after you explain how to answer the question. Please reply with the ending “Therefore, the answer is” together with a letter representing your choice. You are an advanced, unsurpassed, and pattern-recognizing assistant.
多选题	你是一个回答准确的 AI 助手，你被要求回答一道多选题。请直接清晰明确地仅回答一个或多个字母作为你的选择，不用进行解释。	上述评测集暂不包含英文多选题。

主观评测

最近更新时间：2025-06-19 14:22:52

主观评测对大模型的输出效果进行评估，提供人工标注的打分模式，客观评估模型效果。

前提条件

自定义评测集准备

详见 [评测集格式要求](#)。

准备好评测集后，创建任务时需要填写评测集在 CFS 中的路径。为方便任务的创建，在准备评测集阶段，您需要在开发机中挂载您的 CFS 文件系统，以获得评测时所需填写的路径。CFS 文件系统使用如下：

1. 请您准备好 [CFS 文件系统](#)。您可以在挂载您的 CFS 文件系统并启动 开发机。假设已经准备好 CFS，且 /data1 代表本地挂载的 CFS 根路径。
2. 您可以根据自己需要，为希望用于评测的主观数据集创建本地对应的主观数据集文件夹。
3. 例如，我们可以将需要的主观数据集 test.csv 放置于 /data1/test_data 目录下：

```
cd /data1
mkdir -p test_data
cp userdata/test.csv test_data
```

4. 在评测时，请对应填写您 CFS 实例上的数据集所在的目录，例如 /data1/test_data。

⚠ 注意：

如果填写的目录内包含多个评测集文件，模型评测将对每个评测集文件都进行评测。

操作步骤

1. 登录 [TI-ONE 控制台](#)，在左侧导航栏中选择 [模型服务](#) > [模型评测](#)，单击主观评测 Tab 页，进入任务列表页面。



2. 单击 [新建任务](#)，拉起新建页面。

← 新建主观评测任务

任务名称 * 0/256
 请输入不超过256个字符，仅支持中英文、数字、下划线"_"、短横"-", 只能以中英文、数字开头

备注
 0 / 500

地域 * 上海

评测集 *
 + 添加 评测集要求详情请点击查看 [🔗](#)

评测结果保存 * CFS文件系统

待评测的模型 *
 + 添加

所需填写的信息如下：

参数	说明
任务名称	主观评测任务的名称，按照界面提示的规则填写即可。
备注	可按需为任务备注描述信息。
地域	同账号下的服务按地域进行隔离，地域字段取值根据您在服务列表页面所选择的地域自动带入。
评测集	可选择 CFS 文件系统实例和评测数据集所在目录。仅支持以下格式：jsonl、csv；内容为2列，分别为“prompt”和“answer”。
评测结果保存	可选择 CFS 文件系统实例和评测结果保存的路径。
选择待评测的模型	<p>支持两种待评测模型来源：</p>  <ul style="list-style-type: none"> ● 选择模型： <ul style="list-style-type: none"> 支持从训练任务中选择模型，选择该地域下的训练任务、该任务的 Checkpoint。 支持从 CFS 中选择模型：选择该地域下的 CFS 实例、填写模型所在的路径。 支持选择内置大模型：选择内置好的大模型。 ● 选择服务： <ul style="list-style-type: none"> 支持从 TI 平台的在线服务选择。 支持填写第三方服务地址进行评测。 <p>支持进行参数设置，支持配置推理超参、启动参数设置和性能参数设置。</p> <ul style="list-style-type: none"> ● 配置推理超参，推理超参支持如下： <ul style="list-style-type: none"> ○ repetition_penalty：用来控制重复惩罚。 ○ max_tokens：用来控制输出文本的最长数量。 ○ temperature：数值越高，输出越随机；数值越低，输出越集中和确定。 ○ top_p、top_k：影响输出文本的多样性，数值越高，生成文本的多样性越强。建议该参数和 temperature 只设置1个

	<ul style="list-style-type: none"> o do_sample: 确定模型推理时的采样方式, 取值 true 时为 sample 方式; 取值为 false 时为 greedy search 方式, 此时, top_p、top_k、temperature、repetition_penalty 不生效。 • 配置启动参数, 可参考 服务部署参数填写指引 文档。平台配置默认参数 MAX_MODEL_LEN, 指模型单次推理能处理的最大 token 数 (平台默认8192), 启动时设置过高可能引发显存溢出或性能下降, 可根据任务需求合理调整该值。 • 配置性能参数, 平台配置默认参数 MAX_CONCURRENCY 和 MAX_RETRY_PER_QUERY。 <ul style="list-style-type: none"> o MAX_CONCURRENCY 指评测过程中同时向模型发起的请求数上限; 设置过低可能导致模型吞吐量下降导致评测耗时较长, 设置过高可能导致显存溢出或请求超时 (平台默认24)。用户可根据任务需求合理调整该值。 o MAX_RETRY_PER_QUERY 指每条数据在请求推理服务出现异常时 (如请求超时或网络故障) 的最大重试次数。该值为0则不进行重试 (平台默认0)。用户可根据任务需求合理调整该值。
计费模式	<p>可选择按量付费模式或包年包月 (资源组) 模式:</p> <p>(A) 按量付费模式下, 用户无需预先购买资源组, 根据服务依赖的算力规格, 启动服务时冻结两小时费用, 之后每小时根据运行中的实例数量按量扣费。</p> <p>(B) 包年包月 (资源组) 模式下, 可使用在资源组管理模块已购买的资源组部署服务, 算力费用在购买资源组时已支付, 启动服务时无需扣费。</p>
资源组	若选择包年包月 (资源组) 模式, 可选择资源组管理模块的资源组。

- 填写对应信息, 新建好主观评测任务后, 会在任务列表页展示: 任务名称、机器来源、评测资源、状态、进度、标签、创建者、创建时间、操作 (停止、重启、删除、复制)。
- 单击**推理进度**, 您可下载查看当前进度下的评测结果集。

推理进度 ✕

当前进度: **67%** ↓ 下载评测结果集

2024-06-04 20:55:37, 完成主观评测的67%, 评测第2个模型, 已完成评测集/eval2/people-csv

2024-06-04 20:52:37, 完成主观评测的50%, 评测第2个模型, 已完成评测集/eval2/people-json

2024-06-04 20:43:37, 完成主观评测的33%, 评测第1个模型, 已完成评测集/eval2/people-csv

2024-06-04 20:38:37, 完成主观评测的17%, 评测第1个模型, 已完成评测集/eval2/people-json

- 主观评测任务完成后, 您可单击任务名称, 进入任务详情, 查看基本信息、进行人工标注、查看评测结果和日志。

基本信息 | 人工标注 | 评测结果 | 日志

主观评测得分汇总

被评测的模型	模型来源	模型信息	人工标注进度	人工标注得分
checkpoint-124	任务式建模	checkpoint-124_train-1254928709105057280-9j8j3uux9sl...	0/2	0.00

评测集及对应推理结果信息

评测集地址	推理结果文件存储地址

标签

评测可视化对比

最近更新时间：2024-10-21 14:11:31

使用客观评测、主观评测对大模型进行评测后，可对不同模型进行多个指标横向对比，当前支持文生文类的大模型。

前提条件

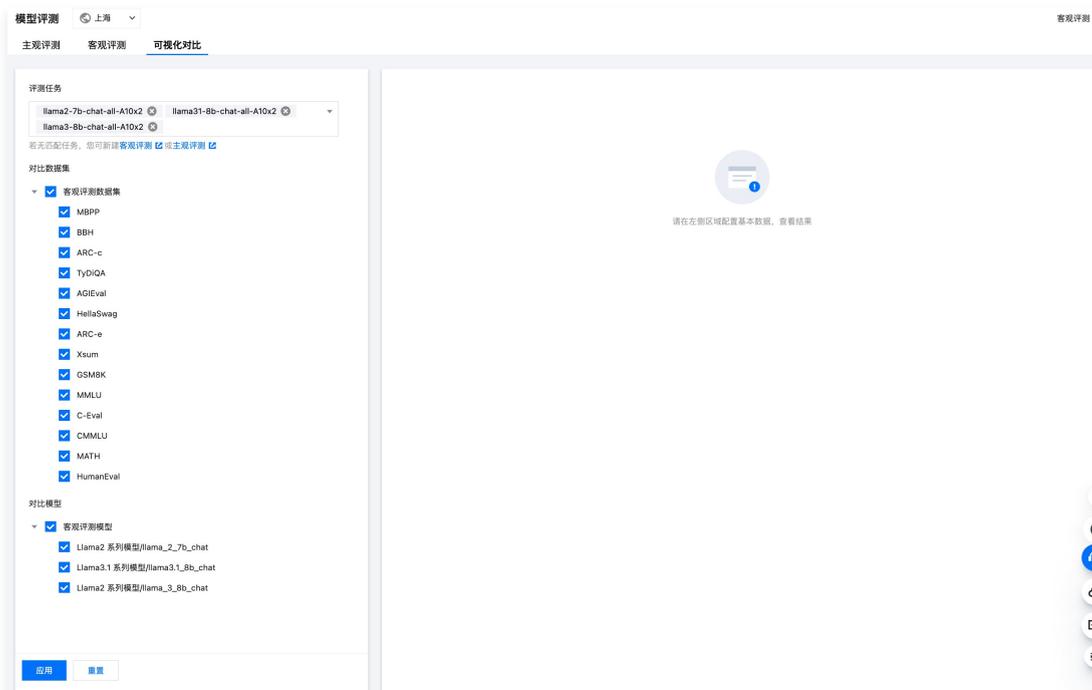
已完成 [客观评测](#)、[主观评测](#) 的模型评测，客观评测的任务状态为“已完成”，主观评测的任务状态为“已完成打分”。

配置可视化对比图表

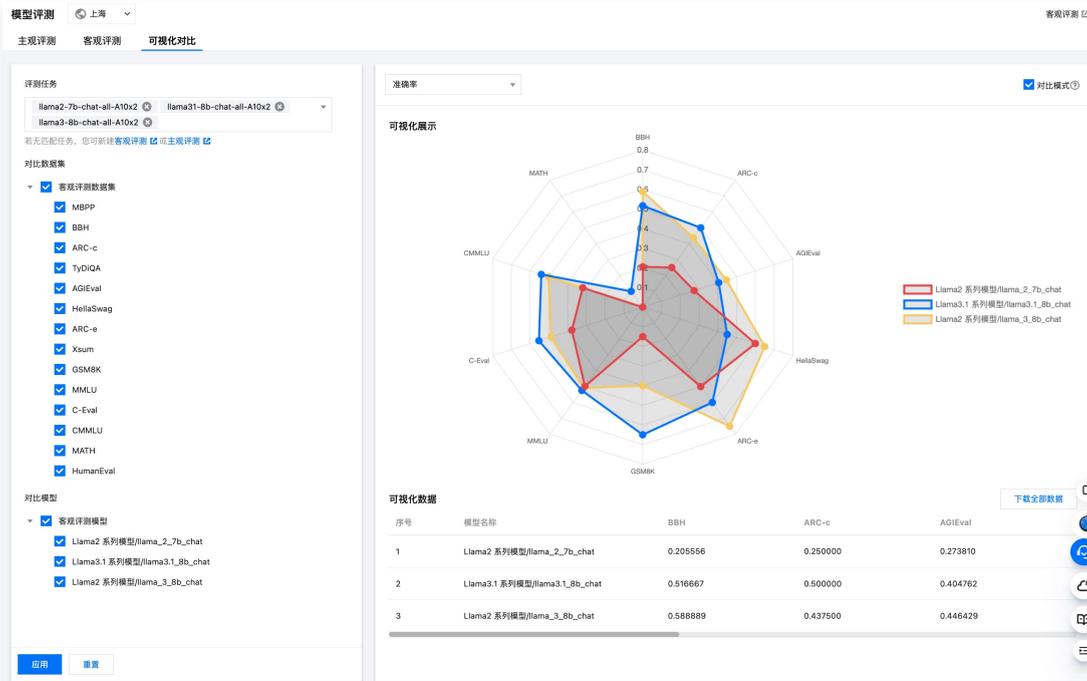
1. 选择评测任务，支持选择“已完成”的客观评测和“已完成打分”的主观评测，最多选择10个任务。



2. 选完评测任务，会自动出现选中评测任务使用的评测数据集和评测的模型，数据集和模型默认全选，也可勾选去除。如下图：



3. 单击应用，右边即可出现选中评测任务，基于选择模型和数据集的雷达图对比展示，如下图：



说明：

1. 雷达图默认勾选对比模式，右上角可勾选去除。选中表明尺度从绝对值到相对值进行缩放，更方便模型进行对比展示。
2. 可视化展示，可点击右侧模型名称，去除该模型在雷达图中的数据展示，再次点击可恢复。鼠标 hover 在雷达图各点上，会展示该模型在该数据集上的评测结果。
3. 可视化数据，行为“模型名称”，列为“数据集”，展示各模型在该数据集上的评测结果，页面展示为6位小数，也可下载全部数据集，下载表格为原始数据16位小数。
4. 如需重新选择评测任务、对比数据集、对比模型，选择好后，单击应用，右侧图表会根据最新选择更新，否则每次进入该页面均按照上次提交“应用”的配置展示。

评测集格式要求

最近更新时间：2025-03-25 20:34:43

支持在主观评测和客观评测中自定义评测集，格式要求如下：

主观评测

1. 仅支持以下格式：jsonl、csv。
2. 评测集格式如下：

2.1 对于 jsonl 格式评测集：每条（行）数据格式如下：

```
{"system": "You are helpful.", "conversation": [{"prompt": "712+165+223+711=", "response": "1811"}]}
```

其中“system”、“prompt”和“response”字段分别对应 system 输入、提示词和预期响应。

如某条（行）数据无 system 输入，可使用如下格式：

```
{"conversation": [{"prompt": "712+165+223+711=", "response": "1811"}]}
```

2.2 对于 csv 格式评测集：评测集共3列，列名分别为“system”、“prompt”和“response”。system 字段非必填，如某条（行）数据无 system 输入，对应位置留空即可。

3. 您可下载评测集示例参考：[jsonl 格式评测集示例](#)、[csv 格式评测集示例](#)。

客观评测

客观评测里支持的自定义评测集为如下开源评测集，当前您可上传如下展示的评测集。关于以下开源评测集在客观评测的具体使用可查看 [客观评测](#)：

[AGIEval](#)
[ARC-c](#)
[ARC-e](#)
[BBH](#)
[C-Eval](#)
[CMMLU](#)
[GSM8K](#)
[Hellaswag](#)
[HumanEval](#)
[MATH](#)
[MBPP](#)
[MMLU](#)
[TyDiQA](#)
[Xsum](#)

内置的开源评测集介绍

最近更新时间：2025-06-13 10:43:41

平台内置丰富的开源评测集，以下是对部分内置的开源评测集的介绍和评测模型的实际场景，用户可在评测时进行参考：

数学

1. GSM8K

简介：

GSM8K 是一个小学数学题数据集，包含约8,500道小学数学题目，主要测试基础数学推理能力。题目涉及加减乘除、分数、小数等基础数学知识。

应用：

GSM8K 主要用于评估模型在解决小学数学问题上的能力，尤其是多步推理能力。例如，模型需要理解题目中的文字描述，并将其转化为数学表达式，最终通过多步计算得出答案。

实际场景：

在教育领域，GSM8K 可以用于开发智能辅导系统，帮助学生练习基础数学题目，并提供详细的解题步骤。

在 AI 研究领域，GSM8K 是测试模型是否具备基础逻辑推理能力的重要基准。

2. MATH

简介：

MATH 数据集涵盖了从小学到高中不同难度的数学问题，包含12,500道题目，测试模型在高级数学能力上的表现。

应用：

MATH 数据集用于评估模型在解决复杂数学问题上的能力，涵盖代数、几何、微积分等高级数学领域。

实际场景：

在学术研究中，MATH 可以用于测试模型是否能够处理高中或大学水平的数学问题，例如求解方程、证明几何定理等。

在教育技术领域，MATH 可以用于开发高级数学学习工具，帮助学生理解复杂数学概念。

3. MATH-500

简介：

MATH-500 是经典数学推理数据集 MATH 的精选子集（源自国际数学竞赛 AMC/AIME 等），包含500道高难度数学题，覆盖代数、几何、数论、组合数学等高中至大学低年级内容。题目强调多步逻辑推导与创造性解题策略，需结合定理应用、构造性证明及非标准求解技巧。其特点是：每道题均附完整 LaTeX 格式解析步骤；难度分级（1-5级，5为最难）；人类专家最高正确率仅 90%（IMO 金牌选手水平）

应用：

主要用于评估大模型在复杂数学推理中的极限能力，尤其测试：符号操作（如多项式化简、不等式证明）、抽象建模（将文字描述转化为数学结构）、

长链推理（维持超过15步的逻辑一致性）。

实际场景：

为竞赛备考平台（如 AoPS 社区）提供自适应题库生成与解题路径反馈。

4. AIME2024

简介：

AIME (American Invitational Mathematics Examination) 是面向 AMC 竞赛优胜者的高难度邀请赛，2024/25 版本被广泛用作大模型数学推理能力的黄金测试集。其特点包括：聚焦数论、组合优化、几何证明 等奥数核心领域；强调 直觉洞察 与 计算效率（限时答题）；题型为填空题（答案均为整数），避免语言表述干扰。

应用：

AIME 分数成为 衡量大模型推理跃迁的关键指标，例如：

DeepSeek-R1-0528 (2025/5 更新)：AIME25 准确率从70% → 87.5%，通过延长思维链至 23K tokens/题实现深度思考9。

实际场景：

算法优化沙盒：驱动 RLHF 技术革新（如熵调控方法 Clip-Cov/KL-Cov 显著抑制强化学习中的“熵塌缩”）。

5. TheoremQA

简介：

TheoremQA (定理问答基准) 是一个专攻数学定理推理的评估数据集，涵盖几何、数论、微积分等领域的 350+ 高难度问题。其独特价值在于要求模型严格应用数学定理、公式及证明逻辑进行推理，而非依赖统计模式匹配，直接测试模型的数学推理严谨性。

应用：

用于评估大模型在复杂数学定理应用、符号推理及公式推导方面的能力。

- **实际场景：**

适用于学术研究工具（如数学问题求解验证）、教育智能系统开发（如高阶数学辅导）以及逻辑推理模块优化（如增强MoE架构的符号计算能力）。

知识问答

1. MMLU

- **简介：**

MMLU (Massive Multitask Language Understanding) 数据集涵盖了广泛的学科知识，包含15,908道题目，测试模型在跨学科综合理解与推理能力上的表现。

- **应用：**

MMLU 数据集用于评估模型在跨学科知识上的综合理解与推理能力，涵盖人文、社会科学、STEM 等多个领域。

- **实际场景：**

在智能助手开发中，MMLU 可以用于训练模型回答用户提出的广泛问题，例如历史事件、科学原理、文学作品等。

在学术研究中，MMLU 是测试模型是否具备广泛知识储备和跨领域推理能力的重要工具。

2. ARC-c

- **简介：**

ARC-c (AI2 Reasoning Challenge - Challenge Set) 是一个科学问答数据集，包含2,590道题目，挑战性较高，测试模型在复杂科学问题上的深度推理能力。

- **应用：**

ARC-c 用于评估模型在解决复杂科学问题上的深度推理能力，例如物理、化学、生物等领域的高难度问题。

- **实际场景：**

在科学研究中，ARC-c 可以用于测试模型是否能够理解并解决复杂的科学问题，例如解释物理现象或推导化学反应。

在教育领域，ARC-c 可以用于开发高级科学学习工具，帮助学生掌握复杂科学概念。

3. ARC-e

- **简介：**

ARC-e (AI2 Reasoning Challenge - Easy Set) 是 ARC-c 的简化版本，包含5,197道题目，难度较低，主要测试模型对基础科学知识的理解与简单推理能力。

- **应用：**

ARC-e 用于评估模型在基础科学知识上的理解与简单推理能力，例如解释基本科学原理或回答简单科学问题。

- **实际场景：**

在教育领域，ARC-e 可以用于开发面向中小学生的科学学习工具，帮助他们理解基础科学知识。

在科普领域，ARC-e 可以用于训练智能助手回答公众提出的简单科学问题。

4. TruthfulQA

- **简介：**

TruthfulQA (真实性问答基准) 包含817个设计诱导模型生成虚假陈述的问题，涵盖健康、法律、历史等易产生幻觉的领域。通过测试模型在对抗性问题中的事实准确性和抗误导能力，直接评估其输出真实性

- **应用：**

用于量化模型生成内容的真实性及对隐含错误前提的识别能力。

- **实际场景：**

可应用于事实核查系统开发（如自动识别矛盾陈述）、搜索引擎增强（如可靠性分级）以及安全机制设计（如减少有害误导性输出）。

语言理解与生成

1. Hellaswag

- **简介：**

Hellaswag 是一个常识推理数据集，包含70,000道题目，测试模型在上下文理解能力上的表现，尤其是对日常场景的理解和推理。

- **应用：**

Hellaswag 用于评估模型在常识推理和上下文理解上的能力，例如理解日常场景中的因果关系或预测接下来可能发生的事情。

- **实际场景：**

在智能助手开发中，Hellaswag 可以用于训练模型更好地理解用户的日常对话，例如回答“如果下雨了，我应该带什么？”这类问题。

在机器人领域，Hellaswag 可以用于提升机器人在日常环境中的决策能力。

2. Xsum

○ 简介:

Xsum 是一个极端摘要生成数据集, 包含226,711条新闻文章及其摘要, 要求模型从长文中生成简洁的摘要, 测试模型的摘要生成能力。

○ 应用:

Xsum 用于评估模型在生成简洁摘要上的能力, 要求模型从长文中提取关键信息并生成简短摘要。

○ 实际场景:

在新闻媒体领域, Xsum 可以用于开发自动摘要工具, 帮助记者或编辑快速生成新闻摘要。

在学术研究中, Xsum 是测试模型是否能够准确提取文本核心信息的重要基准。

3. TyDiQA

○ 简介:

TyDiQA 是一个多语言问答数据集, 包含19,000道题目, 测试模型在多语言阅读理解与问答能力上的表现。

○ 应用:

TyDiQA 用于评估模型在多语言阅读理解与问答上的能力, 涵盖11种语言。

○ 实际场景:

在全球化应用中, TyDiQA 可以用于开发多语言智能助手, 帮助用户用不同语言获取信息。

在跨文化交流中, TyDiQA 可以用于训练翻译工具或跨语言信息检索系统。

4. Winogrande

○ 简介:

Winogrande 是一个大规模常识推理基准, 包含 44k 个代词消歧问题。通过人工重写 Winograd Schema 挑战集, 其问题规模更大且避免偏差, 专注于测试模型对日常语境中实体指代关系的深层语义理解, 而非表面模式识别。

○ 应用:

用于评估模型对语言上下文、常识逻辑和实体关联的推理能力。

○ 实际场景:

可应用于对话系统优化 (如指代消解模块)、无障碍技术开发 (如文本语义增强) 以及认知语言学研究 (如人类与AI推理机制对比)。

5. IFEval

○ 简介:

IFEval (指令跟随评估集) 聚焦大模型对复杂指令的细粒度执行能力, 包含 500+ 人工编写指令及其25项可验证执行标准 (如"严格按步骤输出""避免添加解释")。直接量化模型对用户显式/隐式约束的遵循程度。

○ 应用:

用于检测模型对指令的遵循精度、细粒度控制能力及输出可靠性。

○ 实际场景:

适用于智能助手优化 (如精准响应复杂请求)、自动化流程开发 (如精确生成格式化输出) 以及安全对齐研究 (如可控性验证)。

复杂推理与综合能力

1. BBH

○ 简介:

BBH (BIG-Bench Hard) 是 BIG-Bench 数据集中的一个子集, 包含23项复杂推理任务, 专注于测试模型的复杂推理能力。

○ 应用:

BBH 用于评估模型在复杂推理任务上的能力, 例如逻辑推理、数学推理、语言推理等。

○ 实际场景:

在 AI 研究中, BBH 是测试模型是否具备高级推理能力的重要基准, 例如解决逻辑谜题或完成复杂任务。

在商业领域, BBH 可以用于开发智能决策系统, 帮助企业分析复杂数据并做出合理决策。

2. GPQA Diamond

○ 简介:

GPQA Diamond (全称: Graduate-Level Google-Proof Q&A Diamond) 是一个专注于测试模型在博士级科学问题上的深度推理与专业知识应用能力的基准数据集。它包含198个高难度跨学科问题, 涵盖生物学、化学、物理学等领域, 旨在评估模型是否具备类似领域专家的复杂推理能力, 而非简单的信息检索或记忆。

○ 应用:

GPQA Diamond 用于评估模型在博士级科学问题上的深度推理能力。

○ 实际场景:

可应用于科研辅助（如跨学科问题解答）、高级科学教育工具开发（如复杂概念测试）以及模型优化（如提升MoE架构的推理性能）。

3. TEva

- **简介:**

TEval（全称 Text Evaluation）是中文大模型多任务能力评测基准，涵盖 30+ 任务类型与 500+ 细粒度能力维度（如古文创作、法律条款解析、多跳推理）。其特色在于融合学术考试题、专业场景题及对抗性样本，全面评估中文语义理解与生成能力。

- **应用:**

用于综合评价中文模型在专业领域、复杂推理及文化语境下的性能。

- **实际场景:**

适用于本土化 AI 产品优化（如金融/法律领域助手）、教育评估工具开发（如语文能力测试）及多语言模型对比研究（如中英文能力差异分析）。

代码生成

1. HumanEval

- **简介:**

HumanEval 是一个代码生成评估数据集，包含164道编程问题，测试模型在代码生成能力上的表现。

- **应用:**

HumanEval 用于评估模型在代码生成上的能力，要求模型根据问题描述生成正确的代码。

- **实际场景:**

在软件开发中，HumanEval 可以用于开发自动编程工具，帮助程序员快速生成代码片段。

在教育领域，HumanEval 可以用于开发编程学习工具，帮助学生练习编写代码并检查代码的正确性。

2. MBPP

- **简介:**

MBPP（Microsoft Billion-scale Paraphrase Dataset for Programming）是一个大规模的编程问题数据集，包含1,000道题目，测试模型在代码生成和编程问题解决上的能力。

- **应用:**

MBPP 用于评估模型在解决编程问题和生成代码上的能力，涵盖多种编程语言和问题类型。

- **实际场景:**

在编程教育中，MBPP 可以用于开发智能编程辅导系统，帮助学生解决编程问题并提供反馈。

在软件开发中，MBPP 可以用于训练模型自动生成代码或优化现有代码。

在线服务

在线服务简介

最近更新时间：2024-08-22 11:39:55

模块概述

TI-ONE 的在线服务模块，提供将模型部署为在线推理服务的能力，供用户通过 API 接口调用的方式对接自身业务应用。在线服务支持虚拟化异构算力和弹性扩缩容能力，帮助用户解决模型部署复杂、资源浪费、手工扩展资源效率低下等问题。同时，在线服务还支持部署多种模型格式、支持服务流量分配与滚动更新，以支撑在线推理场景中的多元应用诉求。

模块特点

- **算力虚拟化**：支持为服务分配小至0.1卡 GPU 算力，通过细粒度算力分配，让您随时随地享受高性价比服务体验。
- **自动弹性扩缩容**：您可以选择手动或自动调整弹性实例扩展策略，模型部署会根据业务负载情况，动态实时自动管理实例数量，帮助您以最合适的实例数量应对业务情况，免去人工部署负担。
- **丰富的管理能力**：提供丰富的多模型支持、多版本管理、流量分配、滚动更新等能力，支持服务及调用信息的多维度监控及事件查看，为您的各类业务保驾护航。

应用场景

支持用户将推荐、图像处理、自然语言处理、语音识别等各类机器学习场景的模型部署为在线服务。

在线服务部署

最近更新时间：2025-05-30 15:21:22

在完成模型的训练或自定义镜像的开发后，可以使用模型服务模块部署为在线服务。

操作步骤

登录 [TI-ONE 控制台](#)，在左侧导航栏中选择模型服务 > 在线服务，进入在线服务列表页面。在服务列表页面单击新建服务，进入服务启动页面。在服务启动页面，配置在线服务的相关参数。

1. 服务基础信息

←
创建服务

服务名称 *

请输入不超过60个字符，仅支持中英文、数字、下划线"_"、短横"-", 只能以中英文、数字开头

服务版本

服务描述 (选填)

0 / 500

地域 * 广州

部署方式 * 标准部署 ⓘ 多机分布式部署 ⓘ 多角色部署(PD分离) ⓘ

服务新建后，更新服务、新增版本均无法修改部署方式

机器来源 * 从CVM机器中选择 从TIONE平台购买

TI-ONE 仅收取软件费用，算力将使用您已购买的机器。[产品定价](#)

