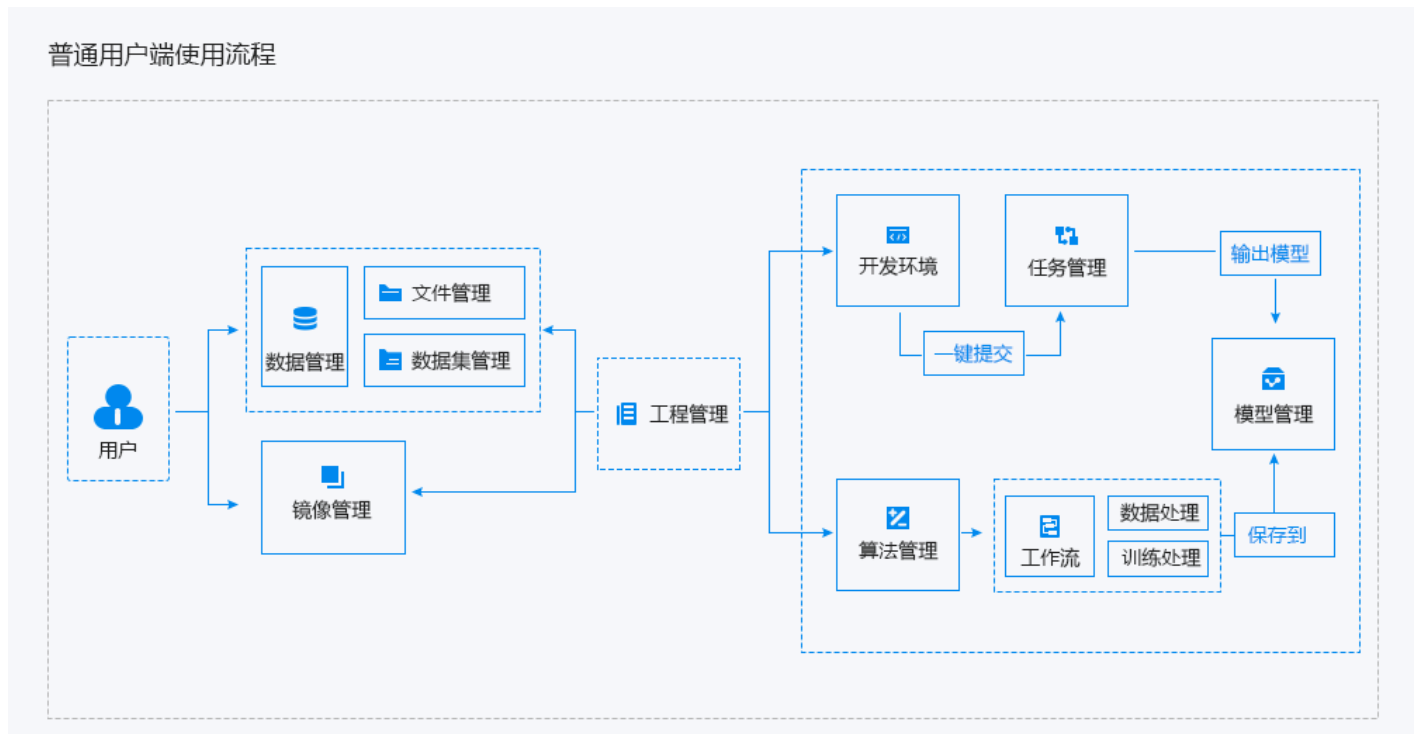


AIStation-普通用户



整体介绍

软件介绍

人工智能开发平台 AIStation，提供智能的 AI 容器化部署以及更具效率的分布式训练。

AIStation 是浪潮面向人工智能企业训练场景的人工智能开发资源平台，可实现容器化部署、可视化开发、集中化管理等，为用户提供极致高性能的 AI 计算资源，实现高效的计算力支撑、精准的资源管理和调度、敏捷的数据整合及加速、流程化的 AI 场景及业务整合，有效打通开发环境、计算资源与数据资源，提升开发效率。

用户通过 AIStation 平台能够创建不同的深度学习框架环境，可以自由的进行模型的开发，通过命令行方式进行调试模型，然后通过开发平台快速提交到训练平台，达到开发训练一体化解决方案。

本平台可以帮助用户实现如下功能：

提供多种数据使用方式

平台提供了开发环境中可以使用用户自己的数据集方式，平台提供共有数据集方式，该数据集统一存放到共享目录下，用户可以按需选择不同的数据集，该数据集由管理员统一维护。

在线模型开发功能

平台默认提供了 jupyter 功能，且每个用户创建的开发环境都自带 jupyter 方便用户进行模型的开发，且自动带全屏功能，相当于一个独立的 IDE 开发环境。

框架环境多种连接方式

深度学习框架运行环境支持 web 版本 shell 直接连接，在该页面上用户可以使用任何相关的命令操作，满足命令行操作习惯的用户使用。

深度学习框架运行环境支持本地 shell 连接方式，通过在开发列表中直接复制 ssh 连接方式，自动连接到开发环境中。

多种深度学习训练任务模式

平台提供单机训练任务、分布式训练任务、MPI 训练任务三种类型，用户根据自身的业务需求进行灵活选择。

多种资源自动匹配

平台提供集群不同 Gpu 卡类型自动识别技术，在调度中会根据业务需求进行自动调度到相同类型的 Gpu 卡上，也支持不同类型的 Gpu 卡调度。

任务容错全自动化

平台提供了多种容错方式，自动识别网络中断、服务器宕机、Gpu 卡丢失的情况，自动会把作业重新运行，如果有 checkpoint 会自动恢复等容错方式，保证用户的任务高可靠的运行。

系统要求

浪潮 AIStation 人工智能开发平台需要以下软件环境支撑：

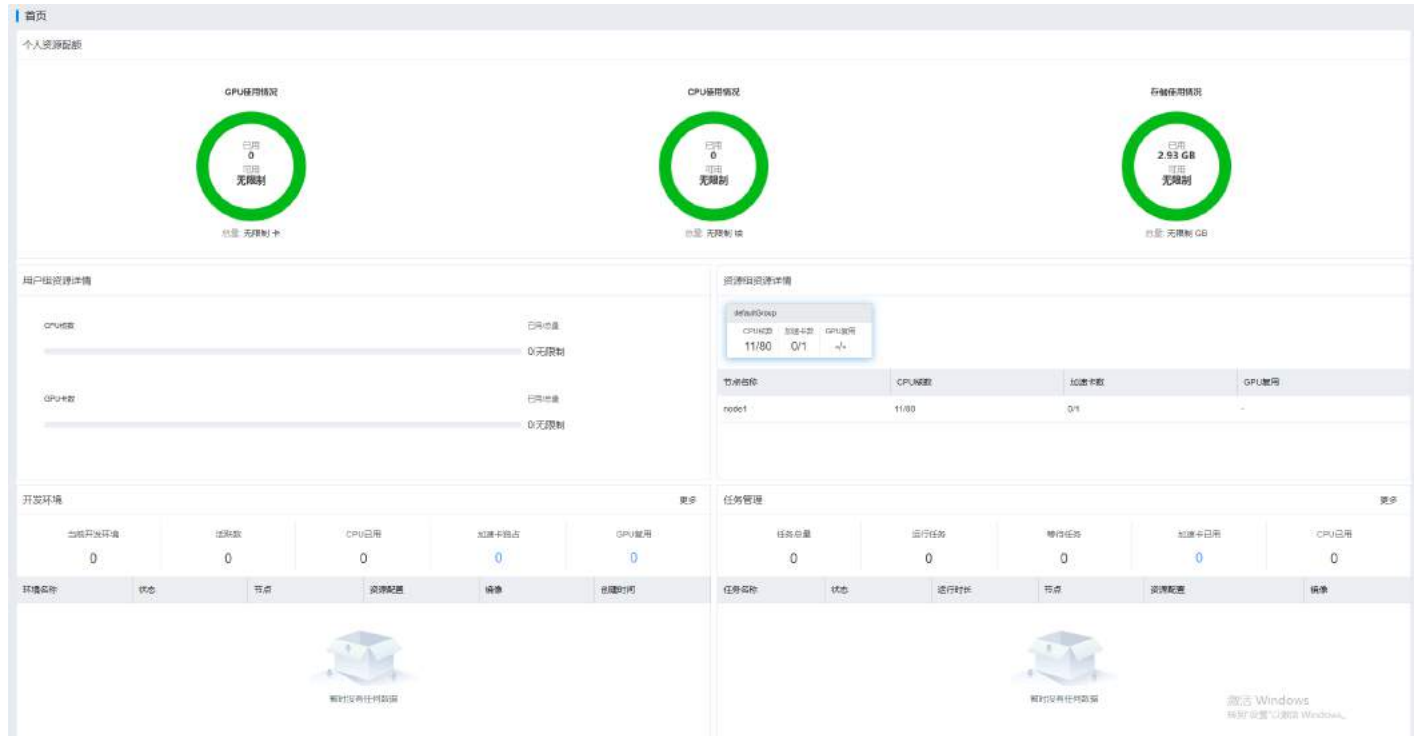
客户浏览端：

操作系统：windows 10 以上版本。

浏览器：chrome80.0 及以上版本。

首页

普通用户登录系统之后可以在首页中直观的看到与个人相关资源的使用情况以及用户个人创建开发环境和训练平台的运行情况，如下图所示：



注意：页面刷新频率为 30s

上图中包含的具体信息详细说明如下：

1. 个人资源配额显示当前用户个人资源的使用情况，包括：



GPU 使用情况：总量（创建用户时分配的资源大小，可以设置为无限制）、已用（状态为正在运行与排队中的开发环境 + 训练平台使用 GPU 卡的和）、可用（总量-已用）

MLU 使用情况：总量（创建用户时分配的资源大小，可以设置为无限制）、已用（状态为正在运行与排队中的开发环境 + 训练平台使用 MLU 卡的和）、可用（总量-已用）

CPU 使用情况：总量（创建用户时分配的资源大小，可以设置为无限制）、已用（状态为正在运行与排队中的开发环境 + 训练平台使用 CPU 的和）、可用（总量-已用）

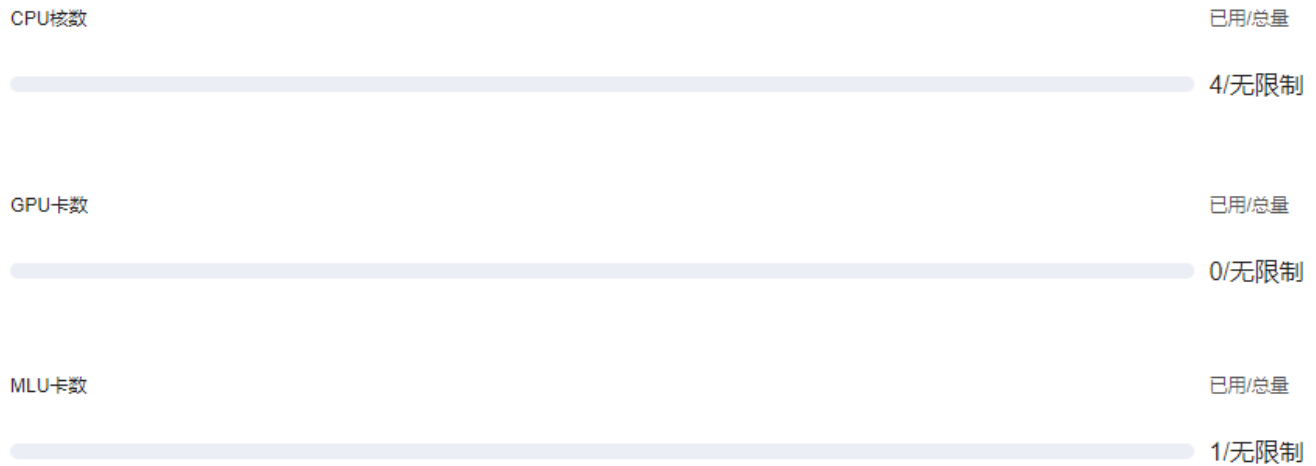
存储使用情况：总量（创建用户时分配的资源大小，可以设置为无限制）、已用（统计当前用户的用户

目录在节点中的实际使用空间)、可用(总量-已用)

注意: 当总量为无限制时, 可用也为无限制; 当已用超过总量时, 可用显示为 0; 当集群中没有 MLU 节点的时候, MLU 使用情况将不会展示。

2. 用户组资源详情(逻辑资源)显示当前用户所在用户组资源的使用情况, 包括:

用户组资源详情



CPU 核数: 已用(当前用户所在用户组, 状态为正在运行与排队中的开发环境 + 训练平台使用 CPU 的和)、总量(管理员创建用户组时分配的 CPU 核数大小, 可以配置为无限制)

GPU 卡数: 已用(当前用户所在用户组, 状态为正在运行与排队中的开发环境 + 训练平台使用 GPU 的和)、总量(管理员创建用户组时分配的 GPU 卡大小, 可以配置为无限制)

MLU 卡数: 已用(当前用户所在用户组, 状态为正在运行与排队中的开发环境 + 训练平台使用 MLU 的和)、总量(管理员创建用户组时分配的 MLU 卡大小, 可以配置为无限制)

颜色说明: 0-49%: 绿色; 50%-79%: 橙色; 80%-100%: 红色。

注意: 当集群中没有 MLU 节点的时候, MLU 卡数将不会展示。

3. 资源组资源详情(物理资源), 显示当前用户所在资源组以及资源组内节点的资源使用情况, 包括: 资源组使用情况

资源组资源详情

train_gpu_762			train_mlu522		defaultGroup			defaultGroup_MLU	
CPU核数	加速卡数	GPU复用	CPU核数	加速卡数	CPU核数	加速卡数	GPU复用	CPU核数	加速卡数
11/96	0/16	-/-	5/48	1/4	0/0	0/0	-/-	0/0	0/0

节点名称	CPU核数	加速卡数	GPU复用
node1	10/32	0/8	-
node211	1/64	0/8	-

CPU 核数：已用（统计当前资源组下实际使用的 CPU 核数，包括组件使用）、总量（统计当前资源组下所有节点的实际 CPU 核数）

加速卡数：已用（统计当前资源组下实际使用的加速卡数，如果同一个卡被多个任务使用则只统计一次，已用不会超过总量）、总量（统计当前资源组下所有节点的实际加速卡数）

共享模式下包括 GPU 复用、GPU 显存复用、A100 复用

GPU 复用：已用（统计当前资源组下所有任务使用的 GPU 共享数量）、总量（该资源组下 GPU 复用的个数），如果不是共享则显示“-”

GPU 显存复用：已用（统计当前资源组下所有任务使用的 GPU 显存大小）、总量（该资源组下 GPU 显存复用大小）

A100 复用：已用（按照 A100mig 规格统计当前资源组下所有任务使用的 GPU 数量）、总量（按照 A100mig 规格统计该资源组下 GPU 个数）

节点使用情况

节点名称：当前用户所在资源组内包含的节点的名称

CPU 核数：已用（统计当前节点下 CPU 实际使用的数量，包括节点组件中使用的资源，向上取整，不能超过总量）、总量（该节点下 CPU 总核数）

加速卡数：已用（统计当前节点下实际使用的加速卡数，如果同一个卡被多个任务使用则只统计一次，已用不会超过总量）、总量（统计当前节点下的实际加速卡数）

共享模式下包括 GPU 复用、GPU 显存复用、A100 复用

GPU 复用：已用（统计当前节点下所有任务使用的 GPU 共享数量）、总量（该节点下 GPU 复用的个数），如果不是共享则显示“-”

GPU 显存复用：已用（统计当前节点下所有任务使用的 GPU 显存大小）、总量（该节点下 GPU 显存复用大小）

A100 复用：已用（按照 A100mig 规格统计当前节点下所有任务使用的 GPU 数量）、总量（按照 A100mig

规格统计该节点下 GPU 个数)

4. 开发环境，显示当前用户所创建开发环境的运行情况，最多显示 5 条数据，其它任务通过查看“更多”跳转到完整任务列表

开发环境 更多

当前开发环境	活跃数	CPU已用	加速卡独占	GPU复用	
0	0	0	0	0	
环境名称	状态	节点	资源配置	镜像	创建时间



暂时没有任何数据

在该区域可以显示：

- 当前开发环境：未被删除的开发平台数量
- 活跃数：当前用户所属不是停止、排队中的开发环境数量
- CPU 已用：当前用户所属活跃中的开发环境所占用的 CPU 数量
- 加速卡独占：当前用户所属活跃中并且使用加速卡整卡的开发环境的加速卡的数量
- GPU 复用：当前用户所属活跃中并且不使用 GPU 整卡的开发环境的 GPU 卡的数量
- 列表信息：环境名称、状态、节点、资源配置、镜像、创建时间
- 快捷键：点击【更多】跳转到【开发环境】

CPU已用	加速卡独占	GPU复用
0	0	0

GPU	0
MLU	0



5. 训练任务，显示当前用户所创建训练平台的运行情况，最多显示 5 条数据，其它任务通过查看“更多”跳转到完整任务列表

训练任务 更多

任务总量	运行任务	等待任务	加速卡已用	CPU已用
1	1	0	1	4

任务名称	状态	运行时长	节点	资源配置	镜像
mlu	运行中	3分 34秒	node2	worker*1: MLU290-32GB:1, CPU:4, MEM:0GB	100.2.126.198:5000/te...

在该区域可以显示：

- 任务总量：当前用户创建的所有任务信息的数量
- 运行任务：状态为运行中、镜像拉取中、数据集拉取中的任务数量
- 等待任务：状态为排队中的任务数量
- 加速卡已用：状态为运行中、数据集下载中、镜像拉取中的任务使用的加速卡数量之和
- CPU 已用：状态为运行中、数据集下载中、镜像拉取中的任务使用的 CPU 核数数量之和
- 列表信息：任务名称、状态、运行时长、节点、资源配置、镜像
- 快捷键：点击【更多】跳转到【任务管理】-【训练任务】

任务	加速卡已用	CPU已用
0	1	4
GPU	0	
MLU	1	2.126.19

工程管理

创建工程

用户点击【创建工程】，进入创建工程页面。用户填写工程信息进行创建

创建工程
×

* 名称

描述 4/300

工程信息查看

用户可以选择列表中的工程，点击工程名称，即展开该工程的下拉信息页面。工程信息以标签页形式进行展示：

1. 基本信息：创建时间、更新时间、拥有者、描述
2. 开发环境：展示最近 10 条属于该工程的开发环境任务及其状态，点击名称打开详细信息，其它任务

通过查看“更多”跳转到完整任务列表

3. 训练任务：展示最近 10 条未完成的属于该工程的训练任务及其状态，点击名称打开详细信息，其它任务通过查看“更多”跳转到完整任务列表

4. 数据处理任务：展示最近 10 条未完成的属于该工程的数据处理任务及其状态，点击名称打开详细信息，其它任务通过查看“更多”跳转到完整任务列表

5. workflow 任务：展示最近 10 条属于该工程的 workflow 任务及其状态，点击名称打开详细信息，其它任务通过查看“更多”跳转到完整任务列表



编辑工程

用户点击右侧【编辑】按钮修改工程的描述信息



创建工程任务

用户在已创建的工程下，可以创建任务，这些任务都属于该工程。工程任务包括：

1. 开发环境
2. 训练任务
3. 数据处理任务
4. workflow 任务



这些任务的创建，可参见开发环境、任务管理、工作流管理相关内容

删除工程

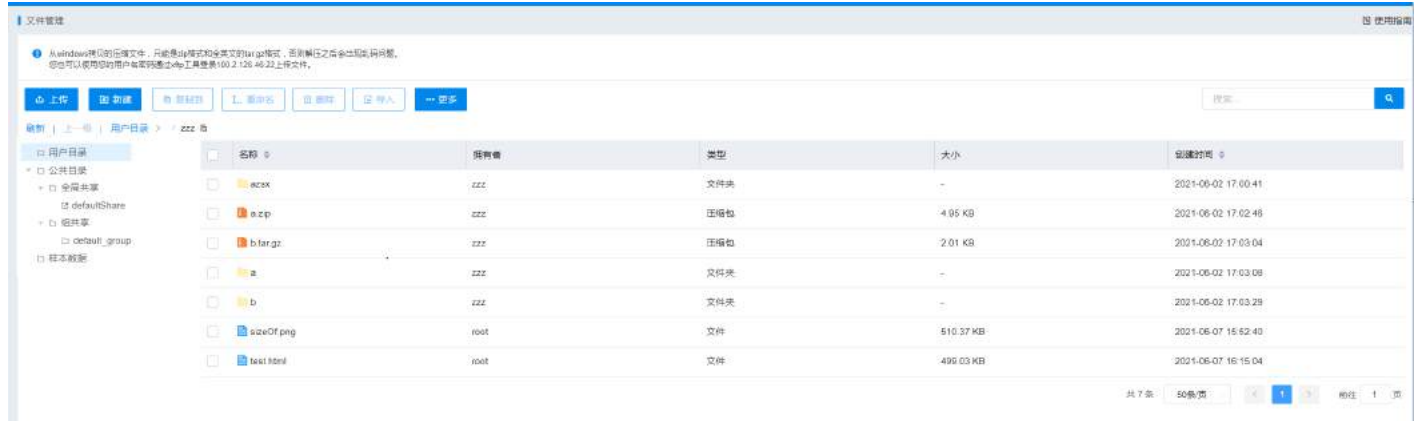
可以删除没有关联任务的工程

数据管理

文件管理

文件列表

1. 单击【文件管理】，查看用户目录、公共目录和样本数据，默认显示普通用户家目录文件列表，公共目录包括全局共享和组共享，用户目录和公共目录的文件列表显示项均为名称、拥有者、类型、大小、创建时间。



文件管理用户目录可以对接多存储后，用户目录左侧的目录树将使用存储名分级展示如下图，其中主存储展示在第一个。



2. 文件列表上方有快捷文件操作按钮，包括上传、新建、复制到、重命名、删除、导入，点开更多，可以查看其余文件操作按钮。



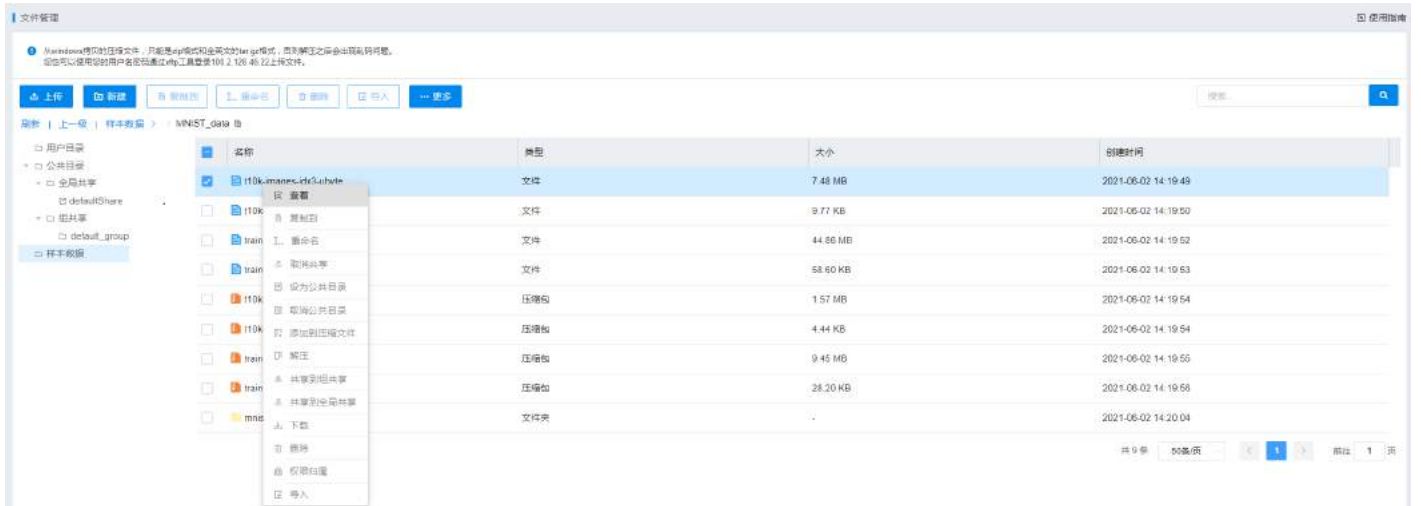
3. 单击【文件管理】->【公共目录】下的文件夹，显示全局共享和组共享文件，普通用户可以查看所属公共目录下文件。全局共享下有默认的 defaultShare 目录，管理员可以在全局共享下，将目录设为公共目录后，普通用户即可查看。普通用户的组共享目录只有一个，即为绑定的用户组，文件名称与用户组名相同。