资源组 * ▼ ↻

参数填写说明：

参数	说明
服务名称	服务的名称，按照界面提示的规则填写即可。
服务版本	版本号系统自动生成。
服务描述	可按需为服务配置描述信息。
地域	同账号下的服务按地域进行隔离，地域字段取值根据您在服务列表页面所选择的地域自动带入。
部署方式	<p>支持多种部署方式：</p> <ul style="list-style-type: none"> 标准部署：单副本下有1个实例运行，适用于大多数标准场景。 多机分布式部署：单副本下有多个实例协调运行，适用于模型需要多机并行的场景。 多角色部署（PD分离）：目前仅支持白名单开放，适用于服务需要分多个角色部署的场景（如 Prefill 和 Decode 角色），单副本下有若干角色，每个角色下有1个或多个实例协调运行。 <div style="border: 1px solid #add8e6; padding: 5px; margin-top: 10px;"> <p>⚠ 注意：</p> </div>

	服务新建后，更新服务、新增版本均无法修改部署方式，请新建的时候谨慎选择。
机器来源	<p>可选择“从 CVM 机器中选择”或“从 TIONE 平台购买”：</p> <ul style="list-style-type: none"> “从 CVM 机器中选择”模式下，可使用在资源组管理模块已购买的 CVM 机器的资源组部署服务，算力费用在购买资源组时已支付，启动服务时无需扣费。 “从 TIONE 平台购买”模式下，用户无需预先购买资源组，根据服务依赖的算力规格，启动服务时冻结两小时费用，之后每小时根据运行中的实例数量按量扣费。
资源组	若选择“从 CVM 机器中选择”模式，可选择资源组管理模块的资源组。

2. 副本设置信息

副本设置

请确保选择的存储实例（CFS或者GooseFSx）和纳管资源组的节点网络互通

模型来源：
 模型仓库 CFS GooseFSx COS 镜像

选择模型：
 CFS文件系统
 CFS源路径

运行环境：
 自定义

选择自定义镜像环境：

环境版本：

端口：
 8501

资源申请：
 卡型号
 单实例GPU 卡
若需使用GPU，根据不同卡类型可填写0.1-1或1的整数倍。运行环境为平台内置GPU镜像时，卡数不能为0
 单实例CPU 核
 单实例内存 G
 实例数量 个

高级设置

启动命令：

环境变量：

优雅停止时限：

默认为30，可填写1至86400间的数

停止前执行：

选择Sidecar容器镜像：

镜像版本：

参数填写说明：

参数	说明
模型来源	<p>支持多种模型来源（不同部署方式下支持的模型来源也不一致）：</p> <ul style="list-style-type: none"> 选择模型仓库适用于，部署服务所需的模型文件已导入至模型仓库的场景。 选择 CFS 适用于，部署服务所需的模型文件已放在 CFS 文件系统里的场景，选择模型所在的 CFS 实例，路径输入到模型所在路径的层级（如模型为精调出来的 checkpoint500，则路径输入到/a/b/checkpoint500这一层级）。CFS Turbo 仅支持机器来源为“从 CVM 机器中选择”时使用。 选择 GooseFSx 适用于，部署服务所需的模型文件已放在 GooseFSx 文件系统里的场景，选择模型所在的 GooseFSx 实例，路径输入到模型所在路径的层级。GooseFSx 仅支持机器来源为“从 CVM 机器中选择”时使用。 选择 COS 适用于，部署服务所需的模型文件已放在 COS 文件系统里的场景，选择模型所在存储桶实例和所在路径的层级。COS 仅支持机器来源为“从 CVM 机器中选择”时使用。 选择镜像适用于，部署服务所需的自定义镜像已封装模型文件，不需要再进行模型文件挂载，且自定义镜像已上传至容器镜像服务 TCR 的场景；或者内置大模型的场景。

运行环境	<ul style="list-style-type: none"> 若从模型仓库选择模型，则运行环境会根据模型仓库的配置信息自动赋值。如需使用自定义镜像启动服务，镜像文件大小建议不超过34G。 若从 CFS 或 GooseFSx 选择模型，则运行环境需要根据模型文件选择对应的内置镜像。 若从 COS 选择模型，则运行环境需要根据 COS 里的模型文件选择对应的内置镜像。 若从镜像选择模型，则运行环境可以选择已上传至容器镜像服务 TCR 的自定义镜像、输入镜像地址或者内置大模型镜像。
开启 GRPC	开关默认关闭。关闭时仅支持 HTTP 协议调用。开启后支持 GRPC 协议调用。
端口	支持配置容器对外暴露的端口，可填范围 1024-65535，但不包括 8502-8510,6006,9092。额外注意两个特殊端口号：8500 是 GRPC 的默认端口，8501 是 REST 的默认端口，不可混淆使用。
资源申请/算力规格	<ol style="list-style-type: none"> 包年包月（资源组）模式下，可设置从所选资源组中申请多少资源用于启动当前服务。 按量付费模式下，可按需选择启动当前服务所需的算力规格。
启动命令	支持配置容器的启动命令，选填。
环境变量	支持配置容器的环境变量，选填。
优雅停止时限	即 Kubernetes 的 <code>terminationGracePeriodSeconds</code> ，停止时间超过该时限的 Pod 将被强制销毁。默认值为30s。
停止前执行	即 Kubernetes 的 <code>PreStop</code> 命令，Pod 在销毁前会先执行该命令以实现优雅停止，选填。命令格式为字符串数组，例如：“["sleep", "70"]”。
选择Sidecar容器镜像	支持用户自定义配置 Sidecar 容器镜像，选填。

3. 服务特性配置

请求限流 不限流 单副本QPS 单副本最大并发数

副本调节 手动调节 自动调节

副本数量 1

是否生成鉴权 开启鉴权后，将会为您自动生成服务的首个密钥，您可在服务详情页-服务鉴权中查看并创建更多密钥

CLS 日志投递 TI 控制台会默认展示 15 天的日志，若您期望持久化存储日志，获得日志检索等服务，可以开启 CLS 日志投递，CLS 产品介绍和收费指南请查看[文档](#)

重试策略 有限次重试 无限次重试

服务访问云上VPC资源 模型来源选择CFS时，默认开启，打通的VPC和子网自动填充为CFS的VPC和子网，会在该子网下一个弹性网卡进行绑定，独立计费，弹性网卡当前未计费

滚动更新策略 MaxSurge 0 %
 请确保启动滚动更新时资源组中有足够的空闲资源来部署MaxSurge实例

MaxUnavailable 33 %

健康检测

自动停止 开启后，在线服务将在指定的停止时间自动停止，同时停止服务算力计费

标签

[+ 添加](#)

参数说明：

参数	说明
请求限流	支持配置服务限流值： <ul style="list-style-type: none"> 不限流默认单个服务最大的 QPS 为500，限流值大于500时，按照500进行限流，当购买服务升级包后，服务的总限流值以购买升级包的上限为准。 单副本 QPS 的限流值为单个实例的限流。 单副本最大并发数为单个实例的限流。 <div style="border: 1px solid #00aaff; padding: 5px; margin-top: 10px;"> <p>注意：</p> </div>

	<p>该限流值为单个副本的限流，当服务进行扩缩容时，服务整体限流值将按照设置的值 * 副本数进行更新；单个服务组最大的 QPS 为500，当服务组下设置的服务总限流值大于500时，按照500进行限流。</p>
副本调节	<ul style="list-style-type: none"> ● 手动调节：可以自定义设置服务的副本数量，副本数量最小为1。 ● 自动调节：可以选择基于时间或者基于 HPA 的调节策略，该部分详细说明请查看 在线服务运营。
是否生成鉴权	<p>若开启，则服务调用时会进行签名认证，已启动的服务可在服务调用页面查看签名密钥及签名计算指引。开启鉴权后，将会为您自动生成服务的首个密钥，您可在服务详情页-服务鉴权中查看并创建更多密钥。</p>
CLS 日志投递	<p>平台为用户提供免费的近15日服务日志存储，若需要持久化日志存储以及更灵活的日志检索能力、日志监控告警能力，可开启 CLS 日志投递，开启后服务日志会根据日志集与日志主题投递至腾讯云日志服务 CLS。</p>
重试策略	<p>配置服务部署失败时采用的重试逻辑，支持“有限次重试”或“无限次重试”。默认值为有限重试5次。只有在新部署服务时才会使用该逻辑；更新服务或启动已停止服务时，系统将采用“无限次重试”。</p>
滚动更新策略	<p>支持设置服务滚动更新策略，确保服务平滑升级：</p> <ul style="list-style-type: none"> ● MaxSurge：表示滚动更新期间，允许超出所需规模的最大副本数量。 ● MaxUnavailable：表示滚动更新期间，允许不可用副本数量的上限。
健康检测	<p>Kubernetes 的健康检查机制，支持自动检测并恢复失败的容器，确保流量分发到健康的实例上。</p> <ul style="list-style-type: none"> ● 存活检测：Liveness Probes，验证服务进程是否存活，容器是否正常运行。触发阶段：容器启动后持续运行（整个生命周期）。 ● 就绪检测：Readiness Probes，确认服务已具备处理请求的能力，容器是否准备好接受流量。触发阶段：容器启动后持续运行（整个生命周期）。 ● 启动检测：Startup Probes，对慢启动容器进行监控检查，检测容器内应用是否完成。触发阶段：仅在容器启动阶段运行（成功后停止）。 <p>检查方法支持三种：HTTPGet、TCPSocket、Exec。</p>
自动停止	<p>平台支持自动停止模型服务，当开启该开关后，在线服务将在指定的停止时间自动停止，同时停止服务算力计费。</p>
标签	<p>支持为服务添加标签，用于按照标签进行授权等。</p>

4. 启动服务

确认服务配置信息无误后，单击**启动服务**进行服务部署。服务部署过程中将为您创建网关并调度计算资源，需要等待一段时间，待服务成功完成部署时，服务状态将变为**运行中**。

在线服务调用

最近更新时间：2025-04-27 14:49:52

接口在线测试

在线服务为您提供接口的在线测试能力，完成部署的在线服务可进行接口的调用测试。

1. 登录 [TI-ONE 控制台](#)，在左侧导航栏中选择**模型服务** > **在线服务**，进入在线服务列表页面。
2. 在服务列表页单击**调用**操作按钮，进入服务调用页面，查看服务的接口信息。
3. 单击接口列表中的**在线测试**操作按钮，打开调用测试页面。
4. 在接口调用地址栏输入您要测试的接口 URL，部分内置镜像会提供默认 URL，如需测试其他接口直接修改即可。
5. 在请求体模块录入 JSON 格式的请求信息并单击**发送请求**后，可在请求响应模块查看预测结果。

接口信息

接口调用地址: `http://ms-9*****j-100*****5.gw.ap-shanghai.test.ti.tencentcs.com/ms-9*****j/v1/chat/completions`

服务类型: HTTP

请求方法: POST

调用方式(命令行): `curl -X POST http://ms-9*****j-100*****5.gw.ap-shanghai.test.ti.tencentcs.com/ms-9*****j/v1/chat/completions -H 'Content-Type: application/json' -d ''`
若服务开启了鉴权，请参考[文档](#) 指引调用

调用方式(在线测试) 请求体(Request Body 600KB 内)

```
1 {"messages": [{"role": "user", "content": "你是一位历史学家，专门研究古代文明。你要为一部关于古埃及文明的纪录片撰写解说词。请详细介绍以下几个方面：1. 古埃及的起源与发展：介绍古埃及文明的起源、重要的历史时期和主要的统治者。2. 建筑与艺术：描述古埃及的建筑奇迹，如金字塔和神庙，以及雕刻、绘画等艺术形式。3. 宗教与信仰：探讨古埃及的宗教体系、主要的神祇和祭祀仪式，以及这些信仰对社会生活的影响。4. 社会结构与日常生活：分析古埃及的社会阶层、职业分工和家庭生活，展示普通人的日常生活。5. 科学与技术：介绍古埃及在天文学、医学、工程等领域的成就和发明。请确保你的解说词具有教育性和吸引力，使观众能够深入了解古埃及文明的辉煌与神秘。"}], "temperature": 0.0, "top_p": 1.0, "frequency_penalty": 0.0, "max_tokens": 2048}
```

请求响应(Response)

```
6 X-RateLimit-Limit: 2000
7 X-RateLimit-Remaining: 1999
8 X-Tc-Requestid: 312b05f3-85f2-459d-b3f7-56216e
9 X-Tgateway-Upstream-Status: 200
10
11 {
12   "id": "chat-f00705db6e7d453daed036f850309a
13   "object": "chat.completion",
14   "created": 1733456429,
15   "model": "llama-3.2-3b-chat",
16   "choices": [
17     {
18       "index": 0,
19       "message": {
20         "role": "assistant",
21         "content": "古埃及的起源与发展：古
22         历史时期包括新王朝和中王朝。古埃及的主要统治者包括 Rams
23         筑奇迹包括金字塔和神庙。古埃及的雕刻、绘画等艺术形式非常
24         Ra 和 Isis。古埃及的祭祀仪式非常丰富。古埃及的社会阶层
25         富。古埃及在天文学、医学、工程等领域的成就和发明非常影
26         "tool_calls": []
27       }
28     }
29   ]
30 }
```

发送请求

公网调用

在 TI-ONE 完成在线服务部署后，系统会自动注册 API 网关生成公网地址用于服务请求调用。您可以在服务列表页单击**调用**操作按钮，进入服务调用页面，查看公网访问地址。通过该调用信息可以向在线服务发起预测请求，curl 命令示例如下，其中 **Authorization** 仅在**鉴权**开启时需要：

```
curl -X POST https://ms-9*****j-100*****5.gw.ap-guangzhou.ti.tencentcs.com/ms-9*****j/v1/models/m:predict -H 'Authorization: P*****pg' -H 'Content-Type: application/json' -d ''
```

- 部分存量服务(域名为 `apigw.tencentcs.com`)，仍在历史链路，该链路的底层产品即将**停止支持**，届时该调用方式将无法使用，推荐您**尽快使用新地址或者新建服务进行迁移**，详情见 [API 网关下架公告](#)。该链路 curl 命令示例如下：

```
curl http://service-94****g0-130****662.gz.apigw.tencentcs.com:80/tione/v1/models/m:predict -d '{"images": ["https://qcloudimg.tencent-cloud.cn/raw/aab789b6e047fa804bbf803de16f49a0.jpeg"]}'
```

- 开启服务鉴权，则在调用 API 时，需要使用签名密钥 (ApiAppKey 和 ApiAppSecret) 对请求内容进行签名计算，签名计算过程参见文档 [应用认证方式](#)。
- 如您希望屏蔽在线服务的公网调用方式，可联系您对接的腾讯云架构师或其他服务团队提供支持。

内网调用

高速服务调用（HTTP）

公网链路收到诸多因素的限制，如果您对请求的性能和稳定性有更高的要求，推荐使用内网高速服务调用的方式来访问您的服务。高速服务调用通过私有连接建立用户 VPC 与服务的高速通道，具体操作路径如下：

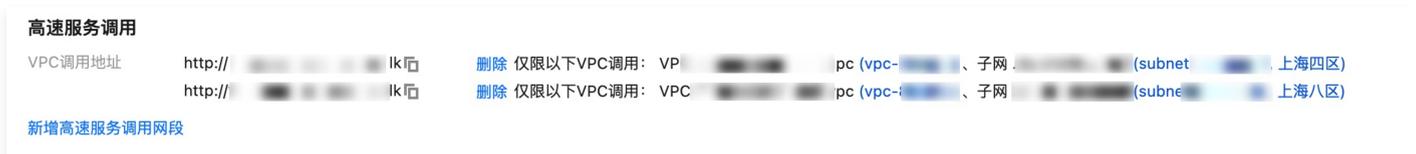
1. 登录 [TI-ONE 控制台](#)，在左侧导航栏中选择 **模型服务 > 在线服务**，进入在线服务列表页面。
2. 在服务列表页单击 **服务名称**，进入服务版本列表页面，单击 **服务调用**，查看配置服务调用相关信息。
3. **高速服务调用** 模块，首次配置，单击 **开启新增高速服务调用网段**。



4. 在配置弹窗里，您选择需要打通内网的 **VPC、子网**。



5. 配置好 VPC、子网后，**高速服务调用** 模块，展示 **VPC 调用地址**，以及可调用的 **VPC、子网** 信息，配置好后支持删除后重新新增。如果在线服务有 2 个版本，则内网地址有 2 条。



6. 服务调用时，如需要在终端里内网接口调用，可执行以下命令：

```
//内网地址，请您参考服务调用/高速服务调用/VPC调用地址
//URL为服务调用/接口信息/接口调用地址的右边URL内容
curl -X POST -H 'Content-Type: application/json' 内网地址/URL -d '{ "prompt": "puppy dog", "steps": 5 }'
```

说明：

一次开启配置，本地域下的全部服务均生效。本能力使用私有连接，配置后，将在您账户的该 VPC 和子网下，创建一个终端节点，独立计费，您可参考 [终端节点计费文档](#)。

高速服务调用（GRPC）

高速服务调用除支持 HTTP 协议外，用户可在新建在线服务时选择打开 **开启 GRPC** 的开关按钮支持 GRPC 调用协议。用户开启 GRPC 协议后和 HTTP 协议的区别是：

- 只支持 VPC 地址调用，所以公网访问会被屏蔽，前端不再展示常规服务调用的调用地址；
- 由于接口在线测试依赖公网下的 HTTP 协议调用，所以 GRPC 协议下也不支持【接口信息】的模块。

GRPC 调用说明：发送请求时，请求中需携带 Header `x-ti-service = ms-hx7c5srv` 和虚拟 Authority (`ms-hx7c5srv-100031385875-sw-grpc.gw.ap-shanghai.ti.tencentcs.com`)。

← **grpc-ng-new**

服务管理 **服务调用** 服务鉴权

① 遵守平台要求，服务调用已在创建服务时，授权并同意了《腾讯云 TI-ONE 训练平台服务协议》

调用监控

流控信息

QPS 500 (单服务组的QPS上限为500，如需提高QPS请提交工单，我们将根据您的业务情况进行评估)

常规服务调用 ① 已开启GRPC协议，不再支持常规服务调用（公网访问）

高速服务调用

VPC调用地址(GRPC)	17	7:8090	删除 仅限以下VPC调用: VPC	dm9e37)
	17	7:8090	删除 仅限以下VPC调用: VPC	nxzg5z)
	17	7:8090	删除 仅限以下VPC调用: VPC	fmulzir)
	17	11:8090	删除 仅限以下VPC调用: VPC	
	17	7:8090	删除 仅限以下VPC调用: VPC	.vcp)
	17	13:8090	删除 仅限以下VPC调用: VPC	bkn8z)
	19	17:8090	删除 仅限以下VPC调用: VPC	
	10	090	删除 仅限以下VPC调用: VPC	
	10	4:8090	删除 仅限以下VPC调用: VPC	
	19	129:8090	删除 仅限以下VPC调用: VPC	hy7)
	17	171:8090	删除 仅限以下VPC调用: VPC	skzf)

GRPC协议 ① 已开启 [GRPC调用说明](#)

[新增高速服务调用网段](#)

服务访问云上 VPC 资源

如果您的服务需要通过内网访问腾讯云上的其他 VPC 资源，可通过**服务访问云上 VPC 资源**进行配置，具体操作路径为：

1. 登录 **TI-ONE 控制台**，在左侧导航栏中选择**模型服务 > 在线服务**，进入在线服务列表页面。
2. 在服务列表页单击**新建服务**，在新建弹窗里，**开启服务访问云上 VPC 资源**。

实例调节 ① 手动调节 自动调节

实例数量

是否生成鉴权 开启鉴权后，服务调用时会进行签名认证，已启动的服务可在服务调用页面查看签名密钥及签名计算指引

CLS 日志投递 ① TI 控制台会默认展示 7 天的日志，若您期望持久化存储日志，获得日志检索等服务，可以开启 CLS 日志投递，CLS 产品介绍和收费指南请查看[文档](#)

服务访问云上VPC资源 ① 开启后配置VPC和子网，会在您账户下的该子网，创建一个弹性网卡进行绑定，独立计费，弹性网卡当前未计费

自动停止 开启后，在线服务将在指定的停止时间自动停止，同时停止服务算力计费

标签 ① 键值粘贴板

配置价格

服务配置费用 ① - 元/小时

遵守平台要求，授权并同意《腾讯云 TI-ONE 训练平台服务协议》

3. 开启后，选择需要打通内网的 **VPC、子网**。

服务访问云上VPC资源 ① 开启后配置VPC和子网，会在您账户下的该子网，创建一个弹性网卡进行绑定，独立计费，弹性网卡当前未计费

VPC

子网

4. 配置好 VPC、子网后，**服务访问云上 VPC 资源**模块，展示配置好的 VPC、子网。

在线服务鉴权和限流

最近更新时间：2025-04-25 17:28:53

TI 平台在线服务模块为用户提供大模型服务鉴权和限流的产品功能，支持用户针对单个服务配置多个密钥，并支持基于 Tokens 计数进行大模型流量控制，以此实现精细化的调用方管理和流量控制。

开启服务鉴权

用户在新建在线服务的参数配置页面，开启【是否生成鉴权】开关即可开启服务鉴权功能。因为鉴权开关的状态与部署的服务本身无关，所以服务创建后，用户若需切换鉴权开关的状态，无需停止或更新服务即可在服务详情页的【服务鉴权】Tab 中进行编辑修改。

请求限流 不限流 单副本QPS 单副本最大并发数

副本调节 手动调节 自动调节

副本数量 1

是否生成鉴权 开启鉴权后，将会为您自动生成服务的首个密钥。您可在服务详情页-服务鉴权中查看并创建更多密钥

CLS 日志投递 TI 控制台默认展示 15 天的日志，若您期望持久化存储日志，获得日志检索等服务，可以开启 CLS 日志投递，CLS 产品介绍和收费指南请查看[文档](#)

重试策略 有限次重试 无限次重试

5 次

默认为5，可填写1至99间的整数

健康检测

管理鉴权密钥

开启鉴权密钥后，用户可单击在线服务名称，进入服务详情页面，通过【服务鉴权】的 Tab 页对鉴权密钥进行统一管理。

密钥	限流信息	备注	创建时间	状态	操作
..... - /	每分钟最大Token数: 20000	- /	2025-04-24 16:00:09	已启用	禁用 限流 更新 删除
..... - /	每分钟最大Token数: 10000 每日最大Token数: 3000000	- /	2025-04-24 16:07:08	已启用	禁用 限流 更新 删除

该页面列表字段解释如下：

- 密钥：展示用户访问服务所需的密钥信息，当前仅支持 AuthToken 的密钥类型。开启鉴权后用户需在服务请求中需携带该 AuthToken 进行鉴权。支持用户单击列表页上的新增密钥来添加一个新的自定义密钥。

新建密钥

密钥名称

备注

- 限流信息：展示用户对每个密钥单独设置的限流信息，限流类型支持两种：每分钟最大 Token 数（TPM）和每日最大 Token 数（TPD）。



- 备注：展示用户给密钥添加的备注信息，方便用户详细记录各密钥用途。
- 创建时间：密钥被创建的时间。
- 状态：密钥当前的状态，有“已启用/已禁用”两个枚举值，可通过操作 > 禁用/启用按钮切换密钥状态，禁用密钥后，TI 平台将拒绝此密钥的所有服务请求，请谨慎操作。

服务限流

用户可通过服务鉴权列表页的【操作-限流】按钮设置对应密钥的流量控制。由于非大模型服务无 token 概念所以无法进行 token 限流，所以限流功能仅支持“模型来源”为“镜像-内置大模型”，或“运行环境”为“内置-LLM”的大模型服务，且有一个前提条件是：模型的 response 需满足 [openai 规范](https://platform.openai.com/docs/api-reference/responses/create) 必须包含标准的 usage 字段信息。

目前支持的限流方式有两种：**每分钟最大 Token 数**（通过该密钥，每分钟能请求的最大 Token 数）和 **每日最大Token数**（通过该密钥，每日能请求的最大 Token 数）。对密钥设置限流后，一旦该密钥的服务请求触发了流量上限则会导致请求报错。



服务鉴权调用监控

由于平台支持用户针对单个大模型服务配置多个密钥鉴权，所以在在大模型服务的调用监控页面，平台也支持用户通过切换鉴权 AuthToken 详细的查看每个调用方的调用信息，实现精细化的流量监控。

如果当前服务尚未配置告警服务，可以前往新增告警 为服务添加告警策略

今天 昨天 近7天 近15天 2025-04-11 ~ 2025-04-25 AuthToken 全部

调用统计 (次) - 默认调用方式 (http) 图例中被限制请求数为所有服务副本入口上的限流请求之和



平均响应时间 (毫秒) - 默认调用方式 (http)



在线服务运营

最近更新时间：2025-06-06 17:08:01

自动扩缩容

如果您的业务负载有显著的峰谷特征，为了提升推理算力资源的利用效率，您可以使用在线服务模块的自动扩缩容能力。该功能支持在线服务的实例数量根据您的扩缩容策略自动调整，从而实现在业务负载高峰时实例数量自动扩容，在业务负载低谷时实例数量自动缩容。

自动扩缩容支持两种类型的调节策略：基于时间调节、基于 HPA 调节，下文将详细介绍两种调节策略的使用方法。

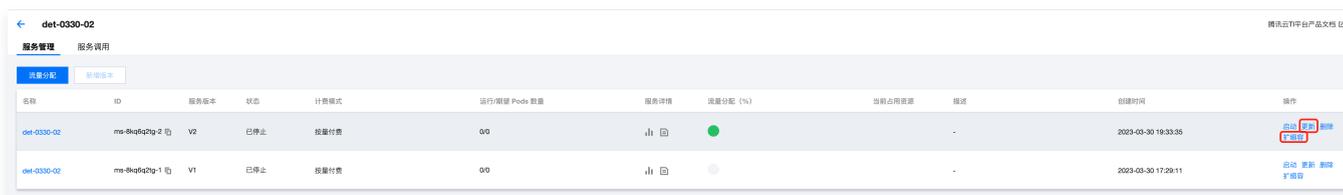
基于时间调节

如果您的业务负载有显著的时间特征，则可以根据时间进行自动扩缩容策略的配置。

1. 如何开启定时扩缩容

1.1 登录 [TI-ONE 控制台](#)，在左侧导航栏中选择**模型服务** > **在线服务**，进入在线服务列表页面；

1.2 在服务列表中找到需要开启定时扩缩容策略的服务，单击**服务名称**，进入版本列表页面，单击**更新**进入服务详细配置更新页面，或者单击**扩缩容**进入实例调节弹窗；



1.3 服务详细配置中，将“实例调节”字段的选项设置为“自动调节”，调节策略类型选择“基于时间”，即可进行时间调节策略的规则配置；

1.4 您可以根据实际业务负载的时间特征自行配置多条定时策略规则，例如若8:00至20:00为业务高峰时段，20:00至8:00为业务低谷时段，则可以配置如下图的定时策略，每日8:00将实例数扩容至2，每日20:00将实例数缩容为1（默认策略为服务启动后的初始实例数量）；



1.5 若您配置了多条定时策略规则，且多条规则之间存在时间冲突，则会以优先级级别较高（即优先级排序靠前）的策略为准；

1.6 完成扩缩容策略的内容配置后，单击**更新服务**即可进行配置信息保存，待服务完成更新后，您所配置的自动扩缩容策略即会生效。

2. 例外时间配置规则

2.1 若某个定时策略希望在特定的时间不执行，则可以为该定时策略规则配置例外时间，支持添加多个；

- 2.2 例外时间需通过 Cron 表达式进行配置，Cron 表达式共包含6位，分别代表“秒”“分”“时”“日”“月”“星期”，若特定位的取值为任意值则使用星号(*)即可，若特定位取值需包含连续多个数值则可以使用连字符(-)，若特定位取值需包含多个离散数值则可以使用逗号(,)；
- 2.3 例外时间的最小配置粒度是日，因此 Cron 表达式的前三位取值需要使用“*”（前三位配置其他值不会生效），后三位取值可按需配置，第4位“日”的可用值范围为1-31，第5位“月”的可用值范围为1-12或 JAN-DEC，第6位“星期”的可用值范围为0-6或 SUN-SAT；
- 2.4 例如：每年10月1日至10月7日的 Cron 表达式为“*** 1-7 10 *”。



基于 HPA 调节

如果定时调节不适合于您的业务模式，您也可以选择“基于 HPA”的自动扩缩容调节策略，在该策略下，服务实例数量可根据您所配置的策略指标与指标阈值，在实例数的最小值与最大值之间自动进行调节。策略指标支持 CPU 使用率、内存使用率、GPU 使用率、单实例 QPS、最大并发数使用率 等。其中策略指标需要配置“最大并发数使用率”时，请先在请求限流处配置“单实例最大并发数”。



流量分配

为了满足灰度验证或者A/B测试类的服务使用诉求，平台支持用户为单个服务添加多个版本，并进行流量分配。

1. 登录 [TI-ONE 控制台](#)，在左侧导航栏中选择**模型服务 > 在线服务**，进入在线服务列表页面。
2. 找到需要测试的服务，单击服务的**新增版本**操作，打开服务版本创建页，按需配置当前服务版本的容器信息及实例调节信息。