4. 单击【文件管理】->【样本数据】下的文件夹，显示样本数据目录，样本数据目录可以查看文件列表，可以查看文件（包括普通文件和图片），可以支持文件搜索，不可进行其他操作。

当样本数据权限开关开启后，该目录只能展示已授权给该用户的样本数据目录。



5. 除样本数据以外，用户目录、全局共享目录、组共享目录，可在文件列表表头对名称和创建时间进行升序或降序排序。



新建文件夹

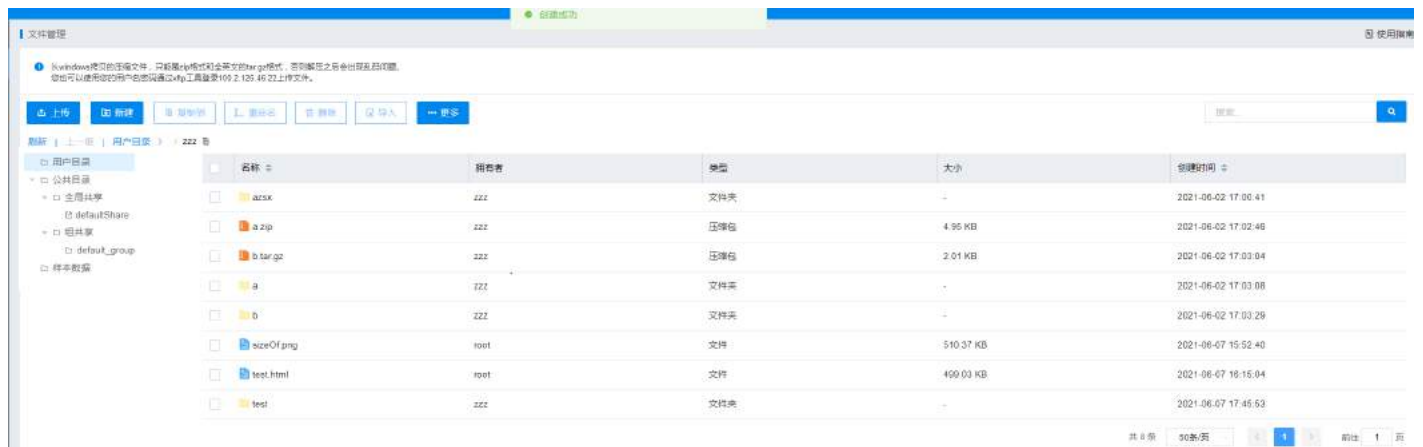
1. 单击列表上方快捷键新建，可以在用户目录新建文件夹。



2. 弹出新建文件夹界面，在名称输入栏，输入新建文件夹名，只能输入汉字、英文字母、数字、点、下划线和连接线，不能以连接线和点开头。



3. 输入合法文件夹名，点击确定，则页面自动刷新，显示新创建的文件夹。



新建文件

1. 单击列表上方快捷键新建，可以在用户目录新建文件。



2. 弹出新建文件界面，在名称输入栏，输入合法文件名（与新建文件夹约束一致）。点击确定，则页面自动刷新，显示新建的文件。

查看

1. 可以查看普通文件和图片，支持查看 50M 以下的文件。
2. 普通文件查看：文件显示区域，选中文件，单击右键，弹出右键菜单，点击查看，或者双击左键，查看文件内容。



弹出的编辑文件界面，可以对文件进行编辑，点击【确定】保存。

3. 查看图片：支持双击和点击查看按钮进行图片预览。默认查看 10 张图片，多张图片采用轮播图方式展示，可放大预览。

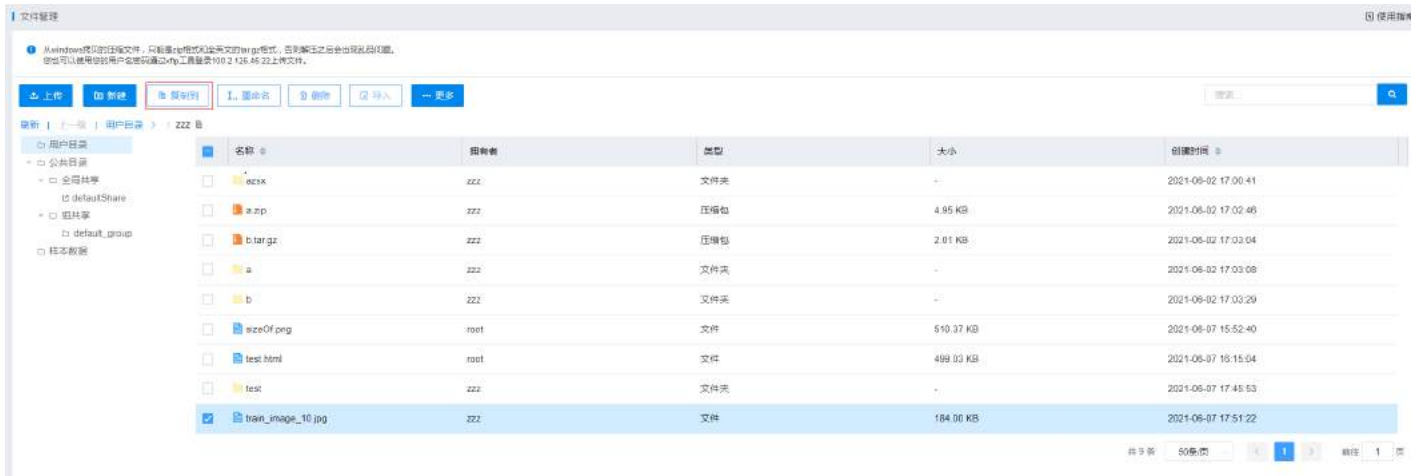
查看图片

✕



复制到

1. 选择文件，单击右键，点击右键菜单的【复制到】或者点击列表上方快捷菜单【复制到】，弹出对话框，选择目标文件夹，文件的处理进度显示在右侧进度列表中。可通过进度列表的目录链接，进入文件列表。需要注意的是：不支持不同存储之间的复制



重命名

选择文件，单击右键，点击右键菜单的【重命名】或者点击列表上方快捷菜单【重命名】，弹出重命名界面，在名称输入栏输入合法的文件名（与新建文件夹约束一致），点击【确定】，页面自动刷新，显示

重命名文件。

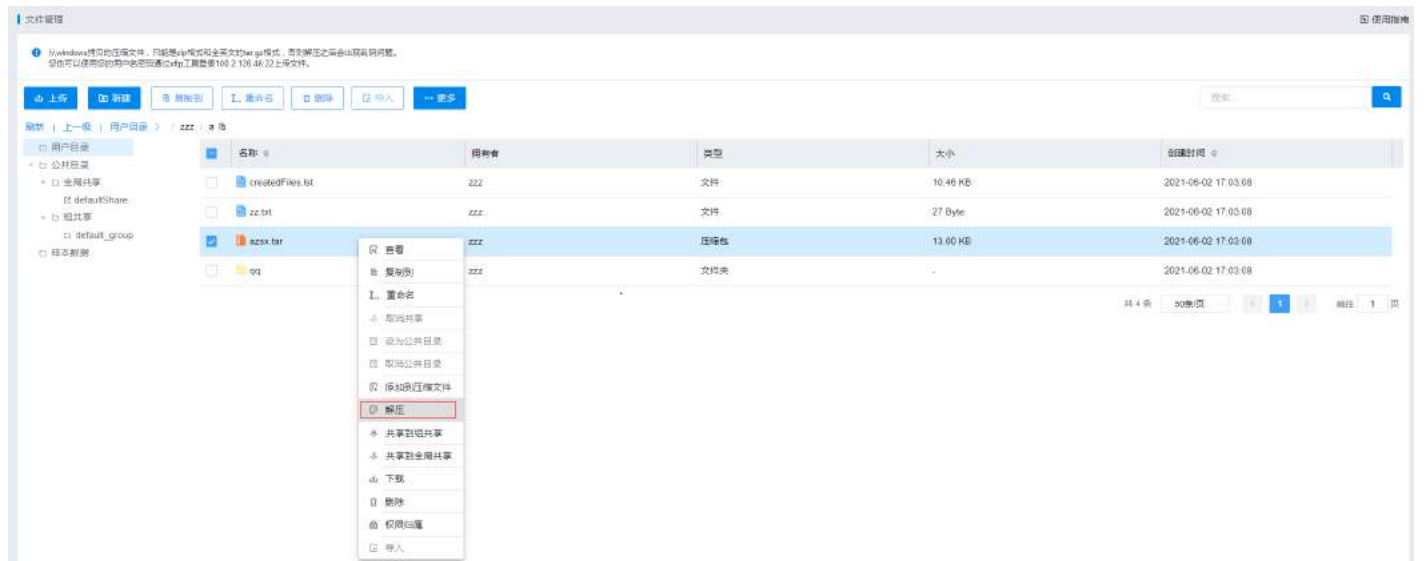
压缩

压缩文件支持批量或单个文件进行压缩，支持压缩格式 tar、tar.gz 和 zip, 首先选择文件，然后单击右键，点击【添加到压缩文件】或者点击列表上方快捷菜单【更多】选择添加到压缩文件，弹出压缩文件界面，在名称输入栏输入合法的压缩文件名，点击【确定】，页面自动刷新，显示压缩的文件，文件的处理进度显示在右侧进度列表中。可通过进度列表的目录链接，进入文件列表。压缩文件只可以在用户目录操作。



解压

选择压缩文件，右键点击【解压】，将压缩包解压到与压缩包同名的文件夹下，页面自动刷新，显示解压文件，文件的处理进度显示在右侧进度列表中。可通过进度列表的目录链接，进入文件列表。



上传

1. 单击列表上方快捷键上传，可以在用户目录上传文件。



2. 点击【上传文件】，弹出加载文件界面，选择需要上传的文件。

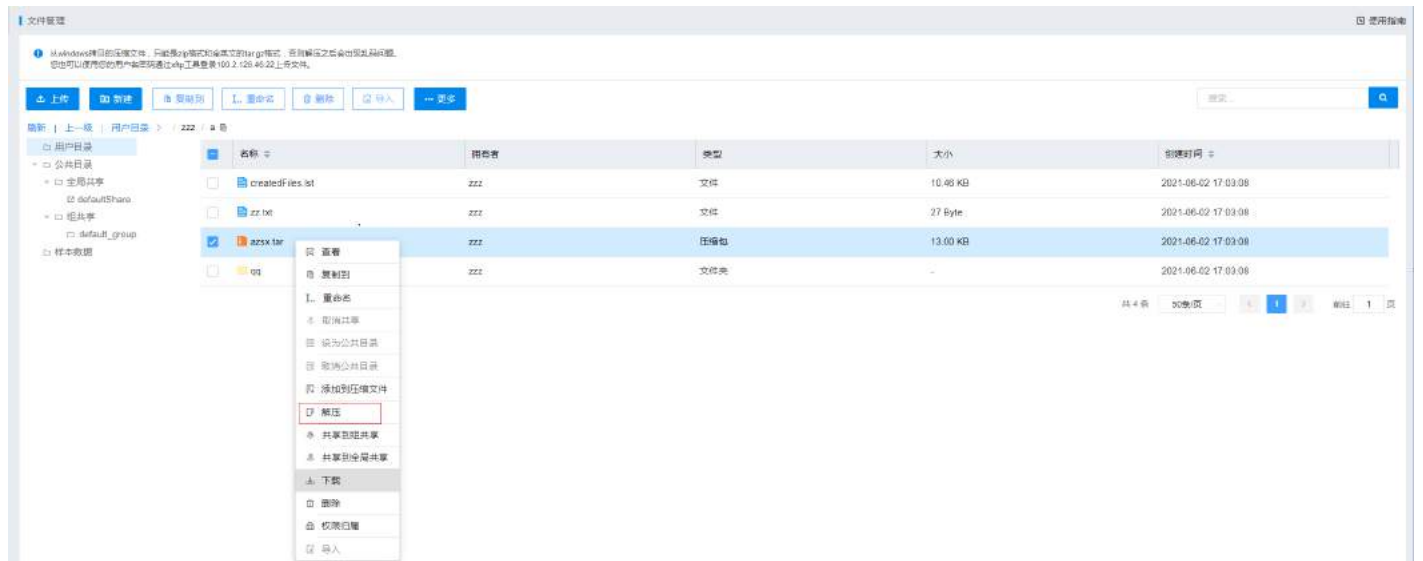
3. 选中要上传的文件，点击打开，文件处理列表中显示上传文件的进度。上传完成，页面自动刷新，列表中显示上传的文件。

4. 上传文件不能超过 1G，如果文件超过 1G 使用 xftp 工具进行上传。



下载

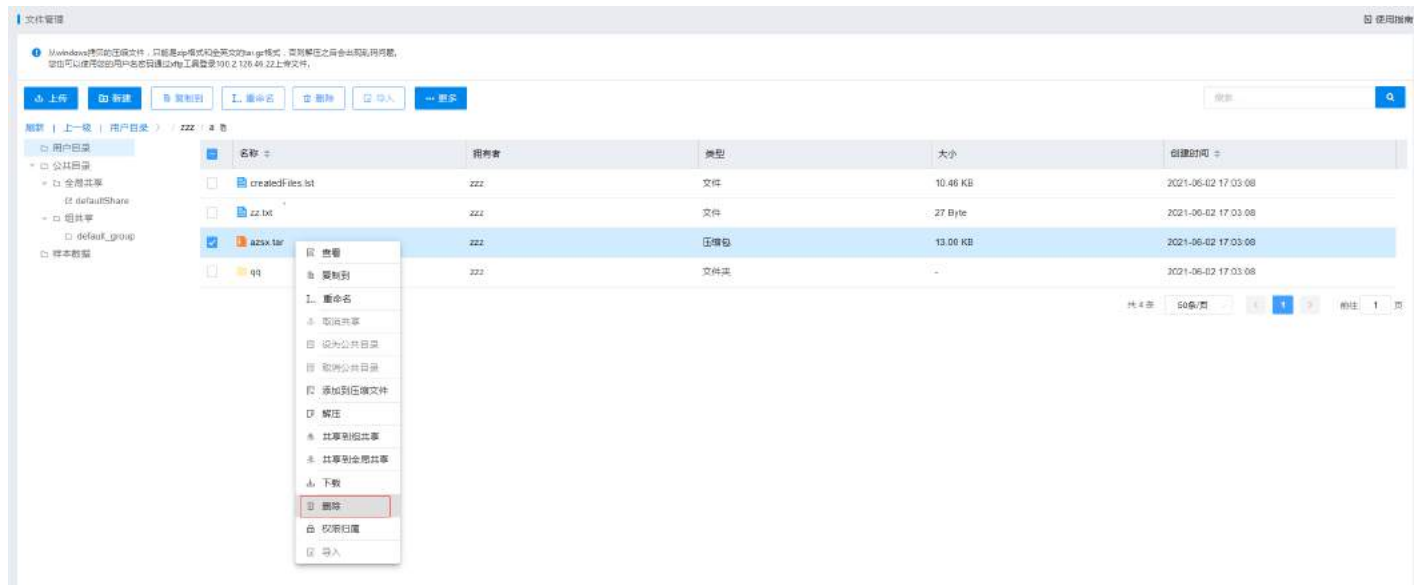
1. 选择下载文件，右键菜单点击【下载】或者点击列表上方快捷菜单【更多】选择下载，只支持单个文件下载. 只可以在用户目录下载文件。



2. 弹出下载文件提示。

删除

1. 删除支持单个文件和批量删除，首先选择删除的文件或文件夹，右键菜单，点击【删除】或者点击列表上方快捷菜单【删除】。

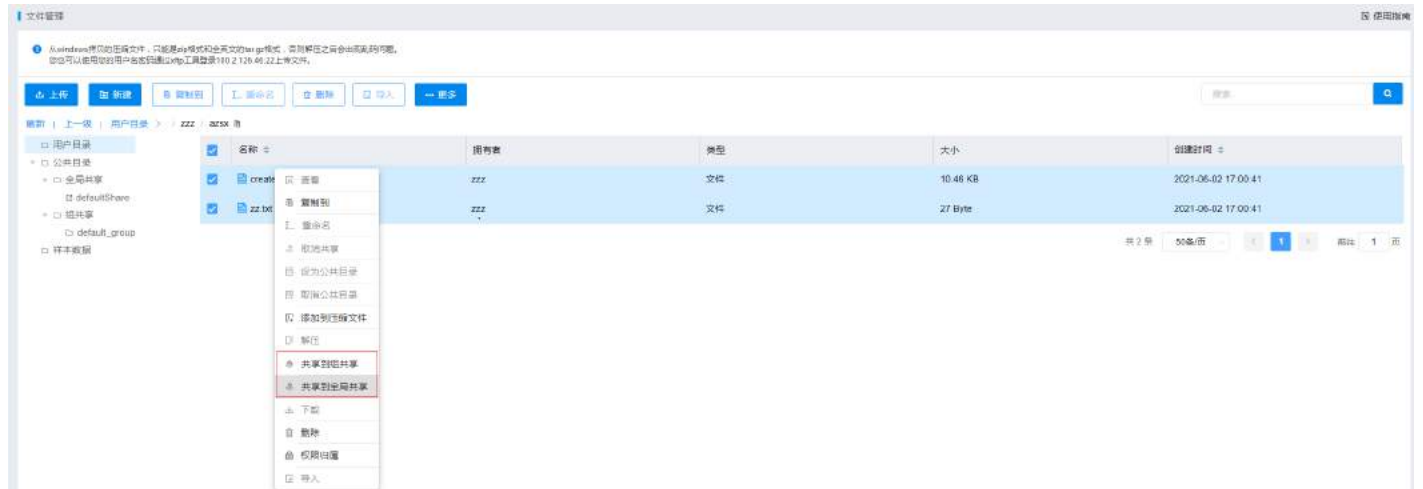


2. 弹出确定删除界面，点击【确定】，删除文件，文件处理列表中显示删除文件的进度。删除完成，页面自动刷新。

共享

1. 用户目录的操作可以共享到组共享或共享到全局共享。选择文件，单击右键，点击右键菜单的【共享到组共享】/【共享到全局共享】或者点击列表上方快捷菜单【更多】选择操作。

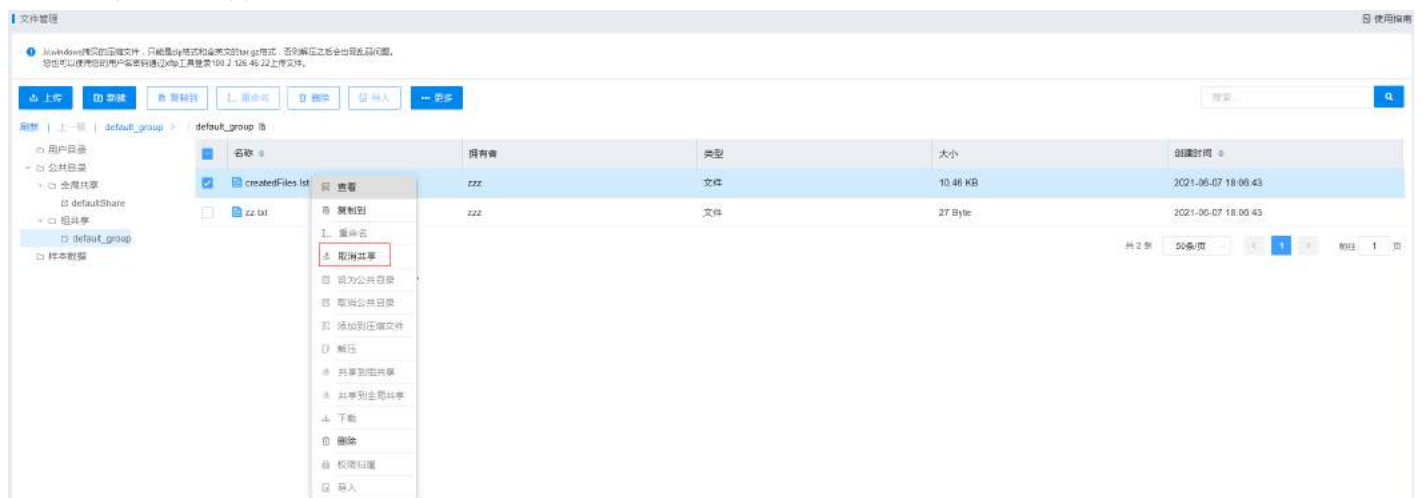
需要注意的是：新添加外置存储下的目录无法进行共享



2. 若只有一个全局共享目录或组共享目录，文件直接共享到该目录下，不必选择目标文件夹。当全局共享或组共享的目录多于一个时，弹出对话框，选择目标文件夹，文件的处理进度显示在右侧进度列表中。可通过进度列表的目录链接，进入文件列表。

取消共享

1. 支持单个和批量取消共享，在全局共享或组共享目录下，选择要取消共享的文件或文件夹，这里只能取消自己共享后的文件，不能取消其他人的共享文件。右键菜单，点击【取消共享】或者点击列表上方快捷菜单【删除】



2. 弹出确定取消共享界面，点击【确定】，文件处理列表中显示取消共享文件的进度。取消共享完成后，

页面自动刷新。

搜索

1. 在搜索框中输入要搜索的关键字，回车或点击搜索按钮，在当前目录下全局搜索。



2. 当搜索结果较多时，可以拖动右侧滚动条，进行展示。

进度操作

1. 进入文件处理列表的操作包括：复制、共享、删除、取消共享、压缩、解压缩、上传。同一用户同一操作只有一个在处理，其余需要排队。

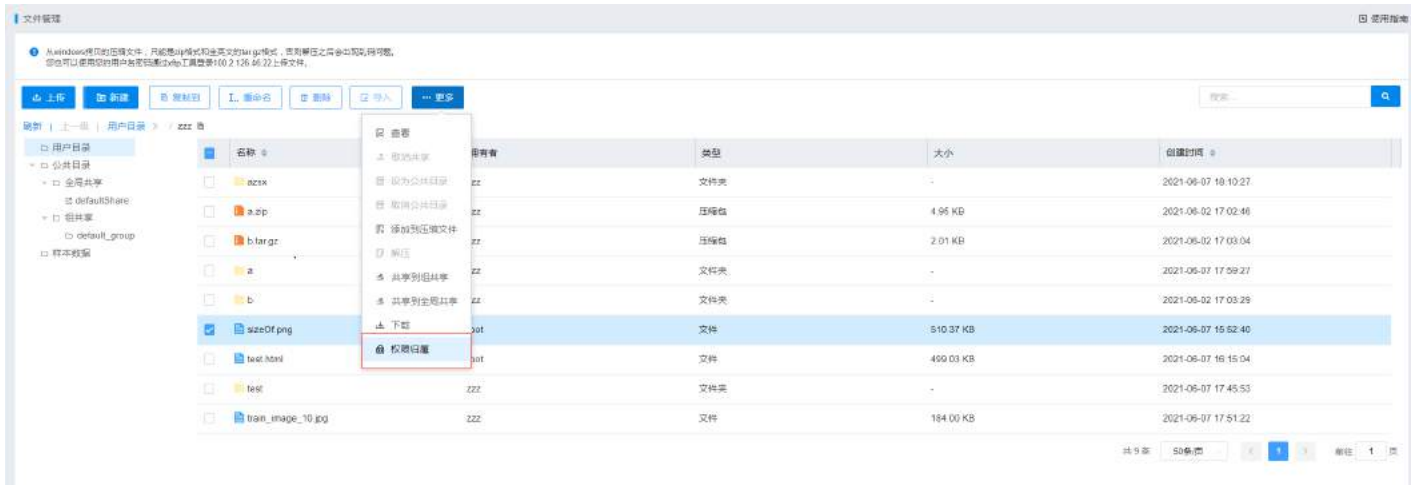


2. 文件处理进度列表可以进行查看、删除、最小化、关闭等操作。最小化后，文件处理进度列表显示为悬浮窗，可以正常切换模块，例如开发环境、训练任务等。可以删除处理中（显示为百分比），排队中，失败，完成的进度。当关闭文件处理进度列表时，需二次确认，关闭后，清除所有完成的任务，如果存在文件处理任务则显示为悬浮窗。