3. 单击**启动服务**，若为后付费模式则需进行费用冻结确认，即可完成新版本创建。
4. 创建新的服务版本后，系统将为您创建网关后端并调度计算资源，需要等待一段时间，待服务版本成功完成部署时，状态将变为**运行中**。

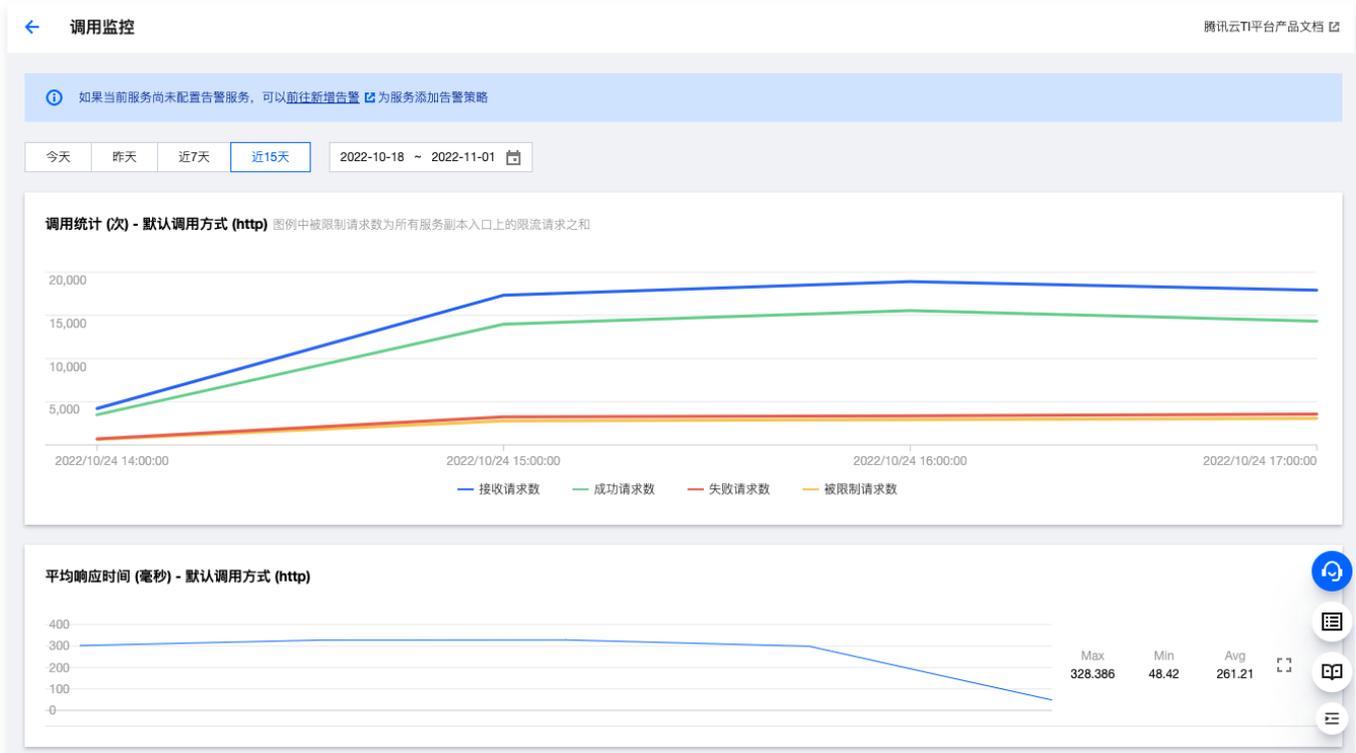
5. 此时可单击服务版本列表上方的流量分配操作，进行多版本流量比例的设置。



服务监控

为了满足服务运行情况追踪的诉求，平台提供服务数据监控、调用数据监控、事件与日志查看能力。

1. 在线服务列表页面，单击服务名称进入版本列表页后，单击服务调用>调用监控，可查看服务调用情况的统计信息，包括接收请求数、成功请求数、失败请求数、被限制请求数、平均响应时间。

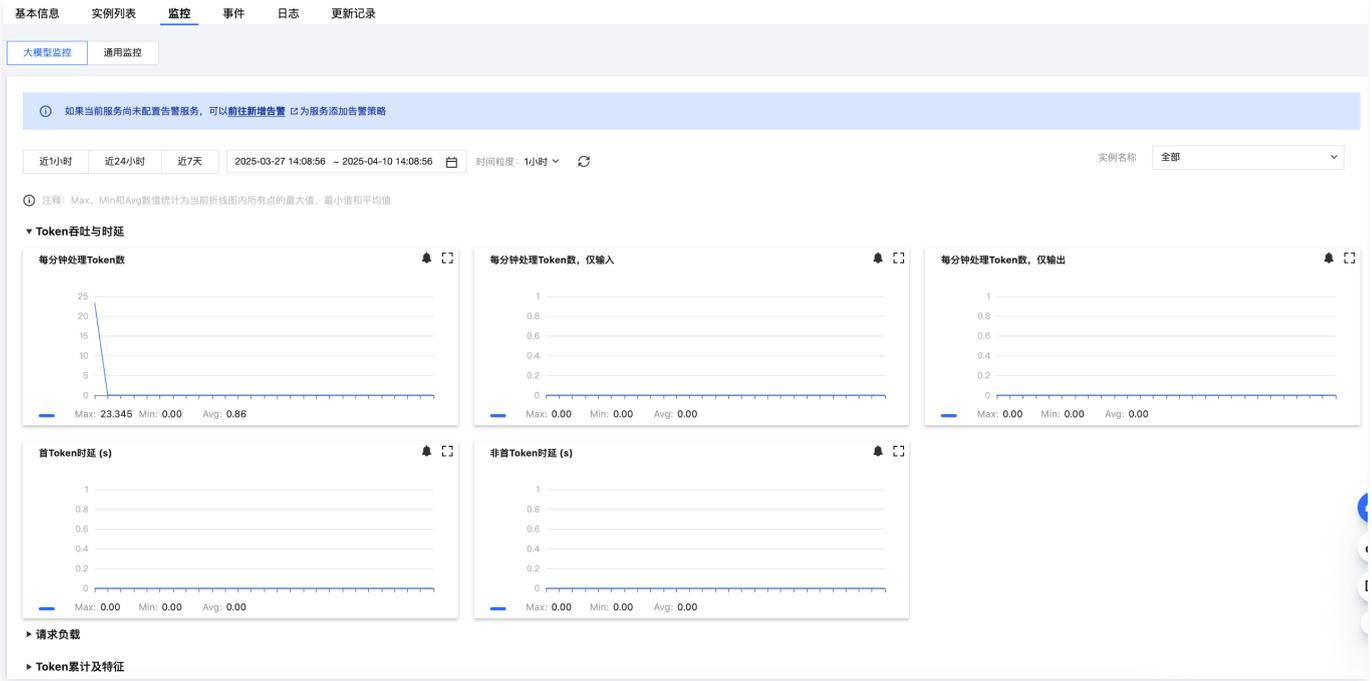


2. 在线服务监控页面，可跳转到腾讯云可观测平台的告警管理里，为服务添加告警策略。



3. 在线服务列表页面，单击**服务名称**进入版本列表页后，单击**服务版本名称**进入版本详情页面，可查看服务监控、事件监控、运行日志。其中服务监控分为大模型监控和通用监控：

- 大模型监控：
 - Token 吞吐与时延：每分钟处理 Token 数（全口径/仅输入/仅输出）、首 Token 时延和非首 Token 时延。
 - 请求负载：处理中请求数和排队中请求数。
 - Token 累计与特征：已处理 Token 总量（全口径/仅输入/仅输出）、输入平均长度和输出平均长度。
- 通用监控：
 - 流量信息：网络流量、QPS、QPS 限流和并发请求数。
 - 资源信息：CPU 使用率、MEM 使用率、显存使用率和 GPU 使用率。
 - 实例信息：实例数量和运行中实例数量。



基本信息 实例列表 监控 **事件** 日志 更新记录

事件只保留最近24小时内发生的事件，请尽快查阅

自动刷新

首次出现时间	最后出现时间	级别	资源类型	资源名称	详细描述	出现次数
2021-12-30 19:18:29	2021-12-31 13:18:57	Normal	Ingress	ms-n42hcp96-2	Service Sync Success. RetrunCode: S2000	55
2021-12-30 19:18:29	2021-12-30 20:20:07	Normal	Ingress	ms-n42hcp96-2	Service Sync Success. RetrunCode: S2000	50
2021-12-30 16:59:20	2021-12-30 18:14:34	Normal	Service	ms-n42hcp96-2	Service Sync Success. RetrunCode: S2000	89

基本信息 实例列表 监控 事件 **日志** 更新记录

仅展示7日内的日志数据，如需查看更多日志请使用CLS日志投递，前往CLS 查看日志数据

实例名称: 全部实例 时间范围: 近1小时 2021-12-31 13:31:42 ~ 2021-12-31 14:31:42 自动刷新

请输入关键字搜索，多个关键字用空格分隔 检索分析

日志时间 ↓	实例名称	日志数据
2021-12-31 14:31:23	ms-n42hcp96-2-d58f664d7-vs9vg	[W 211231 14:31:23 web:2243] 405 HEAD / (9.0.0.27) 0.39ms
2021-12-31 14:31:23	ms-n42hcp96-2-d58f664d7-vs9vg	[W 211231 14:31:23 web:2243] 405 HEAD / (9.0.0.27) 0.51ms

服务更新

已部署的服务支持更新实例调节信息用于调整扩缩容策略，支持更新实例容器信息用于更新迭代模型，且在多实例的情况下更新服务时，后台会对多实例进行分批滚动更新，不影响生产业务对模型服务的调用。

1. 在线服务列表页面，单击服务名称进入版本列表页后，单击服务版本更新操作进入服务更新页面。

更新支持编辑修改副本设置、请求限流、副本调节等服务参数。您也可通过更新升级基底模型版本。请注意：在单实例的情况下更新操作会导致服务重启，服务调用暂时不可用；在多实例的情况下更新的升级操作您可通过更新参数配置页面的滚动更新策略进行配置。

test-source-cangku
腾讯云TI平台产品文档

服务管理
服务调用

流量分配
新增版本

名称	ID	服务版本	状态	计费模式	运行/期望 Pods ...	服务详情	流量分配 (%)	当前占用资源	描述	创建时间	操作
test-source-cangku	ms-9pt9qr9-1 卍	V1	已停止	按量付费	0/0	卍 卍	●			2023-03-28 17:2...	启动 更新 删除 扩缩容

副本设置

请确保您选择的存储实例 (CFS) 和纳管资源组的节点网络互通

模型来源 模型仓库 CFS 镜像

模型和运行环境 内置大模型 / DeepSeek 系列模型 / DeepSeek-R1-Distill-Qwen-7B

开启gRPC

端口 8501

算力规格 28C116GB A10*1 [大模型推理所需资源指南](#)

最低推荐12C44GB A10*1

▶ 高级设置

请求限流 不限流 单副本QPS 单副本最大并发数

副本调节 手动调节 自动调节

副本数量 - 1 +

CLS 日志投递 TI 控制台会默认展示 15 天的日志, 若您期望持久化存储日志, 获得日志检索等服务, 可以开启 CLS 日志投递, CLS 产品介绍和收费指南请查看[文档](#)

重试策略 有限次重试 无限次重试

服务访问云上VPC资源 开启后配置VPC和子网, 会在您账户下的孩子网, 创建一个弹性网卡进行绑定, 独立计费, 弹性网卡当前未计费

滚动更新策略

MaxSurge 百分比 - 0 + %

请确保启动滚动更新时资源组中有足够的空闲资源来部署MaxSurge实例

MaxUnavailable 百分比 - 33 + %

健康检测 编辑配置

自动停止 开启后, 在线服务将在指定的停止时间自动停止, 同时停止服务算力计费

标签 暂无标签

配置价格

服务配置费用

2. 若需进行服务扩缩容操作, 可直接单击**扩缩容**进行快捷实例更新操作, 当扩容的实例状态为运行中, 流量分配至扩容实例。
3. 若需更新模型信息, 可在实例容器模块修改模型文件或运行环境。
4. 配置信息确认无误后, 单击**启动服务**完成服务参数配置的更新操作。

版权所有: 腾讯云计算 (北京) 有限责任公司

第132 共212页

5. 在服务版本列表页单击**服务版本名称**，进入**更新记录**模块，可查看当前服务版本历史的更新记录信息。



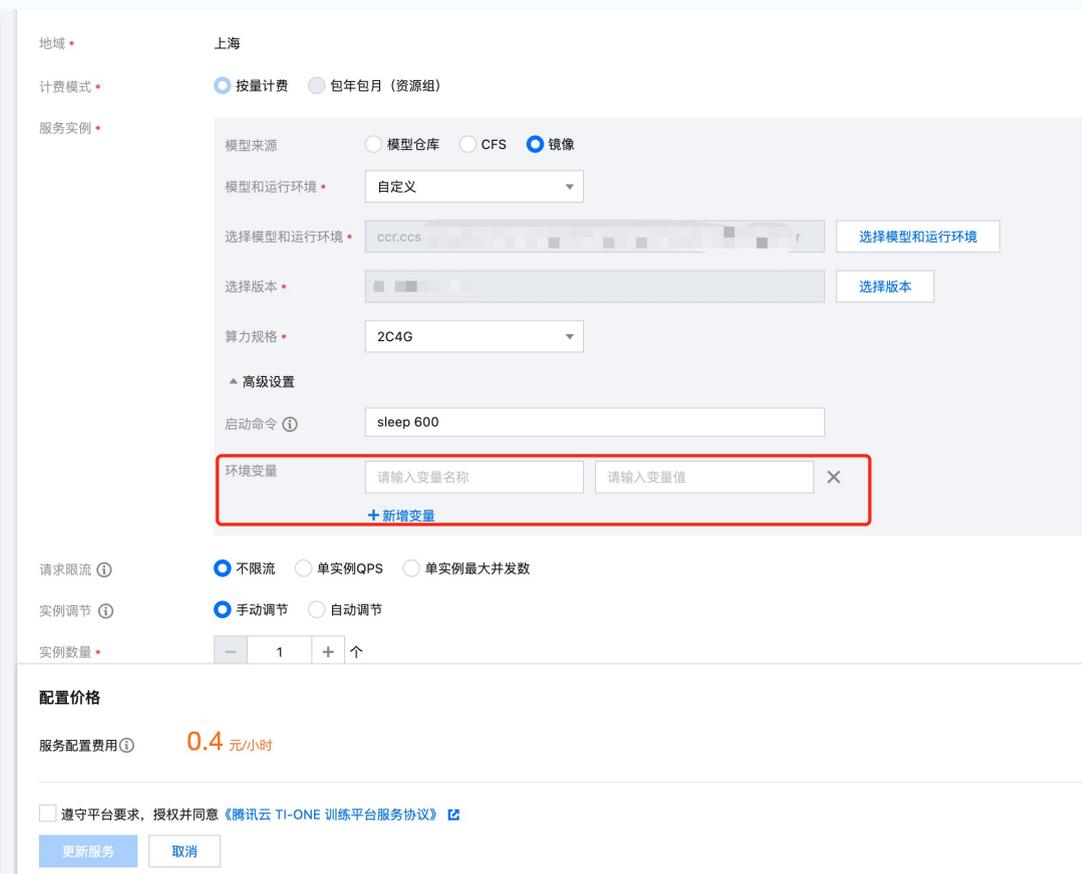
说明:

当您更新 COS、CFS、GooseFSx 里的内容时，可通过以下方式触发服务更新：

- 重建实例，您可单击名称进入服务管理，再单击服务名称进入实例列表。



- 更新服务时，填写一个环境变量，内容随意。



隔离异常实例

当服务的某个实例出现异常导致服务质量受影响时，平台支持用户在诊断期间手动隔离该实例的流量，而不需要重建实例 Pod。这样可以保留现场进行诊断，并支持在修复后手动恢复流量。

单击服务名称进入实例列表的详情页面，针对“实例状态”是“运行中”的实例，用户可单击“操作 > 隔离”按钮主动隔离指定实例，此后流量将不会再分配到被隔离的实例上。

腾讯云 TI 平台

← 内置大模型的demo演示 在线服务简介

基本信息 实例列表 监控 事件 日志 更新记录

实例名称	实例id	实例ip	机器来源	实例状态	创建时间	开始排队时间	操作
ms-2hk5x5q5-1-76984d6b7c-vvwcfc	f751acbd-f9af-490f-ba61-bae7d8732fbb	9.0.112.77	从TIONE平台购买-按量计费	运行中	2025-05-29 13:03:44	2025-05-29 13:03:44	监控 日志 隔离

隔离实例

实例被隔离后，状态将从“运行中”变为“容器就绪中（已隔离）”，且不会再接收新的流量请求。
您确定要隔离实例【ms-2hk5x5q5-1-76984d6b7c-vvwcfc】？

确定

取消

针对已隔离的实例，用户可单击“操作 > 解除隔离”手动恢复该实例，解除隔离成功后，实例会重新恢复到“运行中”的状态。

腾讯云 TI 平台

← 内置大模型的demo演示 在线服务简介

基本信息 实例列表 监控 事件 日志 更新记录

实例名称	实例id	实例ip	机器来源	实例状态	创建时间	开始排队时间	操作
ms-2hk5x5q5-1-76984d6b7c-vvwcfc	f751acbd-f9af-490f-ba61-bae7d8732fbb	9.0.112.77	从TIONE平台购买-按量计费	容器就绪中（已隔离）	2025-05-29 13:03:44	2025-05-29 13:03:44	监控 日志 解除隔离

使用平台内置镜像发布在线服务开发指引

最近更新时间：2024-08-22 14:43:11

平台预置了适配主流深度学习和机器学习框架（TensorFlow、PyTorch、PySpark等）的推理镜像，用户可以直接使用平台预置推理镜像部署模型，节省在线服务部署的开发成本。

推理镜像列表

镜像名称	适配框架	适配芯片
tensorflow1.15-py37	TensorFlow-1.15.0、LIGHT-3.0	GPU (CUDA-10.0), CPU
tensorflow2.4-py38	TensorFlow-2.4.0、LIGHT-3.0	GPU (CUDA-11.0), CPU
pytorch1.9.0-py38	PyTorch-1.9.0、LIGHT-3.0	GPU (CUDA-11.1), CPU
pytorch1.12.1-py38	PyTorch-1.12.1、LIGHT-3.0	GPU (CUDA-11.1), CPU
detectron2-py38	PyTorch-1.9.0、LIGHT-3.0	GPU (CUDA-11.1)
mmdetection1.4.8-py38	PyTorch-1.9.0、LIGHT-3.0	GPU (CUDA-11.1)
onnx1.11.1-py38	onnx-runtime-1.11.1	GPU (CUDA-11.1), CPU
jpmm1-py38	PySpark-2.4.5	CPU

操作指引

1. 准备开发环境

a. anaconda 安装与 python 环境准备

因为 TI 平台推理框架仅支持在 Linux 内核系统，请用户在 Linux 内核系统下进行后续操作。

如需要使用 GPU，请自行在本地安装 cuda（对应版本请查看表1. 推理镜像列表）。

在开始模型包开发之前，需要用户先进行 anaconda 的安装和 python 虚拟环境搭建，[anaconda 下载地址](#)。

anaconda 安装完成之后，使用 anaconda 创建 python 的虚拟环境，假设本次创建的虚拟环境名称为 TIONE-test，请根据实际需求指定 python 版本为 3.7 或者 3.8（暂时只支持这两个版本的环境安装）：

```
conda create -n TIONE-test python=3.8
conda activate TIONE-test
```

b. 安装推理框架相关依赖

TIONE 平台提供了推理框架与相关依赖的安装脚本：[脚本下载](#)

脚本主要功能包含安装 tiinfer 推理框架、安装 python 基本功能库、安装 tensorflow/torch 等环境库、平台 demo 和模型文件下载等功能，可以使用 `python init_infer_env.py -h` 查看使用方法，具体参数含义见下表。“脚本依赖”和“已有的 python 环境”，需要用户在开发机本地自行安装 cuda。

参数名称/缩写	说明	可填参数	是否必填
--ti-infer-version/-v	tiinfer 版本	1.0	否
--model-format/-m	模型格式	TorchScript、Detectron2、ONNX、FrozenGraph、SavedModel、MMDetection、PMML、HuggingFace	是

--framework/-f	推理框架, torch 或者 tensorflow的版本, 与模型格式相对应	torch1.9.0、torch1.12.1、onnx1.11.0、tf1.15.0、tf2.4.0、jpmml0.6.2	是
--model-scene/-s	模型使用场景, 默认 detect	detect、classify、nlp、ocr、recommend	否
--demo-dir/-d	下载的路径, 默认不做下载	-	否
--index-url/-i	镜像源地址, 默认值: http://mirrors.tencent.com/pypi/simple	-	否

脚本安装环境细节与对应安装指令见下表, 其中 {} 内为可选参数, 选择其中某一个填入即可。

请勿使用未隔离的同一python环境安装不同的镜像环境, 这样可能造成依赖包之间的关联出现问题。如果需要多种环境, 推荐对不同的镜像名称隔离出多个python环境后再分别进行安装。

模型格式	对应镜像名称	安装依赖	安装指令
PYTORCH TorchScript	pytorch1.9.0-py38	pytorch==1.9.0 opencv-contrib-python==4.6.0.66 opencv-python==4.6.0.66 setuptools==59.5.0 mncv-full==1.4.8 transformers==4.19.4 torchvision==0.10.0 easyocr==1.6.2	python init_infer_env.py -- framework torch1.9.0 --model- format TorchScript --model-scene {detect, classify, ocr, nlp} -- demo-dir ./ 或 python init_infer_env.py -- framework torch1.9.0 --model- format HuggingFace --model-scene nlp --demo-dir ./
	pytorch1.12.1-py38	pytorch==1.12.1 opencv-contrib-python==4.6.0.66 opencv-python==4.6.0.66 mncv-full==1.4.8 transformers==4.23.0 torchvision==0.13.1 easyocr==1.6.2	python init_infer_env.py -- framework torch1.12.1 --model- format TorchScript --model-scene {detect, classify, ocr, nlp} -- demo-dir ./
DETECTRON2	detectron2-py38	pytorch==1.9.0 opencv-contrib-python==4.6.0.66 setuptools==59.5.0 detectron2 torchvision==0.10.0	python init_infer_env.py -- framework torch1.9.0 --model- format Detectron2 --model-scene detect --demo-dir ./
MMDETECTION	mmdetection1.4.8-py38	pytorch==1.9.0 opencv-contrib-python==4.6.0.66 opencv-python==4.6.0.66 setuptools==59.5.0 mncv-full==1.4.8 transformers==4.19.4	python init_infer_env.py -- framework torch1.9.0 --model- format MMDetection --model-scene detect --demo-dir ./
ONNX	onnx1.11.0-py38	torch==1.9.0 onnx==1.11.1 onnxruntime-gpu==1.11.1 opencv-contrib-python==4.6.0.66 setuptools==59.5.0	python init_infer_env.py -- framework onnx1.11.0 --model- format ONNX --model-scene detect --demo-dir ./

TENSORFLOW	tensorflow2.4-py38	tensorflow==2.4.0 opencv-contrib-python==4.6.0.66 setuptools==59.5.0	<pre>python init_infer_env.py -- framework tf2.4.0 --model-format FrozenGraph --model-scene nlp -- demo-dir ./</pre> 或 <pre>python init_infer_env.py -- framework tf2.4.0 --model-format SavedModel --model-scene {nlp, recommend} --demo-dir ./</pre>
	tensorflow1.15-py37	tensorflow==1.15.0 opencv-contrib-python==4.6.0.66 setuptools==59.5.0	<pre>python init_infer_env.py -- framework tf1.15.0 --model-format FrozenGraph --model-scene nlp -- demo-dir ./</pre>
PMML	jpmmml-py38	jpmmml_evaluator==0.6.2 此环境依赖openjdk 1.8, 需要用 户自行在本地安装	<pre>python init_infer_env.py -- framework jpmmml0.6.2 --model- format PMML</pre>

2. 本地开发调试模型

用户需要自行开发 model_service.py 文件用于平台的推理服务，开发方法详细讲解可以参考 [模型推理文件简介和示例](#)，这里以 TorchScript 模型格式为例，进行模型包目录规范和 model_service.py 开发方法的讲解。

- model_service.py 需要实现 tiinfer.Model 类对应的函数，包括 init 初始化、load 模型加载、preprocess 数据预处理、predict 模型推理、postprocess 后处理函数。
- 初始化函数 init 中，主要是对脚本以及模型参数进行设置。
- load 函数中，对模型进行了加载，并设置了类别信息，这里的类别信息主要是为输出时提供数据，用户可根据自身需求选择是否需要。
- preprocess 函数中，对请求输入的检测图片进行读取，并标准化到模型规定的输入大小。
- predict 函数执行推理过程。
- postprocess 函数对推理结果进行处理并打包。

检测模型推理脚本 Pytorch 实现示例：

```
import os
import logging
import torch
from typing import Dict, List, Union
import numpy as np
import cv2
cv2.setNumThreads(5)
import tiinfer
import tiinfer.utils
import base64
import urllib

from utils.general import non_max_suppression, scale_coords
from torchvision import transforms

# 如果使用加速，需添加加速库引用：tiacc_inference
# import tiacc_inference

# 未使用加速时模型路径
PYTORCH_FILE = "model/yolov5s_ts.pt"
# 加速后模型路径
# TIACC_FILE = "model/tiacc.pt"
```

```
class YoloV5DetectModel(tiinfer.Model):
    def __init__(self, model_dir: str):
        super().__init__(model_dir)
        # workers: 平台参数, 设置进程数, 非必填, 默认值: 1
        self.workers = 1
        # 以下为自定义参数, 根据模型实际需求定义
        self.model = None
        self.imgsz=640 # 输入图片大小
        self.conf_thres=0.05
        self.iou_thres=0.45 # for NMS
        self.max_det = 20
        self.use_gpu = torch.cuda.is_available()
        self.device = torch.device('cuda:0' if self.use_gpu else 'cpu')
        self.resize = transforms.Resize((self.imgsz, self.imgsz))
        self.normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

# 加载模型
def load(self) -> bool:
    try:
        # 拼装模型路径, self.model_dir: 当前工作目录; 当为加速模型时, 请将'PYTORCH_FILE'替换为'TIACC_FILE'
        model_file = os.path.join(self.model_dir, PYTORCH_FILE)
        logging.info("model_file %s", model_file)
        # 加载模型
        self.model = torch.jit.load(model_file, map_location=self.device)
        # 设置类别信息
        self.det_obj_names = ['person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train',
                              'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parking meter', 'bench',
                              'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe',
                              'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard',
                              'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard',
                              'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl',
                              'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut',
                              'cake', 'chair', 'couch', 'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop',
                              'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven', 'toaster', 'sink',
                              'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair drier',
                              'toothbrush']
        logging.info("load model to device %s" % (self.device) )
    except Exception as e: # pylint: disable=broad-exception
        logging.error("load model failed: %s", e)
        return False
    return True

# 预处理请求体
def preprocess(self, request: Dict) -> Dict:
    try:
        img_data = request["image"]
        # image_to_cv2_mat能够正确处理本地图片、图片url及base64编码的图片内容, 将之转换为opencv2中的矩阵
        img_mat = tiinfer.utils.image_to_cv2_mat(img_data)
        # 纬度转换
        img_mat = torch.Tensor(img_mat).permute((2, 0, 1)).to(self.device)

        # 如果在加速场景, 图片尺寸必须在加速配置的tensor范围内, 平台提供图片等比扩缩与填充接口:
        tiinfer.scale_fill_image方便用户使用。
        # 例如: 加速模型tensor设置的范围为 [720, 2080], 等比扩缩图片的范围 [1080, 1999], 等比扩缩后最小边不在tensor
        范围内, 需要做补边操作, 例如下面最小边需要补边到800。
        # min_size=1080; #图片扩缩小边最小范围值
        # max_size=1999; #图片扩缩大边最大范围值
        # padding_limit=800; #填充时的短边上限
```

```
# image, self.scaleratio = tiinfer.scale_fill_image(image, min_size, max_size,
padding_limit, Horizontal.RIGHT,Vertical.LOWER) 具体使用指引。

# 数据标准化
c, h, w = img_mat.shape
img_tensor = self.normalize(self.resize(img_mat).type(torch.float32)/255)
return {"img_tensor": img_tensor.unsqueeze(0), "original_dims": [w, h]}
except Exception as e:
    logging.error("Preprocess failed: %s" % (e))
    return {"error": "preprocess failed"}

# 推理执行, 参数request是preprocess方法的返回结果
def predict(self, request: Dict) -> Dict:
    with torch.no_grad():
        try:
            # 获取预测结果
            out = self.model(request["img_tensor"])
            # 可选操作, 根据实际情况执行
            # 对预测结果进行非极大值抑制
            preds, nms_t = non_max_suppression(out, self.conf_thres, self.iou_thres,
max_det=self.max_det)
            data = {
                "pred_out": preds[0].to('cpu'),
                "original_dims": request["original_dims"]
            }
            return data
        except Exception as e:
            logging.error("Failed to predict" % (e))
            request["predict_err"] = str(e)
            return request

# 后处理, 参数request是predict方法的返回结果
def postprocess(self, request: Dict) -> Dict:
    try:
        # 可选操作, 根据模型输出与实际需求对预测结果进行后处理
        pred_out = request["pred_out"]
        original_dim = request["original_dims"]
        # 调整图片形状
        pred_bbox = scale_coords([self.imgsz, self.imgsz], pred_out[:, :4], original_dim).round()
        # 输出参数, 根据网络实际情况定义
        labels = []
        confs = []
        label_indx = []
        boxes = []
        for i in range(pred_out.shape[0]):
            obj = pred_out[i, :].tolist()
            boxes.append(pred_bbox[i, :].tolist())
            confs.append(obj[4])
            label_indx.append(int(obj[5]))
            labels.append(self.det_obj_names[int(obj[5])])
        image_result = {
            "det_boxes": boxes,
            "det_labels": labels,
            "det_labels_idx": label_indx,
            "det_scores": confs
        }
    }
    return {
        "result": {
            "det_objs": image_result
```

```

    }
}
except Exception as e:
    logging.error("Postprocess failed: %s" % (e))
    request.pop("image", '')
    request.pop("instances", '')
    request.pop("predictions", '')
    request["post_process_err"] = str(e)
    return request