用户目录权限归属

只有普通用户可以进行用户目录权限归属操作，只能操作自己用户目录下的文件和文件夹。

选择一个或多个用户目录下的文件和文件夹，点击列表上方快捷菜单【更多】选择【权限归属】操作，弹出二次确认页面，点击确定，可以将选中文件的权限归属为当前用户。



完成上述操作后，文件列表的拥有者显示为当前用户。

用户目录导入

新增外置存储后，对于平台之前已存在的用户，如果该存储上用户家目录不存在，可以通过用户目录导入在新增外置存储上创建用户家目录。对于已存在用户家目录的存储，该按钮置灰。

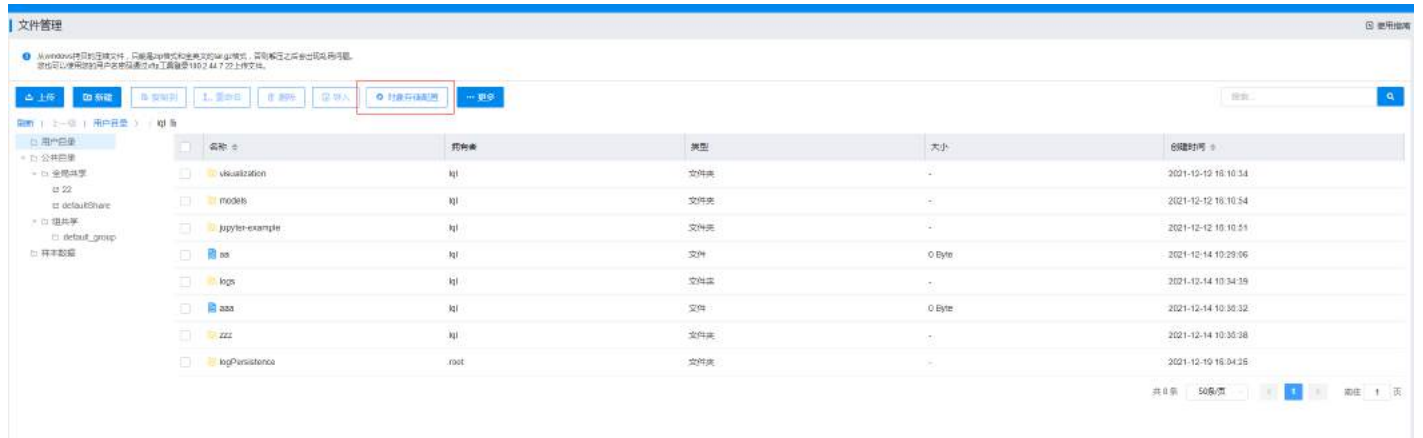


导入成功后，自动刷新，进入该用户家目录下



对象存储配置

对接对象存储配置时，才会有此功能。点击【配置对象存储按钮】，填入配置项信息。每个用户只能配置一个，可以多次配置，以最新配置的为准。不对配置项做校验，正确展示文件列表，错误，提示配置信息问题



数据集管理

适用存储

数据集管理只适用于共享存储 (NFS、Beegfs、Lustre)。对象存储 (Ceph、HDFS) 等不支持使用数据集管理功能

创建数据集

1. 点击列表右上方【创建数据集】按钮，可以创建一个数据集，必填项为名称、导入数据路径、数据类型。数据集名称由系统自动生成成为 dataset-后加一个 4 位随机字符，也可以自定义修改。导入数据路径选择文件管理中的样本数据，数据类型包含图片，文本，音频，视频，其他，描述信息可不填，如果填写需不超过 256 个字符。创建数据集时，会自动创建一个 V001 的版本，该版本来源于选择的导入数据集路径。每个用户最多可以创建 100 个数据集。

创建数据集
✕

*** 名称**

*** 导入数据路径** 📁

*** 数据类型** 图片 音频 文本 视频 其他

描述0/256

取消
确定

选择数据
✕

上一级 当前路径

- 📁 用户目录
- 📁 公共目录
- 📁 全局共享
 - 📁 defaultShare
 - 📁 组共享
 - 📁 zyhGroup
- 📁 样本数据

☑	名称 ↕	拥有者	类型	大小	创建时间 ↕
☑	📁 visualization	zyhgrp	文件夹	-	2021-12-17 17:3...
☐	📁 models	zyhgrp	文件夹	-	2021-12-17 16:5...
☐	📁 jupyter-example	zyhgrp	文件夹	-	2021-12-17 16:5...

取消
确定

创建版本

支持修改数据的处理方式，您可以点击创建版本，在该数据集下创建一个新的版本，数据集来源可以选择原始数据集或者已发布的数据集版本。创建完成后，可以对每个版本进行文件增删改操作，并发布使用。发布后的数据集版本不能再进行文件增删改操作。如果想再次进行文件操作，必须创建一个新的数据集版本。创建版本时，数据集名称自动带入，不可修改，来源可以选择导入的数据路径或者已发布的版本。版本由系统自动生成，按顺序填充，从 V001 累加到 V999。版本名称可以自定义修改，不可重名。版本名称只能包含数字字母下划线连接线，不能以下划线和连接线开头，最多 32 个字符。创建的版本会自动进入进度列表中，进度列表显示版本、数据集、操作类型、存储名称、状态、操作等。

创建版本 ×

* 名称	dataset-6sw9
* 来源	请选择
* 版本	<div style="border: 1px solid #ccc; padding: 5px;"><div style="background-color: #e0f0ff; padding: 2px;">/MNIST_pytorch</div><div style="padding: 2px;">V001</div><div style="padding: 2px;">V002</div><div style="padding: 2px;">V003</div></div>
描述	<div style="border: 1px solid #ccc; height: 40px;"></div> <div style="text-align: right; font-size: small;">0/256</div>

取消 确定

修改数据（增删改文件）

未发布的版本可以进行修改数据，操作类型包括：导入、删除、编辑文件。1. 导入数据可以选择文件管理中的文件，不能导入创建数据集时选择的原始数据。如果导入重名文件，提示是否覆盖。点击版本名称进入文件列表页面，点击右上方的【导入】按钮，弹出文件管理页面，选择要导入的数据，点击确定。导入的数据会自动进入传输列表中，进度列表显示导入的文件名、类型、操作类型、大小、存储名称、进度值、操作等。

版本 | 数据管理 | 使用指南

上一级 当前路径: data001-noy6/V024

显示 导入 删除 返回列表

名称	类型	大小	存储时间
<input type="checkbox"/> train001.txt	文件	1.27 KB	2021-12-15 15:07:05
<input type="checkbox"/> UJADUtils.java	文件	1.70 KB	2021-12-15 15:08:35
<input type="checkbox"/> 110k-images-1024-ubyte	文件	7.48 MB	2021-09-28 16:11:38
<input type="checkbox"/> 110k-labels-1024-ubyte	文件	9.77 KB	2021-09-28 16:11:35
<input type="checkbox"/> train-images-1024-ubyte	文件	44.86 MB	2021-09-28 16:11:41
<input type="checkbox"/> train-labels-1024-ubyte	文件	58.60 KB	2021-09-28 16:11:42
<input type="checkbox"/> 110k-images-1024-ubyte.gz	文件	1.57 MB	2021-09-28 16:11:43
<input type="checkbox"/> 110k-labels-1024-ubyte.gz	文件	4.44 KB	2021-09-28 16:11:44
<input type="checkbox"/> train-images-1024-ubyte.gz	文件	9.45 MB	2021-09-28 16:11:45
<input type="checkbox"/> train-labels-1024-ubyte.gz	文件	28.20 KB	2021-09-28 16:11:45
<input type="checkbox"/> msvc	文件夹	-	2021-09-28 16:11:54
<input type="checkbox"/> logPersistence	文件夹	-	2021-12-04 01:00:03

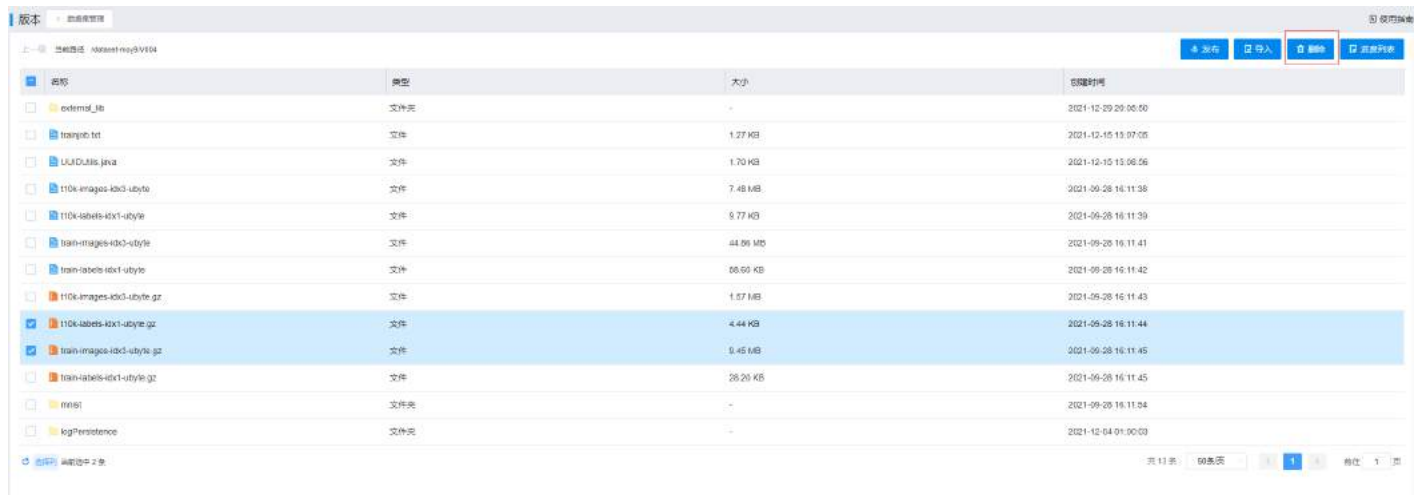
共 17 条 | 刷新 | 1 | 前往 | 1 | 页

进度列表

<input type="checkbox"/>	文件(夹)名	类型	操作类型	大小	存储名称	状态	操作
<input type="checkbox"/>	www.tar.gz	压缩包	导入	690.40 MB	master	0%	🗑

当前选中 0 条 | 共 1 条 | 50条/页 | 1 | 前往 | 1 | 页

2. 删除数据时，选中文件列表中要删除的文件，点击右上角的【删除】按钮即可。



3. 编辑文件时，点击文件名称打开可编辑的文件，输入修改内容，点击确定即可。

编辑文件

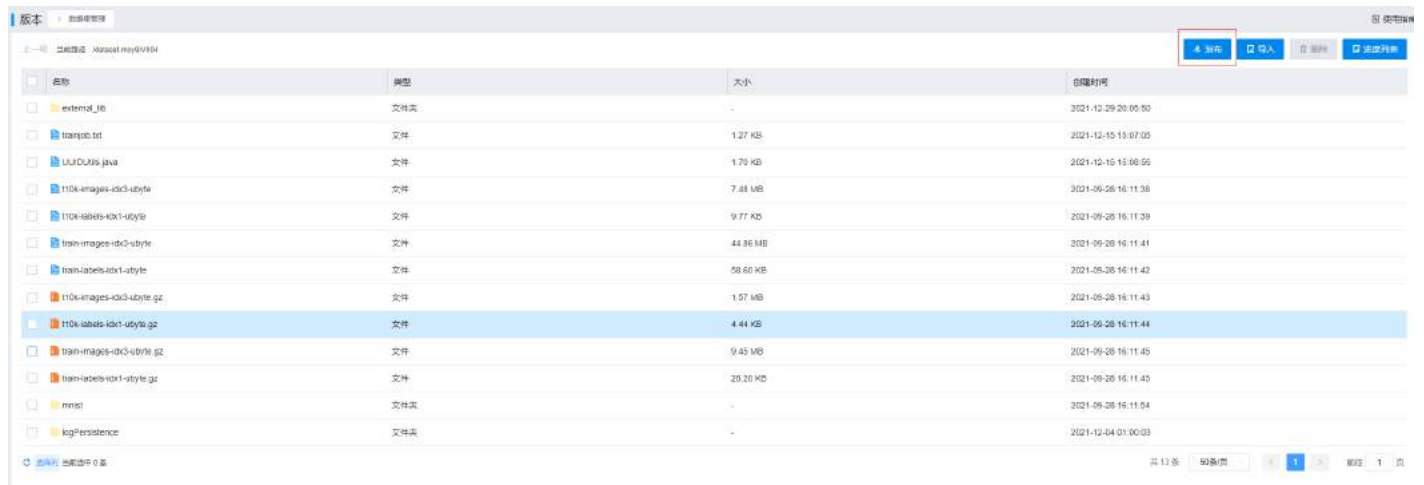


取消 确定

版本发布

未发布的数据集，可以点击操作栏的发布按钮进行发布。也可以在该版本的文件列表页面点击【发布】按钮操作。





版本视图

发布的数据集版本可以进行版本视图查看，追溯来源过程。



版本删除

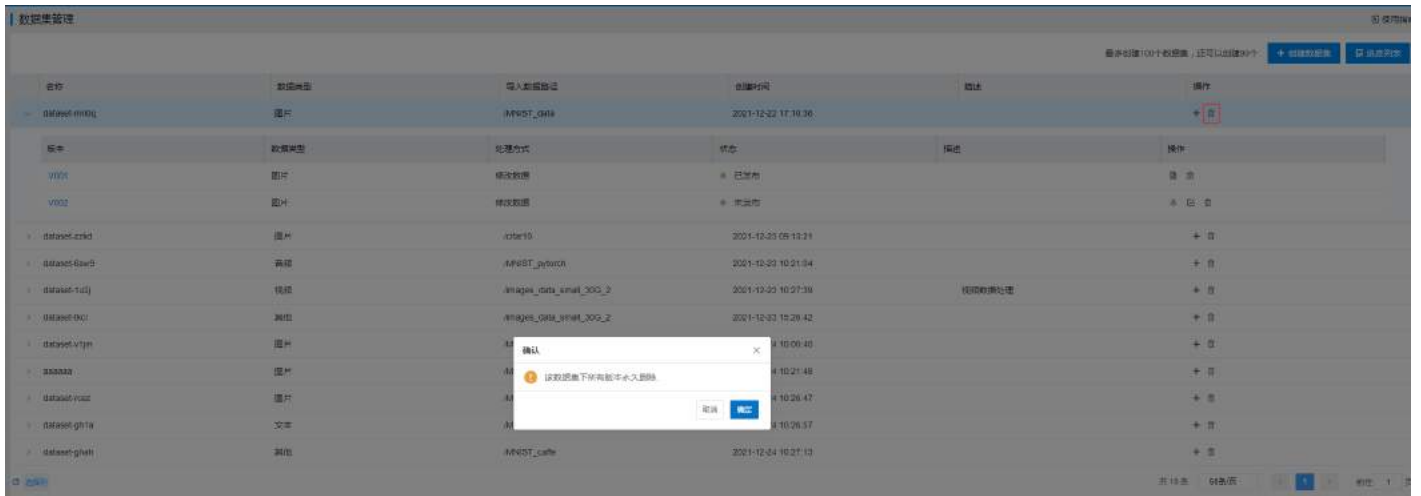
普通用户可以对发布和未发布的数据集版本进行删除。发布的数据集版本删除为逻辑删除，底层和数据库保留信息，只是将该版本标记为删除状态，便于其他模块（如开发环境、训练任务等）使用该数据集版本时能够追溯到使用的文件。点击操作栏的操作按钮执行删除操作。

名称	数据类型	导入数据路径	创建时间	描述	操作
dataset-mn0g	图片	/MNIST_data	2021-12-22 17:10:36		+ 目
dataset-zzhd	图片	/cifar10	2021-12-23 09:13:21		+ 目
dataset-6a9d	音频	/MNIST_pytorcs	2021-12-23 10:21:04		+ 目
dataset-1v2j	视频	/images_data_small_30G_2	2021-12-23 10:27:39	视频数据处理	+ 目

版本	数据类型	处理方式	状态	描述	操作
V001	视频	修改数据	* 未发布	视频数据处理 V001	+ 目
V002	视频	修改数据	* 已发布	添加2017年的视频数据	+ 目
V003	视频	修改数据	* 未发布	视频数据处理 V002	+ 目

数据集删除

点击数据集操作栏的删除按钮进行数据集删除，该操作会将数据集下的所有版本永久删除，无法恢复。



文件列表展示

点击版本名称，展示该版本下所有的文件列表。

名称	类型	大小	创建时间
110k-images-idx3-ubyte	文件	7.48 MB	2021-09-28 16:11:38
110k-labels-idx1-ubyte	文件	5.77 KB	2021-09-28 16:11:39
train-images-idx3-ubyte	文件	44.86 MB	2021-09-28 16:11:41
train-labels-idx1-ubyte	文件	58.60 KB	2021-09-28 16:11:42
110k-images-idx3-ubyte.gz	文件	1.57 MB	2021-09-28 16:11:43
110k-labels-idx1-ubyte.gz	文件	4.44 KB	2021-09-28 16:11:44
train-images-idx3-ubyte.gz	文件	0.45 MB	2021-09-28 16:11:45
train-labels-idx1-ubyte.gz	文件	28.20 KB	2021-09-28 16:11:46
mnist	文件夹	-	2021-09-28 16:11:54
log/persistence	文件夹	-	2021-12-04 01:00:03

文件查看

可以查看普通文件和图片，支持查看 50M 以下的文件。点击文件名称打开即可。

发布数据集使用

发布数据集只支持节点缓存，不支持更新。以开发环境为例，进行说明。选择数据时，选择数据集管理，弹出所有数据集列表，展开可以显示所有已发布的版本，只能选择其中的一个版本，点击确定。

The screenshot shows the '创建开发环境' (Create Development Environment) interface. On the left, there are configuration options for environment ID, image, network, acceleration type, CPU, and GPU. On the right, there are resource allocation options for user and personal environments, and a table of available nodes.

节点	CPU核数	加速卡数	GPU型号	加速卡型号
node1	7&154	1/2	+	NVIDIA-A30

Below the configuration is a '选择数据' (Select Data) dialog box. It displays a list of datasets with columns for name, data type, import path, creation time, and description. The 'dataset-1u2j' dataset is selected.

名称	数据类型	导入数据路径	创建时间	描述
dataset-mn0q	图片	/MNIST_data	2021-12-22 17:10:36	
dataset-zzkd	图片	/cifar10	2021-12-23 09:13:21	
dataset-6sw9	音频	/images_data_small_30G_2	2021-12-23 10:21:04	
dataset-1u2j	视频	/images_data_small...	2021-12-23 10:27:39	视频数据处理
dataset-tkci	其他	/images_data_small...	2021-12-23 15:26:42	

Below the dataset list, there is a pagination control showing '共 10 条' (Total 10 items), '5条/页' (5 items per page), and page numbers 1 and 2. The '1' button is highlighted. At the bottom right, there are '取消' (Cancel) and '确定' (Confirm) buttons.

传输列表操作

1. 进入传输列表的操作包括：创建版本和导入。同一用户同一操作只有一个在处理，其余需要排队。
2. 传输列表可以进行查看、删除、关闭等操作。

开发环境

开发环境列表

点击左侧导航栏中的开发环境，打开开发环境列表页面，页面顶部展示开发环境汇总信息，列表中展示开发环境的基本信息，列表右侧展示相关操作按钮，同时可以根据所属工程等进行筛选，可以选中多个开发环境点击右上角删除按钮进行批量删除，如下：

环境名称	状态	运行时长	剩余时长	资源配置	节点	镜像	所属工程	类型	创建时间	操作
20211228154653	运行中	22时 47分	-22时 46分	GPU:0 CPU:1	node1(0,-)	100.2.44.80:5000/c...	-	开发环境	2021-12-28 15:46:08	[操作按钮]
20211228154646	运行中	22时 47分	-22时 46分	GPU:0 CPU:1	node1(0,-)	100.2.44.80:5000/c...	-	开发环境	2021-12-28 15:46:03	[操作按钮]

点击顶部密码设置按钮设置开发环境 shell 密码，支持随机密码 (默认)、固定密码，如下

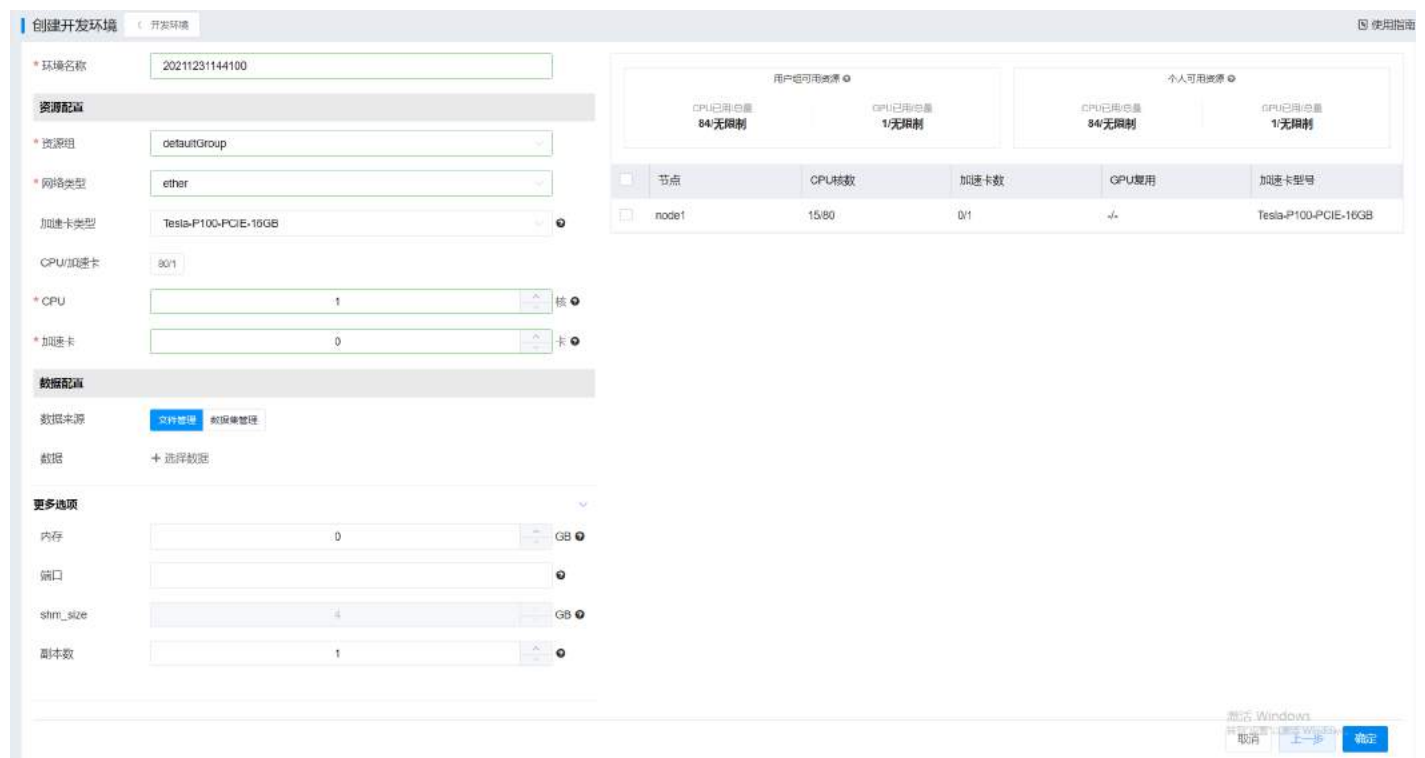
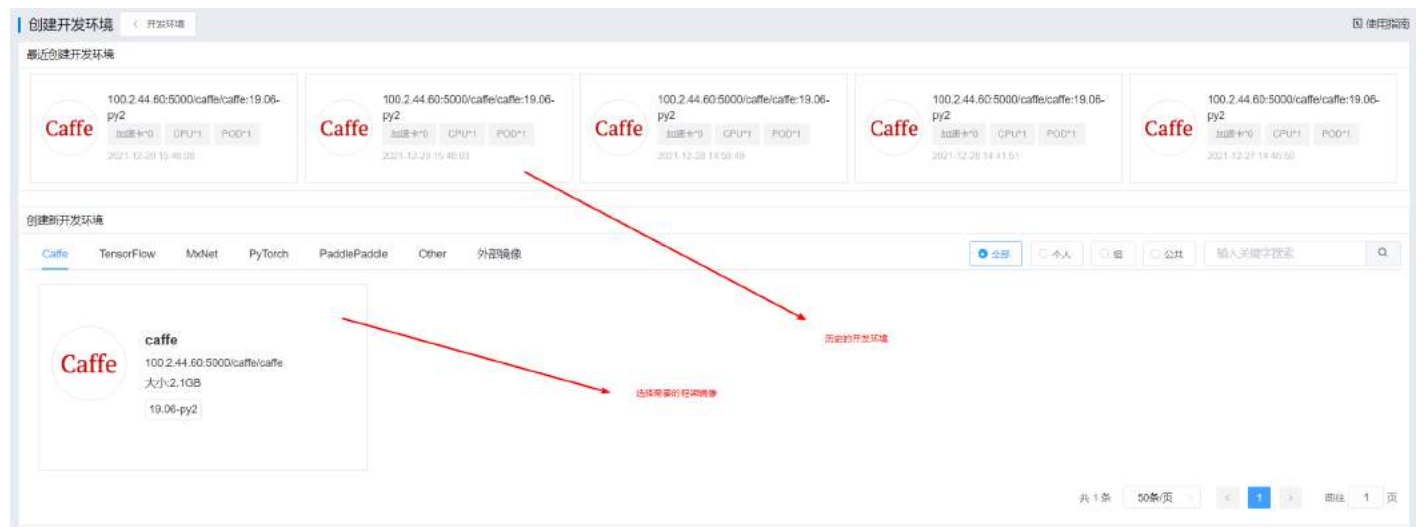
密码设置