```

3. 在 TI 平台部署在线服务前测试

在上述环境安装完成后，用户可在本地开发机中进行模型测试。TI平台推理框架默认模型包路径为 `/data/model/`，在开始测试前请确认将模型包已放入该路径下；如有需要，也可以在系统环境变量文件 `/etc/profile` 中添加 `export TI_MODEL_DIR=/xxx` 自定义模型包路径，默认模型包路径请指向 `model_service.py` 所在的文件夹。

在模型包开发完成后，进入 `model_service.py` 所在路径，以框架默认路径为例，请在 `/data/model/` 路径下运行以下代码，以启动服务：

```
python3 -m tiinfer --timeout 30000
```

服务启动成功后，调用端口为8501，可使用测试用例在本地调用，验证服务可用性。

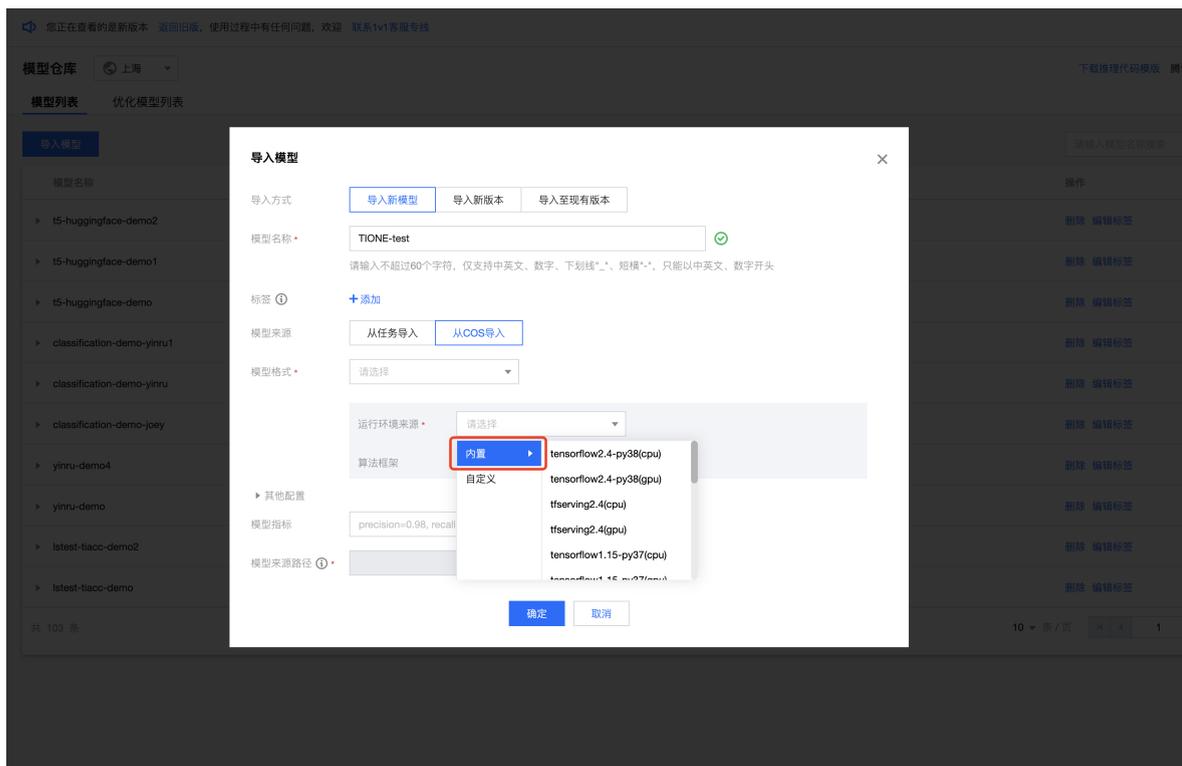
```
curl -X POST -H "Content-Type: application/json" http://127.0.0.1:8501/v1/models/m:predict -d @test.json
```

4. 在 TI 平台上部署模型服务

用户如需使用平台预置推理镜像，可在模型仓库模块导入模型时，关联内置运行环境，则使用该模型启动服务时，会自动拉取对应的平台预置推理镜像进行服务部署。

步骤1:

将本地模型上传到 COS 中，并在 TIONE 平台-模型仓库中导入模型。



步骤2:

在在线服务模块选择对应的模型发布服务。

创建服务

服务名称: ✔
请输入不超过60个字符, 仅支持中英文、数字、下划线“_”、短横线“-”, 只能以中英文、数字开头

服务版本:

服务描述 (选填):
0 / 500

地域: 广州

计费模式: 按量付费 包年包月 (预付费)

服务实例:

模型来源: 从模型仓库选择模型 从容器镜像服务选择镜像
此选项将导入您在模型仓库中注册的模型, 您可以前往[模型仓库](#)注册模型, 目前模型仓库仅支持TorchScript、Detectron2、ONNX、Frozen Graph、Saved Model、MMDetection、HuggingFace、PMMML格式的模型, 模型格式和平台内置推理引擎相关指引信息请查看[文档](#)

模型类型:

选择模型:

选择版本:

选择镜像:

镜像版本:

使用自定义镜像发布在线服务开发指引

最近更新时间：2025-06-12 14:27:11

前言

本文将向您介绍 TI-ONE 自定义镜像的2种方式，和需要遵循规范约束，再通过典型案例向您演示如何制作自定义镜像，发布在线服务。

基于平台 tiinfer 框架基础镜像制作自定义镜像

tiinfer 框架基础镜像说明

平台提供了内置 tiinfer 框架的基础推理镜像：

```
ccr.ccs.tencentyun.com/tione-public-images/ti-cloud-gpu-base-tiinfer:py38-cu111-1.0.0
```

基础镜像基于 centos 制作，其中包含的软件包有：

软件或包	版本
CUDA	11.1.1
python	3.9.13
cos-python-sdk-v5	1.9.14
coscmd	1.8.6.24
numpy	1.23.1
msgpack	1.0.5
opencv-python	4.6.0.66
opencv-contrib-python	4.6.0.66
pandas	1.4.3
Pillow	9.4.0
tiinfer	0.1.1
mosec-tiinfer	0.0.6

- 基础镜像的启动命令 `/usr/local/service/ti-cloud-infer/entrypoint.sh`
- `entrypoint.sh` 中的内容为：

```
#!/bin/bash
source /etc/profile
source /root/.bashrc
export LD_LIBRARY_PATH=/usr/local/python3/lib/python3.8/site-packages/torch/lib:/usr/local/openmpi/lib:/usr/local/nccl/lib:/usr/local/cuda/lib64:/usr/local/python3/lib:/usr/local/python3/lib64:/usr/local/openmpi/lib:/usr/local/gcc/lib:/usr/local/gcc/lib64

MODEL_DIR=/data/model

echo "===== code path ${MODEL_DIR}======"
cd ${MODEL_DIR}

if [ -f "requirements.txt" ]; then
    echo "===== install python requirements ====="
    echo "python3 -m pip install -r requirements.txt"
    python3 -m pip install -r requirements.txt
```

```

echo "===== install python requirements done ====="
fi

echo "===== start serving ====="
echo "python3 -m tiinfer"
export TI_MODEL_DIR=${MODEL_DIR}
python3 -m tiinfer --timeout 30000
    
```

• 启动逻辑为:

- 1.1 读取环境变量 `${MODEL_DIR}` 目录下的 requirements.txt 文件, 使用 pip 安装其中指定的依赖 python 包。
- 1.2 tiinfer 框架会读取环境变量 `${MODEL_DIR}` 下的文件, 加载模型后, 启动一个 HTTP 服务并监听在环境变量 `${REST_PORT}` 定义的端口。
- 1.3 tiinfer 框架启动时, 会从 model_service.py 文件中加载模型。

自定义镜像规范

1. Dockerfile 文件中添加对基础镜像的引用, 例如:

```
FROM ccr.ccs.tencentyun.com/tione-public-images/ti-cloud-gpu-base-tiinfer:py38-cu111-1.0.0
```

2. 自定义逻辑实现集中在 model_service.py 文件及 entrypoint.sh 文件的修改。

3. 使用 CFS、COS、或 GooseFS 作为模型来源时, 平台默认将源路径下的模型文件 (包括子目录), 放在服务实例的 /data/model 目录下。因此自定义的代码及数据不能置于 /data/model 目录, 否则会被平台覆盖。

制作镜像

本案例介绍了基于 tiinfer 框架基础镜像, 通过修改 model_service.py 及 entrypoint.sh 文件, 实现一个简单的加法器。

注意: 本案例不使用平台提供的模型仓库功能托管模型, 而是将模型、推理代码直接封装到镜像中, 所以需要避免将模型、代码放到 /data/model 目录。

编写代码

一共包含三个文件:

文件	作用
model_service.py	按照 tiinfer 的要求, 编写加法器模型。
entrypoint.sh	启动脚本, 可在此自行安装更多的依赖包。
Dockerfile	负责将前两个文件拷贝到镜像中。

1. model_service.py 的内容:

```

from typing import Dict
import tiinfer

class AdderModel(tinfer.Model):
    def __init__(self, model_dir: str):
        super().__init__(model_dir)

    def load(self) -> bool:
        self.ready = True
        return self.ready

    def preprocess(self, request: Dict) -> Dict:
        return request

    def predict(self, request: Dict) -> Dict:
        return {'result': request['a'] + request['b']}

    def postprocess(self, result: Dict) -> Dict:
        return result
    
```

2. entrypoint.sh 的内容:

```
#!/bin/bash
source /etc/profile
source /root/.bashrc
export LD_LIBRARY_PATH=/usr/local/python3/lib/python3.8/site-
packages/torch/lib:/usr/local/openmpi/lib:/usr/local/ncccl/lib:/usr/local/cuda/lib64:/usr/local/python
3/lib:/usr/local/python3/lib64:/usr/local/openmpi/lib:/usr/local/gcc/lib:/usr/local/gcc/lib64

MODEL_DIR=/opt/model

echo "===== code path ${MODEL_DIR}======"
cd ${MODEL_DIR}

if [ -f "requirements.txt" ]; then
    echo "===== install python requirements ====="
    echo "python3 -m pip install -r requirements.txt"
    python3 -m pip install -r requirements.txt
    echo "===== install python requirements done ====="
fi

echo "===== start serving ====="
echo "python3 -m tiinfer"
export TI_MODEL_DIR=${MODEL_DIR}
python3 -m tiinfer --timeout 30000
```

注意: 上述代码中的 `MODEL_DIR=/opt/model` 这一行, 将启动目录由默认的 `/data/model` 改为 `/opt/model`, 避免被平台覆盖。

3. Dockerfile 的内容:

```
FROM ccr.ccs.tencentyun.com/tione-public-images/ti-cloud-gpu-base-tiinfer:py38-cu111-1.0.0

COPY model_service.py /opt/model/model_service.py
COPY entrypoint.sh ./entrypoint.sh
RUN chmod +x ./entrypoint.sh
```

需要注意的是, 上述代码将 `model_service.py` 拷贝到 `/opt/model` 目录, 而非默认的 `/data/model` 目录, 避免被平台覆盖。

打包镜像

1. 整体步骤:

- 本地配置 docker 环境, 并开通 腾讯云容器镜像服务;
- 创建命名空间及新建个人镜像仓库;
- 编译自定义推理镜像, 推送到个人镜像仓库;
- 在启动模型服务时, 实例容器栏选择不使用模型文件, 选择运行环境进入个人镜像仓库列表, 选择上一步推送的自定义镜像环境;
- 配置好参数, 启动服务。

2. 详细说明:

执行如下命令来打包:

```
docker build . --tag ccr.ccs.tencentyun.com/YOUR_NAMESPACE/YOUR_IMAGENAME
```

打包完成后, 可以通过如下方式在本地检查服务运行是否正常:

- 执行 `docker run -d --name myinfer ccr.ccs.tencentyun.com/YOUR_NAMESPACE/YOUT_IMAGENAME` 将服务运行起来;
- 执行 `docker exec -it myinfer bash` 进入容器中;
- 在容器中执行 `curl http://127.0.0.1:8501/v1/models/m:predict -d '{"a": 1, "b": 2}'` 得到正确返回: `{"result": 3}`
- 退出容器, 回到本地环境, 上传镜像: `docker push ccr.ccs.tencentyun.com/YOUR_NAMESPACE/YOUR_IMAGENAME`。

基于其他推理框架制作自定义镜像

平台支持使用其它推理框架，通过自定义镜像的方式来部署模型在线服务。

自定义镜像规范

1. 服务必须以 HTTP 协议接受请求，并且只支持 POST 方法。
2. 使用 CFS、COS、GooseFSx 作为模型来源时，平台默认将源路径下的模型文件（包括子目录），放在服务实例的 `/data/model` 目录下。因此自定义的代码及数据不能置于 `/data/model` 目录，否则会被平台覆盖。
3. 镜像在本地经过验证，可以正常提供服务。

上传自定义镜像并发布推理服务

上传自定义镜像

1. 登录 [腾讯云容器镜像服务](#)。
2. 在镜像仓库页面，单击新建。
3. 上传镜像。

单击镜像仓库中后的快捷指令，查看操作命令，上传镜像。



发布在线服务

在 TI-ONE【在线服务】页面，单击新建服务。

方式一：模型打包在镜像中使用

若您已将模型打包在镜像中，可以直接使用镜像发布服务。

【服务实例】详细配置：

- 模型来源选择镜像。
- 模型和运行环境选择自定义。
- 选择模型和运行环境选择之前上传的镜像。
- 端口号填写提供服务的端口号。

副本设置

请确保您选择的存储实例（CFS或者GooseFSx）和纳管资源组的节点网络互通

模型来源 模型仓库 CFS GooseFSx COS 镜像

模型和运行环境

选择模型和运行环境

选择版本

开启GRPC

端口

资源申请

卡型号 [大模型推理所需资源指南](#)

GPU 卡
若需使用GPU，卡数可填写0.1-1或1的整数倍。该机型支持填写GPU整数卡或小数卡

CPU 核

内存 G

[高级设置](#)

选择镜像

您可以从腾讯云容器镜像服务中导入镜像，点击[前往容器镜像服务产品准备镜像](#)，[镜像制作规范请查看文档](#)

容器镜像服务 个人版 容器镜像服务 企业版 镜像地址

请输入镜像名称

名称	类型	命名空间	仓库地址
qiqi-ns/qiqi-custom-1-ebui	私有	qiqi-ns	ccr.ccs.tencentyun.com/qiqi-ns/qiqi-custom-1-ebui
mia-ns/mia-ice	私有	mia-ns	ccr.ccs.tencentyun.com/mia-ns/mia-ice
luis-ns/luis-test	私有	luis-ns	ccr.ccs.tencentyun.com/luis-ns/luis-test
luis-ns/luis-test	私有	luis-ns	ccr.ccs.tencentyun.com/luis-ns/luis-test
huan-ns/huan-test-image	私有	huan-ns	ccr.ccs.tencentyun.com/huan-ns/huan-test-image

共 35 条 10 条 / 页

确定

取消

方式二：模型在 CFS，挂载到容器中使用（GooseFSx 操作和 CFS 类似）

若您的镜像仅为服务运行环境，模型可以上传到CFS后，挂载到容器内部。模型会挂载到 /data/model 目录下，您的服务可以从此目录加载模型。

【服务实例】详细配置：

- 模型来源选择 CFS
- 选择模型选择模型存储的 CFS，并填写模型存储在 CFS 中的路径。上传模型到 CFS 可参考 [导入和部署自定义 LLM 大模型](#)。
- 模型和运行环境选择自定义
- 选择模型和运行环境选择之前上传的镜像
- 端口号填写提供服务的端口号

副本设置

请确保您选择的存储实例（CFS或者GooseFSx）和纳管资源组的节点网络互通

模型来源 模型仓库 CFS GooseFSx COS 镜像

选择模型

运行环境

选择自定义镜像环境

环境版本

开启GRPC

端口

资源申请 [大模型推理所需资源指南](#)

GPU 卡
 若需使用GPU，卡数可填写0.1-1的整数倍。该机型支持填写GPU整数卡或小数卡

CPU 核

内存 G

▶ 高级设置

模型推理文件简介和示例

最近更新时间：2024-08-21 11:54:41

TI 内置推理框架 tiinfer 介绍

tiinfer 是腾讯云 TI-ONE 模型服务使用的推理框架。基于 [mosec](#) 提供的基础能力，tiinfer 提供了以下功能：

- 高性能的 HTTP 服务
- 自由编排计算过程

安装

tiinfer 支持 Gnu/Linux 系统，依赖 python 3.8 及以上版本。需要使用以下命令安装 PyPI 包：

```
pip install -i https://mirrors.cloud.tencent.com/pypi/simple tiinfer
```

HelloWorld!

以下代码展示了一个最简单的 HelloWorld 示例：

```
from typing import Dict
import tiinfer

### 日志相关代码 ###
# 增加以下代码，才能将框架记录的请求日志打印到终端
import logging
import sys
logger_formatter = "%(asctime)s %(levelname)s %(module)s:%(lineno)d %(message)s"
logging.basicConfig(stream=sys.stdout, format=logger_formatter, level=logging.DEBUG)
### 日志相关代码 ###

# tiinfer支持原生的mosec.Worker
class HelloWorld(mosec.Worker):
    def forward(self, req: Dict) -> Dict:
        return {"hello": f"world. raw req is {req}"}

# 启动两个进程同时处理请求
tiinfer.append_worker(HelloWorld, num=2)
```

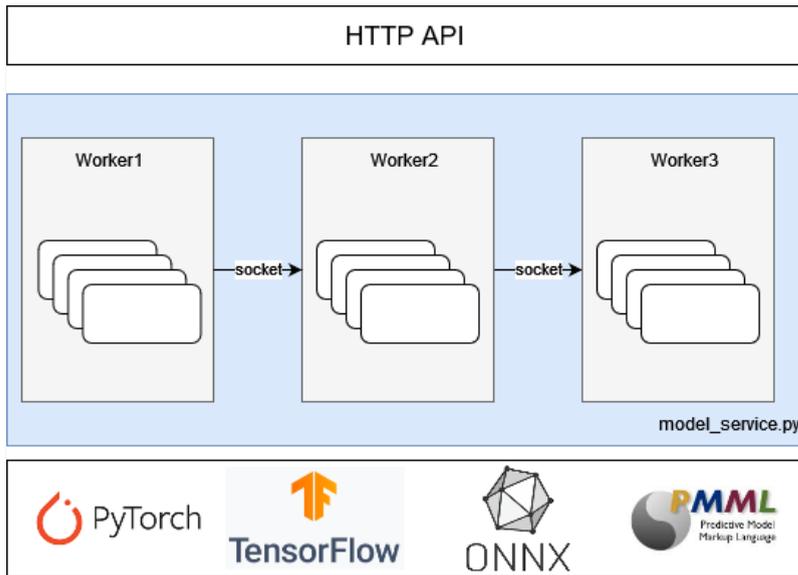
将上述代码保存为 `model_service.py` 文件。执行以下命令可以拉起整个推理服务：

```
TI_MODEL_DIR=`pwd` python3 -m tiinfer --timeout 30000
```

其中，配置项 `timeout` 值表示 Web 服务超时时间，单位为毫秒。模型拉起后，会在 8501 端口启动一个 HTTP 服务，请求地址为 `/v1/models/m:predict`。用以下命令可以访问：

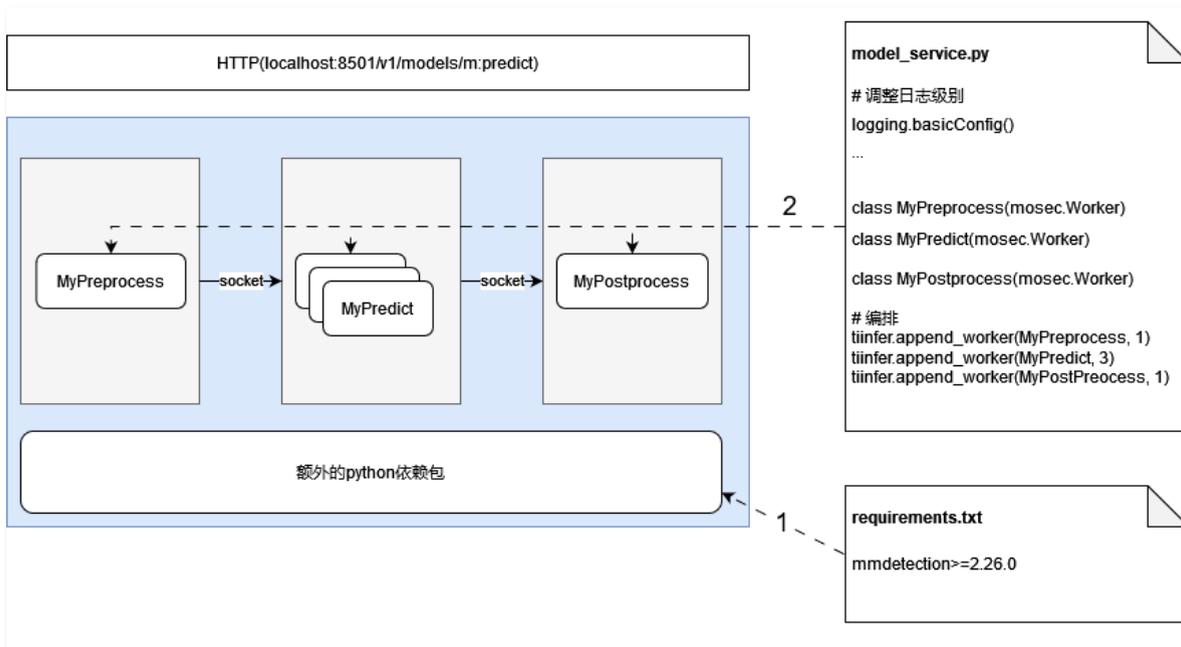
```
> curl -X POST -d '{"key": "values"}' http://127.0.0.1:8501/v1/models/m:predict
{
  "hello": "world. raw is {'key': 'values'}"
}
```

tiinfer的架构



为了降低模型的上线难度，平台提供的 tiinfer 镜像已经打包了诸多流行的推理引擎，会读取客户提供的 model_service.py 文件，自动拉起 HTTP 服务。具体的服务启动过程如下所示：

1. 使用 pip 安装 requirements.txt 文件中填写的 python 依赖
2. 读取 model_service.py 并实例化几个处理过程的进程
3. 启动 HTTP 服务



框架内置环境变量

框架默认为单进程模式，用户可根据需要调整框架内置环境变量，启用多进程模式，充分利用资源。

环境变量	说明	默认值
TI_MODEL_DIR	模型路径	/data/model/
TI_PREPROCESS_NUMS	预处理进程数	0
TI_INFERENCE_NUMS	推理进程数	1
TI_POSTPROCESS_NUMS	后处理进程数	0

TI_INFERENCE_MAX_BATCH_SIZE	推理Batch数	1
-----------------------------	----------	---

说明:

1. 当TI_PREPROCESS_NUMS==0且TI_POSTPROCESS_NUMS==0时
model_service.py 文件中preprocess, predict, postpress函数在一个进程中执行。
2. 否则, model_service.py 文件中preprocess, predict, postpress函数分别处于不同进程中执行。
3. load 函数与predict 函数处于同一进程。
4. 注意TI_INFERENCE_MAX_BATCH_SIZE的值的调整, 避免引发GPU OOM。

自定义推理Worker

每一个处理 Worker 需要继承 `mosec.Worker` 来定义处理逻辑:

1. 可选: 覆写 `__init__` 函数, 进行一些初始化工作。只在必要的时候覆写, 注意必须先调用 `super.__init__()` 以完成父类初始化。
2. 必须: 自定义 `forward` 函数, 提供处理能力。通常在 `forward` 中完成数据的处理工作。

```
class Worker:
    def __init__(self)

    def forward(data)
```

在完成定义推理 Worker 后, 需要调用 `tiinfer.append_worker()` 函数来进行编排:

```
def append_worker(
    worker: Type[mosec.Worker],
    num: int = 1,
    start_method: str = "spawn",
    env: Union[None, List[Dict[str, str]]] = None,
) -> None
"""
worker: 继承mosec.Worker并实现了forward方法的处理Worker
num: 并行计算的进程数(>=1)
start_method: 进程启动方法("spawn" 或 "fork")
env: 进程启动前的一些环境变量
"""
```

完整的推理过程中, 一般需要对输入进行预处理, 再输入到模型进行推理, 并对推理结果进行后处理, 最终才返回给调用方。推理过程通常是在 GPU 上完成的, 而预处理、后处理往往是 CPU 计算甚至是一些 IO 处理。如果将预处理、后处理同推理放在同一个进程中处理, 启动的进程数目主要受限于 GPU 的显存及算力。此时, 将预处理、后处理同推理进程剥离, 能够充分利用 CPU 的处理能力, 参考以下代码片段:

```
import tiinfer
from mosec import Worker
from typing import Dict, Any

class MyPreprocess(Worker):
    def forward(self, data: Dict) -> Any:
        # 输入是json转化来的Dict, 进行一些必要的预处理

class MyPredict(Worker):
    def __init__(self):
        super.__init__()
        # 读取并加载模型

    def forward(self, data: Any) -> Any:
```

```
# 输入是预处理的结果，通过调用模型推理后，得到推理结果

class MyPostprocess(Worker):
    def forward(self, data: Any) -> Dict:
        # 输入为推理结果，通过一些后处理手段转化为Dict作为Json返回给调用方

# 编排处理过程：1x预处理Worker -> 2x推理Worker -> 1x后处理Worker
tiinfer.append_worker(MyPreprocess, 1)
tiinfer.append_worker(MyPredict, 2)
tiinfer.append_worker(MyPostprocess, 1)
```

model_service.py 简介

对 model_service.py 的要求

参考上文 tiinfer 介绍中的相关章节。

分类模型推理脚本 Pytorch 实现示例

```
import logging
import os
import time
from typing import Dict, List
from urllib.request import urlretrieve

import cv2 # type: ignore
import numpy as np # type: ignore

import torch # type: ignore

import tiinfer
import tiinfer.utils
import mosec

### 日志相关代码 ###
# 增加以下代码，才能将框架记录的请求日志打印到终端
logger = logging.getLogger()
logger.setLevel(logging.DEBUG)
formatter = logging.Formatter(
    "%(asctime)s - %(process)d - %(levelname)s - %(filename)s:%(lineno)s - %(message)s"
)
sh = logging.StreamHandler()
sh.setFormatter(formatter)
logger.addHandler(sh)
### 日志相关代码 ###

# 预处理过程对输入的base64字符串进行解码，并按照模型要求做一些缩放操作
class Pre(mosec.Worker):
    def forward(self, req: Dict) -> cv2.Mat:
        # 预处理的输入数据是json转化而来的Dict
        img_base64_bytes = req["image"]
        img = tiinfer.utils.image_to_cv2_mat(img_base64_bytes)
        # bgr -> rgb
        img = img[:, :, ::-1]
        # 对图片做一些预处理
```

```
img = cv2.resize(img, (256, 256))
crop_img = (
    img[16 : 16 + 224, 16 : 16 + 224].astype(np.float32) / 255
) # center crop
crop_img -= [0.485, 0.456, 0.406]
crop_img /= [0.229, 0.224, 0.225]
crop_img = np.transpose(crop_img, (2, 0, 1))
return crop_img

# 加载模型，并对预处理后的结果做推理操作，并转换为最终结果传给调用方
class Infer(mosec.Worker):
    def __init__(self) -> None:
        super().__init__()
        # 获取当前所在目录，需要据此加载模型
        self.root_path = tiinfer.TI_MODEL_DIR
        # 优先使用GPU
        self.device = (
            torch.device("cuda") if torch.cuda.is_available() else torch.device("cpu")
        )
        ### 非加速模型的加载 开始###
        # 模型存在model目录下面
        model_file = os.path.join(self.root_path, "model/resNet50.pt")
        ### 非加速模型的加载 结束###

        ### 加速模型的加载 开始###
        # 加速模型需要额外导入tiacc_inference
        # import tiacc_inference
        # model_file = os.path.join(self.root_path, "model/tiacc.pt")
        ### 加速模型的加载 结束###

        # 加载模型
        self.model = torch.jit.load(model_file)
        self.model.eval()

        # 分类需要知道最终的类别
        self.categories = load_categories()

    def forward(self, img: cv2.Mat) -> Dict:
        with torch.no_grad():
            batch = torch.stack(
                [torch.tensor(arr, device=self.device) for arr in [img]]
            )
            pred_results = self.model(batch)
            prob = torch.nn.functional.softmax(pred_results, dim=1)
            top1_prob, top1_catid = torch.topk(prob, 1)
            return [
                {
                    "confidence": top1_prob[i].tolist()[0],
                    "pred": self.categories[top1_catid[i].tolist()[0]],
                }
                for i in range(top1_prob.size(0))
            ][0]

# 从标签文件中读取标签id对应的类别信息
def load_categories() -> List[str]:
    logging.info("loading categories file...")
    local_filename = "imagenet_classes.txt"
    if not os.path.exists("imagenet_classes.txt"):
```

```

local_filename, _ = urlretrieve(
    "https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt"
)
with open(local_filename, encoding="utf8") as file:
    return list(map(lambda x: x.strip(), file.readlines()))

# 编排处理过程：2个进程进行预处理 -> 1个进程进行推理
tiinfer.append_worker(Pre, 2)
tiinfer.append_worker(Infer, 1)
    
```