随机密码

自定义密码

创建开发环境

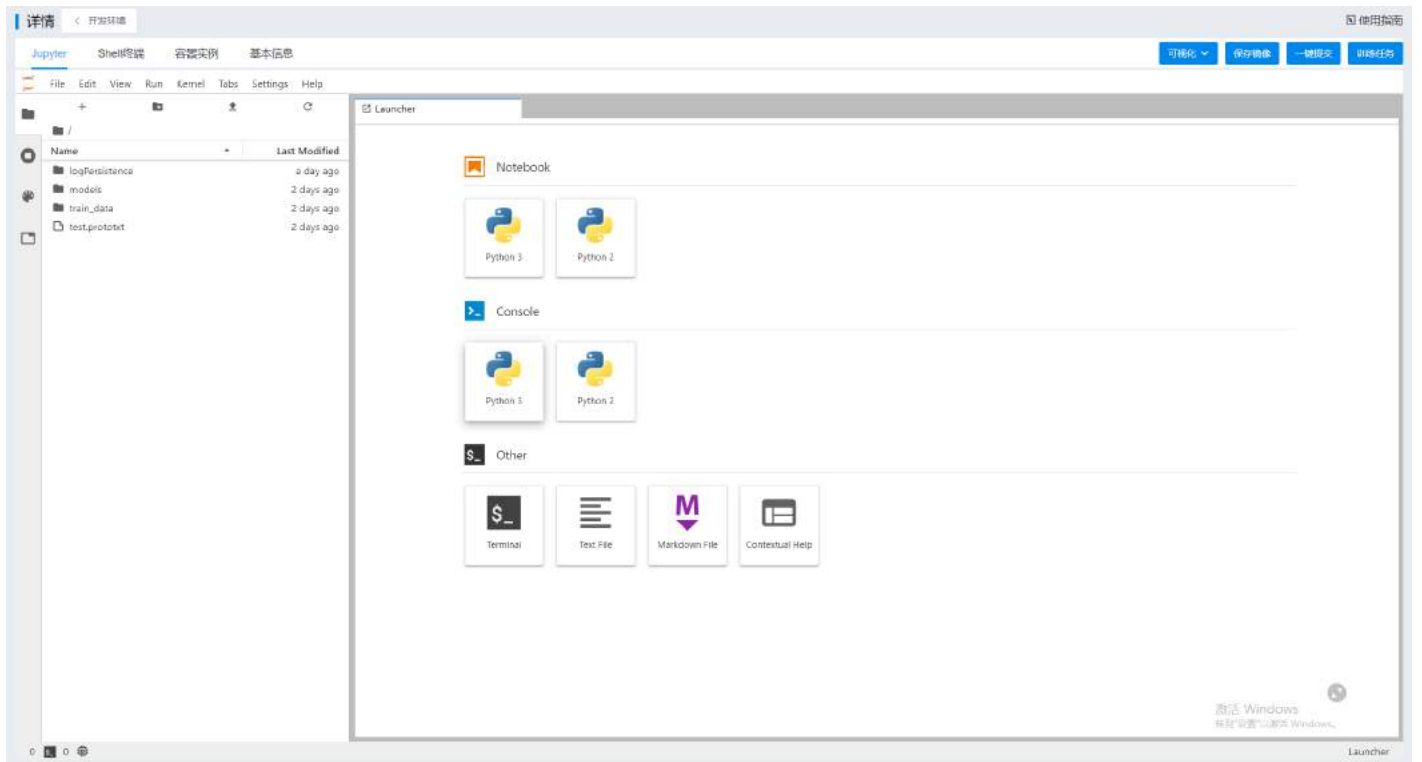
用户点击页面中的创建按钮创建开发环境，支持从头创建开发环境 (目前支持 Caffe、TensorFlow、Mxnet、Pytorch、PaddlePaddle 和 other 框架)，也支持基于历史的开发环境创建，用户可以选择资源组、资源配置、数据等信息，同时也可以指定节点，如下图：



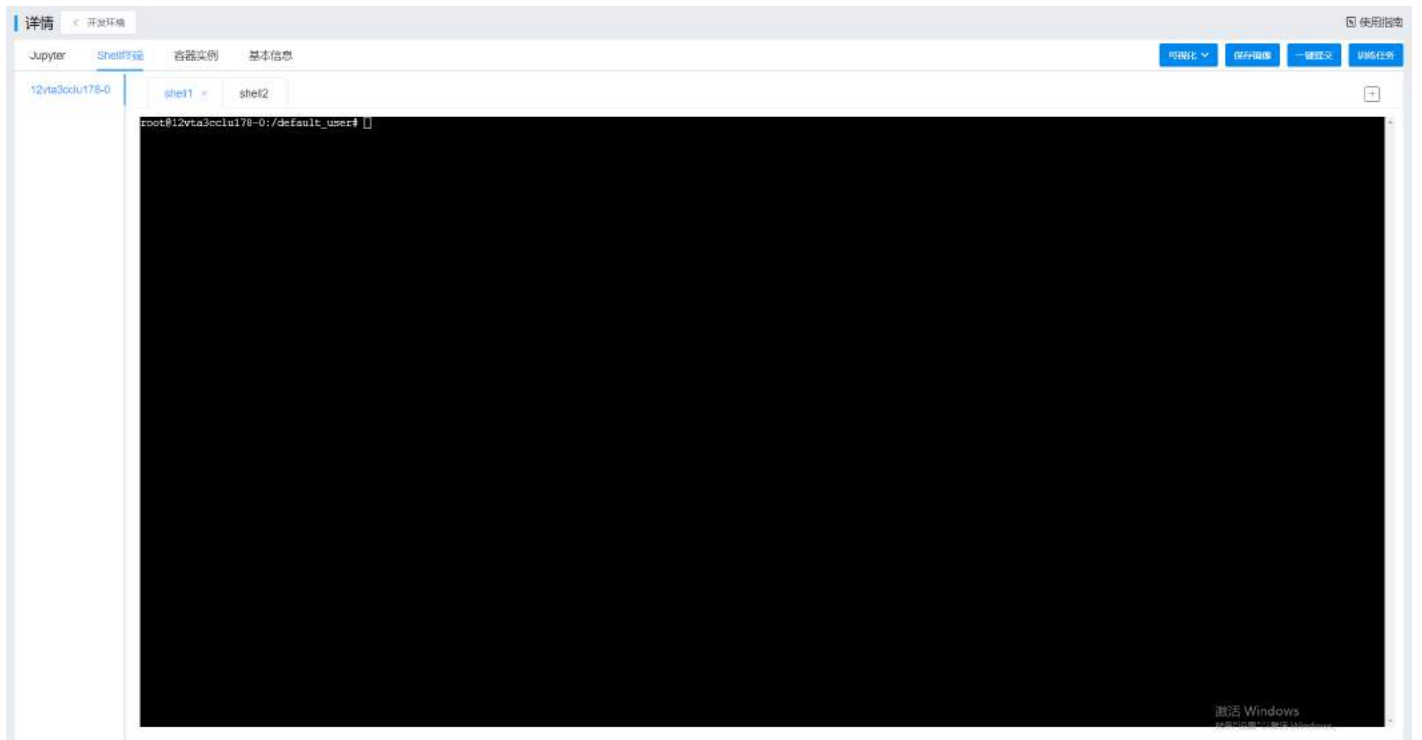
开发环境详情

用户在开发环境列表中，点击运行中的开发环境的名称，可以进入该开发环境的详情中，在详情页中，可以使用 Jupyter 对脚本进行编辑调试，可以使用 web shell 终端连接环境 (支持多窗口)，可以在容器实例中查看容器实例列表、性能监控，可以查看基本信息，同时也支持可视化 (目前支持 TensorBoard、Visdom、Netscope)、将当前开发环境保存为镜像，一键提交训练任务，查看训练任务等功能，如下图：

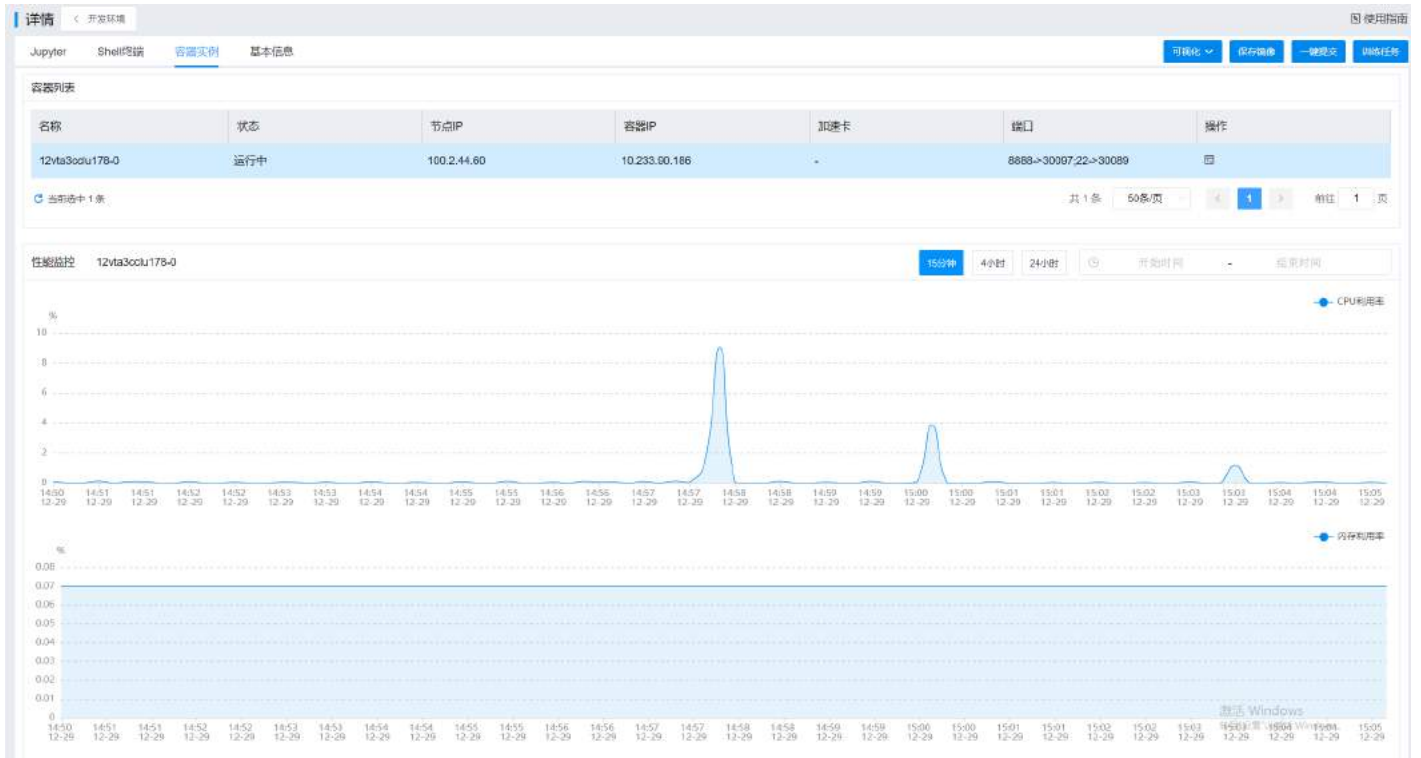
Jupyter:



Shell 终端:



容器实例:

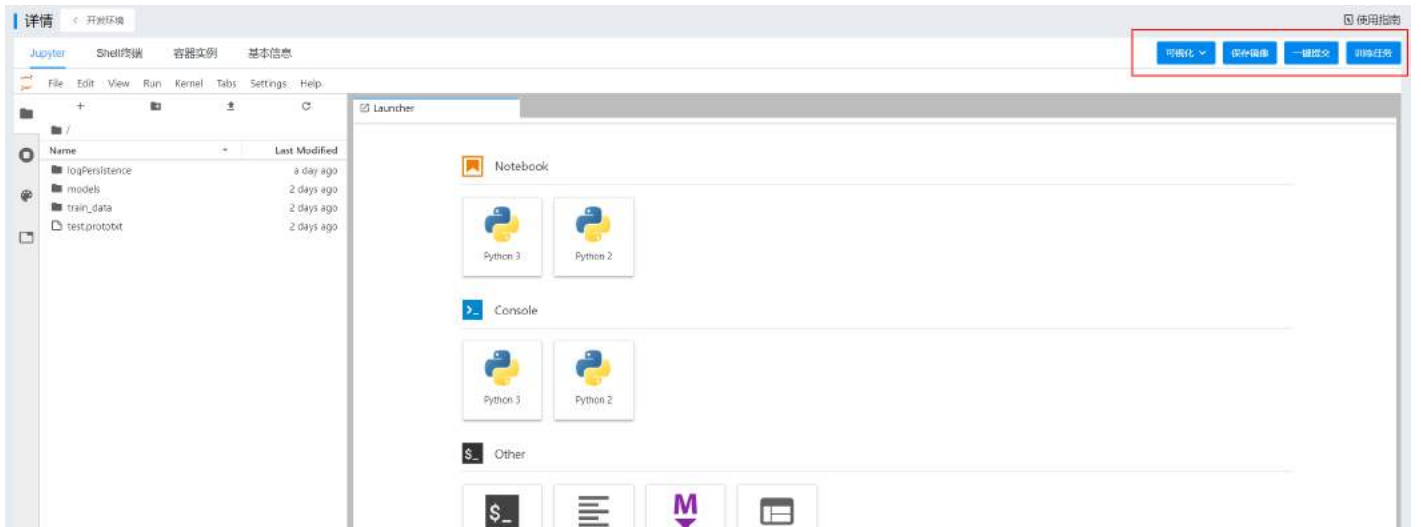


基本信息:

The screenshot shows the '基本信息' (Basic Information) tab for the container instance. It contains a table with the following details:

环境名称	20211228154653	创建时间	2021-12-28 15:46:08	状态	运行中
镜像	100.2.44.60:5000/caffe/caffe:19.06-py2				
资源组	defaultGroup	加速卡	GPU.0	CPU	1
副本数	1	shm_size	4GB	数据路径	

其他功能，包括可视化、保存镜像、一键提交、训练任务



开发环境操作功能

用户在列表中的操作栏中，可以对开发环境进行操作，目前支持以下操作：

SSH： 点击查看开发环境的 ssh 配置

克隆： 克隆一个相同配置的开发环境

启动： 将停止的开发环境启动

暂停、恢复： 用户可以将一个运行中的开发环境暂停 (暂停后底层数据会删除)，点击恢复后可以重新运行该开发环境

删除： 可以删除开发环境

资源调整： 可以对运行中的开发环境进行动态资源调整，修改资源配置



环境名称	状态	运行时长	剩余时长	资源配置	节点	镜像	所属工程	类型	创建时间	操作
20211228154653	运行中	23时 32分	-23时 31分	GPU:0 CPU:1	node1(0,-,)	100.2.44.60:5000/c...	-	开发环境	2021-12-28 15:46:08	ssh, 克隆, 启动, 暂停/恢复, 删除
20211228154648	运行中	23时 32分	-23时 31分	GPU:0 CPU:1	node1(0,-,)	100.2.44.60:5000/c...	-	开发环境	2021-12-28 15:46:03	ssh, 克隆, 启动, 暂停/恢复, 删除
20211228154653	已暂停	23时 34分	-23时 33分	GPU:0 CPU:1	node1(0,-,)	100.2.44.60:5000/c...	-	开发环境	2021-12-28 15:46:08	ssh, 克隆, 启动, 暂停/恢复, 删除
20211228154648	运行中	23时 34分	-23时 33分	GPU:0 CPU:1	node1(0,-,)	100.2.44.60:5000/c...	-	开发环境	2021-12-28 15:46:03	ssh, 克隆, 启动, 暂停/恢复, 删除

任务管理

创建训练任务

本手册以 tensorflow 单机任务为例子进行说明。功能说明：用户通过平台提供的训练任务功能，能够自动创建一个新的训练任务，创建成功后自动在任务管理列表展示该任务。操作步骤：**A：** 进入任务管理模块，点击页面“创建”按钮，弹出填写任务信息页面，如下图：



任务名称	状态	运行时长	节点	资源配置	镜像	启动内容	部署类型	镜像	所属工程	任务类型	提交时间	紧急任务	操作
20211227144136	运行中	11分 33秒	node1(0,-,)	Cards:0, CPU:1, ...	caffe	sleep 1d	单机	100.2.44.60:5000/...	-	训练任务	2021-12-28 14:5...	否	删除, 暂停, 启动

B： 填写任务详细信息：

名称： 任务名字（只接受英文字母、数字和下划线，不能以下划线开头）。

镜像： 在第一个窗口选择 tensorflow 框架名称，在第二个窗口选择框架版本。

外部镜像：勾选该选项后，用户可以自定义输入镜像名称。

资源组：选择资源组。

加速卡类型：选择资源组内相应的加速卡类型。

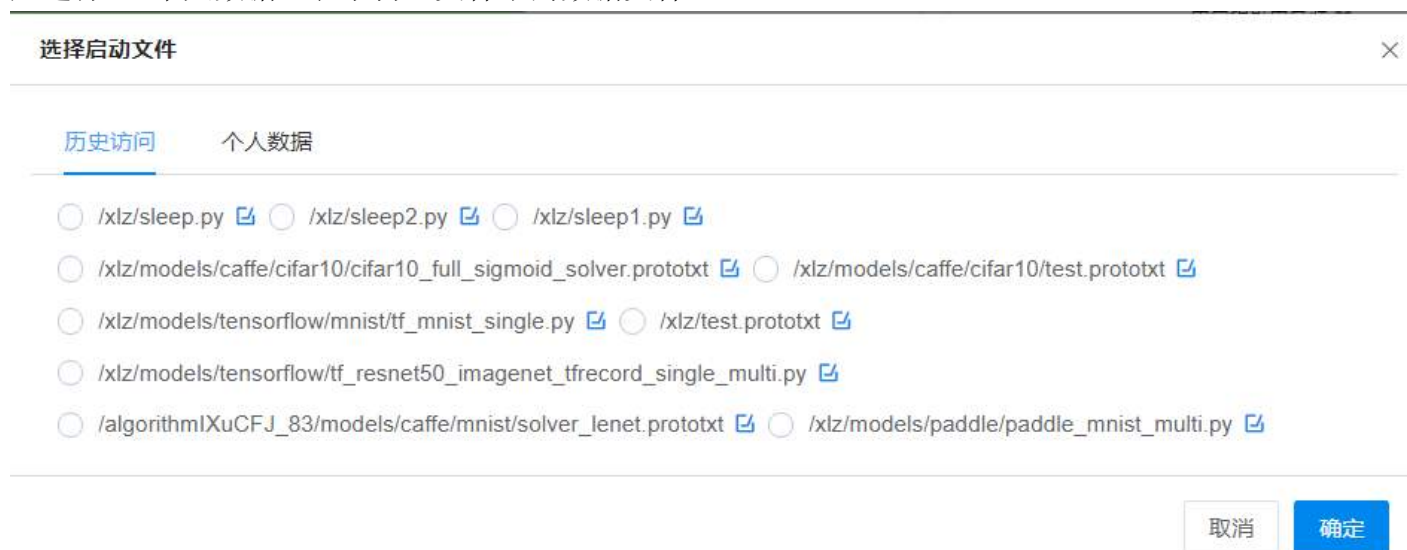
CPU/加速卡：选择 worker 节点的 CPU/加速卡资源配置方案，当配额方案是“自定义”时，会弹出加速卡和 CPU 窗口，可以自定义设置资源配置方案。

py 脚本：点击窗口后第一个按钮，弹出“选择启动文件”窗口，选择 tensorflow 单机训练脚本，

脚本示例路径：/xlz/models/tensorflow/mnist/tf_mnist_single.py

说明：xlz 这个表示用户的家目录，最终以实际的用户名为准。选定后点击确定；

点击第二个按钮，会弹出“选择启动文件标签”窗口，选定后点击确定。“选择启动文件”中有两个子选项，分别为“历史访问”、“个人数据”。“历史访问”表示以前使用过的启动脚本，展示在此处以便用户选择。“个人数据”表示自己文件中的数据文件。



选择启动文件
✕

历史访问
个人数据

当前路径 ↗

<input type="checkbox"/>	名称	拥有者	类型	大小	创建时间
<input type="checkbox"/>	app.py	xlz	文件	0 Byte	2021-12-13 20:39:34
<input type="checkbox"/>	sleep.py	xlz	文件	71 Byte	2021-12-23 14:20:34
<input type="checkbox"/>	sleep2.py	xlz	文件	73 Byte	2021-12-29 16:29:19
<input type="checkbox"/>	models	root	文件夹	-	2021-11-24 10:38:48
<input type="checkbox"/>	logPersistence	root	文件夹	-	2021-12-30 09:40:35
<input type="checkbox"/>	test11	root	文件夹	-	2021-12-22 11:02:28
<input type="checkbox"/>	testrae22	root	文件夹	-	2021-12-27 15:30:09
<input type="checkbox"/>	test111	root	文件夹	-	2021-12-27 16:06:36

共 8 条 50条/页
< 1 > 前往 1 页

取消 确定

命令：用户点击下图中方框中的按钮会切换到命令行模式，可以自定义自己的启动命令。

启动配置

* py脚本 命令模式

* 执行目录

启动配置

* 命令模式 脚本模式

执行目录：选择执行训练脚本的目录，执行目录可以选择自己目录下的任何文件夹。

脚本参数：在“脚本参数”输入框可以输入 python 脚本所跟随的参数，例如“-data_dir /MNIST_data -data_dir2 /MNIST_data2”

数据配置：点击数据配置右侧的选择数据集，提供 2 种数据配置方式：



The screenshot shows a web interface for data configuration. At the top, there is a grey header with the text "数据配置". Below the header, there are two tabs: "文件管理" (File Management) and "数据集管理" (Dataset Management). The "文件管理" tab is currently selected and highlighted in blue. Below the tabs, there is a label "数据" (Data) and an empty input field with a folder icon on the right side.