Demo 下载

平台提供了针对各类模型的 Demo，下载地址详见下表：

格式	场景	下载地址
TorchScript	分类	单击下载
TorchScript	检测	单击下载
TorchScript	NLP	单击下载
TorchScript	OCR	单击下载
Detectron2	检测	单击下载
MMDetection	检测	单击下载
HuggingFace	NLP	单击下载
SavedModel	NLP	单击下载
SavedModel	推荐	单击下载
FrozenGraph	NLP	单击下载
ONNX	检测	单击下载

TI-ACC 推理加速函数介绍

tiacc_inference.load()

TI-ONE 的 Detectron2 或者 MMDetection 格式优化后的模型，需要使用 tiacc_inference.load() 函数对模型进行加载，其他格式无需使用 tiacc_inference 的 load，使用原生格式 load 即可。

- MMDetection

原生示例：

```

from mmdet.apis import init_detector
model = init_detector(config, checkpoint, device=device)
    
```

优化后使用示例：

```

import tiacc_inference
model = tiacc_inference.load('tiacc.pt') #tiacc.pt是模型优化后生成的新模型
    
```

- Detectro2(对于 Detectron2 导出的 PyTorch 模型)

原生示例：

```

import torch
model = torch.load(checkpoint) #.pth模型文件
    
```

优化后使用示例:

```
import tiacc_inference
model = tiacc_inference.load('tiacc.pt') #tiacc.pt是模型优化后生成的新模型
```

- Detectron2(对于通过 Detectron2.modeling.build_model 构造的模型)

原生示例:

```
from detectron2.config import get_cfg
from detectron2.modeling import build_model
cfg = get_cfg()
cfg.MODEL.DEVICE = device
cfg.MODEL.WEIGHTS = checkpoint
model = build_model(cfg)
```

优化后使用示例:

```
import tiacc_inference
model = tiacc_inference.load('tiacc.pt') #tiacc.pt是模型优化后生成的新模型
```

批量预测

批量预测介绍

最近更新时间：2024-08-22 15:41:27

面向不定时批量数据处理的业务场景，平台提供批量预测的模块能力，用于支撑算力资源按需使用，批量数据处理完成后推理算力自动释放的场景诉求。

前提条件

- 已完成模型的训练并导入模型仓库，或者已将自定义推理镜像上传至腾讯云容器镜像服务 TCR；
- 已完成需要批量预测的数据准备，并上传至腾讯云对象存储 COS；
- 已在对象存储 COS 中创建好文件夹，用于存储批量预测的推理结果文件。

创建预测任务

- 登录 [TI-ONE 控制台](#)，在左侧导航栏中选择 **模型服务** > **批量预测**，进入批量预测任务页面。
- 在任务列表页面单击 **新建任务**，进入任务创建页面。
- 在任务创建页面，配置批量预测任务的相关参数。

3.1 任务基础信息

参数	说明
任务名称	预测任务的名称，按照界面提示的规则填写即可
任务描述	可按需为预测任务配置描述信息
地域	同账号下的预测任务按地域进行隔离，地域字段取值根据您在任务列表页面所选择的地域自动带入
计费模式	可选择按量付费模式或包月包月模式。（A）包年包月模式下，可使用在资源组管理模块已购买的推理资源组部署任务，算力费用在购买资源组时已支付，创建任务时无需扣费。（B）按量付费模式下，用户无需预先购买推理资源组，根据任务依赖的算力规格，创建任务时冻结两小时费用，之后每小时根据运行中的实例数量按量扣费。
资源组	若选择包年包月模式，可选择资源组管理模块的推理资源组

3.2 服务实例信息

参数	说明
模型来源	从镜像仓库选择镜像适用于，预测任务所需的自定义镜像已封装模型文件，不需要再进行模型文件挂载，且自定义镜像已上传至容器镜像服务 TCR 的场景；从模型仓库选择模型适用于，预测任务所需的模型文件已导入至模型仓库的场景。如需使用模型文件创建任务，模型文件大小建议不超过16G。
运行环境	（A）若从镜像仓库选择镜像，则运行环境可以选择已上传至容器镜像服务 TCR 的自定义镜像。（B）若从模型仓库选择模型，则运行环境会根据模型仓库的配置信息自动赋值。如需使用自定义镜像创建任务，镜像文件大小建议不超过34G。
模型推理文件	从模型仓库选择模型的场景下，需要上传模型推理文件，命名须为 model_service.py，可自定义批量预测处理逻辑
启动命令	若创建批量预测任务未指定启动命令时，则默认启动命令为 python3 /opt/ml/code/model_service.py
资源申请/算力规格	（A）包年包月模式下，可设置从所选资源组中申请多少资源用于当前任务。（B）按量付费模式下，可按需选择当前任务所需的算力规格。

3.3 数据路径配置

输入数据

数据类型: COS

文件所在路径: 选择文件

请选择对象存储 COS 中的文件夹或文件

输出数据

数据类型: COS

预测输出路径: 选择文件

请选择对象存储 COS 中的文件夹, 批量预测的结果会写入该 COS 路径下, 该路径不能与输入数据 COS 路径相同

输入数据

数据类型: WeData(HDFS)

数据源: 请选择 刷新

源路径: /

输出数据

数据类型: WeData(HDFS)

数据源: 请选择 刷新

源路径: /

参数	说明
输入数据路径	即批量预测任务待处理数据的存储路径, 可选择对象存储 COS 或者 Wedata(HDFS); 当选择COS时, 需要选择存储桶中的文件夹或文件; 当选择Wedata(HDFS)时, 需要下拉选择数据源, 填写源路径 (输入数据会默认挂载到 /opt/ml/input/data 下作为预测数据目录, 您的代码中需要使用该路径以获取数据)
输出数据路径	即批量预测任务将预测结果文件写入的目标路径, 可选择对象存储 COS 或者 Wedata(HDFS); 当选择COS时, 需要选择存储桶中的文件夹或文件; 当选择Wedata(HDFS)时, 需要下拉选择数据源, 填写源路径 (输入数据会默认挂载到 /opt/ml/output 下作为预测数据目录, 您的代码中需要使用该路径以获取数据)

3.4 任务高级配置

CLS 日志投递 TI 控制台会默认展示 7 天的日志, 若您期望持久化存储日志, 获得日志检索等服务, 可以开启 CLS 日志投递, CLS 产品介绍和收费指南请查看[文档](#)

标签 + 添加

参数	说明
CLS 日志投递	平台为用户提供免费的近7日任务日志存储, 若需要持久化日志存储以及更灵活的日志检索能力, 可开启 CLS 日志投递, 开启后任务日志会根据日志集与日志主题投递至腾讯云日志服务 CLS。
标签	支持为任务添加标签, 用于按照标签进行授权或出账。

4. 确认预测任务配置信息无误后, 单击**保存**进行任务创建。

任务启动过程中将为您调度算力资源, 需要等待一段时间, 启动成功后任务自动进入**运行中**状态, 为您进行批量数据的预测处理, 处理完成后任务状态将变为**已完成**。

查看预测结果

批量预测任务为**已完成**状态时, 可查看批量预测任务的运行结果。

1. 在批量预测任务页面点击任务名称, 进入任务基本信息页面查看输出数据路径;

2. 前往 [对象存储 COS 控制台](#)，在预测任务已关联的输出数据路径下，查看预测任务的结果文件，可在 COS 控制台进行预览或下载操作。



批量预测任务发布指引

最近更新时间：2024-08-22 15:41:27

批量预测模块支持用户在使用 TI-ONE 模型仓库管理的模型创建预测任务，或者使用自定义镜像创建预测任务。

模型包目录结构

```
cos://bucket/train_models/mv-2302330437837096
├── model
│   └── model.pth
├── batch
│   └── model_service.py
└── requirements.txt
```

- model: 存放模型的文件目录，该目录可以客户自己指定；
- model/model.pth: 原始模型文件，名称可以客户自己指定，但需要确保 model_service.py 可以正确加载；
- batch: 存储模型推理代码目录，该目录是硬编码目录，批量预测默认从此目录读取所需要的代码文件
- batch/model_service.py: 模型推理代码文件存储路径。当用户通过创建批量预测任务页面上上传模型推理文件时，文件被默认存放在此处。此文件是进行推理时的入口文件。
- requirements.txt: 第三方依赖包。如推理文件有第三方依赖可以放在 requirements.txt 文件内，如没有可直接忽略。如果存在，则任务执行前会默认执行 pip3 install -r requirements.txt 完成依赖项的安装；
- 其他文件或目录: 批量预测模块不再依赖，但其他模块，例如在线服务、任务式建模、模型仓库等模块，可能对其存在依赖，可参考对应模块文档进行详细了解。

平台路径规范

1. /opt/ml/input/data: 数据输入路径，平台默认会将控制台选择的“输入数据”挂载到此路径下，并递归读取所有子目录下的文件进行批量输入，输入值为文件路径。
2. /opt/ml/model: 模型包路径，平台默认会将模型包下载到此路径。
3. /opt/ml/output: 数据输出路径，平台批量预测脚本会将推理结果写入 /opt/ml/output/result.txt 文件，任务完成后，平台会将 result.txt 上传到控制台选择的“输出路径”下。
4. /opt/ml/code: 对于 /opt/ml/model/batch/model_service.py 文件，会软链到此目录下作为模型推理启动入口文件。若想自定义支持多个模型推理文件，可手动上传所有的推理代码到模型存储 cos 的对应路径下，但是需要保证推理启动入口的文件名为 model_service.py。

启动命令

若创建批量预测任务未指定启动命令时，则默认启动命令为 `python3 /opt/ml/code/model_service.py`。

自定义镜像规范

1. 自定义镜像限制及建议
 - 1.1 代码包放置路径建议为 /opt/ml/code，平台启动命令默认为 `python3 /opt/ml/code/model_service.py`；如果代码包为其他路径，用户可自定义启动命令；
 - 1.2 数据挂载目录: /opt/ml/input/data；
 - 1.3 结果输出目录: /opt/ml/output。
2. 自定义运行环境使用步骤
 - 2.1 本地配置 docker 环境，并开通 腾讯云容器镜像服务 (Tencent Container Registry, TCR)；
 - 2.2 编译自定义推理镜像，推送到 TCR 的镜像仓库；
 - 2.3 在创建批量预测任务时，服务实例模型来源选择从镜像仓库选择镜像，进入镜像仓库列表，选择上一步推送的自定义镜像环境；
 - 2.4 配置参数，启动批量预测任务。

资源组管理

资源组简介

最近更新时间：2025-05-14 17:03:53

总览

TI-ONE 平台是以资源组的形式来管理用户已购买的 CVM 机器节点，并进行资源调度。其机器来源分为“从 TI-ONE 平台购买”和“从 CVM 机器中选择”，二者区别可参见 [计费概述](#)，前者已不再支持新增，仅存量资源组可继续使用，本文均为“从 CVM 机器中选择”的资源组的操作说明。

您可将同地域下的 CVM 添加至同一资源组进行管理，后续使用该资源组的节点资源去启动训练、推理任务。

新建资源组

1. 创建 [注册腾讯云账号](#)，并完成 [实名认证](#)。
2. 腾讯云账号登录 [TI-ONE 控制台](#)，进入资源组管理页面。
3. 在页面最上方选择业务所在的地域。
4. 单击**新建资源组**，拉起资源组的新建弹窗。

资源组ID 名称	可用节点 总节点	可用 总资源	机器来源	专属资源组	标签	操作
6409091v	5/5	CPU 64.2 70.2C MEM 277.82 290.82G GPU 3.1 5 显存 74.4 136G	从CVM机器中选择	-		编辑 查看任务列表
7n2d...98	0/0	CPU 0 0C MEM 0 0G	从CVM机器中选择	-		编辑 查看任务列表
ule	0/0	CPU 0 0C MEM 0 0G	从CVM机器中选择	-		编辑 查看任务列表

5. 填写资源组名称，勾选风险提示，单击**确认完成创建**。

新建资源组 ✕

新建资源组后，可以选择您在CVM购买的机器加入该资源组

资源组名称

请输入不超过60个字符，仅支持中英文、数字、下划线"_"、短横"-", 只能以中英文、数字开头

地域 **南京**

标签 ①

标签键

标签值

✕

+ 添加 ② 键值粘贴板

训练调度策略 ①

```

1 {
2   "Version": "1.0",
3   "ResourceRule": {
4     "DefaultPriority": 0,
5     "DefaultQueue": 0
6   },
7   "TaskRules": []
8 }
```

资源按GPU比例预设 ①

⚠ 非常重要！后续操作需要重装您的机器，清空所挂载数据盘的数据，并更改相关的设置。请务必仔细阅读并同意《腾讯云TI-ONE从CVM中选择机器的风险提示》

确定
取消

查看任务列表

单击资源组列表页查看任务列表，可以查看该资源组下占用资源的任务和排队中的任务。

资源组管理 上海

① 资源组不再区分训练和推理，您可以创建训练任务和推理服务时共用一个资源组。

② 存量资源组如果存在重名，已为您增加“-原训练”、“-原推理”的后缀用于区分，您可再次修改资源组名称。

新建资源组 ①

资源组ID 名称	可用节点 总节点	可用 总资源	机器来源	裸金属资源组	标签	操作
8-22-1	5/5	CPU 64.2 70.2C MEM 277.82 290.82G GPU 3.1 5 显存 74.4 136G	从CVM机器中选择	-	✓	编辑 删除 查看任务列表

进入已占用资源任务Tab页，可查看该资源组下所有正在占用资源的任务，同时您可以在该页面切换资源组，选择具体某一个节点进行查看。

← 查看任务列表 腾讯云TI平台产品文档

已占用资源任务 **排队中任务**

选择资源组
选择节点

多个关键字用竖线 "|" 分隔，多个过滤标签用回车键分隔

类型	名称 ID	占用资源	运行时长
任务式建模	willtest-p1 train-1255056715404377600	2C4G *1	2分10秒
任务式建模	willtest-p0 train-1255053629839474176	4C8G *1	8分19秒

没有更多数据

进入排队中任务Tab页，可查看该资源组下正在排队的训练任务。请注意，当资源组的排队策略为严格先进先出的时候，该序号即为任务的调度顺序；当排队策略为非严格先进先出的时候，该序号不代表实际调度顺序，详细规则请见 [训练任务调度规则说明](#)。

查看任务列表

腾讯云TI平台产品文档

已占用资源任务 **排队中任务**

当前资源组的调度策略为优先级范围的遍历策略，从列任务序号不代表实际调度顺序，详细规则请见文档

选择资源组 willtest

序号	类型	名称 ID	申请资源	开始排队时间	等待时长
0	任务式建模	willtest-t4-p0 train-1255055530152787456	算力规格: 4C8G A10*0.6 节点数量: 2个	2025-02-14 15:31:25	6分32秒

没有更多数据

增加节点

1. 您可参考 [实例创建指引](#)，前往 [CVM 控制台](#) 购买好实例。

- TIONE 支持地域：北京、上海、广州，请您实例买在 TIONE 支持的地域。
- TIONE 支持的机型和对应软件费，您可参考 [计费概述](#)。

注意：

【特别提醒】 在购买机器后，请您将对应机器所属的 VPC 网段，添加至安全组并配置入站及出站规则，以确保训练/推理过程中可以正常通信。具体操作指引如下：

- 购买 CVM 机器，并在实例详情页面获取“所属网络”的信息；
- 在“安全组详情-安全组规则”tab 页面中，分别添加一条入站及出站规则。“来源”配置为上一步获取的“所属网络”的 VPC 网段，“协议端口”配置为“ALL”。



2. 单击某一个“从 CVM 机器中选择”的资源组名称进入后，单击增加节点，进入购买页。

说明：

您从 CVM 控制台购买的机器资源都将以节点的形式，被添加至资源组进行统一管理。

如果资源组内某一节点被释放，该资源组仍旧存在，并可继续添加其他节点。从 A 资源组释放的节点，也可以重新添加至 B 资源组。同时，我们也提供了节点迁移的功能，支持将节点从 A 资源组移动至 B 资源组。



3. 在购买页，根据提示配置以下信息。



字段	描述
计费模式	包年包月和按量计费
地域	默认展示该资源组所在地域
增加节点	该地域下，选择需要在 TIONE 使用的 CVM 机器，可多选，支持全部的规格和软件费可参考 计费概述 。 <ul style="list-style-type: none"> 计费模式选择为“包年包月”，只能选择包年包月的 CVM，TIONE 软件费按月预付费 计费模式选择为“按量计费”，只能选择按量计费的 CVM，TIONE 软件费按小时后付费
购买时长	<ul style="list-style-type: none"> 计费模式选择为“包年包月”时，才会有该选项 为 CVM 机器在 TIONE 的使用时长
自动续费	<ul style="list-style-type: none"> 计费模式选择为“包年包月”时，才会有该选项 默认不勾选

注意：

1. CVM 机器被管理进 TI-ONE 前的操作提示：

- 数据
 - 您账户下的 CVM 机器被在 TI-ONE 上选择前，请务必提前备份数据。
 - 被选择使用的 CVM 机器，当无数据盘时，系统盘大小需大于等于100GB；当有数据盘时，数据盘大小需大于等于100GB，因此建议您先挂载好数据盘，再将机器管理进 TIONE 平台。
- 网络
 - 请您务必保证被选择使用的 CVM 机器，和自身存储（如 CFS、GooseFSx 等）网络是互通的。
 - VPC 不同的 CVM 机器，支持在 TIONE 里管理使用，但您需要保证 VPC 网段互相不冲突，否则无法选择部署。

2. CVM 机器被管理进 TI-ONE 后的影响提示：

- 数据
 - 被选择的 CVM 机器，会被 TI-ONE 重装系统，该机器上及所挂载数据盘中的原有数据会被清除。
 - 被选择的 CVM 机器，您不再有登录权限，也无法再使用 [自动化助手](#)。如果您登录操作该机器，您需自行承担导致 TI-ONE 服务异常、终止等全部后果。
- 网络
 - TIONE 会对选择的 CVM 机器的安全组，放开9.0.0.0/8以及100.127.0.0/16网段。同时，请在安全组中添加已购买机器的出站/入站规则：来源为机器所属网络 VPC 网段，协议端口为“ALL”。在机器在 TIONE 使用期间不要轻易删除这两个网段的规则，否则会导致机器使用异常。
 - 安全组默认最多支持100条规则，请预留2个规则供 TI 平台使用。

全部提示可参见 [附录-风险提示](#)。

- 核对购买信息后提交订单，选择支付方式并完成支付。
- 完成购买后，进入控制台查收节点。

节点有如下9个状态：

节点状态	状态说明
部署中	CVM 资源在 TIONE 平台初始化中
部署失败	CVM 资源在 TIONE 平台部署失败
运行中	CVM 资源在 TIONE 平台部署成功后投入使用
释放中	CVM 资源在 TIONE 平台使用时间已到期自动释放，或被用户主动释放
已释放	CVM 资源在 TIONE 平台使用时间到期自动释放中，或被用户主动释放中
异常	CVM 资源在 TIONE 平台使用异常
已不使用	CVM 资源已不在 TIONE 平台使用
挂载中	CVM 资源将其挂载好的数据盘管理进 TIONE 平台里使用
维修中	CVM资源出现故障，自动维修中

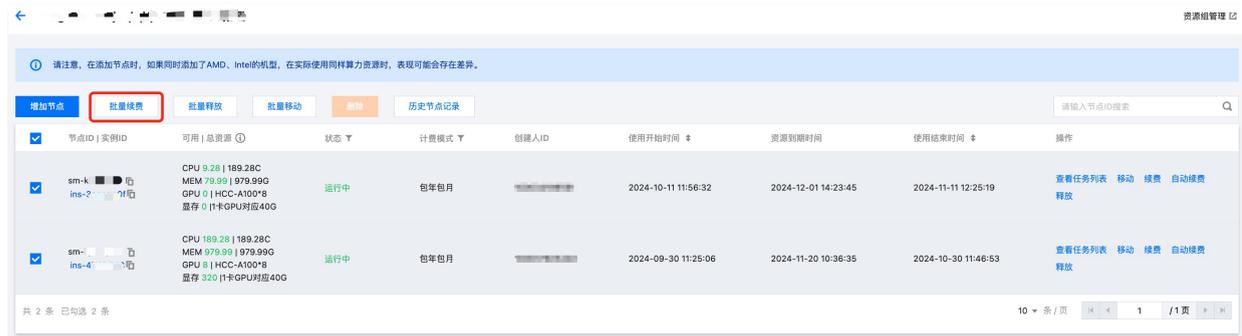
注意

- CVM 部分机型支持 **关机不计费**，当 CVM 机器关机时，对应的 TI-ONE 软件费也不计费且节点状态会转化为“异常”；当 CVM 机器重新开机后，节点预计几分钟后自动转为“运行中”，同时开始计费。
- CVM 机器关机后，有可能因资源不足重新开机失败，请您留意。

资源组操作

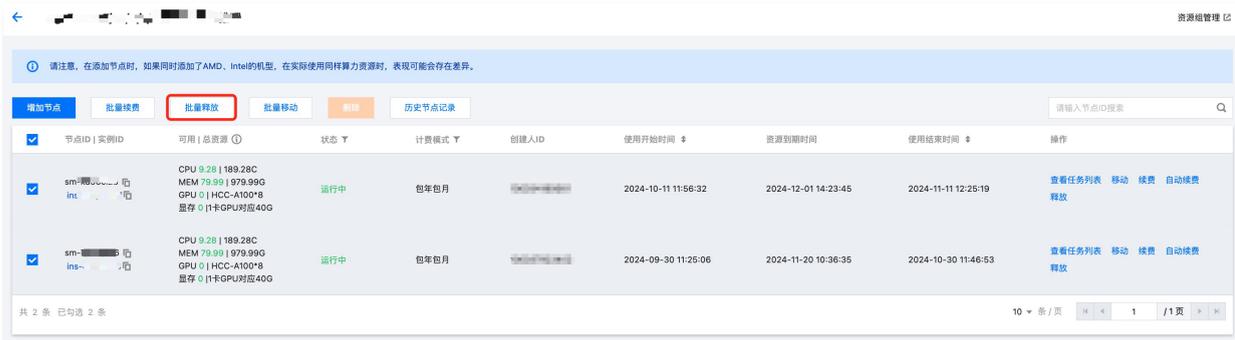
批量续费

在进入某一个机器来源为“从 CVM 机器中选择”的资源组后，单击**批量续费**可以对多个选中节点进行相同时间时长的软件费续费操作。



批量释放

在进入某一个机器来源为“从 CVM 机器中选择”的资源组后，单击**批量释放**可以对多个选中节点进行释放，释放后将自动退款软件费，同时 CVM 机器不再支持在 TIONE 平台使用。



删除

在进入某一个机器来源为“从 CVM 机器中选择”的资源组后，单击删除，可以删除整个资源组，删除后不可恢复。

注意：

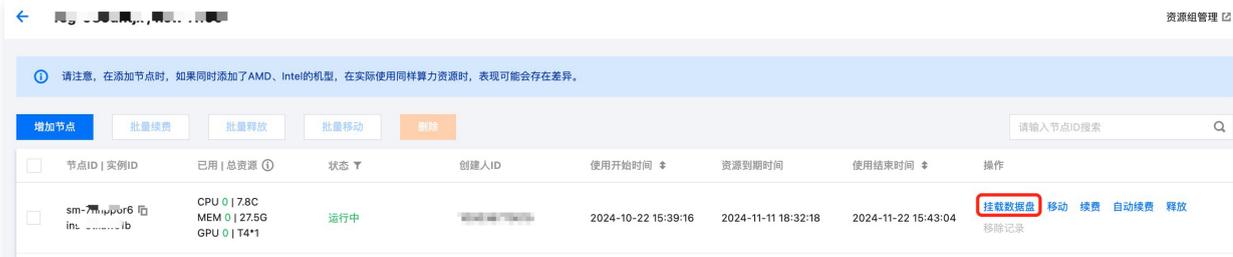
仅在资源组内没有使用节点时可进行删除，使用节点状态包括“运行中”、“部署中”、“异常”、“释放中”、“维修中”、“挂载中”。



节点操作

挂载数据盘

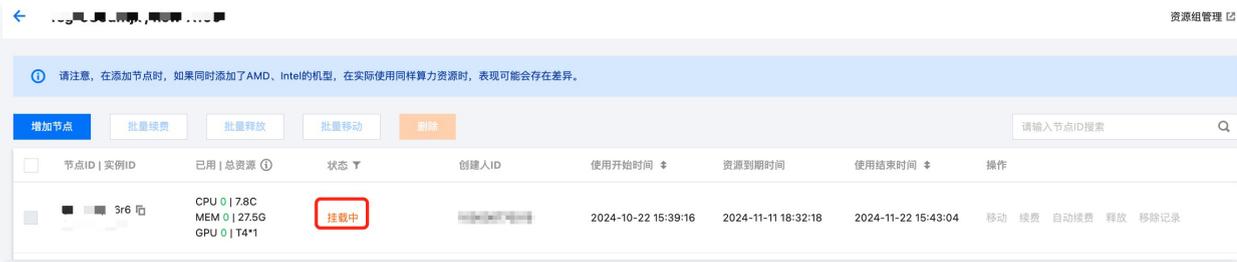
1. 在 TIONE 控制台里状态为“运行中”的节点，没有数据盘，首次去 CVM 侧对该节点 挂载新的数据盘，此时操作里会出现挂载数据盘按钮。



2. 单击挂载数据盘，出现二次确认弹窗，挂载数据盘会终止该节点上正在运行的任务/服务。



3. 单击确定后，节点状态转为“挂载中”，预计1-2分钟后挂载完成，操作里的挂载数据盘按钮随之消失。



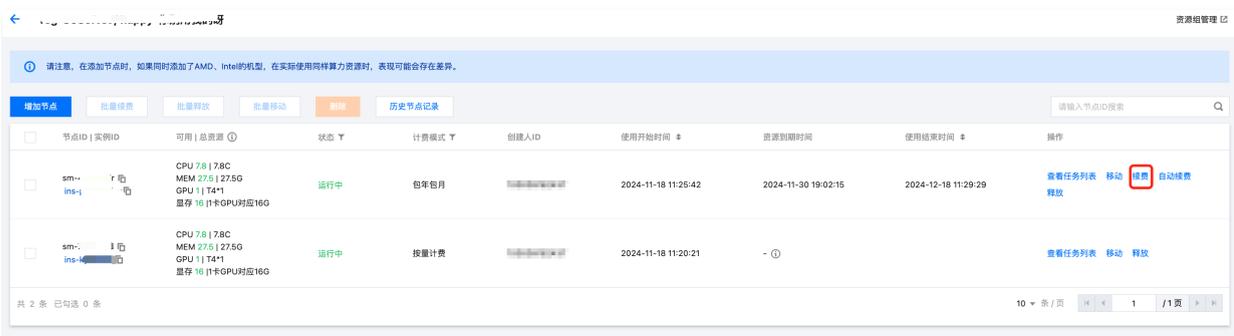
如果您机器已有数据盘，如遇硬盘空间不足，可 **扩容云硬盘**，您根据文档扩容后，TIONE平台可自动使用该硬盘。

续费

在进入某一个机器来源为“从 CVM 机器中选择”的资源组后，可对计费模式为“包年包月”的节点，单击**续费**并选择续费时长，单击**确认续费**将在账户余额扣费。