1. 文件管理

在弹出的路径选择窗口中选择要使用的数据集。支持选择多个数据集。

选择数据 ×

平台可用数据列表,用于训练的输入数据

当前路径

已选择

- 用户目录
- 公共目录
 - 全局共享
 - defaultShare
 - 组共享
 - default_group
 - 样本数据

<input type="checkbox"/>	名称	类型	大小	创建时间
<input type="checkbox"/>	MNIST_data	文件夹	-	2021-11-30 19:54:45
<input type="checkbox"/>	MNIST_caffe	文件夹	-	2021-12-14 10:54:06
<input type="checkbox"/>	MNIST_pytorch	文件夹	-	2021-11-30 19:16:17
<input type="checkbox"/>	cifar10	文件夹	-	2021-09-28 16:12:34
<input type="checkbox"/>	cifar10_caffe	文件夹	-	2021-09-28 16:13:01
<input type="checkbox"/>	images_data_small...	文件夹	-	2021-09-29 15:40:59
<input type="checkbox"/>	images_data_small...	文件夹	-	2021-09-29 15:45:20

共 7 条 前往 页

2. 数据集管理

在数据集列表中选择可用数据集

选择数据
✕

	名称	数据类型	导入数据路径	创建时间	描述
▼	dataset-bpou	图片	/MNIST_data	2021-12-10 09:31:00	

版本	数据类型	处理方式	状态	描述
V001	图片	修改数据	● 已发布	

选择列
共 1 条
50条/页
< 1 >
前往 1 页

取消
确定

更新数据集说明：勾选后，平台自动会对缓存的数据集进行识别，如果部分数据集文件发生变化，平台会实现增量更新，如果缓存中没有数据集，会全量下载数据集。如果缓存中的数据集正在使用，则不能进行更新操作。

数据集使用方式说明：有“节点缓存”和“直接使用”两种方式。“节点缓存”表示将数据集缓存到节点，“直接使用”表示使用共享存储中的数据集。

注意：数据集也可以来自于用户目录、公共目录（全局共享和组共享），数据集可以选择多个。数据集示例路径：/MNIST_data

点击“更多配置”可以显示以下信息选项：

内存：配置训练任务 worker 节点所需要的内存，当设置为 0 时表示无限制（需要小于 worker 所在主机目前剩余内存量）

日志路径：训练日志输出路径，点击窗口后的按钮，选择相应路径后点击确定。

目录挂载：可供挂载的公共目录。

shm_size：容器 shm_size 参数，默认为 4GB。

部署类型：训练任务部署类型，选择“单机”。

在右侧区域显示资源组下的节点信息，这里可以自定义运行的节点，比如需要在 ainode53 上运行该任务，可以直接勾选，这样平台会默认调度到该节点上。节点列表中还可以看到每个节点上资源的情况。如果不选择节点，则平台会自动选择剩余资源满足的节点运行该任务。

C：信息填写完之后，点击“确定”按钮创建任务，任务展示在任务管理列表中。

停止训练任务

功能说明：用户通过平台提供的停止训练任务功能，能够停止一个正在运行的训练任务。 workflows 相关类型的任务不支持该操作。

操作步骤：A：进入任务管理模块，选中一个正在运行的训练任务，点击停止按钮，如下图：



B：页面显示停止成功表示该操作成功。

启动训练任务

功能说明：用户通过平台提供的启动训练任务功能，能够启动一个停止的训练任务。 workflows 相关类型的任务不支持该操作。

操作步骤：A：进入任务管理模块，选中一个停止的训练任务，点击“启动”按钮，如下图：



B：任务重新启动说明操作成功。

重新提交训练任务

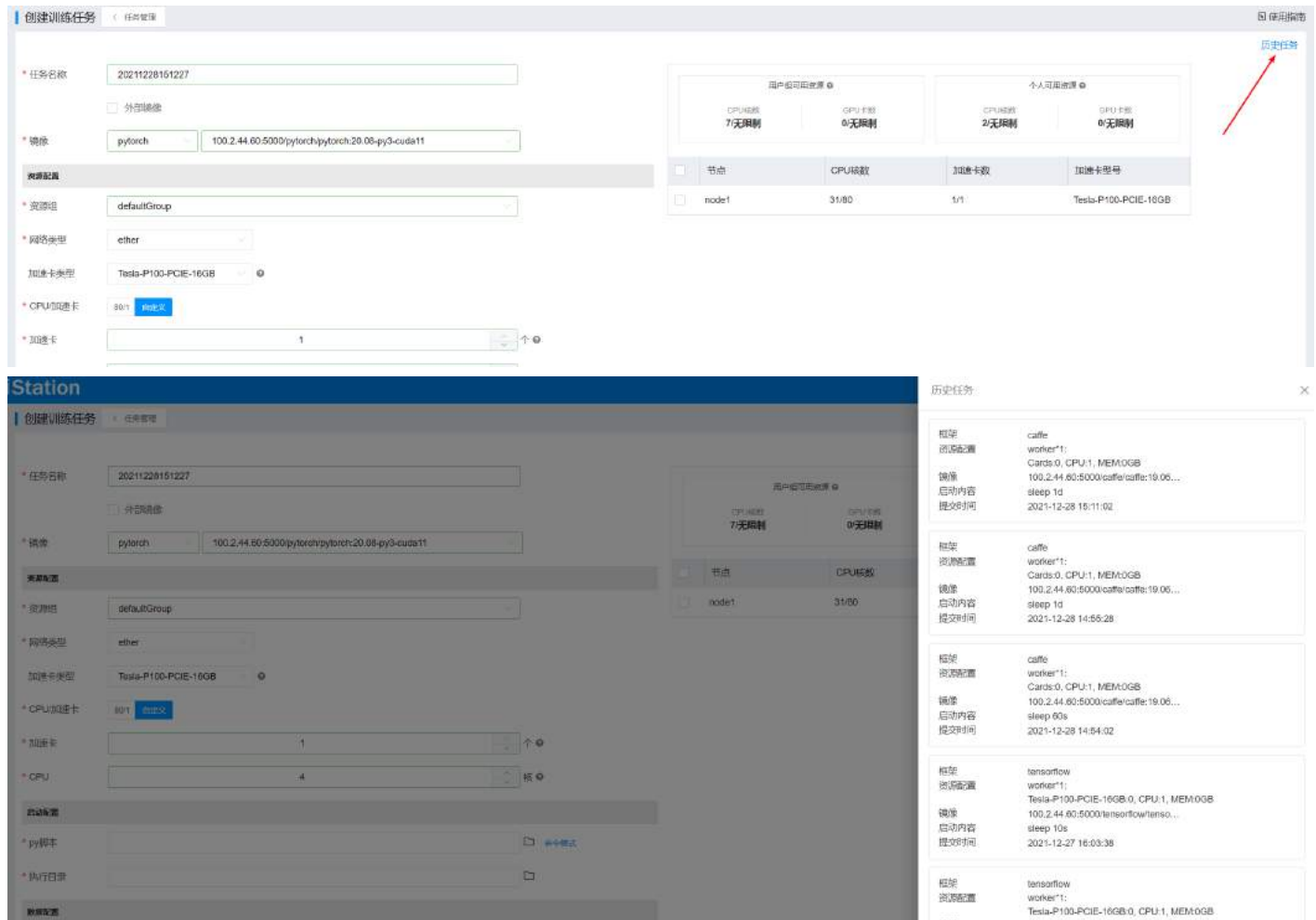
功能说明：用户通过平台提供的重新提交训练任务功能，能够重新提交一个训练任务。 workflows 相关类型的任务不支持该操作。

操作步骤：A：进入任务管理模块，选中一个训练任务，点击重新提交按钮，如下图：



通过历史记录提交训练任务

功能说明：用户通过平台提供的通过历史记录提交训练任务功能。操作步骤：**A**：进入任务管理模块，点击创建按钮，进入任务信息填写窗口，点击“历史记录”按钮，如下图：



B：选中相应的历史任务后，点击后将自动填充历史任务信息：框架类型、镜像、加速卡、CPU、内存、启动文件、数据集。

删除训练任务

功能说明：用户通过平台提供的删除训练任务功能，能够删除一个训练任务。 workflows 相关类型的任务不支持该操作。

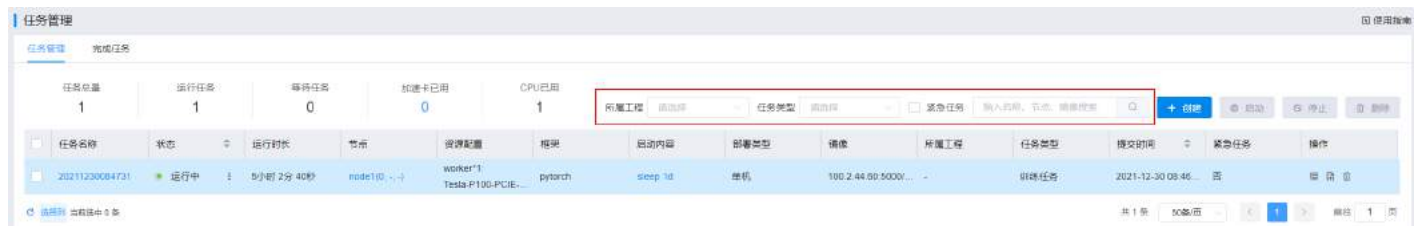
操作步骤：**A**：进入任务管理模块，选中一个训练任务，点击删除按钮，如下图：



B: 页面显示删除成功表示该操作成功。

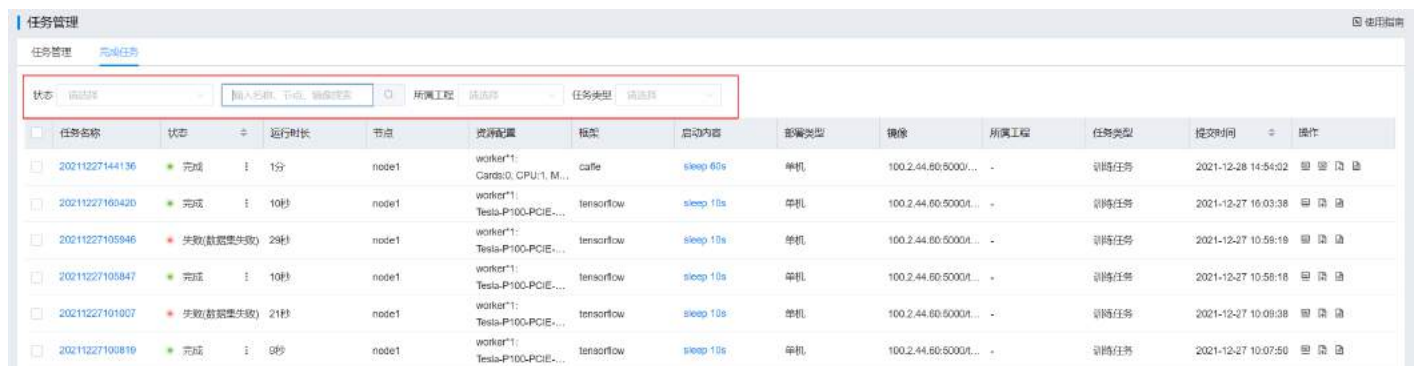
筛选未完成任务

功能说明：用户通过筛选功能筛选未完成的任務。操作步骤：A: 进入任务管理模块，点击未完成任务列表，通过筛选功能进行筛选，如下图：



筛选完成任务

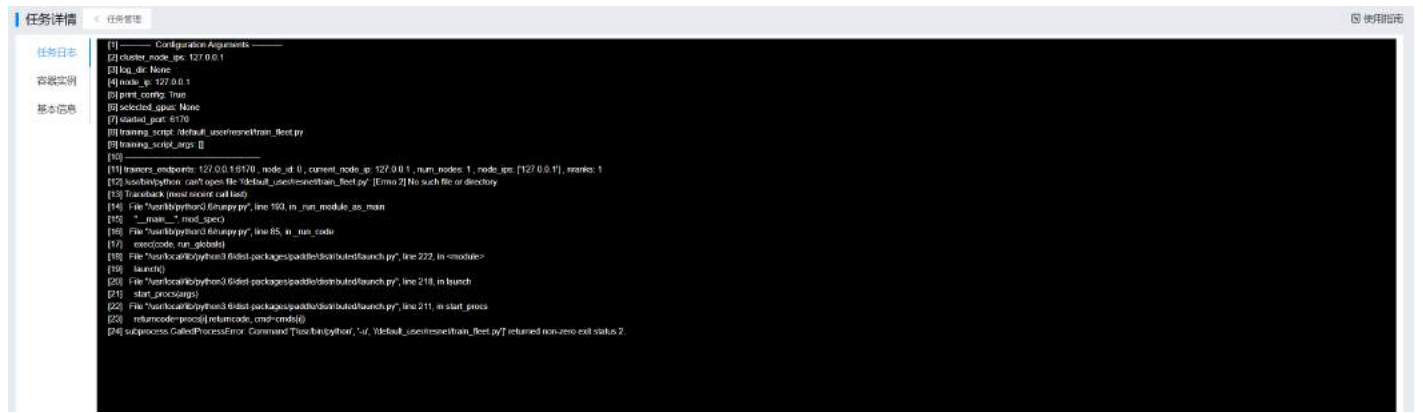
功能说明：用户通过筛选功能筛选完成的任務。操作步骤：A: 进入任务管理模块，点击完成任务列表，通过筛选功能进行筛选，如下图：



查看训练日志

功能说明：用户通过平台提供的查看任务日志功能，能够查看具体的训练日志。

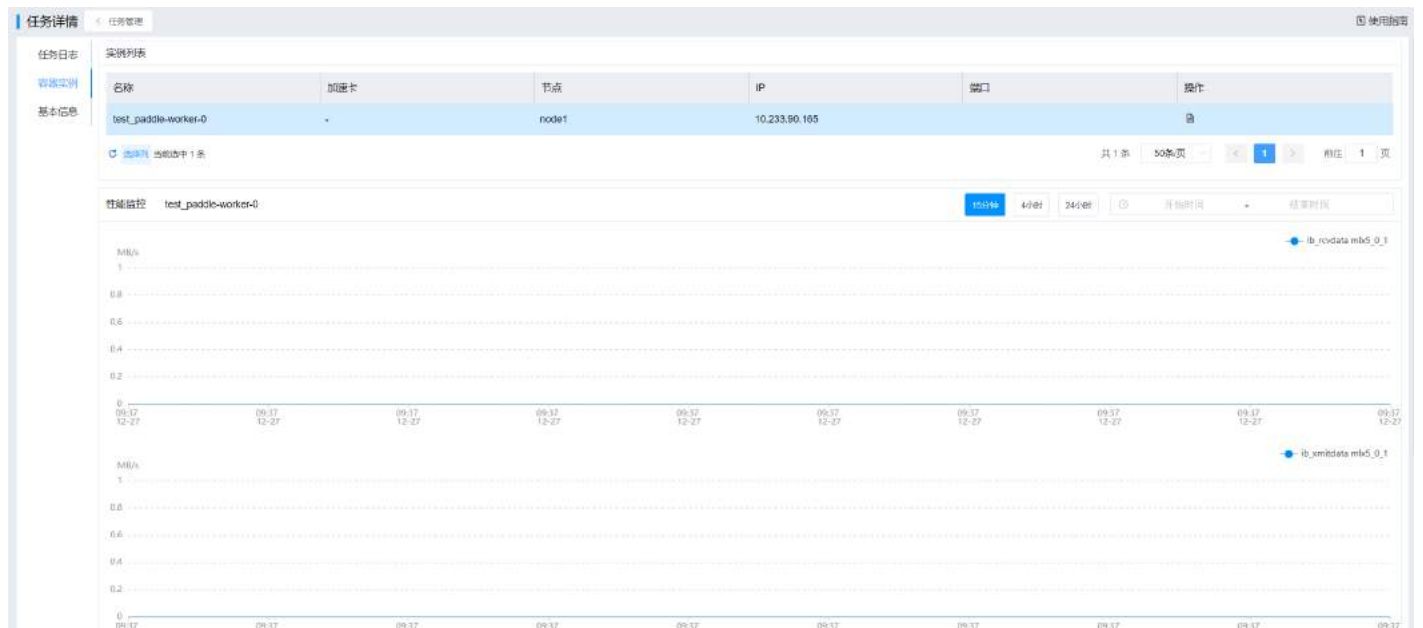
操作步骤：A: 进入任务管理模块，点击任务名字跳转到任务详情页面，点击“任务日志”按钮可以查看训练日志，如下图：



查看容器实例

功能说明：用户通过平台提供的查看任务容器实例功能，能够查看任务的容器实例信息和监控信息。

操作步骤：**A**：进入任务管理模块，点击任务名字跳转到任务详情页面，点击“容器实例”按钮可以查看容器实例信息，如下图：



查看任务基本信息

功能说明：用户通过平台提供的查看任务基本功能，能够查看任务的基本信息。

操作步骤：**A**：进入任务管理模块，点击任务名字跳转到任务详情页面，点击“基本信息”按钮可以查看任务基本信息，如下图：

任务名称	Testa-P100-PCI...	部署类型	worker	资源组	defaultGroup	shm_size	4GB
容器实例	镜像	100.2.44.60:5000/paddlepaddle/paddle:1.5.1-cuda10-py36	启动文件	-	-	-	-
基本信息	脚本参数	-	日志路径	-	-	-	-
	执行目录	-	命令	unset NCCL_LAUNCH_MODE && python -u -m paddle.distrib...	-	-	-
	数据	/default_user/train_data	资源配置	worker*1: Tesla-P100-PCIe-16GB:1, CPU:1, MEM:0GB	-	-	-

任务可视化

功能说明：用户通过平台提供的可视化功能，能够查看任务的训练日志。

操作步骤：**A**：进入任务管理模块，点击相关任务可视化按钮，如下图：

任务名称	状态	运行时长	节点	资源配置	框架	启动目录	部署类型	镜像	所属工程	任务类型	提交时间	紧急任务	操作
20211227144130	运行中	8分 38秒	node1(0, -)	worker*1: Cards:0, CPU:1, ...	caffe	sleep 1d	单机	100.2.44.60:500...	-	训练任务	2021-12-28 15:1...	否	返回 刷新 删除

B：如果在创建任务的时候没有选择日志路径，在此处将会再次提示用户选择，如下图：

选择日志路径 ✕

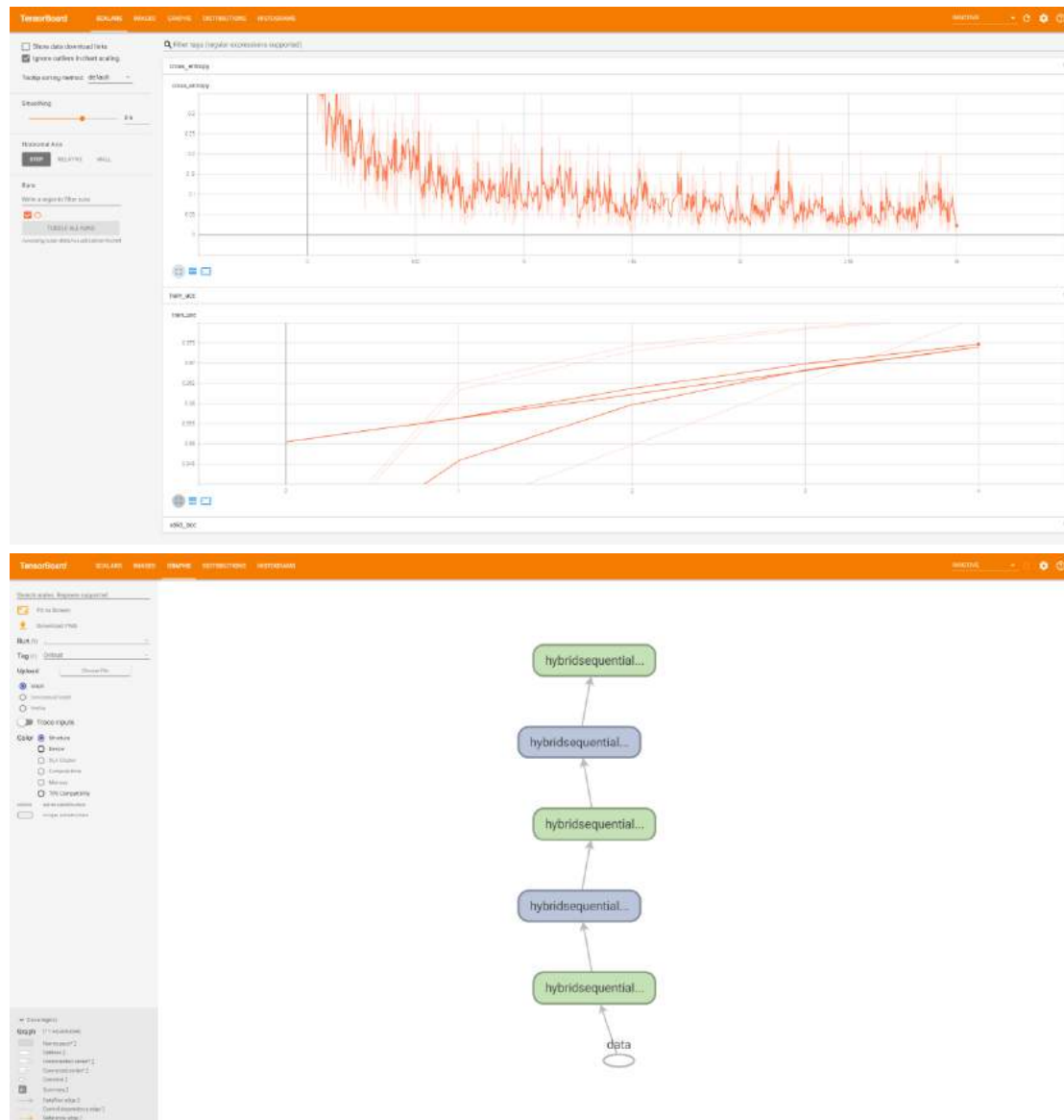
当前路径

<input type="checkbox"/>	名称	拥有者	类型	大小	创建时间
<input type="checkbox"/>	models	default_user	文件夹	-	2021-12-27 09:06:05
<input type="checkbox"/>	train_data	default_user	文件夹	-	2021-12-27 09:08:15
<input type="checkbox"/>	logPersistence	root	文件夹	-	2021-12-28 15:17:57
<input type="checkbox"/>	test.prototxt	default_user	文件	0 Byte	2021-12-27 09:16:40

共 4 条 50条/页 < 1 > 前往 1 页

取消 确定

C: 点击确定后，弹出可视化窗口，如下图:



提交紧急任务

如果用户有提交紧急任务的权限，当点击创建按钮，创建任务时，可以看到紧急任务的开关。

数据配置

数据来源 文件管理 数据集管理

数据

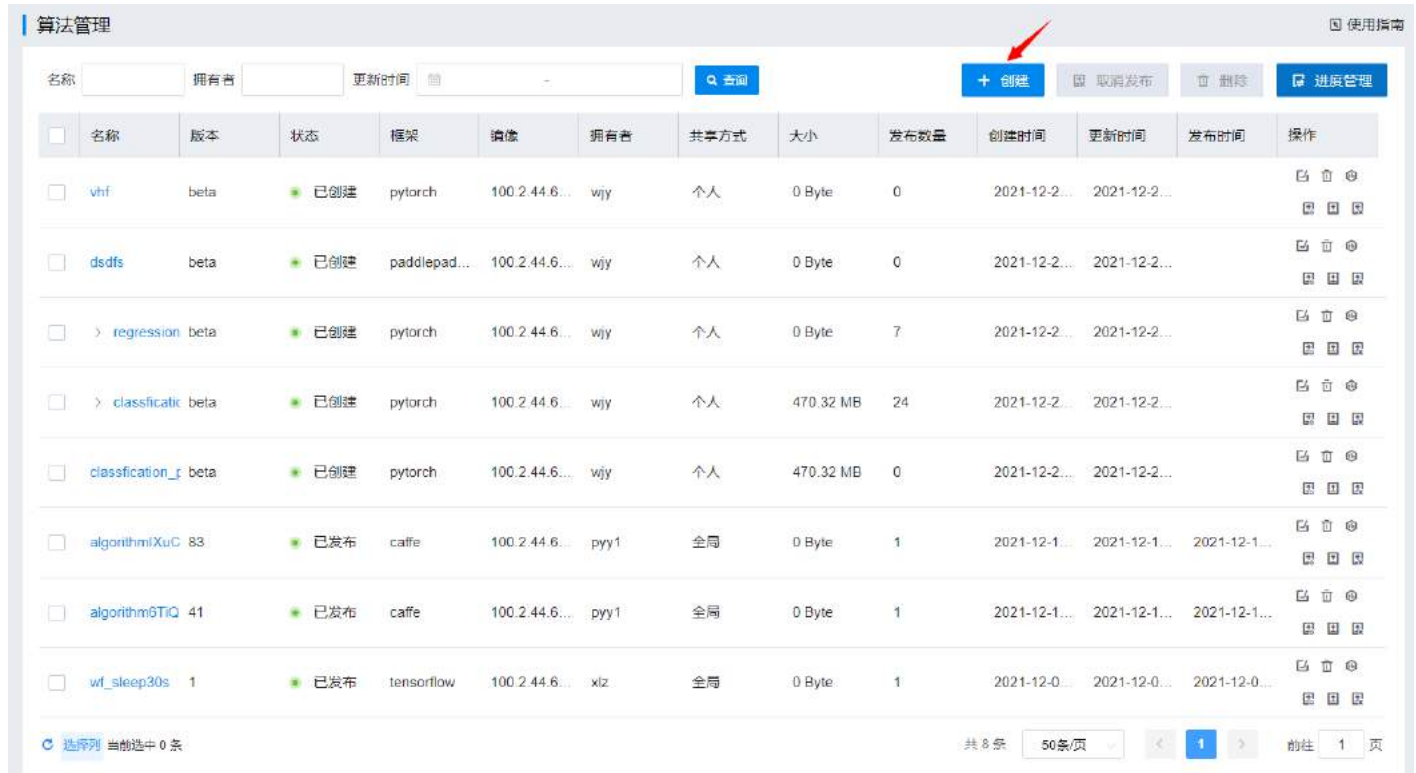
紧急任务 关闭 开启

点击“确定”按钮，提交一个紧急任务。

算法管理

创建算法

用户点击【创建】，进入创建算法页面。用户可以根据需要，自己填写算法信息。



算法管理

名称: 拥有者: 更新时间: 使用指南

名称	版本	状态	框架	镜像	拥有者	共享方式	大小	发布数量	创建时间	更新时间	发布时间	操作
<input type="checkbox"/> vhf	beta	已创建	pytorch	100.2.44.6...	wjy	个人	0 Byte	0	2021-12-2...	2021-12-2...		<input type="button" value="查看"/> <input type="button" value="删除"/> <input type="button" value="分享"/>
<input type="checkbox"/> dsdfs	beta	已创建	paddlepad...	100.2.44.6...	wjy	个人	0 Byte	0	2021-12-2...	2021-12-2...		<input type="button" value="查看"/> <input type="button" value="删除"/> <input type="button" value="分享"/>
<input type="checkbox"/> > regression	beta	已创建	pytorch	100.2.44.6...	wjy	个人	0 Byte	7	2021-12-2...	2021-12-2...		<input type="button" value="查看"/> <input type="button" value="删除"/> <input type="button" value="分享"/>
<input type="checkbox"/> > classification	beta	已创建	pytorch	100.2.44.6...	wjy	个人	470.32 MB	24	2021-12-2...	2021-12-2...		<input type="button" value="查看"/> <input type="button" value="删除"/> <input type="button" value="分享"/>
<input type="checkbox"/> classification_f	beta	已创建	pytorch	100.2.44.6...	wjy	个人	470.32 MB	0	2021-12-2...	2021-12-2...		<input type="button" value="查看"/> <input type="button" value="删除"/> <input type="button" value="分享"/>
<input type="checkbox"/> algorithmXuC	83	已发布	caffe	100.2.44.6...	pyy1	全局	0 Byte	1	2021-12-1...	2021-12-1...	2021-12-1...	<input type="button" value="查看"/> <input type="button" value="删除"/> <input type="button" value="分享"/>
<input type="checkbox"/> algorithm6TiQ	41	已发布	caffe	100.2.44.6...	pyy1	全局	0 Byte	1	2021-12-1...	2021-12-1...	2021-12-1...	<input type="button" value="查看"/> <input type="button" value="删除"/> <input type="button" value="分享"/>
<input type="checkbox"/> wf_sleep30s	1	已发布	tensorflow	100.2.44.6...	xlz	全局	0 Byte	1	2021-12-0...	2021-12-0...	2021-12-0...	<input type="button" value="查看"/> <input type="button" value="删除"/> <input type="button" value="分享"/>

共 8 条 前往 页

创建算法页面，算法版本都是 beta，镜像可以使用平台镜像或者外部镜像。脚本模式需要在个人目录选择算法执行脚本和执行目录，且执行脚本在执行目录下，命令模式需要输入执行命令。运行参数是执行脚本或命令需要的运行参数，可以选择性填写运行算法需要的 CPU 和加速卡资源数量。部署类型，根据镜像框架类型，自主选择。支持的部署类型：

—单机（worker: 1），所有框架

—PS/Worker（PS: 1-1000，Worker: 1-1500），框架: tensorflow

—MPI（Worker: 1-1500），非 paddlepaddle 框架

—Master/Worker（Master: 1，Worker: 1-1500），框架: pytorch

—Server/Worker（Server: 1-1000，Worker: 1-1500），框架: mxnet

—collective:（Worker: 1-1500），框架: paddlepaddle

创建算法

名称: classification

版本: beta

外部镜像

镜像: pytorch

执行脚本: /wjy/classification/classification.py

执行目录: /wjy/classification

运行参数: batch_size: 50, data_input: /wjy/dataset/images

CPU: 1

加速卡: 0

部署类型: MPI

Worker个数: 2

描述: create classification algorithm

使用已发布的算法快速创建

名称	版本	大小	拥有者	描述
algorithm1XuCFJ	83	0 Byte	pyy1	
algorithm8TiQp8	41	0 Byte	pyy1	
wf_sleep30s	1	0 Byte	xlz	

共 3 款 20 条/页 1 页

取消 确定

创建算法页面，用户可以使用已发布的算法进行快速创建，点击对应的已发布算法，创建信息会自动填充，用户可以根据需要，进行修改。

创建算法 < 算法管理 使用指南

* 名称: classification_published

* 版本: beta

外部镜像

* 镜像: pytorch 100.2.44.60:5000/pytorch/pytorch

* 执行脚本: /classification_3/classification.py

* 执行目录: /classification_3

运行参数

batch_size: 50

data_input: /wjy/dataset/images

CPU: 1

加速卡: 0

* 部署类型: 单机 MPI Master/Worker

* Worker个数: 2

描述: create classification algorithm

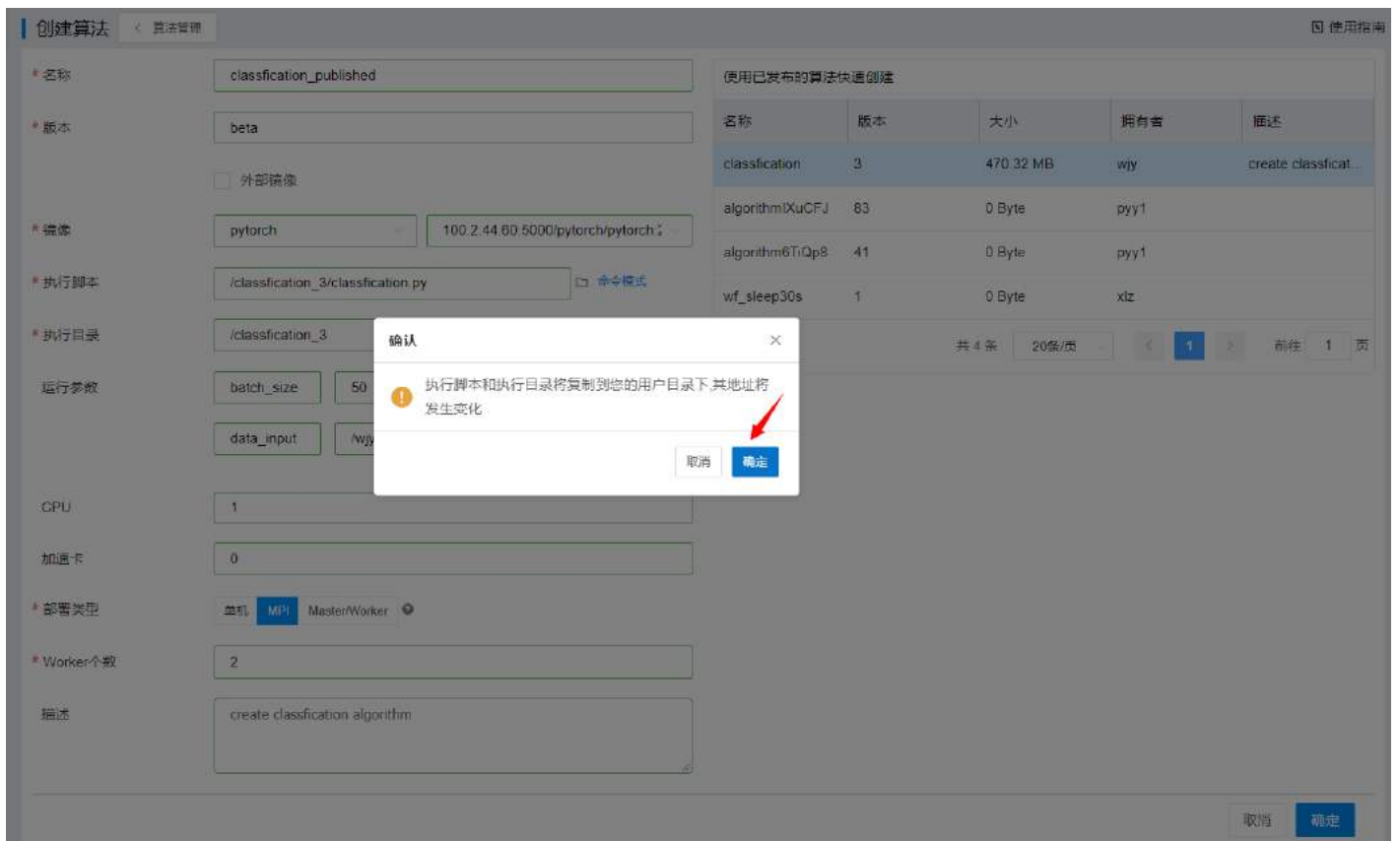
使用已发布的算法快速创建

名称	版本	大小	所有者	描述
classification	3	470.32 MB	wjy	create classificat...
algorithmIXuCFJ	83	0 Byte	ppy1	
algorithm6TiQp0	41	0 Byte	ppy1	
wf_sleep30s	1	0 Byte	xiz	

共 4 条 20条/页 < 1 > 前往 1 页

取消 确定

脚本模式，如果用户未自己选择脚本和目录，平台会将已发布的算法目录，拷贝到用户个人目录，以创建的算法名称_版本命名。用户自己选择，则不再进行拷贝，以用户选择为准。



编辑算法

用户点击【修改】按钮，进入编辑算法界面，除算法名称和版本外，可以根据需要修改相关信息。

编辑算法 < 算法管理 使用指南

* 名称: classification

* 版本: beta

外部镜像

* 镜像: pytorch 100.2.44.60:5000/pytorch/pytorch:20.08-py3-cuda11

* 执行脚本: /wjy/classification/classification.py 命令模式

* 执行目录: /wjy/classification

运行参数: batch_size: 50, data_input: /wjy/dataset/images

CPU: 1

加显卡: 0

* 部署类型: 单机 MPI Master/Worker

* Master个数: 1

* Worker个数: 2

描述: edit classification algorithm

取消 确定

算法列表

用户点击【算法管理】，显示算法信息主列表，同名算法通过主子列表展示，主列表优先显示 beta 版本的算法信息，子列表展示 10 条最新的数据，用户可查看全局、自己组内及个人的算法信息。主列表操作栏有修改、删除、训练、发布管理、发布和取消发布功能。

Algorithm Management Interface (Top View):

- Search filters: Name (名称), Owner (所有者), Update Time (更新时间).
- Buttons: + Create, Refresh, Filter, Export.
- Table Headers: Name (名称), Version (版本), Status (状态), Framework (框架), Model (模型), Owner (所有者), Sharing Method (共享方式), Size (大小), Number of Instances (发布数量), Create Time (创建时间), Update Time (更新时间), Release Time (发布时间), Action (操作).
- Table Content: Lists various algorithms like 'vrf', 'dads', 'regression', 'classification', etc., with their respective details.

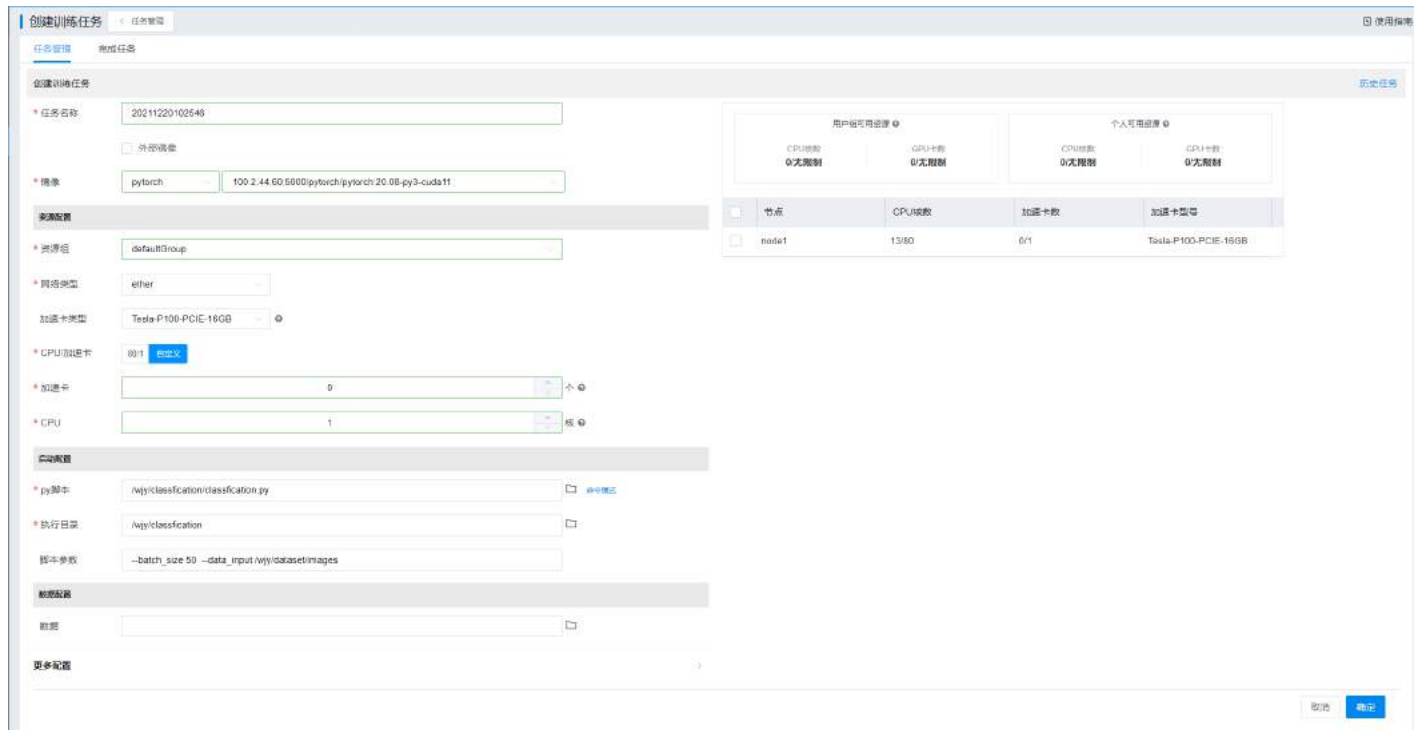
用户可以根据条件查询算法信息，查询条件包括：算法的名称、算法的拥有者，以及算法的更新时间范围。根据条件查询的算法信息，不再以主子列表进行展示，而是分页展示符合条件的查询结果。

Algorithm Management Interface (Bottom View):

- Search filters: Name (名称) set to 'class', Owner (所有者).
- Buttons: + Create, Refresh, Filter, Export.
- Table Headers: Same as the top view.
- Table Content: Shows a subset of algorithms, including 'classification' entries.
- Pagination: Shows '共 26 条' (Total 26 items) and '10 条/页' (10 items per page).

算法训练

用户点击【训练】按钮，跳转到创建训练任务界面，并自动填充算法相关信息



发布算法

用户点击发布按钮，可以发布算法，可以将个人算法发布到组或者全局，发布到组，只有同组成员可以查看，发布到全局，所有用户都可以查看。只有已创建状态的算法发布时，需要填写发布版本，发布版本只能是 1 到 999999 的整数。发布成功后算法状态变为已发布。



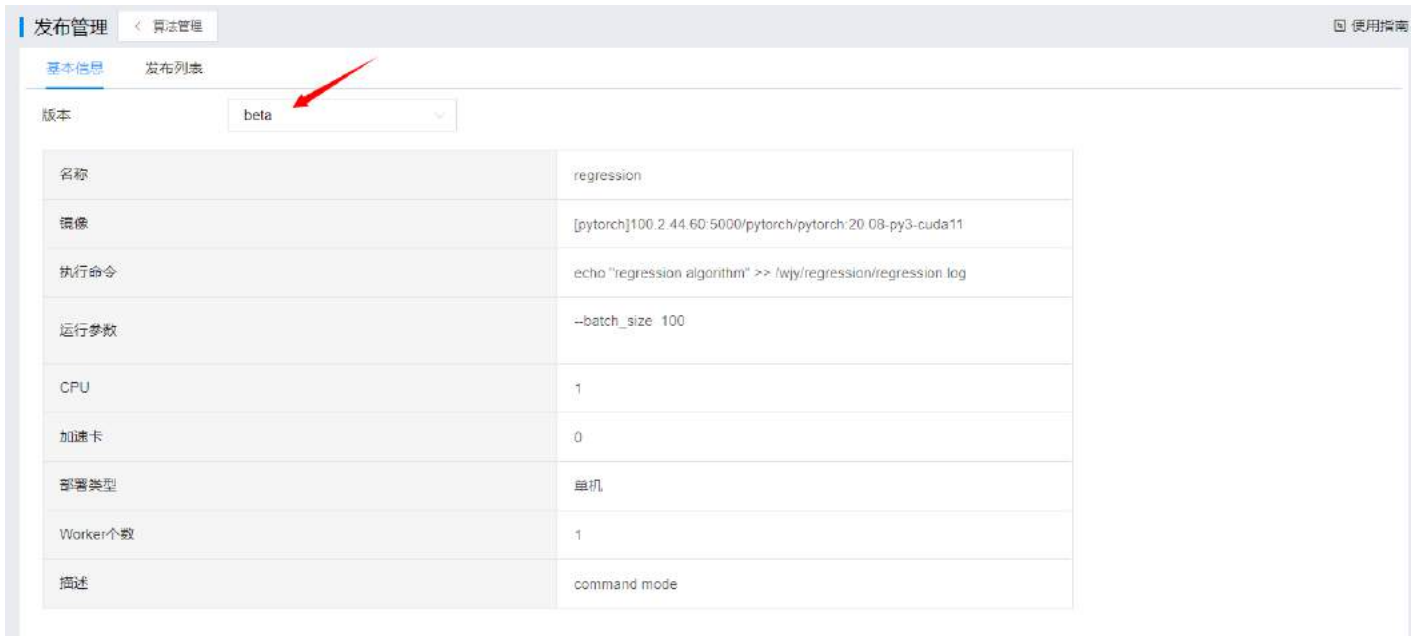
取消发布算法

用户可以取消已发布的算法，取消发布的算法，成为个人算法, 成功后算法状态变为取消发布。



算法发布管理

用户点击发布管理按钮或算法名称，进入发布界面-> 基本信息界面，可以查看算法的详细信息，点击版本的下拉列表，可以查看同名不同版本的算法详细信息



用户在发布界面，点击发布列表，可以查看同名算法的已发布列表，列表操作包含修改、删除、训练、发布和取消发布按钮



删除算法

用户可以删除符合条件的算法，已发布和取消发布状态的算法，不支持批量删除。

算法管理 使用帮助

名称: 拥有者: 更新时间: + 创建 取消发布 删除 修改管理

名称	版本	状态	框架	镜像	拥有者	共享方式	大小	发布数量	创建时间	更新时间	发布时间	操作
regression	beta	已创建	pytorch	100.2.44.60.5000p...	wjy	个人	0 Byte	2	2021-12-20 10:38:55	2021-12-20 10:41:58		
regression	2	已发布	pytorch	100.2.44.60.5000p...	wjy	全局	0 Byte	-	2021-12-20 10:38:55	2021-12-20 10:42:29	2021-12-20 10:42:29	
regression	1	已发布	pytorch	100.2.44.60.5000p...	wjy	组	0 Byte	-	2021-12-20 10:38:55	2021-12-20 10:42:26	2021-12-20 10:42:26	
classification	beta	已创建	pytorch	100.2.44.60.5000p...	wjy	个人	470.32 MB	2	2021-12-20 10:18:17	2021-12-20 10:31:51		
classification	1	已发布	pytorch	100.2.44.60.5000p...	wjy	全局	470.32 MB	-	2021-12-20 10:18:17	2021-12-20 10:32:29	2021-12-20 10:32:29	
classification	3	已发布	pytorch	100.2.44.60.5000p...	wjy	组	470.32 MB	-	2021-12-20 10:18:17	2021-12-20 10:21:23	2021-12-20 10:21:23	
classification_published	beta	已创建	pytorch	100.2.44.60.5000p...	wjy	个人	470.32 MB	0	2021-12-20 10:27:51	2021-12-20 10:27:59		
algorithmXUCFJ	83	已发布	caffe	100.2.44.60.5000c...	pyy1	全局	0 Byte	1	2021-12-14 16:27:54	2021-12-14 16:27:58	2021-12-14 16:27:58	
algorithm8TQpB	41	已发布	caffe	100.2.44.60.5000c...	pyy1	全局	0 Byte	1	2021-12-14 16:27:31	2021-12-14 16:27:35	2021-12-14 16:27:35	
wf_sleep30s	1	已发布	tensorflow	100.2.44.60.5000te...	xiz	全局	0 Byte	1	2021-12-09 15:57:38	2021-12-09 15:59:55	2021-12-09 15:59:55	

共 6 条 00 数据页 1 页



进度管理

命令模式的算法，没有进度列表。进度列表，主要针对操作算法目录场景：

—操作类型【创建】：已发布-> 创建算法，使用已发布的算法，创建新的算法。

—操作类型【编辑】：已发布-> 创建-> 创建失败，编辑创建失败的算法；已创建-> 发布-> 发布失败，编辑发布失败的算法。

—操作类型【发布】：已创建-> 发布，使用已创建算法发布算法；已创建-> 发布-> 发布失败-> 发布，使用已创建算法发布算法失败后，发布状态为发布失败的算法。

—操作类型【删除】：已发布-> 删除和取消发布-> 删除，删除已发布和取消发布状态的算法。

进度管理

✕

🗑 删除

<input type="checkbox"/>	名称	版本	操作类型	状态	进度	失败原因	操作
<input type="checkbox"/>	classification_p...	20	发布	● 成功	100%	-	🗑 🔄
<input type="checkbox"/>	classification_p...	1	发布	● 成功	100%	-	🗑 🔄
<input type="checkbox"/>	classification_p...	beta	创建	● 成功	100%	-	🗑 🔄
<input type="checkbox"/>	classification	30	发布	● 成功	100%	-	🗑 🔄
<input type="checkbox"/>	classification	2	发布	● 成功	100%	-	🗑 🔄
<input type="checkbox"/>	classification	1	发布	● 成功	100%	-	🗑 🔄

🔄 当前选中 0 条

共 6 条

50条/页

<

1

>

前往

1

页

失败重试

进度管理列表，操作类型：创建、编辑和发布失败的进度，如果状态为失败，可以进行失败重试，再次进行操作。

进度管理

✕

删除

<input type="checkbox"/>	名称	版本	操作类型	状态	进度	失败原因	操作
<input type="checkbox"/>	classification_p...	beta	创建	失败	-	-	删除 刷新
<input type="checkbox"/>	classification_p...	20	发布	成功	100%	-	删除 刷新
<input type="checkbox"/>	classification_p...	1	发布	成功	100%	-	删除 刷新
<input type="checkbox"/>	classification	30	发布	成功	100%	-	删除 刷新
<input type="checkbox"/>	classification	2	发布	成功	100%	-	删除 刷新
<input type="checkbox"/>	classification	1	发布	成功	100%	-	删除 刷新

当前选中 0 条

共 6 条

50条/页

<

1

>

前往

1

页

workflow管理

创建工作流

用户点击【创建】，进入创建工作流页面。用户可以根据需要，自己填写工作流信息。工作流创建，主要有以下步骤

1. 基本信息
2. 数据处理
3. 算法
4. 训练处理

workflow basic information: users fill in workflow name, select resource group and network type, running mode, etc.

data processing: users configure data-related parameters, including data source, data usage, data processing, etc.

workflow algorithm: users can choose "my algorithm" and "shared algorithm" for this workflow. For algorithm content, see the algorithm management part introduction.