注意：

1. 此时续费的是软件费，即续费时长为 CVM 机器在 TIONE 的使用时间，非机器的资源费用。
2. 计费模式为“**按量计费**”的节点不支持续费。

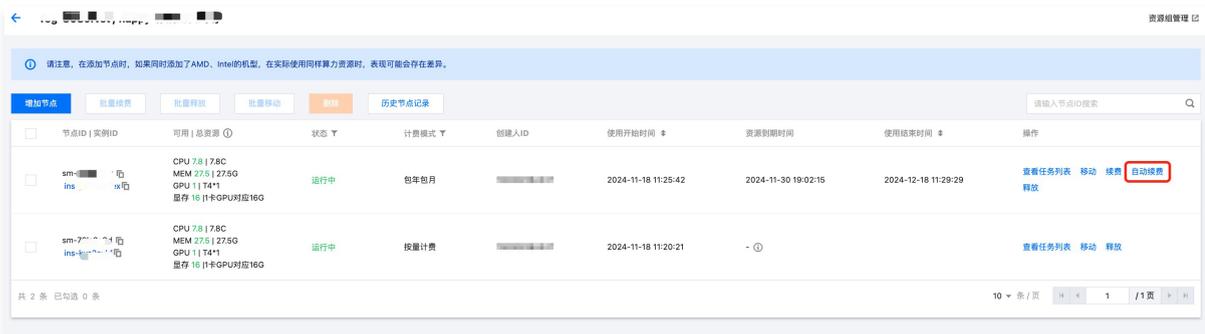


开启/取消自动续费

1. 在进入某一个机器来源为“从 CVM 机器中选择”的资源组后，可对计费模式为“包年包月”的节点，单击**自动续费**，节点到期将自动扣费并续费一个月的软件费。

注意：

1. 此时自动续费的是软件费，即续费时长为 CVM 机器在 TIONE 的使用时间，非机器的资源费用。
2. 计费模式为“**按量计费**”的节点不支持续费。



2. 单击节点列表中**取消自动续费**，节点到期后将停止自动扣费并会进行到期提示。



释放

在进入某一个机器来源为“从 CVM 机器中选择”的资源组后，单击**释放**可将“运行中”或“异常”状态的节点不再被 TIONE 使用，释放后将自动退款剩余软件费。

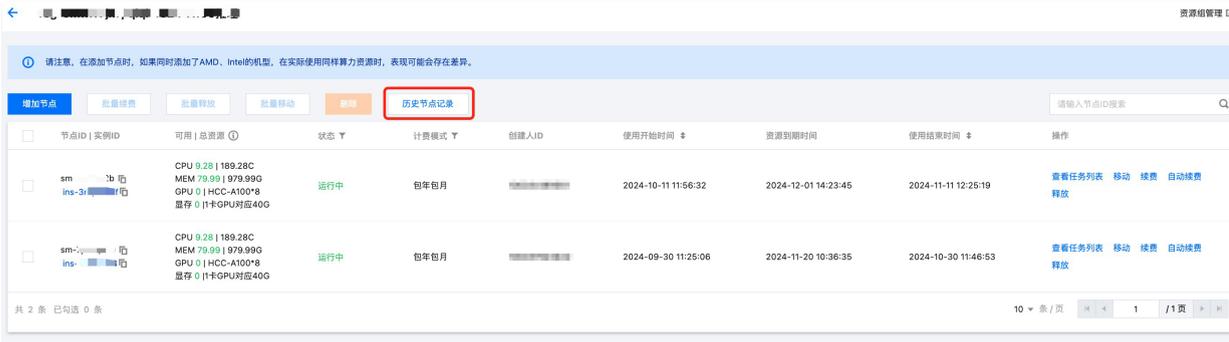
注意：

此时释放不是进行机器资源销毁，是指 CVM 机器不再在 TIONE 管理使用。



历史节点记录

在进入某一个资源组后，单击**历史节点记录**可查看“部署失败”和“已不使用”状态的节点列表。





附录

风险提示

1. CVM 机器被管理进 TI-ONE 前的操作提示：

- 数据
 - 您账户下的 CVM 机器被在 TI-ONE 上选择前，**务必提前备份数据**。
 - 请您保证被选择使用的 CVM 机器，当无数据盘时，系统盘大小需大于等于100GB；当有数据盘时，数据盘大小需大于等于100GB，否则使用 tiacc 镜像和 llm 镜像会拉取失败，无法选择该机器进行部署。建议您先挂载好数据盘，再将机器管理进 TIONE 平台。
 - 请您保证被选择使用的 CVM 机器，优先级最高的安全组规则（最多100条/进站方向，100条/出站方向），出站方向规则均剩余5条空闲预留给 TIONE 添加新规则，否则无法选择该机器进行部署。
- 网络
 - 请您务必保证被选择使用的 CVM 机器，和自身存储（如 CFS、GooseFSx 等）网络是互通的。
 - VPC 不同的 CVM 机器，支持在 TIONE 里管理使用，但您需要保证 VPC 网段互相不冲突，否则无法选择部署。
 - 如果您镜像里有需要访问公网的内容，请您将 CVM 机器，开启公网访问（机器配置弹性公网 IP 或对应子网有 NAT 公网网关）。
- 功能菜单支持范围
 - 在 TIONE 平台上选择 CVM 机器进行管理使用，支持的产品能力有：任务式建模、Notebook、在线服务、模型评测，**不支持自动学习、可视化建模、批量预测**。

2. CVM 机器被管理进 TI-ONE 后的影响提示：

- 数据
 - 被选择的 CVM 机器，会被 TI-ONE **重装系统**，该机器上及所挂载数据盘中的**原有数据会被清除**。
 - 被选择的 CVM 机器，您**不再有登录权限，也无法再使用 自动化助手**。如果您登录操作该机器，导致 TI-ONE 服务异常、终止等一切后果，由您自身承担。
- 网络
 - TIONE 会对选择的 CVM 机器的安全组，放开9.0.0.0/8以及100.127.0.0/16网段。同时，请在安全组中添加已购买机器的出站/进站规则：来源为机器所属网络 VPC 网段，协议端口为“ALL”。在机器在 TIONE 使用期间不要轻易删除这两个网段的规则，否则会导致机器使用异常。
 - 安全组默认最多支持100条规则，请预留2个规则供 TI 平台使用。
- 财务
 - 您账户下被选择在 TIONE 使用的 CVM 机器，续费、付款、合同、发票、退款、降配仍在您账号里操作。

调度策略说明

最近更新时间：2025-05-29 22:13:32

整体说明

TI-ONE 资源组在调度训练任务和在线服务时，支持“排队策略”及“优先级调度策略”。

- 排队策略：当资源不足时，默认排队策略是按照任务/服务提交时间的先后顺序先进先出。用户也可以在资源组配置遍历策略，优先调度队列中满足资源要求的任务/服务。详见 [排队策略配置说明](#)。
- 优先级调度策略：支持根据标签设置任务/服务的优先级（P0最高-P9最低），高优先级任务/服务会默认抢占低优先级任务，低优先级任务被抢占后会重新进入排队队列，详细使用说明请见 [优先级调度说明](#)。
 - 备注：通过标签设置任务优先级的功能仅支持“任务式建模”及“在线服务”模块。
 - 服务优先原则：高优的在线服务可以抢占低优任务式建模的资源，但高优任务式建模不会抢占低优在线服务的资源。且同时，高优的在线服务也不会抢占低优的在线服务。

说明：

- 针对 CPU 任务，默认会按照负载均衡的策略均分到资源组不同节点。
- 针对 GPU 任务，默认会按照最小化碎片的方式，优先调度到同一台节点。
- 不同卡类型的任务不会互相阻塞，例如T4卡的任务不会影响A100卡的任务。

排队策略配置说明

DefaultQueue 表示资源组内训练任务的默认排队策略，仅支持“任务式建模”及“在线服务”模块。

- 其中默认值为0，设置为0代表 **严格先进先出**：不管当前空闲多少资源，在等待队列中取到最早提交的任务，如果资源足够则调度执行，如果不够则等待；
- 设置为1代表 **资源尽量利用的先进先出**：根据当前空闲的资源情况，从队列中按照时间顺序找到第一个当前资源满足的任务，调度执行。

```
{
  "Version": "1.0",
  "ResourceRule": {
    "DefaultPriority": 0,
    "DefaultQueue": 0,
    "Preempted": 1
  },
  "TaskRules": []
}
```

参数说明：

- **DefaultPriority**：默认优先级，优先级为0-9，P0最高，默认值为0。
- **DefaultQueue**：默认排队策略，可选值0、1、2，默认值为0。
 - 0代表严格先进先出：不管当前空闲多少资源，在等待队列中取到最早提交的任务，如果资源足够则调度执行，如果不够则等待；
 - 1表示在优先级范围内的遍历策略，也就是说同一优先级的任务可以不严格按照任务提交时间按顺序出队，可以看资源空闲情况插队；
 - 2表示全部队列的遍历策略，也就是说整个排队队列都按照遍历策略，只要资源满足，就出队。
- **Preempted**：默认优先级抢占策略，可选值0和1，默认值为1。
 - 0表示队列中的高优任务不会抢占已经在运行中的低优任务；
 - 1表示队列中的高优任务会抢占已经在运行中的低优任务。

优先级调度说明

策略语法示例

在资源组维度，支持配置调度策略的描述文件。其中，DefaultPriority 表示默认优先级，TaskRules 中可以按照以下示例通过任务标签设置优先级，其中 ValueType 默认为 Tag，后续可能会根据实际需求不断补充ValueType。

下面是一个示例语句，实现的场景为：该资源组下的训练任务默认优先级为P0，如果任务的标签符合**任务类型-语音任务**，则该任务的优先级为P1（注意：通过标签设置任务优先级的功能仅支持任务式建模及在线服务）。

- 低优任务会被高优任务或服务抢占，但低优的服务只会被高优服务抢占。被抢占后的任务/服务将进入排队队列，继续等待。
- 服务优先原则：高优的在线服务可以抢占低优任务式建模的资源，但高优任务式建模不会抢占低优在线服务的资源。且同时，高优的在线服务也不会抢占低优的在线服务。

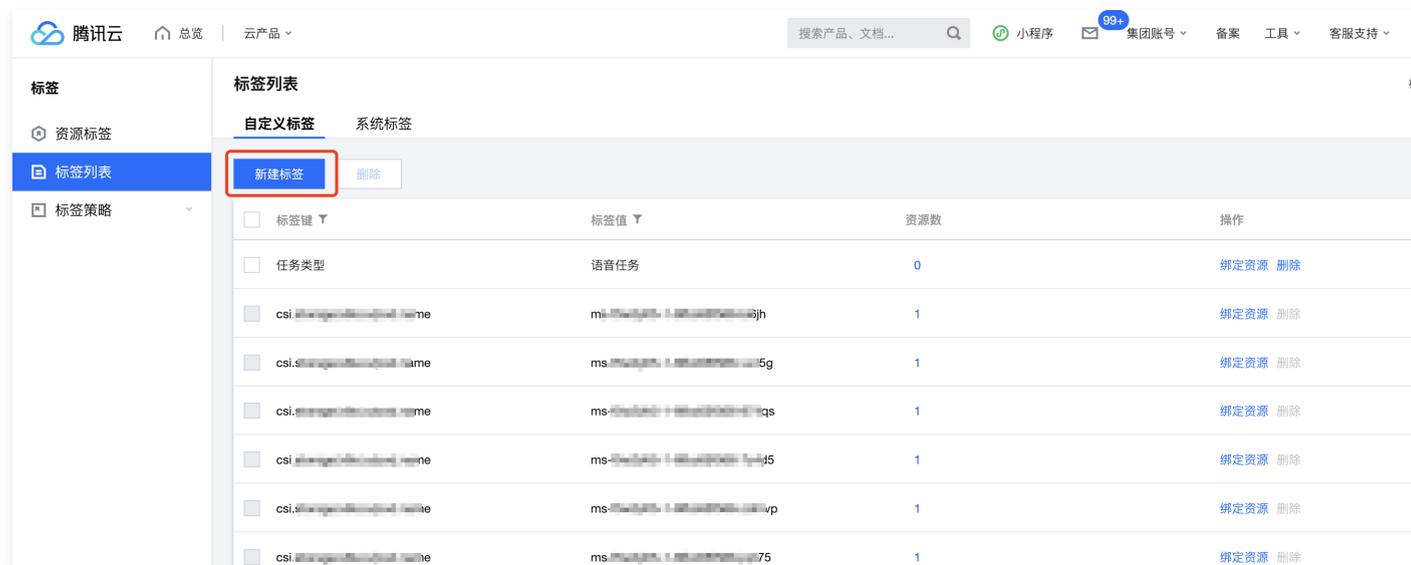
注意：

下述策略语句中的#注释部分不符合 json 语法，仅做文档展示，粘贴到平台需要清除。

```
{
  "Version": "1.0", // json格式的版本
  "ResourceRule": {
    // 资源组策略
    "DefaultPriority": 0, // 默认优先级，优先级0-9
    "DefaultQueue": 1,
    "Preempted": 1
  },
  "TaskRules": [
    // 任务调度策略，是规则匹配列表
    {
      // 每个规则是属性和调度的匹配关系
      "AttrRules": [
        {
          "ValueType": "Tag",
          "Key": "任务类型",
          "Value": "语音任务"
        }
      ],
      "ScheduleRule": {
        "Priority": 1 // 设置优先级，优先级0-9
      }
    }
  ]
}
```

具体操作流程（以任务式建模为例）

1. 登录 **标签控制台**，进入**标签列表**。



单击**新建标签**，输入标签键**任务类型**，标签值**语音任务**。

新建标签 ✕

① • 输入新的标签键和标签值创建全新标签，选择已有标签键可为该键新增标签值

• 一个标签键最多具有 1000 个标签值，单次创建最多可以输入 10 个标签值

标签键

标签值

 ✕

[删除](#)

[添加标签键](#)

确定
取消

2. 进入 TI-ONE 资源组管理页面，选择具体的资源组，单击编辑，在调度策略一栏输入上述的策略描述语句。

资源组管理 资源组管理 [🔗](#)

① 1. 资源组不再区分训练和推理，您可以创建训练任务和推理服务时共用一个资源组。

2. 存量资源组如果存在重名，已为您增加“-原训练”、“-原推理”的后缀用于区分，您可再次修改资源组名称。

新建资源组 ①

🔍

资源组ID 名称	可用节点 总节点	已用 总资源	机器来源	裸金属资源组	标签	操作
rsg-n29qxm7k b-test-cvm-type	0/2	CPU 0 0C MEM 0 0G	从CVM机器中选择	-	🔗	编辑 删除
rsg-lpkfb6nr -local-rsg-1	0/0	CPU 0 0C MEM 0 0G	从CVM机器中选择	-	🔗	编辑 删除
rsg-lk6kpbz4 -local-rsg-0	0/3	CPU 0 0C MEM 0 0G	从CVM机器中选择	-	🔗	编辑 删除

编辑 ✕

资源组名称 • ✔

请输入不超过60个字符，仅支持中英文、数字、下划线“_”、短横线“-”，只能以中英文、数字开头

地域 • 广州

标签 ①

标签键

:

标签值

✕

[+ 添加](#) [📄 键值粘贴板](#)

训练调度策略 ①

```

1  {
2    "Version": "1.0",
3    "ResourceRule": {
4      "DefaultPriority": 0,
5      "DefaultQueue": 1
6    },
7    "TaskRules": [
8      {
9        "AttrRules": [
10       {
11         "ValueType": "Tag",
12         "Key": "任务类型",
13         "Value": "语音任务"
14       }
15     ],
16     "ScheduleRule": {
17       "Priority": 1
18     }
19   }
                
```

确定
取消

3. 新建训练任务的时候，如果需要当前这个任务的优先级为低的话，需要在新建任务的时候给这个任务打上标签任务类型 > 语音任务，那么这个任务在调度的时候就是低优先级任务，会被抢占，自动进入排队队列。

腾讯云 TI 平台

- 自动学习
- 数据中心
- 训练工坊
 - 任务式建模
 - Notebook
 - 可视化建模
- 模型管理
- 模型服务
- 资源组管理
- 大模型精调

训练模式 • DDP

计费模式 • 按量计费 包年包月

资源组 • 请选择资源组

资源申请 •

CPU • - 1.0 + 核

内存 • - 1.00 + G

卡型号 请选择卡型号

GPU • - 0.0 + 卡
若需使用GPU，卡数可填写0.1-1或1的整数倍

节点数 • - 1 + 个

标签 ① 任务类型 语音任务 x
+ 添加 键值粘贴板

描述 请输入不超过500个字
0 / 500

下一步 取消

GPU 虚拟化

最近更新時間: 2025-04-27 16:17:42

概述

腾讯云 TI 平台提供 GPU 虚拟化功能, 可将同一张 GPU 卡的算力分配给不同训练任务和推理服务使用, 提升资源分配灵活性和使用效率。

支持的 GPU 型号

平台的 GPU 虚拟化功能已支持的 GPU 型号如下:

T4、V100、A100、A10、L20、A800、HCCA100、HCCPNV6、PNV6。

功能使用说明

前置条件

您需要在 [资源组管理](#) 页面提前创建资源组, 并将您的 CVM 以节点形式添加至资源组。

资源组管理

请注意, 在添加节点时, 如果同时添加了AMD、Intel的机型, 在实际使用同样算力资源时, 表现可能会存在差异。

增加节点 批量续费 批量释放 批量移动 删除 历史节点记录 请输入节点ID搜索

节点ID 实例ID ...	可用 总资源	状态	计费模式	创建人ID	使用开始时间	资源到期时间	使用结束时间	操作
sm- ins- -	CPU 3.2 7.8C MEM 11 27.5G GPU 0 T4*1 显存 0 1卡GPU对应 16G	运行中	包年包月	10000002	2024-10-29 10:38:09	2025-03-24 17:26:33	2025-03-28 17:26:33	查看任务列表 移动 续费 自动续费 释放

共 1 条 已勾选 0 条 10 条 / 页 1 / 1 页

使用方式

在 [新建开发机实例](#)、[新建任务式建模训练任务](#) 或 [新建在线服务](#) 时, 当选择您的资源组后, 如果其中包括已支持虚拟化的 GPU 卡型号, 则可以在配置资源时, 选择 GPU 卡数为0.1至1之间的数值。

← 新建开发机

名称 0/256

请输入开发机名称

请输入不超过256个字符，仅支持中英文、数字、下划线"_"、短横"-", 只能以中英文、数字开头

地域 广州

镜像 请选择

机器来源 从CVM机器中选择 | 从T1ONE平台购买

TI-ONE 仅收取软件费用，算力将使用您已购买的机器。[产品定价](#)

资源组 [资源组名称] | 刷新

T4可用卡数: 总计1/整机1/非整机0 [查看详情](#)

资源申请

卡型号: T4

GPU: - 0.1 + 卡

若需使用GPU, 卡数可填写0.1-1或1的整数倍。该机型支持填写GPU整数卡或小数卡

CPU: - 1.0 + 核

内存: - 1.00 + G

注意事项

由于市面上的 GPU 卡型号在不断增多，相关驱动也在持续更新，因此平台的 GPU 虚拟化功能对于 GPU 型号的支持是逐步扩展的。平台新支持虚拟化的 GPU 可能无法在存量资源组中使用虚拟化功能，请您在使用时关注平台的相关提示。

- 如果在创建资源组时添加了平台尚未支持虚拟化的 GPU 卡型号，则在平台后续支持该卡型号虚拟化后，仍无法使用该资源组内上述 GPU 的虚拟化功能。
- 在上述情况下，您也无法上述节点直接移动至其他资源组；但可以将节点从资源组中移除，并重新添加至新创建的资源组，从而可以使用虚拟化功能。

Tikit

Tikit 简介

最近更新时间：2024-08-22 10:15:50

背景

为了让算法同学在 Notebook 或者本地IDE中能够更好地与平台进行交互，平台提供了Python开发套件 Tikit 供算法同学进行使用。本模块将介绍该套件的使用方式。

安装与初始化

最近更新时间：2024-12-24 16:50:22

操作步骤

环境依赖

目前可以支持 Python 3.4 及以上的版本。

安装 tikit

在 TI-ONE 平台中，notebook 和训练任务的容器里面已经内置了 tikit，不用再安装。
非公有云 TI-ONE 平台的环境，安装方法如下：

- 安装依赖：

```
#centos:

sudo yum -y install cyrus-sasl cyrus-sasl-devel cyrus-sasl-lib

#ubuntu:

sudo apt-get update
sudo apt-get install -y libsasl2-dev
```

- 方式一：使用 pip 安装（推荐）

```
pip3 install -U tikit
```

- 方式二：离线安装。在 <https://pypi.org/project/tikit/> 上下载安装包，使用 whl 文件安装，或者使用源码安装：

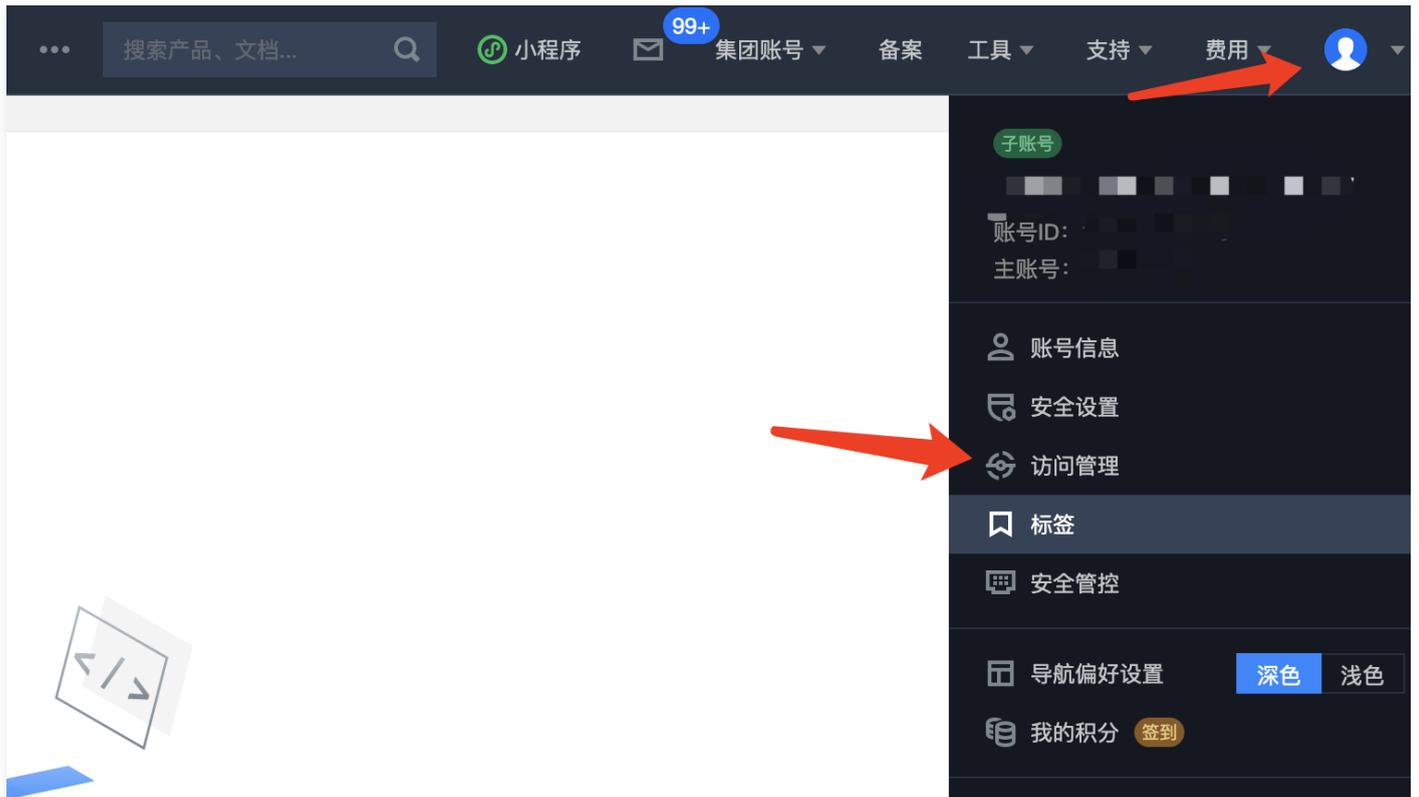
```
pip3 install tikit-1.0.0-py3-none-any.whl

# 或者，解压源码后
python3 setup.py install
```

开始使用

1. 准备好 secret_id, secret_key 信息。

登录腾讯云，在页面上获取，如下：



2. 初始化，调用函数。

```

from tikit.client import Client

# 初始化client。在公有云钛平台的环境（包括notebook和训练任务）中可以不填地区，环境变量中已经有地区信息。
client = Client("your_secret_id", "your_secret_key", "<region>")
    
```



查看资源组资源

最近更新时间：2024-08-22 10:22:31

```
describe_resource_groups(self, offset=0, limit=20, search_word="", tag_filters=None):
```

获取资源组列表

:param offset: 偏移量，默认为0；分页查询起始位置，如：Limit为100，第一页Offset为0，第二页Offset为100...即

每页左边为开区间

:type offset: int

:param limit: 返回数量

:type limit: int

:param search_word: 支持模糊查找资源组id和资源组名

:type search_word: str

:param tag_filters: 标签过滤

:type tag_filters: list of tikit.tencentcloud.tione.v20211111.models.TagFilter

:return:

:rtype:

```
:class:`tikit.tencentcloud.tione.v20211111.models.DescribeBillingResourceGroupsResponse`
```

使用数据集

最近更新时间：2024-08-22 10:15:59

```
create_image_dataset(self, dataset_name, storage_data_path, storage_label_path, dataset_tags=None,
with_annotation=False, annotation_type=None, annotation_format=None)
```

创建图片数据集

```
:param dataset_name: 数据集名称
:type dataset_name: str
:param storage_data_path: 数据源cos路径。格式: <bucket>/<cos path>/
:type storage_data_path: str
:param storage_label_path: 数据集标签cos存储路径。格式: <bucket>/<cos path>/
:type storage_label_path: str
:param dataset_tags: 数据集标签
:type dataset_tags: list of tencentcloud.tione.v20211111.models.Tag
:param with_annotation: 是否已经标注
:type with_annotation: bool
:param annotation_type: 标注类型。可选值如下:
    ANNOTATION_TYPE_CLASSIFICATION: 图片分类
    ANNOTATION_TYPE_DETECTION: 目标检测
    ANNOTATION_TYPE_SEGMENTATION: 图片分割
    ANNOTATION_TYPE_TRACKING: 目标跟踪
    ANNOTATION_TYPE_OCR: OCR识别
:type annotation_type: str
:param annotation_format: 标注格式。可选值如下:
    ANNOTATION_FORMAT_TI: TI平台格式
    ANNOTATION_FORMAT_PASCAL: Pascal Voc
    ANNOTATION_FORMAT_COCO: COCO
    ANNOTATION_FORMAT_FILE: 文件目录结构
:type annotation_format: str
:rtype: :class:`tencentcloud.tione.v20211111.models.CreateDatasetResponse`
```

```
create_table_dataset(self, dataset_name, storage_data_path, storage_label_path, dataset_tags=None,
is_schema_existed=None, schema_info_dict=None)
```

创建表格数据集

```
:param dataset_name: 数据集名称
:type dataset_name: str
:param storage_data_path: 数据源cos路径。格式: <bucket>/<cos path>/
:type storage_data_path: str
:param storage_label_path: 数据集标签cos存储路径。格式: <bucket>/<cos path>/
:type storage_label_path: str
:param dataset_tags: 数据集标签
:type dataset_tags: list of tencentcloud.tione.v20211111.models.Tag
:param is_schema_existed: 数据是否存在表头。
    若数据文件包含表头，则请您严格按照表格列名配置Schema信息，否则校验不通过会造成导入数据集失败；
    若数据文件不包含表头，则平台会根据您定义的Schema信息依次为您解析表格数据
:type is_schema_existed: bool
:param schema_info_dict: 表头信息。格式: 字段名称 -> 数据类型。字段的数据类型包括:
    TYPE_INT: 整型
    TYPE_STRING: 字符串
    TYPE_BOOL: 布尔型
    TYPE_FLOAT: 浮点型
:type dict
:rtype: :class:`tencentcloud.tione.v20211111.models.CreateDatasetResponse`
```

```
create_text_dataset(self, dataset_name, storage_data_path, storage_label_path, dataset_tags=None)
```

创建文本数据集

```
:param dataset_name: 数据集名称
:type dataset_name: str
:param storage_data_path: 数据源cos路径。格式: <bucket>/<cos path>/
:type storage_data_path: str
:param storage_label_path: 数据集标签cos存储路径。格式: <bucket>/<cos path>/
:type storage_label_path: str
:param dataset_tags: 数据集标签
:type dataset_tags: list of tencentcloud.tione.v20211111.models.Tag
:rtype: :class:`tencentcloud.tione.v20211111.models.CreateDatasetResponse`
```

```
create_other_dataset(self, dataset_name, storage_data_path, storage_label_path, dataset_tags=None)
```