选择算法 ×

我的算法 共享算法 创建算法

名称	版本	状态	框架	镜像	执行脚本	执行目录	描述	操作
alg_test_wang	beta	● 已创建	pytorch	100.2.44.60:500...	sleep 3600s			📄

取消 确定

我的算法，只可见本用户创建的算法

选择算法 ×

我的算法 共享算法 创建算法

名称	版本	状态	框架	镜像	执行脚本	执行目录	描述	操作
alg_test_wang	beta	● 已创建	pytorch	100.2.44.60:500...	sleep 3600s			📄

取消 确定

共享算法，只可见本用户可见的算法

选择算法 ×

我的算法 共享算法 创建算法

名称	版本	状态	框架	镜像	执行脚本	执行目录	描述	操作
algorithmXuCFJ	83	● 已发布	caffe	100.2.44.60:50...	/algorithmXuC...	/algorithmXuC...		📄
algorithmXuCFJ	82	● 已发布	caffe	100.2.44.60:50...	/algorithmXuC...	/algorithmXuC...		📄
algorithm6TiQp8	40	● 已发布	caffe	100.2.44.60:50...	/algorithm6TiQp...	/algorithm6TiQp...		📄
algorithm6TiQp8	41	● 已发布	caffe	100.2.44.60:50...	/algorithm6TiQp...	/algorithm6TiQp...		📄
ttt	2	● 已发布	tensorflow	100.2.44.60:50...	sleep 30s			📄
ttt	1	● 已发布	tensorflow	100.2.44.60:50...	sleep 30s			📄
test_tf	2	● 已发布	pytorch	100.2.44.60:50...	sleep 10s			📄
test_tf	1	● 已发布	pytorch	100.2.44.60:50...	sleep 10s			📄
wf_test_TF	8	● 已发布	tensorflow	100.2.44.60:50...	/wf_test_TF_8/...	/wf_test_TF_8		📄
wf_test_TF	7	● 已发布	tensorflow	100.2.44.60:50...	/wf_test_TF_7/...	/wf_test_TF_7		📄

取消 确定

我的算法，可以在此进行编辑修改，修改操作可参见算法管理的编辑算法。本用户无法修改共享算法。

创建算法
✕

* 名称

* 版本

外部镜像

* 镜像 pytorch 100.2.44.60:5000/pytorch/pytorch:20.08-py3-cuda11

* 执行命令 脚本模式

sleep 3600s

运行参数 +

CPU

加速卡

* 部署类型 单机 MPI Master/Worker ?

* Worker个数

描述

取消
确定

配置任务参数信息，可参见任务管理配置介绍

创建算法
工作流管理

1 基本配置

2 数据配置

3 默认

4 高级配置

* 算法

名称 test_alg

版本 2

状态 已发布

镜像 (pytorch) 100.2.44.60:5000/pytorch/pytorch:20.08-py3-cuda11

执行命令 sleep 10s

描述

运行参数 +

* 部署类型 单机 MPI Master/Worker ?

* Worker个数

用户可用资源

CPU核数 33/无限制

GPU核数 2/无限制

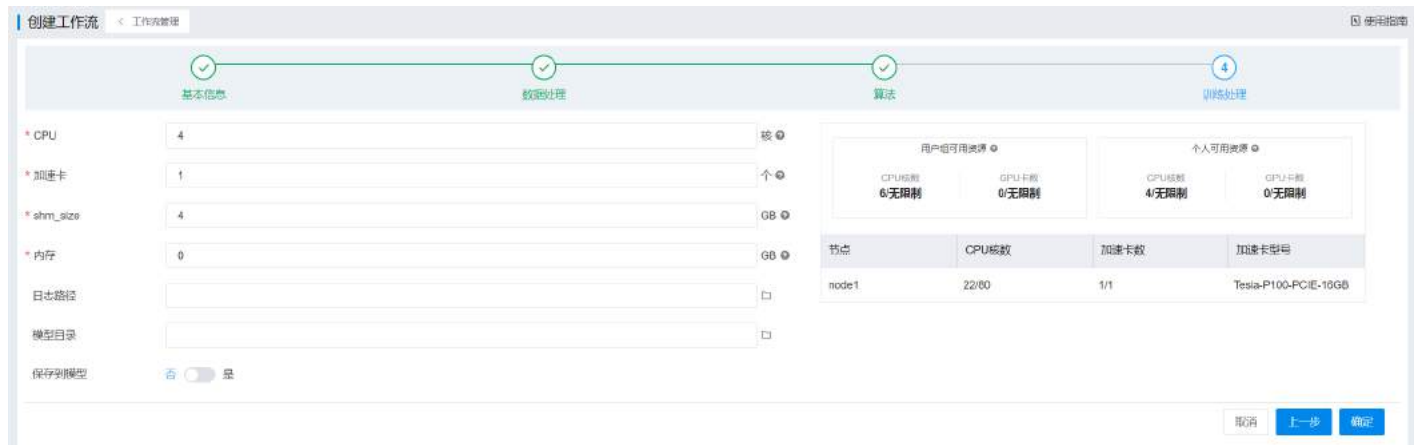
个人可用资源

CPU核数 0/无限制

GPU核数 0/无限制

节点	CPU核数	加速卡数	加速卡型号
node1	38/0	1/1	Tesla-P100-PCIe-16GB

取消
上一步
下一步



运行 workflows

工作流启动方式:

1. 立即运行: 创建和编辑工作流时设置为立即启动运行, 则完成工作流配置后, 该工作流会自动立即运行
2. 非立即运行, 待创建完成工作流后, 在工作流列表操作栏, 手动点击”运行”按钮
3. 周期性运行: 创建和编辑工作流时设置, 或在工作流列表上方选择周期性启动

工作流开始运行



工作流运行完成

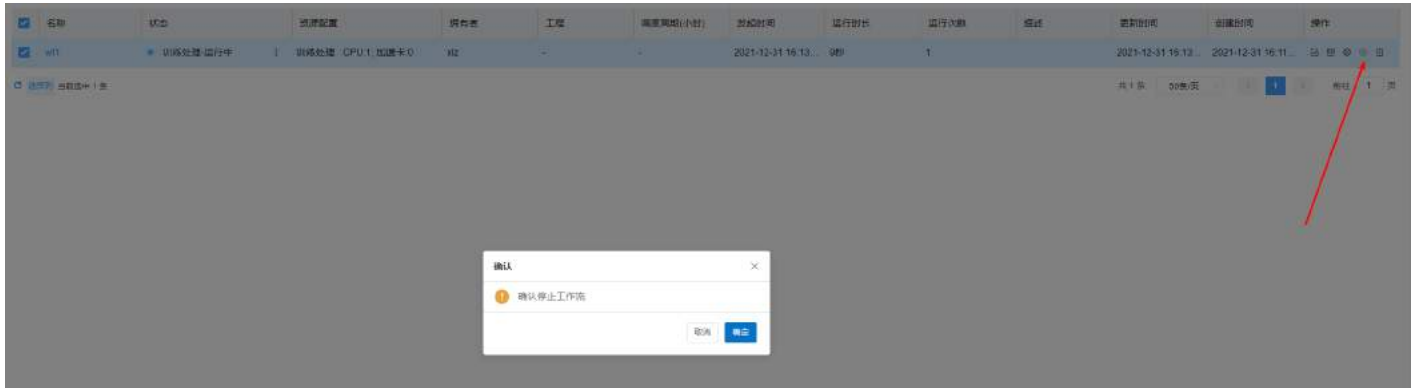


工作流运行完成后, 可进行编辑该工作流, 修改周期运行方式、再次启动或删除操作

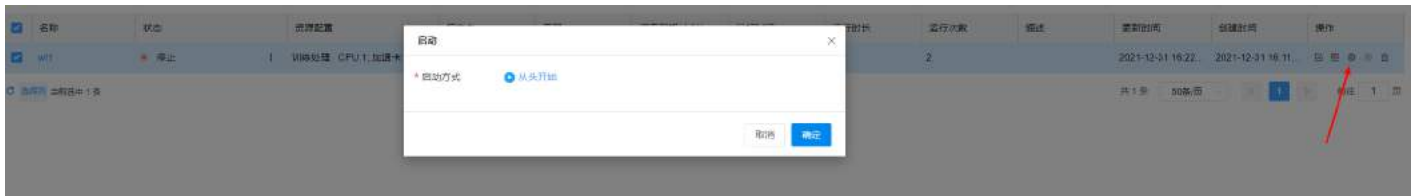


停止/启动工作流

用户可以将运行未完结的工作流停止，停止后，工作流任务以及子任务都会停止，同时也不会再执行周期性运行；

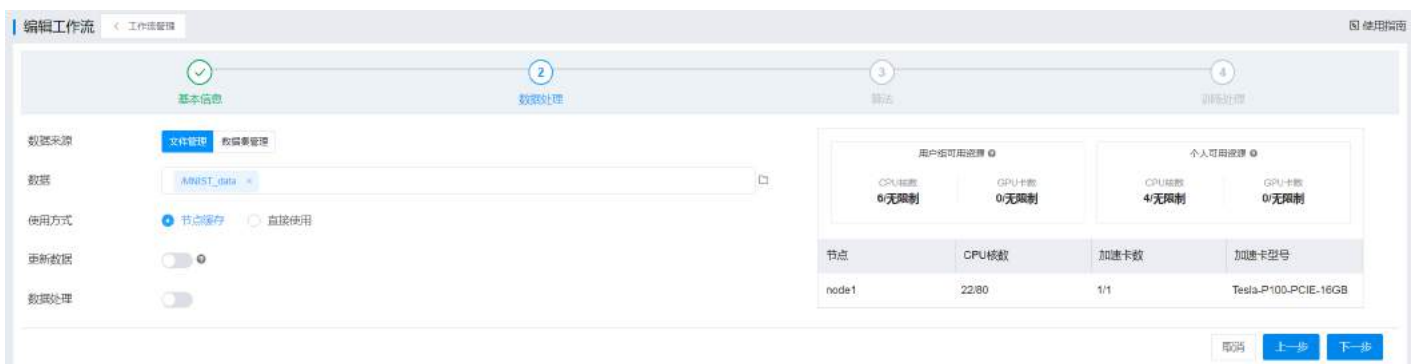


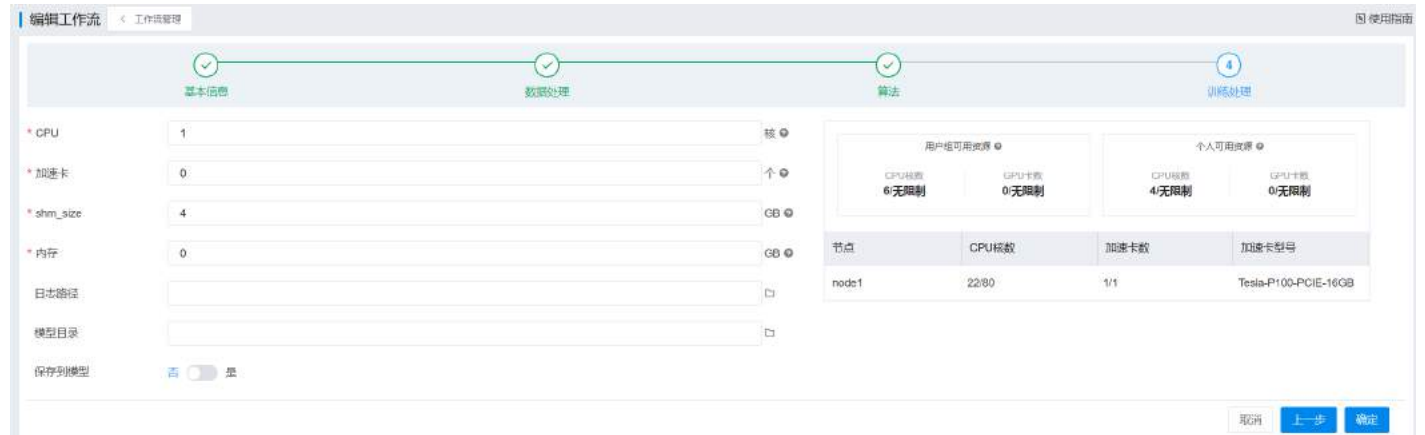
用户可以启动停止的工作流；



编辑工作流

对符合状态条件的工作流进行编辑，有些信息无法修改，如工作流名称等





删除 workflow

对符合状态条件的工作流进行删除



workflow 信息查看

用户可以选择列表中的 workflow，点击 workflow 名称，进入 workflow 详情页面，查看 workflow 基本信息、任务配置信息，以及 workflow 子任务活动状态

workflow详情 < workflow管理

workflow信息

基本信息

名称	imagenet1024_test
周期性启动	否
立即运行	是
资源组	defaultGroup
网络类型	ETHER
描述	自定义数据格式转换+训练作业的workflow
数据	/MNIST_data
数据使用方式	节点缓存
更新数据	否

训练处理

算法	名称	test_tf
	版本	2
	镜像	(pytorch)
		100.2.44.60:5000/pytorch/pytorch:2
		py3-cuda11
	大小	0 Byte
	描述	
	创建时间	2021-12-13 14:45:28

运行参数

CPU	2 核
-----	-----

workflow任务

执行总次数: 2 失败: 0 成功: 2 执行中: 0 停止: 0

imagenet1024_test_2 ● 完成 执行时间: 2021-12-17 11:06:25

训练处理 完成

名称: imagenet1024_test_training_processi

创建时间: 2021-12-17 11:06:25

imagenet1024_test_1 ● 完成 执行时间: 2021-12-17 10:27:24

训练处理 完成

名称: imagenet1024_test_training_processi

创建时间: 2021-12-17 10:27:24

镜像管理

镜像查询

普通用户点击【**镜像管理**】，默认显示所有框架下的镜像，普通用户可查看公共、自己组内及个人的镜像。

The screenshot displays the '镜像管理' (Image Management) interface. At the top, there are tabs for different frameworks: All, PyTorch, TensorFlow, Caffe, MxNet, PaddlePaddle, and Other. Below these are filters for '全部' (All), '个人' (Personal), '组' (Group), and '公共' (Public). A search bar and a dropdown for '最近使用时间(Z-A)' (Sort by Last Used Time) are also present. The main area shows a grid of image cards. Each card includes a framework logo, the image name, creation date, IP address, share status, size, and usage count. The 'pytorch' card in the second row is highlighted with a blue border.

Framework	Image Name	Creation Date	IP Address	Share Status	Size	Usage Count
Caffe	caffe	2021-05-28	100.2.126.15:5000/caffe/caffe	公共	2.1GB	41次
ubuntu	ubuntu18.04	2021-05-28	100.2.126.15:5000/other/ubuntu18.04...	公共	291.5MB	4次
tensorflow	tensorflow	2021-05-28	100.2.126.15:5000/tensorflow/tensorflow	公共	4.8GB	10次
pytorch	pytorch	2021-05-28	100.2.126.15:5000/pytorch/pytorch	公共	3.6GB	6次
tensorflow	tensorflow	2021-05-28	100.2.126.15:5000/mlu/tensorflow	公共	1.1GB	0次
pytorch	pytorch	2021-05-28	100.2.126.15:5000/mlu/pytorch	公共	2.7GB	0次
torch-se...	torch-se...	2021-05-28	100.2.126.15:5000/serving/torch-server	公共	2.8GB	0次
torch-se...	torch-se...	2021-05-28	100.2.126.15:5000/serving/torch-server	公共	697.4MB	0次

普通用户可以按照镜像框架类型查询，选择一个镜像框架（caffe、tensorflow、mxnet、pytorch、paddlepaddle、other），显示该镜像框架下的镜像列表信息，显示信息包括框架名称、镜像名称、分享属性、镜像大小、使用次数、最近使用时间、镜像 tag、上传者、创建时间。

The screenshot shows the '镜像管理' (Image Management) interface. At the top, there are tabs for 'All', 'PyTorch', 'TensorFlow', 'Caffe', 'MxNet', 'PaddlePaddle', and 'Other'. The 'TensorFlow' tab is selected and highlighted with a red arrow. Below the tabs, there are filters for '全部' (All), '个人' (Personal), '组' (Group), and '公共' (Public). A dropdown menu shows '最近使用时间(Z-A)' (Recent usage time (Z-A)). There are buttons for '+ 创建' (Create), '导入' (Import), and '传输列表' (Transfer list). The main content area displays a single image card for 'tensorflow' with the following details: '2021-05-28', '100.2.126.15:5000/tensorflow/tensorflow', '公共' (Public), '4.8GB', '10次' (10 times), '20.09-py3-cuda11', '最近使用时间: 2021-06-03 16:54:59', '上传者: admin', and '备注:' (Remarks). At the bottom right, there is a pagination bar showing '共 1 条' (Total 1 item), '50条/页' (50 items/page), and '前往 1 页' (Go to page 1).

普通用户可以按照分享属性查询，选择一种分享属性（个人、组、公共），显示该分享属性下的镜像列表信息。选择个人属性：筛选对应登录账号上传的且属性为个人的镜像，选择组内属性：筛选属性为组内且属于登录账号所在组的镜像，选择公共：筛选属性为公共的所有镜像。

镜像管理 使用指南

All PyTorch TensorFlow Caffe MxNet PaddlePaddle Other

全部 个人 组 公共

最近使用时间(Z-A)

名称	日期	大小	次数	最近使用时间	上传者
caffe	2021-05-28	2.1GB	41次	2021-06-07 16:22:19	admin
ubuntu1...	2021-05-28	291.5MB	4次	2021-06-04 11:25:43	admin
tensorflow	2021-05-28	4.8GB	10次	2021-06-03 16:54:59	admin
pytorch	2021-05-28	3.6GB	6次	2021-05-31 15:46:12	admin
tensorflow	2021-05-28	1.1GB	0次	2021-05-28 17:22:03	admin
pytorch	2021-05-28	2.7GB	0次	2021-05-28 17:16:17	admin
torch-se...	2021-05-28	2.8GB	0次	2021-05-28 17:02:52	admin
torch-se...	2021-05-28	697.4MB	0次	2021-05-28 17:02:27	admin

普通用户可以按照最近使用时间或镜像大小排序，选择一个排序字段，默认降序排列。

镜像管理 使用指南

分类: All | PyTorch | TensorFlow | Caffe | MxNet | PaddlePaddle | Other

筛选: 全部 | 个人 | 组 | 公共

操作: 创建 | 导入 | 传输列表

排序: 镜像排序 (最近使用时间(Z-A) | 镜像大小(Z-A))

名称	仓库地址	大小	使用次数	最近使用时间	上传者
caffe	100.2.126.15:5000/caffe/caffe	2.1GB	41次	2021-06-07 16:22:19	admin
ubuntu	100.2.126.15:5000/other/ubuntu18.04...	291.5MB	4次	2021-06-04 11:25:43	admin
tensorflow	100.2.126.15:5000/tensorflow/tensorflow	4.8GB	10次	2021-06-03 16:54:59	admin
pytorch	100.2.126.15:5000/pytorch/pytorch	3.6GB	6次	2021-05-31 15:46:12	admin
tensorflow	100.2.126.15:5000/mlu/tensorflow	1.1GB	0次	2021-05-28 17:22:03	admin
pytorch	100.2.126.15:5000/mlu/pytorch	2.7GB	0次	2021-05-28 17:16:17	admin
torch-se...	100.2.126.15:5000/serving/torch-server	2.8GB	0次	2021-05-28 17:02:52	admin
torch-se...	100.2.126.15:5000/serving/torch-server	697.4MB	0次	2021-05-28 17:02:27	admin

普通用户可以进行镜像仓库全局模糊查询，在模糊输入框内输入上传者、镜像 tag 相关信息，显示符合模糊搜索信息的所有镜像列表，支持输入即查询显示功能。

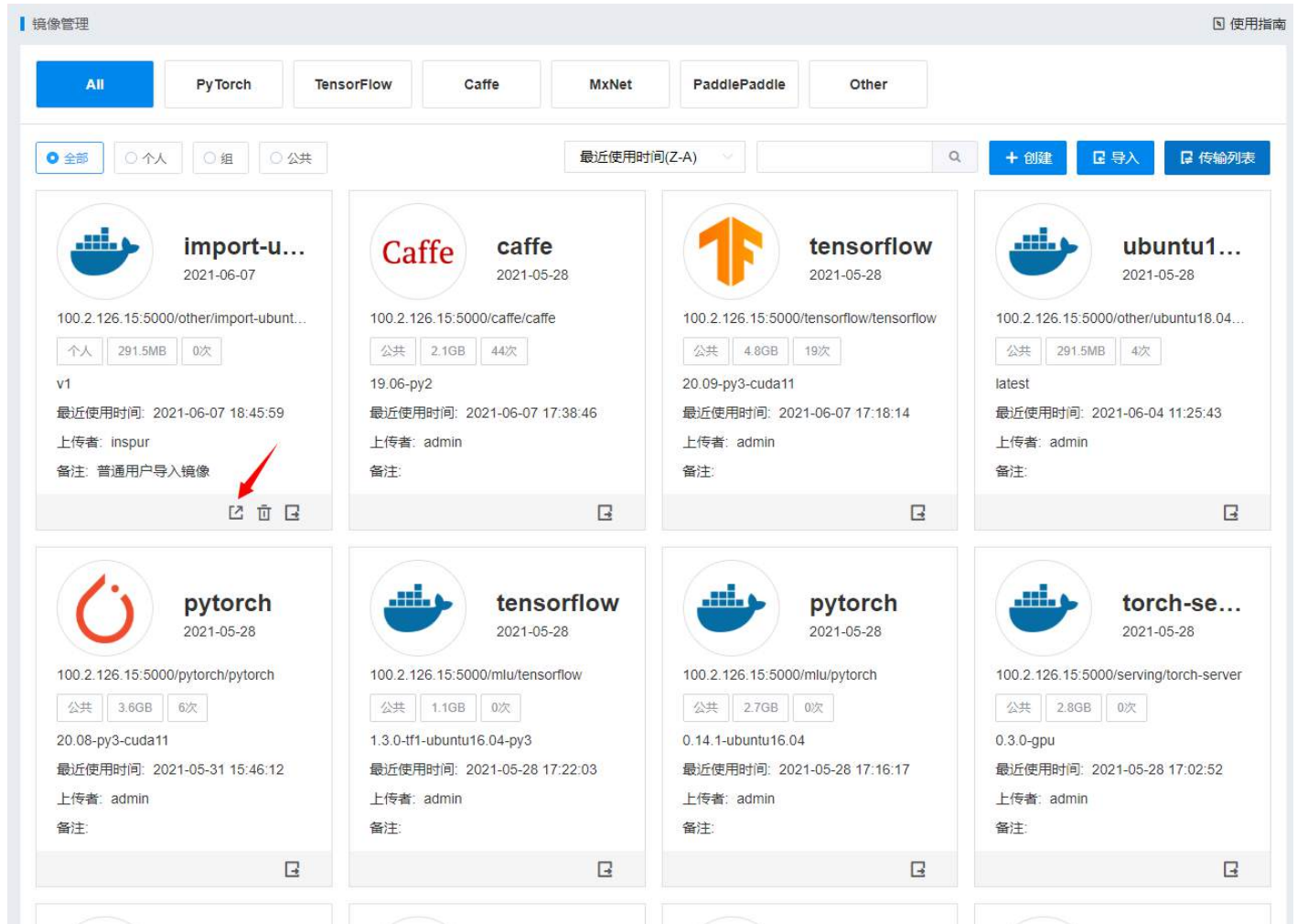
The screenshot shows the '镜像管理' (Image Management) interface. At the top, there are tabs for different frameworks: All, PyTorch, TensorFlow, Caffe, MxNet, PaddlePaddle, and Other. Below these are filters for '全部', '个人', '组', and '公共'. A search bar contains the text 'te', with a red arrow pointing to it. To the right of the search bar are buttons for '+ 创建', '导入', and '传输列表'. The main area displays a grid of six image cards. Each card includes a Docker icon, the image name, upload date, size, and usage count. The cards are: 'ubuntu1...', 'tensorflow' (4.8GB, 10次), 'tensorflow' (1.1GB, 0次), 'tensor-rt' (2.4GB, 0次), 'tensorflo...' (1.2GB, 0次), and 'tensorflo...' (76.6MB, 0次). At the bottom right, there is a pagination control showing '共 6 条', '50 条/页', and '1' page.