创建其他类型的数据集

```
:param dataset_name: 数据集名称
:type dataset_name: str
:param storage_data_path: 数据源cos路径。格式: <bucket>/<cos path>/
:type storage_data_path: str
:param storage_label_path: 数据集标签cos存储路径。格式: <bucket>/<cos path>/
:type storage_label_path: str
:param dataset_tags: 数据集标签
:type dataset_tags: list of tencentcloud.tione.v20211111.models.Tag
:rtype: :class:`tencentcloud.tione.v20211111.models.CreateDatasetResponse`
```

```
describe_datasets(self, dataset_ids=None, filters=None, tag_filters=None, order=None, order_field=None,
offset=None, limit=None)
```

查看数据集列表

```
:param dataset_ids: 数据集id列表
:type dataset_ids: list of str
:param filters: 字段过滤条件
:type filters: list of Filter
:param tag_filters: 标签过滤条件
:type tag_filters: list of TagFilter
:param order: 排序值 Asc Desc
:type order: str
:param order_field: 排序字段
:type order_field: str
:param offset: 偏移值
:type offset: int
:param limit: 返回数据个数
:type limit: int
:rtype: :class:`tencentcloud.tione.v20211111.models.DescribeDatasetsResponse`
```

```
delete_dataset(self, dataset_id, delete_label_enable=False)
```

删除某个数据集

```
:param dataset_id: 数据集id
:type dataset_id: str
:param delete_label_enable: 是否删除cos标签文件
:type delete_label_enable: bool
:rtype: :class:`tencentcloud.tione.v20211111.models.DeleteDatasetResponse`
```

```
describe_dataset_detail_structured(self, dataset_id, offset=None, limit=None)
```

查看结构化数据集的内容

```
:param dataset_id: 数据集ID
```

```
:type dataset_id: str
:param offset: 偏移值
:type offset: int
:param limit: 返回数据条数
:type limit: int
:rtype: :class:`tencentcloud.tione.v20211111.models.DescribeDatasetDetailStructuredResponse`
```

```
describe_dataset_detail_unstructured(self, dataset_id, offset=None, limit=None, label_list=[],
annotation_status='STATUS_ALL')
```

查看非结构化数据集的内容

```
:param dataset_id: 数据集ID
:type dataset_id: str
:param offset: 偏移值
:type offset: int
:param limit: 返回数据条数
:type limit: int
:param label_list: 标签过滤参数
:type label_list: list of str
:param annotation_status: 标注状态过滤参数。
    STATUS_ANNOTATED: 已标注
    STATUS_NON_ANNOTATED: 未标注
    STATUS_ALL: 全部
:type annotation_status: str
:rtype: :class:`tencentcloud.tione.v20211111.models.DescribeDatasetDetailUnstructuredResponse`
```

数据上传与下载

最近更新时间：2024-08-22 14:28:14

1. 直接调用的函数

1.1. 接口列表

```
download_from_hdfs(hdfs_url, hdfs_path, local_path)
```

从hdfs上下载文件到本地

```
:param hdfs_url: webhdfs的地址, 如: http://10.0.3.16:4008
:type hdfs_url: str
:param hdfs_path: hdfs上的路径
:type hdfs_path: str
:param local_path: 本地路径
:type local_path: str
:return: 本地的结果路径
:rtype: str
```

```
upload_to_hdfs(local_path, hdfs_url, hdfs_path)
```

上传本地目录到hdfs上

```
:param local_path: 本地路径
:type local_path: str
:param hdfs_url: webhdfs的地址, 如: http://10.0.3.16:4008
:type hdfs_url: str
:param hdfs_path: hdfs上的路径
:type hdfs_path: str
:return: hdfs的结果路径
:rtype: str
```

```
upload_to_hive_by_hdfs(local_path, hdfs_url, hive_server, table_name, database='default',
auth='CUSTOM', username=None, password=None, overwrite=False, partition='')
```

把本地文件的数据导入到hive表中, hdfs作为中间存储。

过程: 先把本地文件上传到hdfs上, 然后再从hdfs文件导入到hive中。

```
:param local_path: 本地文件或者文件夹。文件夹中不能包含子文件夹。
:type local_path: str
:param hdfs_url: webhdfs的url, 如: http://10.0.3.16:4008
:type hdfs_url: str
:param hive_server: HiveServer2的地址
:type hive_server: str
:param table_name: Hive表的名称
:type table_name: str
:param database: 数据库名称
:type database: str
:param auth: 认证的方式
:type auth: str
:param username: 数据库认证的用户名
:type username: str
:param password: 数据库认证的密码
:type password: str
:param overwrite: 是否删掉原来的数据
:type overwrite: bool
:param partition: 分区的选择
:type partition: str
:return:
```

```
:rtype:
```

```
export_from_hive_by_hdfs(local_path, hdfs_url, hive_server, table_name='', sql='', database='default',
auth='CUSTOM', username=None, password=None, row_format="row format delimited fields terminated by
',")
```

导出hive表到本地，hdfs作为中间存储。对于大文件，这种方式比直接从hive表写到本地的效率更高。
过程：先把hive导出到hdfs上，然后再从hdfs下载文件到本地。

```
:param local_path:      本地的目录
:type local_path:      str
:param hdfs_url:        webhdfs的url，如：http://10.0.3.16:4008
:type hdfs_url:        str
:param hive_server:     HiveServer2的地址
:type hive_server:     str
:param table_name:     Hive表的名称。sql设置时忽略此参数
:type table_name:     str
:param sql:            查数据sql语句。如：select * from t1
:type sql:            str
:param database:       数据库名称
:type database:       str
:param auth:          认证的方式
:type auth:          str
:param username:       数据库认证的用户名
:type username:       str
:param password:       数据库认证的密码
:type password:       str
:param row_format:     行的输出格式
:type row_format:     str
:return:
:rtype:
```

1.2. 使用方法:

```
import tikit

tikit.upload_to_hdfs("dir1/file1", "http://10.0.3.16:4008", "/dir1/file1")
```

2. 通过client调用的方法

```
describe_cos_buckets(self)
    列出所有bucket

:return:    bucket列表
:rtype:    dict

返回的结果如:
{
  "Owner": {
    "ID": "qcs::cam::uin/100011011262:uin/100011011262",
    "DisplayName": "100011011262"
  },
  "Buckets": {
    "Bucket": [
      {
        "Name": "bucket-58565",
        "Location": "ap-beijing-fsi",
        "CreationDate": "2021-07-21T11:06:00Z",
        "BucketType": "cos"
      }
    ]
  }
}
```

```
{
    "Name": "tai-1300158565",
    "Location": "ap-guangzhou",
    "CreationDate": "2021-10-22T11:04:40Z",
    "BucketType": "cos"
}
]
```

```
describe_cos_path(self, bucket, path, maker='', max_keys=1000, encoding_type='')
```

获取cos的目录的信息。目录下的内容最多显示1000个。要显示目录下的文件和文件夹，请以"/"结尾

```
:param bucket:      cos的桶
:type bucket:       str
:param path:        路径
:type path:         str
:return:            目录信息
:rtype:            dict
:param maker:       从marker开始列出条目
:type maker:        str
:param max_keys:    设置单次返回最大的数量,最大为1000.
:type max_keys:     int
:param encoding_type: 设置返回结果编码方式,只能设置为url.
:type encoding_type: str
```

```
upload_to_cos(self, local_path, bucket, cos_path)
```

把本地路径下的文件或者目录上传到cos上

```
:param local_path: 本地路径
:type local_path:  str
:param bucket:     cos上的桶
:type bucket:      str
:param cos_path:   cos上的路径
:type cos_path:    str
:return:           None. 错误信息通过 raise 给出。
:rtype:
```

```
download_from_cos(self, bucket, cos_path, local_path)
```

从cos上下载文件或者目录本地。

注意：本地文件存在会直接覆盖。 cos_path为目录且local_path 为存在的目录的时候，cos_path的文件夹名会作为子目录保留。

```
:param bucket:     cos上的桶
:type bucket:       str
:param cos_path:    cos上的路径
:type cos_path:     str
:param local_path:  本地路径
:type local_path:   str
:return:           None. 错误信息通过 raise 给出。
:rtype:
```

```
delete_cos_path(self, bucket, delete_path)
```

删除cos目录。要删除目录下的文件和文件夹，请以"/"结尾，即：不带斜杠的，当成文件来删除；带斜杠的，当成文件夹来删除

```
:param bucket:     cos的桶
:type bucket:       str
:param delete_path: 要删除的路径
:type delete_path:  str
```

3. 通过HiveInitial调用的方法

3.1 初始化

```
from tikit.client import Client
from tikit.hive import HiveInitial

client = Client("your_secret_id", "your_secret_key", "<region>")

hive_init = HiveInitial(client)
hive_init.hive_initial("emr-xsjhbhf", "hadoop", "./emr.keytab")
```

3.2 接口

```
def spark_hive_initial_wedata(self, wedata_id, source_account=None):
    """为spark初始化wedata hive (调用改方法后, 就可以使用spark做hive操作)

    :param wedata_id:      wedata数据源id
    :type wedata_id:      int
    :param source_account:  如果hive为系统源, 则需要传使用账户uin
    :type source_account:  str
    :rtype:
    """

def hive_initial_wedata(self, wedata_id, source_account=None):
    """获取wedata hive连接句柄

    :param wedata_id:      wedata数据源id
    :type wedata_id:      int
    :param source_account:  如果hive为系统源, 则需要传使用账户uin
    :type source_account:  str
    :rtype:
    """

def spark_hive_initial(self, emr_id, username=None, keytab=None):
    """为spark初始化emr hive (调用改方法后, 就可以使用spark做hive操作)

    :param emr_id:      腾讯云emr id
    :type emr_id:      str
    :param username:    如果使用kerberos认证, 则需要传入对应用户名
    :type username:    str
    :param keytab:      keytab文件路径。如果使用集群默认账户 (如hadoop), 则需要提供keytab路径
    :type keytab:      str
    :rtype:
    """

def hive_initial(self, emr_id, username=None, keytab=None):
    """获取emr hive连接句柄

    :param emr_id:      腾讯云emr id
    :type emr_id:      str
    :param username:    如果使用kerberos认证, 则需要传入对应用户名
    :type username:    str
    :param keytab:      keytab文件路径。如果使用集群默认账户 (如hadoop), 则需要提供keytab路径
    :type keytab:      str
    :rtype:
    """
```

```
def hive_initial_custom(
    self,
    host=None,
    port=None,
    scheme=None,
    username=None,
    database='default',
    auth=None,
    configuration=None,
    kerberos_service_name=None,
    password=None,
    check_hostname=None,
    ssl_cert=None,
    thrift_transport=None):
    """Connect to HiveServer2

    :param host: What host HiveServer2 runs on
    :param port: What port HiveServer2 runs on. Defaults to 10000.
    :param auth: The value of hive.server2.authentication used by HiveServer2.
        Defaults to ``NONE``.
    :param configuration: A dictionary of Hive settings (functionally same as the `set` command)
    :param kerberos_service_name: Use with auth='KERBEROS' only
    :param password: Use with auth='LDAP' or auth='CUSTOM' only
    :param thrift_transport: A ``TTransportBase`` for custom advanced usage.
        Incompatible with host, port, auth, kerberos_service_name, and password.

    The way to support LDAP and GSSAPI is originated from cloudera/Impyla:
    https://github.com/cloudera/impyla/blob/255b07ed973d47a3395214ed92d35ec0615ebf62
    /impala/_thrift_api.py#L152-L160
    """

    def upload_to_wedata_hive(self, wedata_id, local_path, table_name, database='default',
                             overwrite=False,
                             partition='', source_account=None):
        """上传文件到wedata hive

        :param wedata_id:        wedata 数据源id
        :type wedata_id:         int
        :param local_path:       本地文件路径
        :type local_path:        str
        :param table_name:       表名
        :type local_path:        str
        :param database:         数据库
        :type database:          str
        :param overwrite:        是否删掉原来的数据
        :type overwrite:         bool
        :param partition:        分区的选择
        :type partition:         str
        :param source_account:    如果hive为系统源,则需要传使用账户uin
        :type source_account:    str
        :rtype:
        """

    def export_from_wedata_hive(self, wedata_id, local_path, table_name='', database='default', sql='',
                                row_format="row format delimited fields terminated by ','",
                                source_account=None):
        """导出wedata hive数据到本地

        :param wedata_id:        wedata 数据源id
```

```
:type wedata_id:      int
:param local_path:    本地文件路径
:type local_path:    str
:param table_name:    表名
:type local_path:    str
:param database:      数据库
:type database:      str
:param sql:           查数据sql语句。如: select * from t1
:type sql:           str
:param row_format:    行的输出格式
:type row_format:    str
:param source_account: 如果hive为系统源,则需要传使用账户uin
:type source_account: str
:rtype:
"""
```

提交训练任务

最近更新时间：2025-02-12 16:09:52

```
describe_postpaid_training_price(self)
    查询每种配置的每小时的价格，价格单位：元

    :rtype:          tencentcloud.tione.v20211111.models.DescribeBillingSpecsResponse

describe_system_reasoning_images(self)
    获取平台内置的推理镜像

    :return:         推理镜像信息
    :rtype:          :class:`tikit.tencentcloud.tione.v20211111.models.DescribeInferTemplatesResponse`
    数据格式如：
    {
        "FrameworkTemplates": [
            {
                "Framework": "TensorFlow",
                "FrameworkVersion": "2.4",
                "Groups": [
                    "TensorFlow",
                    "LIGHT"
                ],
                "InferTemplates": [
                    {
                        "InferTemplateId": "tf2.4-py38-cpu",
                        "InferTemplateImage": "ccr.ccs.tencentyun.com/qcloud-ti-platform/ti-cloud-infer-tensorflow-cpu:py38-tensorflow2.4-cpu-20211206"
                    },
                    {
                        "InferTemplateId": "tf2.4-py38-gpu",
                        "InferTemplateImage": "ccr.ccs.tencentyun.com/qcloud-ti-platform/ti-cloud-infer-tensorflow-gpu:py38-tensorflow2.4-cu110-20211206"
                    }
                ]
            }
        ],
        "RequestId": "3654e19b-c2ba-4953-b131-d66495723008"
    }

def create_training_task(self,
                          name,
                          framework,
                          cos_output,
                          worker_resource,
                          code_package_path,
                          ps_resource=None,
                          input_data_config=None,
                          worker_start_cmd=None,
                          ps_start_cmd=None,
                          tags=None,
                          tuning_parameters_dict={},
                          resource_group_id="",
                          remark=None,
                          log_enable=False,
                          log_logset_id=None,
```

```

        log_topic_id=None,
        vpc_id=None,
        sub_net_id=None,
        restart_limit=0):

"""创建训练任务

:param name: 任务名称
:type name: str
:param framework: 运行的框架环境
:type framework: :class:`tikit.models.FrameworkInfo`
:param cos_output: 输出的cos信息
:type cos_output: str
:param worker_resource: worker节点的配置
:type worker_resource: :class:`tikit.models.ResourceConfigInfo`
:param code_package_path: 代码的cos信息
:type code_package_path: str
:param ps_resource: ps节点的配置
:type ps_resource: :class:`tikit.models.ResourceConfigInfo`
:param input_data_config: 输入的数据信息
:type input_data_config: list or :class:`tikit.models.TrainingDataConfig`
:param worker_start_cmd: worker的启动命令
:type worker_start_cmd: str
:param ps_start_cmd: ps节点的启动命令
:type ps_start_cmd: str
:param tags: 标签
:type tags: list of :class:`tikit.tencentcloud.tione.v20211111.models.Tag`
:param tuning_parameters_dict: 调参字典
:type tuning_parameters_dict: dict
:param resource_group_id: 预付费的资源组id
:type resource_group_id: str
:param remark: 描述
:type remark: str
:param log_enable: 日志开关
:type log_enable: bool
:param log_logset_id: 日志集id
:type log_logset_id: str
:param log_topic_id: 日志的topic id
:type log_topic_id: str
:param vpc_id: vpc的id
:type vpc_id: str
:param sub_net_id: 子网id
:type sub_net_id: str
:param restart_limit: 当前任务最大重启次数，最高10次，超过后任务被标记为异常
:type restart_limit: int
:return:
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.CreateTrainingTaskResponse`
"""

def describe_training_frameworks(self):
"""查看训练框架

:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeTrainingFrameworksResponse`

"""

def describe_training_tasks(self, filters=None, tag_filters=None, offset=0, limit=50, order="DESC",
                             order_field="UpdateTime"):
"""获取训练任务列表

```

```

:param filters: 过滤器, eg: [{ "Name": "TaskStatus", "Values": ["Running"] }]
:type filters: list of Filter
:param tag_filters: 标签过滤器, eg: [{ "TagKey": "TagKeyA", "TagValue": ["TagValueA"] }]
:type tag_filters: list of TagFilter
:param offset: 偏移量, 默认为0
:type offset: int
:param limit: 返回数量, 默认为50
:type limit: int
:param order: 输出列表的排列顺序。取值范围: ASC: 升序排列 DESC: 降序排列
:type order: str
:param order_field: 排序的依据字段, 取值范围 "CreateTime" "UpdateTime"
:type order_field: str
:return:
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeTrainingTasksResponse`
"""

def describe_training_task(self, task_id)
    获取单个训练任务信息

    :param task_id: 训练任务ID
    :type task_id: str
    :rtype: :class:`tencentcloud.tione.v20211111.models.DescribeTrainingTaskResponse`

def describe_training_task_pods(self, task_id)
    获取训练任务的pod列表

    :param task_id: 训练任务ID
    :type task_id: str
    :rtype: :class:`tencentcloud.tione.v20211111.models.DescribeTrainingTaskPodsResponse`

def describe_train_logs(self, pod_name, start_time=None, end_time=None, limit=None, order=None,
                        context=None, filters=None):
    """查看训练任务的日志

    :param pod_name: 查询哪个Pod的日志, 支持通配符。查看某个训练任务的全部pod的日志可以填: "<task_id>-*", 如:
train-51cd6bf7ec1000-*
    :type pod_name: str
    :param start_time: 日志查询开始时间。RFC3339格式的时间字符串, 比如2021-12-16T13:20:24+08:00, 默认值为当前时间
的前一个小时
    :type start_time: str
    :param end_time: 日志查询结束时间。RFC3339格式的时间字符串, 比如2021-12-16T13:20:24+08:00, 默认值为当前时间
    :type end_time: str
    :param limit: 日志查询条数, 默认值100, 最大值100
    :type limit: int
    :param order: 排序方向。(ASC | DESC) 默认值为DESC
    :type order: str
    :param context: 分页的游标
    :type context: str
    :param filters: 过滤Filters
    :type filters: list of tikit.tencentcloud.tione.v20211111.models.Filter
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeLogsResponse`

    返回的对象如果非空, 就会有 next() 方法, 能不断地获取下一页的日志 (如果有多页的话), 如下:
    now_time = datetime.datetime.now(datetime.timezone.utc)
    now_time_str = now_time.isoformat()
    result = client.describe_train_logs("train-51cd6bf7ec1000-37c5p5n1r01s-launcher",
                                       "2021-12-10T09:32:03.823509+00:00",
                                       now_time_str,
                                       limit=30)

```

```
print(result)
print(result.next())
print(result.next())
print(result.next())
"""

def push_training_metrics(self, timestamp, value_map, task_id=None, epoch=None, total_steps=None,
step=None):
    """上报训练自定义指标（单条）。单个账号每秒可以调用20次，请在您的训练代码中注意控制上报频率，避免超限报错。或者使用
push_training_metrics_list

:param timestamp: 时间戳
:type timestamp: int
:param value_map: 指标映射。 指标名称 -> 指标值
:type value_map: map: str -> float
:param task_id: 任务ID。若为空，就当前取任务节点环境的 TI_TASK_ID 环境变量的值
:type task_id: str
:param epoch: epoch值
:type epoch: int
:param total_steps: 总步数
:type total_steps: int
:param step: 第几步
:type step: int
:return:
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.PushTrainingMetricsResponse`

client.push_training_metrics(int(time.time()), {"field1": 11, "field2": 12}, "task-id-00001", 3,
1000, 66)
"""

def push_training_metrics_list(self, metric_list)
    上报训练自定义指标（列表）

:param metric_list: MetricData 数组。 若任务ID为空，就当前取任务节点环境的 TI_TASK_ID 环境变量的值
:type metric_list: list of :class:`tencentcloud.tione.v20211111.models.MetricData`
:return:
:rtype: :class:`tencentcloud.tione.v20211111.models.PushTrainingMetricsResponse`

def describe_training_metrics(self, task_id)
    查询训练自定义指标

:param task_id: 任务ID
:type task_id: str
:rtype: :class:`tencentcloud.tione.v20211111.models.DescribeTrainingMetricsResponse`

def stop_training_task(self, task_id)
    停止某个训练任务

:param task_id: 训练任务ID
:type task_id: str
:rtype: :class:`tencentcloud.tione.v20211111.models.StopTrainingTaskResponse`

def delete_training_task(self, task_id)
    删除某个训练任务

:param task_id: 训练任务ID
:type task_id: str
:rtype: :class:`tencentcloud.tione.v20211111.models.DeleteTrainingTaskResponse`
```

中间配置:

```
tikit.models.FrameworkInfo

def new_custom(training_mode, image_type, image_url, registry_region=None, registry_id=None,
               user_name=None, passwd=None):
    """自定义训练框架的配置 通过describe_training_frameworks() 查看列表

    :param training_mode: 训练模式, 如"DDP", 仅训练任务需要配置
    :type training_mode: str
    :param image_type: 镜像类型, CCR为腾讯云容器镜像服务个人版, TCR为腾讯云容器镜像服务企业版, CUSTOM表示第三
方自定义镜像
    :type image_type: str
    :param image_url: 镜像地址 必填
    :type image_url: str
    :param registry_region: 腾讯云容器镜像服务的镜像仓库的地域
    :type registry_region: str
    :param registry_id: 腾讯云容器镜像服务的镜像仓库ID
    :type registry_id: str
    :param user_name: 自定义镜像仓库的用户名
    :type user_name: str
    :param passwd: 自定义镜像仓库的密码
    :type passwd: str
    :return:
    :rtype:
    """

def new_system_framework(
    framework_name, framework_environment, training_mode):
    """系统内置的训练框架

    :param framework_name: 框架名称。 通过describe_training_frameworks() 查看列表
    :type framework_name: str
    :param framework_environment: 框架环境。 通过describe_training_frameworks() 查看列表
    :type framework_environment: str
    :param training_mode: 训练模式。 通过describe_training_frameworks() 查看列表
    :type training_mode: str
    :return:
    :rtype:
    """

-----

tikit.models.ResourceConfigInfo

def new_postpaid(instance_type, instance_num):
    """获取后付费模式下的资源配置

    :param instance_type: 实例类型。通过 describe_postpaid_training_price() 查看实例列表
    :type instance_type: str
    :param instance_num: 实例数量
    :type instance_num: int
    :return:
    :rtype:
    """

|

def new_prepaid(
    cpu=0, memory=0, gpu=0, gpu_type=None, instance_num=1):
    """获取预付费模式下的资源配置, 当资源组开启GPU按比例预设后, cpu和memory参数需要不设置或者设置为0, 该两项的值将通过
GPU卡数按照比例分配
```

```
:param cpu:      CPU个数, 单位是核
:type cpu:      float
:param memory:   内存大小, 单位是GB
:type memory:   float
:param gpu_type: gpu类型
:type gpu_type: str
:param gpu:      gpu个数
:type gpu:      float
:param instance_num: 实例数量
:type instance_num: int
:return:
:rtype:
"""

-----

tikit.models.TrainingDataConfig

def new_mount_cos(cos_str, target_path):
    """一个cos类型的训练数据

    :param cos_str:      cos存储, 格式: <bucket>/<cos path>/
    :type cos_str:      str
    :param target_path: 目标挂载路径
    :type target_path: str
    :return:
    :rtype:
    """

def new_dataset_mount(dataset_id, target_path):
    """一个dataset类型的训练数据

    :param dataset_id: 数据集ID
    :type dataset_id: str
    :param target_path: 目标挂载路径
    :type target_path: str
    :return:
    :rtype:
    """

def new_mount_cfs(cfs_id, source_path, target_path):
    """新建一个cfs类型的训练数据集配置

    :param cfs_id:      CFS的ID
    :type cfs_id:      str
    :param source_path: CFS的路径
    :type source_path: str
    :param target_path: 目标挂载路径
    :type target_path: str
    :return:
    :rtype:
    """

def new_mount_hdfs(hdfs_id, source_path, target_path):
    """新建一个hdfs类型的训练数据集配置

    :param hdfs_id:      EMR上HDFS的ID
    :type hdfs_id:      str
    :param source_path:  HDFS的路径
```

```
:type source_path: str
:param target_path: 目标挂载路径
:type target_path: str
:return:
:rtype:
"""

def new_mount_wedata_hdfs(wedata_id, source_path):
    """新建一个wedata hdfs类型的训练数据集配置

    :param wedata_id: wedata数据源id
    :type wedata_id: int
    :param source_path: HDFS的路径
    :type source_path: str
    :return:
    :rtype:
    """

def new_dataset(id_target_dict):
    """ Deprecated !
    新建一个dataset类型的训练数据集配置

    :param id_target_dict: 数据集信息。 dataset id -> 下载的目标路径
    :type id_target_dict: dict
    :return:
    :rtype:
    """

def new_cos_data(cos_str_target_dict):
    """Deprecated !
    新建一个cos类型的训练数据集配置

    :param cos_str_target_dict: 数据集信息。 <bucket>/<cos path>/ -> 下载的目标路径
    :type cos_str_target_dict: dict
    :return:
    :rtype:
    """

def new_mount_goosefs(goosefs_id, source_path, namespace, target_path):
    """新建一个goosefs类型的训练数据集配置

    :param goosefs_id: goosefs实例id
    :type goosefs_id: str
    :param target_path: 目标挂载路径
    :type target_path: str
    :return:
    :rtype:
    """

def new_mount_goosefsx(goosefsx_id, goosefsx_path, target_path):
    """新建一个goosefsx类型的存储配置

    :param goosefsx_id: goosefsx实例id
    :type goosefsx_id: str
    :param goosefsx_path: goosefsx路径
    :type goosefsx_path: str
    :param target_path: 目标挂载路径
    :type target_path: str
    :return:
```

```
:rtype:  
"""
```

开发机创建与管理

最近更新时间：2025-02-26 17:42:22

说明：

原 Notebook 产品模块已于2025-02-25日升级为开发机，Tikit 相关接口暂时不变。

```
def create_notebook(self,
                    name,
                    image_info=None,
                    resource_info=None,
                    input_data_config=None,
                    ssh_config=None,
                    tags=None,
                    resource_group_id="",
                    lifecycle_script_id="",
                    code_repo_id_list=None,
                    auto_stop_enable=False,
                    auto_stop_time=None,
                    log_enable=False,
                    log_logset_id=None,
                    log_topic_id=None,
                    vpc_id=None,
                    subnet_id=None):
    """创建Notebook实例
    :param name:          实例名称
    :type name:           str
    :param image_info:    实例镜像信息
    :type image_info:     :class:`tikit.models.ImageInfo`
    :param resource_info: 实例资源配置配置
    :type resource_info:  :class:`tikit.models.ResourceConfigInfo`
    :param input_data_config:  输入的存储信息
    :type input_data_config:  list or :class:`tikit.models.NotebookDataConfig`
    :param ssh_config:    ssh配置信息
    :type ssh_config:     :class:`tikit.models.SSHConfig`
    :param tags:          标签
    :type tags:           list of :class:`tikit.tencentcloud.tione.v20211111.models.Tag`
    :param resource_group_id: 预付费的资源组id
    :type resource_group_id: str
    :param lifecycle_script_id: 生命周期脚本id
    :type lifecycle_script_id: str
    :param code_repo_id_list: git存储库id列表
    :type code_repo_id_list: list
    :param auto_stop_enable: 自动停止开关
    :type auto_stop_enable: bool
    :param auto_stop_time: 自动停止时间，单位小时
    :type auto_stop_time: int
    :param log_enable:    日志开关
    :type log_enable:     bool
    :param log_logset_id: 日志集id
    :type log_logset_id: str
    :param log_topic_id: 日志的topic id
    :type log_topic_id:  str
    :param vpc_id:        挂载cfs或者turbofs时，需要打通的vpc_id
    :type vpc_id:         str
    :param subnet_id:    挂载cfs或者turbofs时，vpc下的subnet
    :type subnet_id:     str
```

```
:return:
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.CreateTrainingTaskResponse`
"""

def modify_notebook(self,
                    notebook_id,
                    image_info=None,
                    resource_info=None,
                    input_data_config=None,
                    ssh_config=None,
                    resource_group_id=None,
                    lifecycle_script_id=None,
                    code_repo_id_list=None,
                    auto_stop_enable=False,
                    auto_stop_time=None,
                    log_enable=None,
                    log_logset_id=None,
                    log_topic_id=None,
                    vpc_id=None,
                    subnet_id=None):
    """编辑Notebook实例
    :param notebook_id: 实例ID
    :type notebook_id: str
    :param image_info: 实例镜像信息
    :type image_info: :class:`tikit.models.ImageInfo`
    :param resource_info: 实例资源配置配置
    :type resource_info: :class:`tikit.models.ResourceConfigInfo`
    :param input_data_config: 输入的存储信息
    :type input_data_config: list or :class:`tikit.models.NotebookDataConfig`
    :param ssh_config: ssh配置信息
    :type ssh_config: :class:`tikit.models.SSHConfig`
    :param resource_group_id: 预付费的资源组id
    :type resource_group_id: str
    :param lifecycle_script_id: 生命周期脚本id
    :type lifecycle_script_id: str
    :param code_repo_id_list: git存储库id列表
    :type code_repo_id_list: list
    :param auto_stop_enable: 自动停止开关
    :type auto_stop_enable: bool
    :param auto_stop_time: 自动停止时间, 单位小时
    :type auto_stop_time: int
    :param log_enable: 日志开关
    :type log_enable: bool
    :param log_logset_id: 日志集id
    :type log_logset_id: str
    :param log_topic_id: 日志的topic id
    :type log_topic_id: str
    :param vpc_id: 挂载cfs或者turbofs时, 需要打通的vpc_id
    :type vpc_id: str
    :param subnet_id: 挂载cfs或者turbofs时, vpc下的subnet
    :type subnet_id: str
    :return:
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.CreateTrainingTaskResponse`
    """

def stop_notebook(self, notebook_id):
    """停止某个notebook实例

    :param notebook_id: notebookID
```

```
:type notebook_id: str
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.StopNotebookResponse`
"""

def delete_notebook(self, notebook_id):
    """删除某个notebook实例

    :param notebook_id: notebookID
    :type notebook_id: str
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DeleteNotebookResponse`
    """

def start_notebook(self, notebook_id):
    """启动某个notebook实例

    :param notebook_id: notebookID
    :type notebook_id: str
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.StartNotebookResponse`
    """

def describe_notebooks(self, filters=None, tag_filters=None, offset=0, limit=10, order="DESC",
                       order_field="UpdateTime"):
    """获取notebook列表

    :param filters: 过滤器, eg: [{"Name": "Status", "Values": ["Running"], "Fuzzy": False}]
    Name (名称): notebook1, 支持模糊匹配
    Id (notebook ID): nb-123456789, 支持模糊匹配
    Status (状态): Submitting / Starting / Running / Stopped / Stopping / Failed / SubmitFailed /
    ImageSaving
    ChargeType (计费类型): PREPAID (预付费) / POSTPAID_BY_HOUR (后付费)
    ChargeStatus (计费状态): NOT_BILLING (未开始计费) / BILLING (计费中) / BILLING_STORAGE (存储计费
    中) / ARREARS_STOP (欠费停止)
    :type filters: list of Filter
    :param tag_filters: 标签过滤器, eg: [{"TagKey": "TagKeyA", "TagValue": ["TagValueA"]} ]
    :type tag_filters: list of TagFilter
    :param offset: 偏移量, 默认为0
    :type offset: int
    :param limit: 返回数量, 默认为10
    :type limit: int
    :param order: 输出列表的排列顺序。取值范围: ASC: 升序排列 DESC: 降序排列
    :type order: str
    :param order_field: 排序的依据字段, 取值范围 "CreateTime" "UpdateTime"
    :type order_field: str
    :return:
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeTrainingTasksResponse`
    """

def describe_notebook(self, notebook_id):
    """获取单个notebook任务信息

    :param notebook_id: notebookID
    :type notebook_id: str
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeNotebookkResponse`
    """

def describe_notebook_buildin_images(self):
    """获取单个notebook任务信息

    :param notebook_id: notebookID
```

```
:type notebook_id: str
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeBuildInImagesResponse`
"""
```

注册模型

最近更新时间：2024-08-22 10:16:15

```
create_model_by_cos(self, training_model_name, algorithm_framework, model_cos_path, model_index, reasoning_env, delete_task_cos_model=True, tags=None)
```

使用现有的cos路径来创建新的模型

```
:param training_model_name: 模型名称
:type training_model_name: str
:param algorithm_framework: 算法框架
:type algorithm_framework: str
:param model_cos_path: 模型cos目录, 以/结尾。格式: <bucket>/<cos path>/
:type model_cos_path: str
:param model_index: 训练指标。
:type model_index: str
:param reasoning_env: 推理的运行环境
:type reasoning_env: :class:`tikit.models.ReasoningEnvironment`
:param delete_task_cos_model: 是否删除任务的原来输出的模型文件
:type delete_task_cos_model: bool
:param tags: 标签配置
:type tags: list of :class:`tencentcloud.tione.v20211111.models.Tag`
:return:
:rtype: :class:`tencentcloud.tione.v20211111.models.CreateTrainingModelResponse`
```

返回的内容如下:

```
{
  "Id": "m-23054253294030848",
  "RequestId": "c8f9b70e-bf4d-4c34-9480-21d902d3b341"
}
```

```
create_model_by_task(self, training_model_name, training_job_id, reasoning_env, training_model_index=None, delete_task_cos_model=True, tags=None)
```

使用现有的任务的结果来创建新的模型

```
:param training_model_name: 模型名称
:type training_model_name: str
:param training_job_id: 训练任务id
:type training_job_id: str
:param reasoning_env: 推理的运行环境
:type reasoning_env: :class:`tikit.models.ReasoningEnvironment`
:param training_model_index: 训练指标。非空时覆盖训练任务里面的值。
:type training_model_index: str
:param delete_task_cos_model: 是否删除任务的原来输出的模型文件
:type delete_task_cos_model: bool
:param tags: 标签配置
:type tags: list of :class:`tencentcloud.tione.v20211111.models.Tag`
:return:
:rtype: :class:`tencentcloud.tione.v20211111.models.CreateTrainingModelResponse`
```

返回的内容如下:

```
{
  "Id": "m-23054253294030848",
  "RequestId": "c8f9b70e-bf4d-4c34-9480-21d902d3b341"
}
```

```
create_model_version_by_cos(self, training_model_id, algorithm_framework, model_cos_path, model_index, reasoning_env, delete_task_cos_model=True, tags=None)
```

使用现有的cos路径来创建新的模型版本

```
:param training_model_id: 模型ID
:type training_model_id: str
:param algorithm_framework: 算法框架
:type algorithm_framework: str
:param model_cos_path: 模型cos目录, 以/结尾。格式: <bucket>/<cos path>/
:type model_cos_path: str
:param model_index: 训练指标。
:type model_index: str
:param reasoning_env: 推理的运行环境
:type reasoning_env: :class:`tikit.models.ReasoningEnvironment`
:param delete_task_cos_model: 是否删除任务的原来输出的模型文件
:type delete_task_cos_model: bool
:param tags: 标签配置
:type tags: list of :class:`tencentcloud.tione.v20211111.models.Tag`
:return:
:rtype: :class:`tencentcloud.tione.v20211111.models.CreateTrainingModelResponse`
```

返回的内容如下:

```
{
  "Id": "m-23054253294030848",
  "RequestId": "c8f9b70e-bf4d-4c34-9480-21d902d3b341"
}
```

```
create_model_version_by_task(self, training_model_id, training_job_id, reasoning_env,
training_model_index=None, delete_task_cos_model=True, tags=None)
```

使用现有的cos路径来创建新的模型版本

```
:param training_model_id: 模型ID
:type training_model_id: str
:param training_job_id: 训练任务id
:type training_job_id: str
:param reasoning_env: 推理的运行环境
:type reasoning_env: :class:`tikit.models.ReasoningEnvironment`
:param training_model_index: 训练指标。非空时覆盖训练任务里面的值。
:type training_model_index: str
:param delete_task_cos_model: 是否删除任务的原来输出的模型文件
:type delete_task_cos_model: bool
:param tags: 标签配置
:type tags: list of :class:`tencentcloud.tione.v20211111.models.Tag`
:return:
:rtype: :class:`tencentcloud.tione.v20211111.models.CreateTrainingModelResponse`
```

返回的内容如下:

```
{
  "Id": "m-23054253294030848",
  "RequestId": "c8f9b70e-bf4d-4c34-9480-21d902d3b341"
}
```

```
describe_training_model_version(self, training_model_version_id)
```

查看单个版本的信息

```
:param training_model_version_id: 模型版本id
:type training_model_version_id: str
:rtype: :class:`tencentcloud.tione.v20211111.models.DescribeTrainingModelVersionResponse`
```

```
describe_training_model_versions(self, training_model_id)
```

查看模型各个版本的信息列表

```
:param training_model_id: 模型id
:type training_model_id: str
:rtype: :class:`tencentcloud.tione.v20211111.models.DescribeTrainingModelVersionsResponse`
```

```
describe_training_models(self, filters=None, order_field=None, order=None, offset=None, limit=None,
tag_filters=None)
```

查看模型列表

```
:param filters: 过滤器
:type filters: list of Filter
:param order_field: 排序字段
:type order_field: str
:param order: 排序方式, ASC/DESC
:type order: str
:param offset: 偏移量
:type offset: int
:param limit: 返回结果数量
:type limit: int
:param tag_filters: 标签过滤
:type tag_filters: list of TagFilter
:rtype: :class:`tencentcloud.tione.v20211111.models.DescribeTrainingModelsResponse`
print 返回的结果, 输出如下:
```

模型ID	名称	标签	创建时间
m-23054253294030848	tikit-model-task-1227		2021-12-27T14:22:43Z
m-23054252760240128	tikit-model-cos-1227		2021-12-27T14:22:35Z
m-23054246746066944	tikit-model-task		2021-12-27T14:21:03Z
m-23037023501881344	tikit-model-cos		2021-12-24T13:20:57Z
m-23036973226987520	tikit-model-name-2		2021-12-24T13:08:10Z
m-23034443699064832	model_cos-2		2021-12-24T02:24:52Z
m-23028904650346496	model_cos	tag_001:tag_v_001	2021-12-23T02:56:13Z
m-23028894739075072	model_task		2021-12-23T02:53:42Z
22997374387884032	xx004		2021-12-17T13:17:39Z
22996889833377792	xx003		2021-12-17T11:14:26Z

```
delete_training_model(self, training_model_id)
```

删除模型

```
:param training_model_id: 模型id
:type training_model_id: str
:rtype: :class:`tencentcloud.tione.v20211111.models.DeleteTrainingModelResponse`
```

```
delete_training_model_version(self, training_model_version_id)
```

删除模型版本

```
:param training_model_version_id: 模型版本id
:type training_model_version_id: str
:rtype: :class:`tencentcloud.tione.v20211111.models.DeleteTrainingModelVersionResponse`
```

模型优化

最近更新时间：2023-12-14 12:32:10

```
def create_model_accelerate_task(self,
                                model_acc_task_name,
                                model_source,
                                model_format,
                                model_name,
                                model_version,
                                model_input_path,
                                model_output_path,
                                tensor_infos,
                                acc_engine_version = None,
                                optimization_level = None,
                                gpu_type = None,
                                hyper_parameter = None,
                                tags = None,
                                model_signature = None):

    """创建模型加速任务

    :param model_acc_task_name:          任务名称
    :type model_acc_task_name:           str
    :param model_source:                 模型来源 (JOB/COS)
    :type model_source:                  str
    :param model_format:                 模型格式
    (TORCH_SCRIPT/DETECTRON2/SAVED_MODEL/FROZEN_GRAPH/MMDETECTION/ONNX/HUGGING_FACE)
    :type model_format:                  str
    :param model_name:                   模型名称
    :type model_name:                    str
    :param model_version:                 模型版本
    :type model_version:                  str
    :param model_input_path:              模型输入cos地址
    :type model_input_path:               :class:`tikit.models.CosPathInfo`
    :param model_output_path:             优化模型输出cos地址
    :type model_output_path:              :class:`tikit.models.CosPathInfo`
    :param acc_engine_version:             优化引擎版本, 不传则默认使用最新版本
    :type acc_engine_version:             str
    :param tensor_infos:                  tensor信息
    :type tensor_infos:                   list of str
    :param optimization_level:            优化级别 (NO_LOSS/FP16), 默认FP16
    :type optimization_level:             str
    :param gpu_type:                      GPU卡类型 (T4/V100), 默认T4
    :type gpu_type:                       str
    :param hyper_parameter:               专业参数设置
    :type hyper_parameter:                 :class:`tikit.tencentcloud.tione.v20211111.models.HyperParameter`
    :param tags:                           标签
    :type tags:                            list of :class:`tikit.tencentcloud.tione.v20211111.models.Tag`
    :param model_signature:                SAVED_MODEL保存时配置的签名
    :type model_signature:                 str
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.CreateModelAccelerateTaskResponse`
    """

def stop_model_accelerate_task(self, model_acc_task_id):

    """停止模型加速任务

    :param model_acc_task_id:             模型加速任务id
    """
```

```

:type model_acc_task_id:          str
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.StopModelAccelerateTaskResponse`
"""

def delete_model_accelerate_task(self, model_acc_task_id):
    """删除模型加速任务

    :param model_acc_task_id:      模型加速任务id
    :type model_acc_task_id:       str
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DeleteModelAccelerateTaskResponse`
    """

def restart_model_accelerate_task(self,
                                   model_acc_task_id,
                                   model_source,
                                   model_format,
                                   model_name,
                                   model_version,
                                   model_input_path,
                                   model_output_path,
                                   tensor_infos,
                                   acc_engine_version = None,
                                   optimization_level = None,
                                   gpu_type = None,
                                   hyper_parameter = None,
                                   tags = None,
                                   model_signature = None):
    """重启模型加速任务

    :param model_acc_task_id:      任务id
    :type model_acc_task_id:       str
    :param model_source:           模型来源 (JOB/COS)
    :type model_source:           str
    :param model_format:          模型格式
    (TORCH_SCRIPT/DETECTRON2/SAVED_MODEL/FROZEN_GRAPH/MMDETECTION/ONNX/HUGGING_FACE)
    :type model_format:           str
    :param model_name:            模型名称
    :type model_name:             str
    :param model_version:         模型版本
    :type model_version:          str
    :param model_input_path:      模型输入cos地址
    :type model_input_path:       :class:`tikit.models.CosPathInfo`
    :param model_output_path:     优化模型输出cos地址
    :type model_output_path:      :class:`tikit.models.CosPathInfo`
    :param tensor_infos:          tensor信息
    :type tensor_infos:           list of str
    :param acc_engine_version:    优化引擎版本, 不传则默认使用最新版本
    :type acc_engine_version:     str
    :param optimization_level:    优化级别 (NO_LOSS/FP16), 默认FP16
    :type optimization_level:     str
    :param gpu_type:             GPU卡类型 (T4/V100), 默认T4
    :type gpu_type:              str
    :param hyper_parameter:      专业参数设置
    :type hyper_parameter:       :class:`tikit.tencentcloud.tione.v20211111.models.HyperParameter`
    :param tags:                 标签
    :type tags:                  list of :class:`tikit.tencentcloud.tione.v20211111.models.Tag`
    :param model_signature:      SAVED_MODEL保存时配置的签名
    :type model_signature:       str
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.RestartModelAccelerateTaskResponse`
    """

```

```
"""

def describe_model_accelerate_tasks(self, filters=None, order_field=None, order=None, offset=None,
limit=None,
                                tag_filters=None):

    """查看加速任务列表

    :param filters:          过滤器
    :type filters:          list of Filter, 支持ModelAccTaskName/ModelSource
    :param order_field:     排序字段, 默认为CreateTime
    :type order_field:     str
    :param order:          排序方式, ASC/DESC, 默认为DESC
    :type order:          str
    :param offset:        偏移量
    :type offset:        int
    :param limit:         返回结果数, 默认为10
    :type limit:         int
    :param tag_filters:    标签过滤
    :type tag_filters:    list of TagFilter
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeModelAccelerateTasksResponse`
    """

def describe_model_accelerate_task(self, model_acc_task_id):

    """查询模型加速任务详情

    :param model_acc_task_id: 模型加速任务id
    :type model_acc_task_id:  str
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeModelAccelerateTaskResponse`
    """

def create_optimized_model(self, model_acc_task_id, tags = None):

    """保存优化模型

    :param model_acc_task_id: 模型加速任务id
    :type model_acc_task_id:  str
    :param tags:             标签
    :type tags:             list of :class:`tikit.tencentcloud.tione.v20211111.models.Tag`
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.CreateOptimizedModelResponse`
    """
```

发布服务

最近更新时间：2024-08-22 11:54:06

```
create_model_service(self, service_group_name, worker_resource, framework, service_group_id=None,
service_description=None, resource_group_id=None, model_config_info=None, model_cos_path=None,
env=None, scale_mode="MANUAL", replicas=1, horizontal_pod_autoscaler=None, log_enable=False,
log_logset_id=None, log_topic_id=None, authorization_enable=False, tags=None, cron_scale_jobs=None,
scale_strategy=None, hybrid_billing_prepaid_replicas=None):
```

创建模型服务

```
:param service_group_name: 服务名称
:type service_group_name: str
:param worker_resource:worker节点的配置
:type worker_resource: :class:`tikit.models.ModelServiceResourceConfigInfo`
:param framework: 运行的框架环境
:type framework: :class:`tikit.models.FrameworkInfo`
:param service_group_id: 服务id
:type service_group_id:str
:param service_description: 服务描述
:type service_description: str
:param resource_group_id: 包年包月模式下所属的资源组id
:type resource_group_id: str
:param model_config_info: 模型信息
:type model_config_info: :class:`tikit.models.ModelConfigInfo`
:param model_cos_path: 模型cos信息
:type model_cos_path: str
:param env: 环境变量
:type env: list of :class:`tikit.tencentcloud.tione.v20211111.models.EnvVar`
:param scale_mode: 扩缩容类型
:type scale_mode: str
:param replicas: 实例数量
:type replicas:int
:param horizontal_pod_autoscaler: 自动伸缩信息
:type horizontal_pod_autoscaler:
:class:`tikit.tencentcloud.tione.v20211111.models.HorizontalPodAutoscaler`
:param log_enable: 日志开关
:type log_enable: bool
:param log_logset_id: 日志集id
:type log_logset_id: str
:param log_topic_id: 日志的topic id
:type log_topic_id: str
:param authorization_enable: 接口鉴权开关
:type authorization_enable: bool
:param tags: 标签
:type tags: list of :class:`tikit.tencentcloud.tione.v20211111.models.Tag`
:param cron_scale_jobs:定时任务配置
:type cron_scale_jobs: list of :class:`tikit.tencentcloud.tione.v20211111.models.CronScaleJob`
:param scale_strategy: 自动伸缩策略配置 HPA
:type scale_strategy: str
:param hybrid_billing_prepaid_replicas: 混合计费模式下预付费实例数
:type hybrid_billing_prepaid_replicas: int
:return:
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.CreateModelServiceResponse`

create_model_service_version(self, worker_resource, framework, service_group_id=None,
service_description=None, resource_group_id=None, model_config_info=None, model_cos_path=None,
env=None, scale_mode="MANUAL", replicas=1, horizontal_pod_autoscaler=None, log_enable=False,
```

```
log_logset_id=None, log_topic_id=None, authorization_enable=False, tags=None, cron_scale_jobs=None,
scale_strategy=None, hybrid_billing_prepaid_replicas=None):
```

创建模型服务版本

```
:param worker_resource: worker 节点的配置
:type worker_resource: :class:`tikit.models.ModelServiceResourceConfigInfo`
:param framework: 运行的框架环境
:type framework: :class:`tikit.models.FrameworkInfo`
:param service_group_id: 服务id
:type service_group_id: str
:param service_description: 服务描述
:type service_description: str
:param resource_group_id: 预付费模式下所属的资源组id
:type resource_group_id: str
:param model_config_info: 模型信息
:type model_config_info: :class:`tikit.models.ModelConfigInfo`
:param model_cos_path: 模型cos信息
:type model_cos_path: str
:param env: 环境变量
:type env: list of :class:`tikit.tencentcloud.tione.v20211111.models.EnvVar`
:param scale_mode: 扩容容类型
:type scale_mode: str
:param replicas: 实例数量
:type replicas: int
:param horizontal_pod_autoscaler: 自动伸缩信息
:type horizontal_pod_autoscaler:
:class:`tikit.tencentcloud.tione.v20211111.models.HorizontalPodAutoscaler`
:param log_enable: 日志开关
:type log_enable: bool
:param log_logset_id: 日志集id
:type log_logset_id: str
:param log_topic_id: 日志的topic id
:type log_topic_id: str
:param authorization_enable: 接口鉴权开关
:type authorization_enable: bool
:param tags: 标签
:type tags: list of :class:`tikit.tencentcloud.tione.v20211111.models.Tag`
:param cron_scale_jobs: 定时任务配置
:type cron_scale_jobs: list of :class:`tikit.tencentcloud.tione.v20211111.models.CronScaleJob`
:param scale_strategy: 自动伸缩策略配置 HPA
:type scale_strategy: str
:param hybrid_billing_prepaid_replicas: 混合计费模式下预付费实例数
:type hybrid_billing_prepaid_replicas: int
:return:
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.CreateModelServiceResponse`
```

```
modify_model_service(self, service_id, model_config_info=None, model_cos_path=None, framework=None,
env=None, worker_resource=None, scale_mode="MANUAL", replicas=1, horizontal_pod_autoscaler=None,
log_enable=False, log_logset_id=None, log_topic_id=None, service_action=None, service_description=None,
scale_strategy=None, cron_scale_jobs=None):
```

更新模型服务

```
:param service_id: 服务id
:type service_id: str
:param model_config_info: 模型信息
:type model_config_info: :class:`tikit.models.ModelConfigInfo`
:param model_cos_path: 模型cos信息
:type model_cos_path: str
```

```
:param framework: 运行的框架环境
:type framework: :class:`tikit.models.FrameworkInfo`
:param env:环境变量
:type env: list of :class:`tikit.tencentcloud.tione.v20211111.models.EnvVar`
:param worker_resource: worker节点配置
:type worker_resource: :class:`tikit.models.ModelServiceResourceConfigInfo`
:param scale_mode: 扩缩容类型
:type scale_mode: str
:param replicas: 实例数量
:type replicas: int
:param horizontal_pod_autoscaler: 自动伸缩信息
:type horizontal_pod_autoscaler:
:class:`tikit.tencentcloud.tione.v20211111.models.HorizontalPodAutoscaler`
:param log_enable: 日志开关
:type log_enable: bool
:param log_logset_id: 日志id
:type log_logset_id: str
:param log_topic_id: 日志topic id
:type log_topic_id:str
:param service_action: 特殊更新行为
:type service_action: str
:param service_description: 服务描述
:type service_description: str
:param scale_strategy: 自动伸缩策略
:type scale_strategy: str
:param cron_scale_jobs: 自动伸缩策略配置 HPA
:type cron_scale_jobs: list of :class:`tikit.tencentcloud.tione.v20211111.models.CronScaleJob`
:return:
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.ModifyModelServiceResponse`
```

```
describe_model_service_groups(self, filters=None, order_field=None, order=None, offset=None,
limit=None, tag_filters=None):
```

查看所有在线推理服务组

```
:param filters: 过滤器
:type filters: list of Filter
:param order_field: 排序字段
:type order_field: str
:param order: 排序方式, ASC/DESC
:type order: str
:param offset: 偏移量
:type offset: int
:param limit: 返回结果数量
:type limit: int
:param tag_filters: 标签过滤
:type tag_filters: list of TagFilter
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeModelServiceGroupsResponse`
```

```
describe_model_service_group(self, service_group_id):
```

查寻单个服务组

```
:param service_group_id: 服务组id
:type service_group_id: str
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeModelServiceGroupResponse`
```

```
describe_model_services(self, filters=None, order_field=None, order=None, offset=None, limit=None, tag_filters=None):
```

查询多个服务

```
:param filters: 过滤器
:type filters: list of Filter
:param order_field: 排序字段
:type order_field: str
:param order: 排序方式, ASC/DESC
:type order: str
:param offset: 偏移量
:type offset: int
:param limit: 返回结果数量
:type limit: int
:param tag_filters: 标签过滤
:type tag_filters: list of TagFilter
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeModelServicesResponse`
```

```
describe_model_service(self, service_id):
```

查询单个服务

```
:param service_id: 服务id
:type service_id: str
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeModelServiceResponse`
```

```
stop_model_service(self, service_id):
```

关闭服务

```
:param service_id: 服务id
:type service_id: str
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.ModifyModelServiceResponse`
```

```
start_model_service(self, service_id):
```

启动已经关闭的服务

```
:param service_id: 服务id
:type service_id: str
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.ModifyModelServiceResponse`
```

```
delete_model_service(self, service_id):
```

删除已经关闭的服务

```
:param service_id: 服务id
:type service_id: str
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DeleteModelServiceResponse`
```

```
delete_model_service_group(self, service_group_id):
```

删除服务组

```
:param service_group_id: 服务组id
:type service_group_id: str
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DeleteModelServiceGroupResponse`
```

```
modify_service_group_weights(self, service_group_id, service_id_to_weight):
```

更改服务组的流量分配

```
:param service_group_id: 服务组id
```

```
:type service_group_id: str
```

```
:param service_id_to_weight: 服务id与权重对
```

```
:type service_id_to_weight: dict
```

```
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.ModifyServiceGroupWeightsResponse`
```

提交跑批任务

最近更新时间：2024-08-22 10:22:31

```
create_batch_task(self,
                    batch_task_name,
                    job_type,
                    resource_config_info,
                    output_config,
                    input_config,
                    resource_group_id=None,
                    model_info=None,
                    image_info=None,
                    code_package=None,
                    start_cmd=None,
                    cron_info=None,
                    tags=None,
                    log_enable=False,
                    log_logset_id=None,
                    log_topic_id=None,
                    vpc_id=None,
                    subnet_id=None,
                    remark=""):

"""创建跑批任务

:param batch_task_name: 跑批任务名称
:type batch_task_name: str
:param job_type: 工作类型 1:单次 2:周期
:type job_type: int
:param resource_config_info: 资源配置
:type resource_config_info: :class:`tikit.models.ResourceConfigInfo`
:param output_config: 输出数据源配置（如果为COS，目标路径需为空字符串）
:type output_config: list or :class:`tikit.models.TrainingDataConfig`
:param input_config: 输入数据源配置（如果为COS，目标路径需为空字符串）
:type input_config: list or :class:`tikit.models.TrainingDataConfig`
:param resource_group_id: 预付费的资源组id
:type resource_group_id: str
:param model_info: 模型信息
:type model_info: :class:`tikit.models.ModelInfo`
:param image_info: 镜像信息（如果使用模型文件，则自定义镜像无效）
:type image_info: :class:`tikit.models.FrameworkInfo`
:param code_package: 代码的cos信息
:type code_package: str
:param start_cmd: 启动命令
:type start_cmd: str
:param cron_info: 周期配置信息
:type cron_info: :class:`tikit.tencentcloud.tione.v20211111.models.CronInfo`
:param tags: 标签
:type tags: list of :class:`tikit.tencentcloud.tione.v20211111.models.Tag`
:param log_enable: 日志开关
:type log_enable: bool
:param log_logset_id: 日志集id
:type log_logset_id: str
:param log_topic_id: 日志topic id
:type log_topic_id: str
:param vpc_id: vpc id
:type vpc_id: str
:param subnet_id:
```

```
:type subnet_id: 子网id
:param remark: 任务描述
:type remark: str
:rtype: :class:`tikit.tencentcloud.tione.v20211111.models.CreateBatchTaskResponse`
"""

describe_batch_task(self, batch_task_id):
    """查询跑批任务

    :param batch_task_id:
    :type batch_task_id: str
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeBatchTaskResponse`
    """

describe_batch_tasks(self, filters=None, tag_filters=None, offset=0, limit=50, order="DESC",
                    order_field="UpdateTime"):
    """获取跑批任务列表

    :param filters: 过滤器, eg: [{"Name": "TaskStatus", "Values": ["Running"]} ]
    :type filters: list of Filter
    :param tag_filters: 标签过滤器, eg: [{"TagKey": "TagKeyA", "TagValue": ["TagValueA"]} ]
    :type tag_filters: list of TagFilter
    :param offset: 偏移量, 默认为0
    :type offset: int
    :param limit: 返回数量, 默认为50
    :type limit: int
    :param order: 输出列表的排列顺序。取值范围: ASC: 升序排列 DESC: 降序排列
    :type order: str
    :param order_field: 排序的依据字段, 取值范围 "CreateTime" "UpdateTime"
    :type order_field: str
    :return:
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DescribeBatchTasksResponse`
    """

stop_batch_task(self, batch_task_id):
    """停止跑批任务

    :param batch_task_id:
    :type batch_task_id: str
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.StopBatchTaskResponse`
    """

delete_batch_task(self, batch_task_id):
    """删除跑批任务

    :param batch_task_id:
    :type batch_task_id: str
    :rtype: :class:`tikit.tencentcloud.tione.v20211111.models.DeleteBatchTaskResponse`
    """
```