镜像列表支持分页功能展现。

This screenshot shows the same interface as above, but with the 'TensorFlow' tab selected. Only one image card is visible: 'tensorflow' (4.8GB, 10次), with details like '20.09-py3-cuda11' and '最近使用时间: 2021-06-03 16:54:59'. The pagination control at the bottom right shows '共 1 条', '50 条/页', and '1' page, with a red arrow pointing to the '1' page indicator.

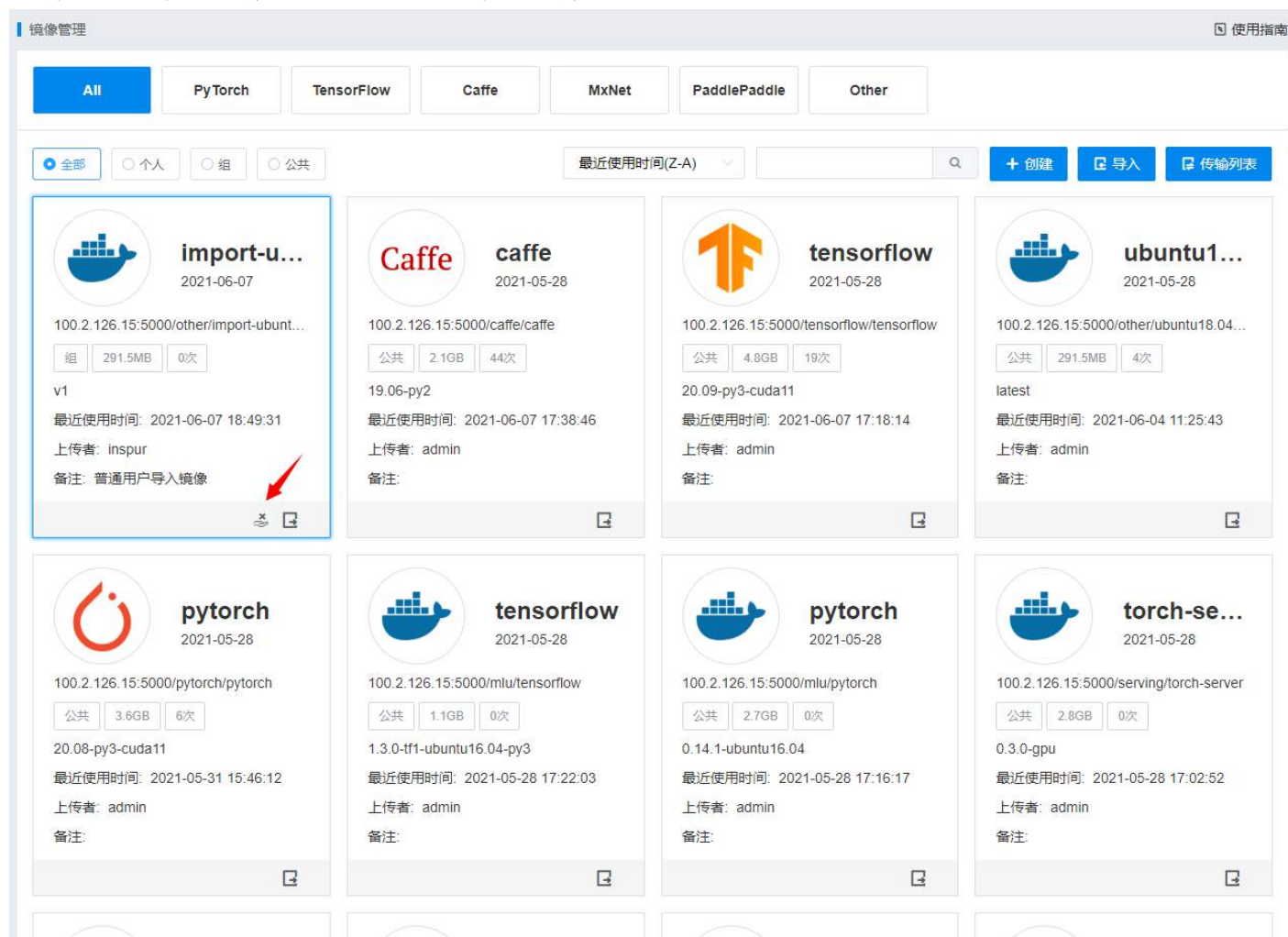
镜像分享

普通用户点击【镜像管理】，个人和自己组内的镜像上有分享按钮，普通用户可以点击该按钮修改该镜像的分享属性。普通用户可以修改自己个人的镜像属性，即普通用户可以把个人修改成组内或公共属性，也可以将以前自己设置的公共属性镜像修改为个人属性。



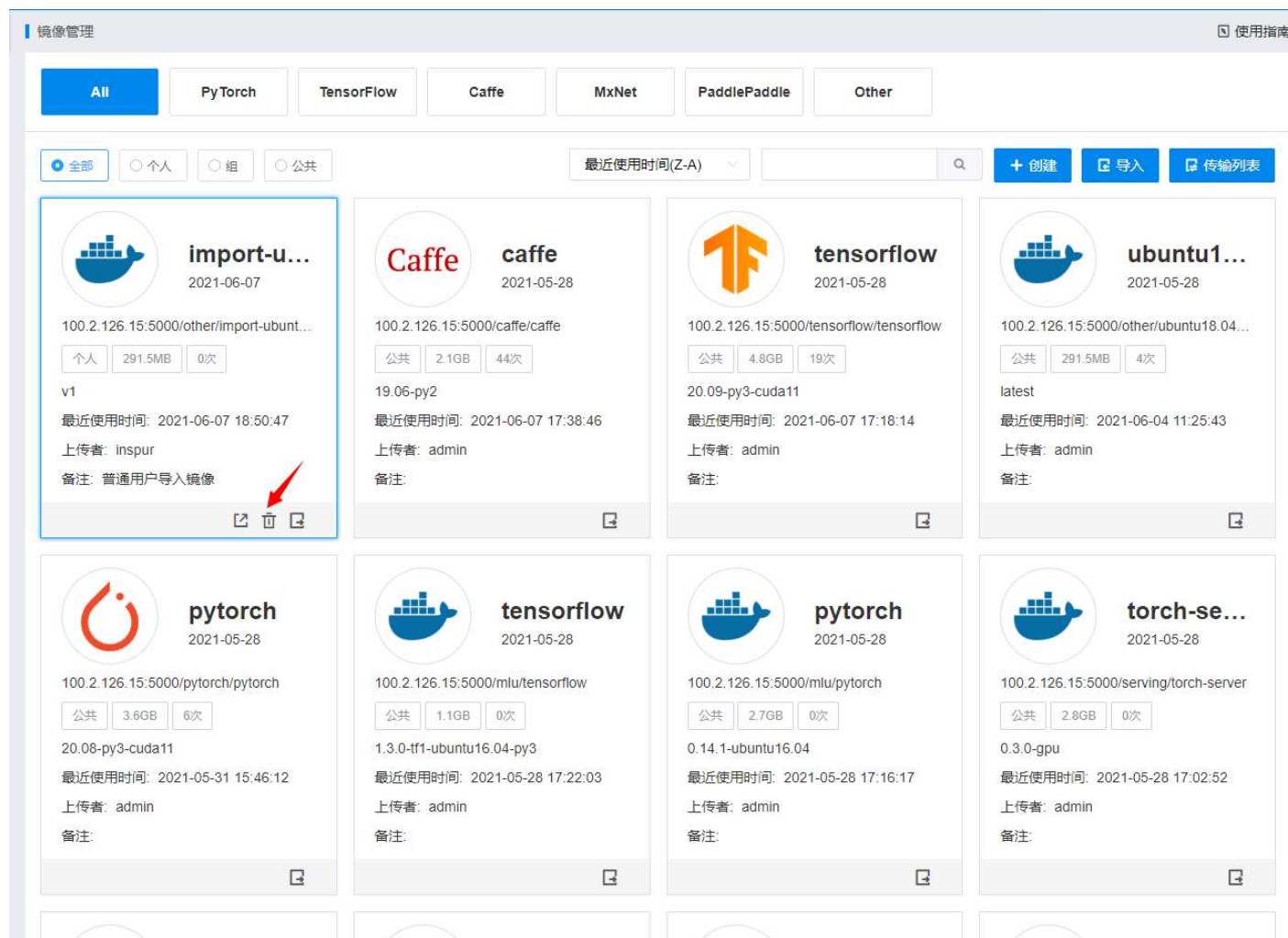
取消分享

普通用户点击【镜像管理】，组和公共分类下能看到自己分享的镜像，点击镜像上的取消分享按钮，可以将自己分享的镜像撤销，重新变成个人镜像。



镜像删除

普通用户点击【镜像管理】，普通用户可以删除个人的镜像，不能删除组内和公共的镜像。



点击镜像上面的删除按钮，弹出确认信息提示，点击确认，删除该镜像。



普通用户可在传输列表中，查看删除的镜像进度信息。点击进度列表的日志文件图标，可以查看删除日志。支持单条或者批量删除状态为成功和失败的进度记录。

传输列表 ×

🗑️ 删除

<input type="checkbox"/>	镜像名称	状态	异常原因	操作类型	进度	排队位置	创建时间	完成时间	操作
<input type="checkbox"/>	100.2.126...	执行中	-	删除镜像	10%	-	2021-06-0...	-	🗑️ 📄
<input type="checkbox"/>	100.2.126...	成功	-	内部镜像...	100%	-	2021-06-0...	2021-06-0...	🗑️ 📄
<input type="checkbox"/>	100.2.126...	成功	-	导出镜像	100%	-	2021-06-0...	2021-06-0...	🗑️ 📄

当前选中 0 条 共 3 条 50条/页 < 1 > 前往 1 页

导出镜像

普通用户点击【镜像管理】，镜像上有导出按钮，普通用户可以点击该按钮导出镜像 tar 包，弹出框内给出默认导出 tar 包名称，用户可自定义修改。

The screenshot displays the '镜像管理' (Image Management) interface. At the top, there are filter tabs for 'All', 'PyTorch', 'TensorFlow', 'Caffe', 'MxNet', 'PaddlePaddle', and 'Other'. Below these are options for '全部' (All), '个人' (Personal), '组' (Group), and '公共' (Public). A dropdown menu shows '最近使用时间(Z-A)' (Sort by last used time Z-A). There are buttons for '+ 创建' (Create), '导入' (Import), and '传输列表' (Transfer List).

The main area shows a grid of image cards. Each card includes an icon, name, date, ID, public status, size, and usage count. A red arrow points to a document icon in the bottom right corner of the first card, which is highlighted in a light grey box.

The modal dialog '导出镜像' (Export Image) is open, showing a text input field for the filename: 'ubuntu18.04-python3.7.5-openssh7.6-jupyterlab1.2.3_latest'. At the bottom right of the dialog are '取消' (Cancel) and '确定' (Confirm) buttons.

普通用户可在传输列表中，查看导出的镜像进度信息，点击进度列表的日志文件图标，可以查看删除日志。如果出现异常，进度列表会显示异常原因。支持单条或者批量删除状态为成功和失败的进度记录。

传输列表 ×

🗑️ 删除

<input type="checkbox"/>	镜像名称	状态	异常原因	操作类型	进度	排队位置	创建时间	完成时间	操作
<input type="checkbox"/>	100.2.126....	执行中	-	导出镜像	40%	-	2021-06-0...	-	🗑️ 📄
<input type="checkbox"/>	100.2.126....	成功	-	删除镜像	100%	-	2021-06-0...	2021-06-0...	🗑️ 📄
<input type="checkbox"/>	100.2.126....	成功	-	内部镜像...	100%	-	2021-06-0...	2021-06-0...	🗑️ 📄
<input type="checkbox"/>	100.2.126....	成功	-	导出镜像	100%	-	2021-06-0...	2021-06-0...	🗑️ 📄

🔄 当前选中 0 条

共 4 条 50条/页 < 1 > 前往 1 页

导出成功后，可以在用户目录下，查看导出的镜像 tar 包。

创建镜像

单击【镜像管理】->【创建】，可以使用 Dockerfile 创建镜像

镜像管理 使用指南

ALL PyTorch TensorFlow Caffe MxNet PaddlePaddle Other

全部 个人 组 公共 最近使用时间(Z-A) [搜索] + 创建 导入 传输列表

名称	大小	使用次数	最近使用时间	上传者
caffe	2.1GB	44次	2021-06-07 17:38:46	admin
tensorflow	4.8GB	19次	2021-06-07 17:18:14	admin
ubuntu18.04	291.5MB	4次	2021-06-04 11:25:43	admin
pytorch	3.6GB	6次	2021-05-31 15:46:12	admin
tensorflow	1.1GB	0次	2021-05-28 17:22:03	admin
pytorch	2.7GB	0次	2021-05-28 17:16:17	admin
torch-se...	2.8GB	0次	2021-05-28 17:02:52	admin
torch-se...	697.4MB	0次	2021-05-28 17:02:27	admin

弹出创建界面，点击文件夹图标



创建

* Dockerfile

建议尽量将Dockerfile放在空目录中或者当前目录下只包含用于Dockerfile制作镜像的文件/文件夹

* 镜像名称 ?

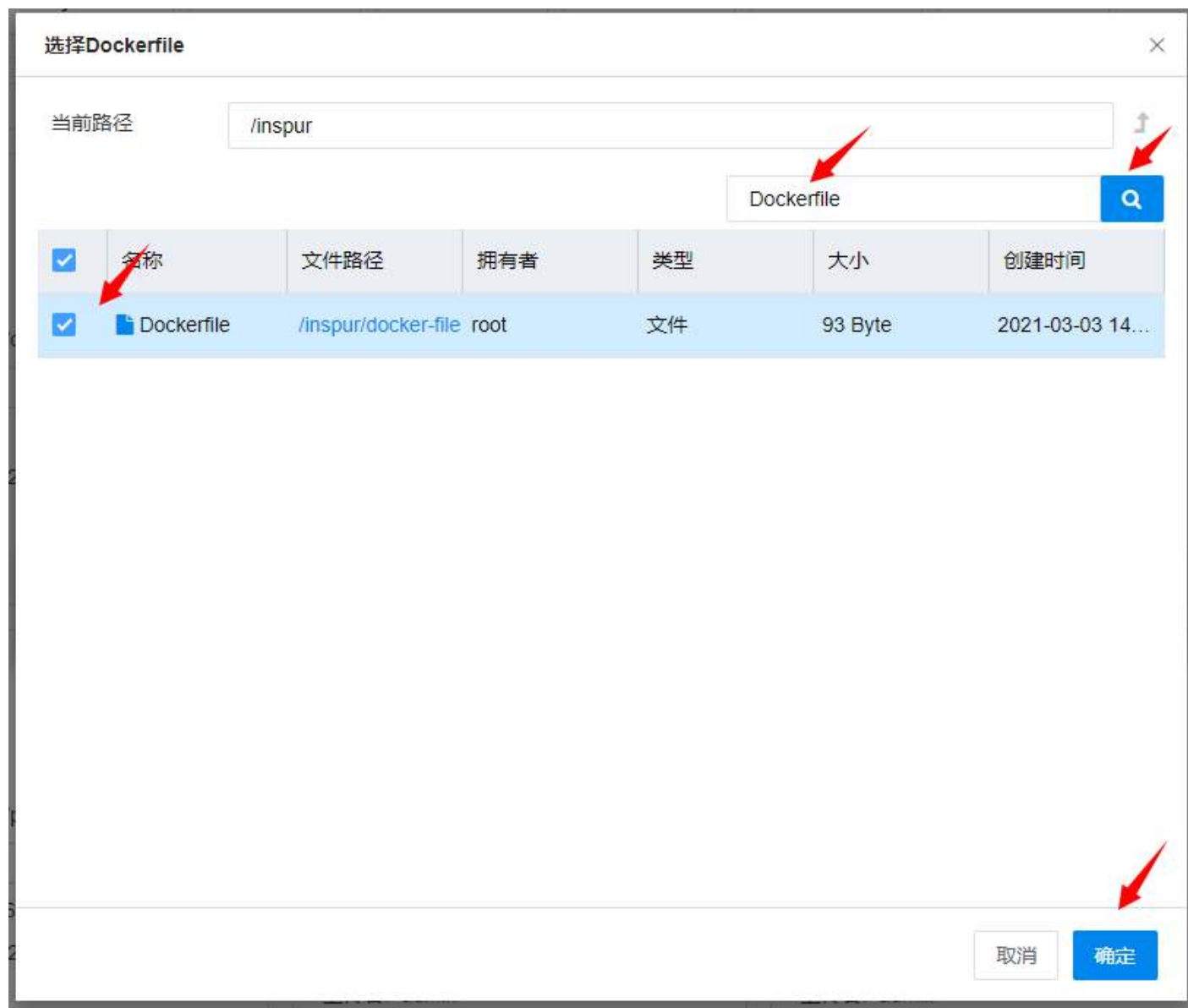
* 镜像类型 请选择

* 标签 ?

备注

取消 确定

注意：镜像名称和标签需要满足 Docker 官方规范，只能包括小写字母、数字、下划线 (_)、连接线 (-)、反斜线 (/)，且只能使用小写字母或数字开头，并且特殊字符（下划线、连接线、反斜线）不能连续使用。
弹出选择 Dockerfile 界面，选择 Dockerfile 文件或者搜索 Dockerfile 文件，点击【确定】



注意：普通用户创建镜像，需要将 Dockerfile 文件放在用户目录下，并且需要在用户目录下单独创建一个目录，用来存放 Dockerfile 文件。

选择完 Dockerfile 文件，返回创建界面，显示 Dockerfile 文件的相对路径，输入镜像名称，镜像标签，备注信息，点击【确定】，镜像名称和标签不支持大写。

创建

* Dockerfile

建议尽量将Dockerfile放在空目录中或者当前目录下只包含用于Dockerfile制作镜像的文件/文件夹

* 镜像名称

* 镜像类型

* 标签

备注

取消 确定

注意：镜像名称和标签需要满足 Docker 官方规范，只能包括小写字母、数字、下划线 (_)、连接线 (-)、反斜线 (/)，且只能使用小写字母或数字开头，并且特殊字符（下划线、连接线、反斜线）不能连续使用。确定后，点击【传输列表】，查看创建镜像的进度信息。点击进度列表的日志文件图标，可以查看创建镜像的日志记录。支持单条或者批量删除状态为成功和失败的进度记录。

传输列表



删除

<input type="checkbox"/>	镜像名称	状态	异常原因	操作类型	进度	排队位置	创建时间	完成时间	操作
<input type="checkbox"/>	100.2.126....	执行中	-	Dockerfile...	30%	-	2021-06-0...	-	
<input type="checkbox"/>	100.2.126....	成功	-	导出镜像	100%	-	2021-06-0...	2021-06-0...	
<input type="checkbox"/>	100.2.126....	成功	-	删除镜像	100%	-	2021-06-0...	2021-06-0...	
<input type="checkbox"/>	100.2.126....	成功	-	内部镜像...	100%	-	2021-06-0...	2021-06-0...	
<input type="checkbox"/>	100.2.126....	成功	-	导出镜像	100%	-	2021-06-0...	2021-06-0...	

当前选中 0 条

共 5 条

50条/页



1

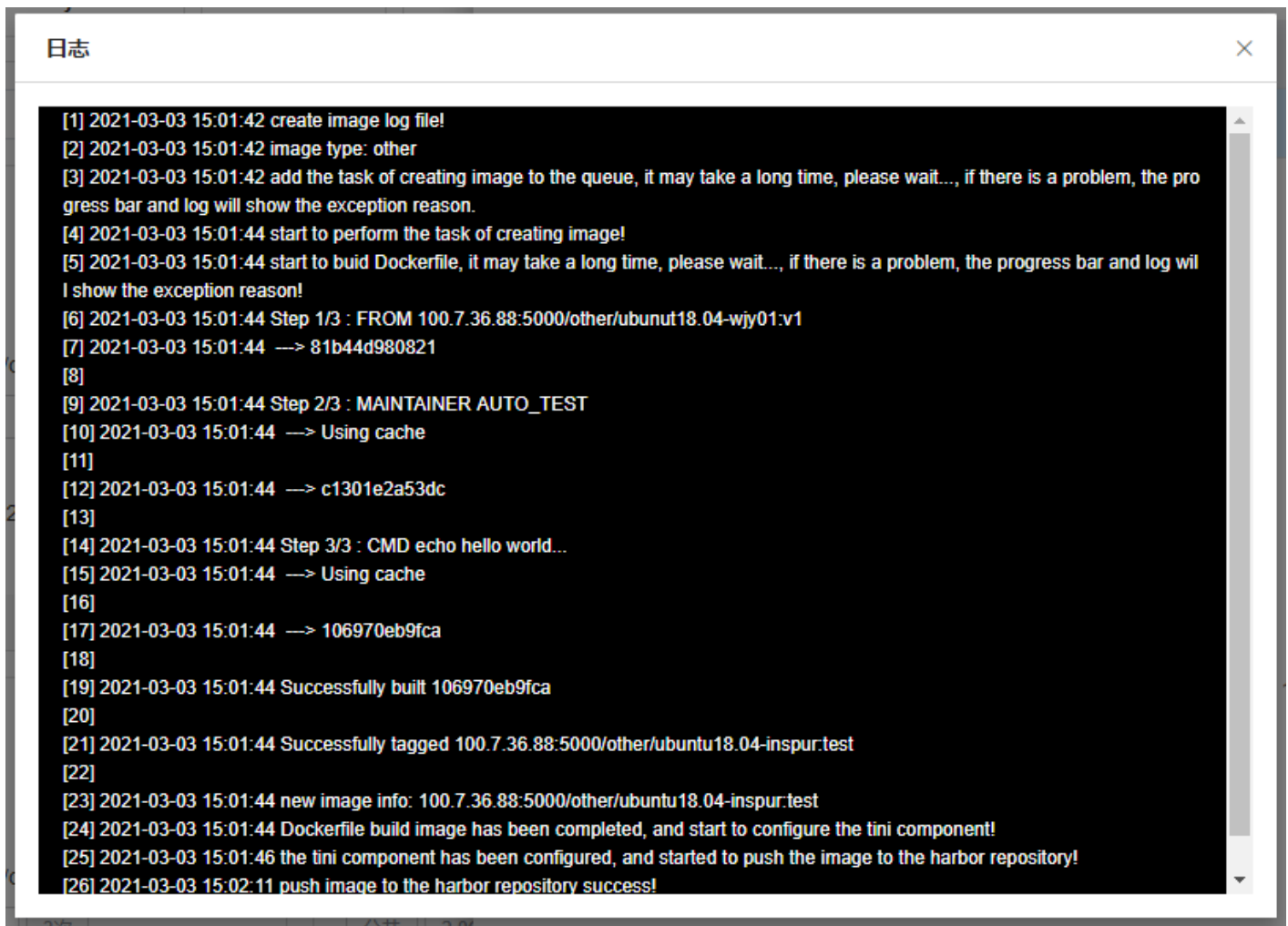


前往

1

页

点击日志文件图标，弹出日志界面，显示日志记录信息。如果 Dockerfile 编写有问题，日志页面会输出异常信息，进度条会显示异常原因，同时后台回滚删除相关操作记录，进度状态会置为失败。



```
日志
[1] 2021-03-03 15:01:42 create image log file!
[2] 2021-03-03 15:01:42 image type: other
[3] 2021-03-03 15:01:42 add the task of creating image to the queue, it may take a long time, please wait..., if there is a problem, the progress bar and log will show the exception reason.
[4] 2021-03-03 15:01:44 start to perform the task of creating image!
[5] 2021-03-03 15:01:44 start to build Dockerfile, it may take a long time, please wait..., if there is a problem, the progress bar and log will show the exception reason!
[6] 2021-03-03 15:01:44 Step 1/3 : FROM 100.7.36.88:5000/other/ubunut18.04-wjy01:v1
[7] 2021-03-03 15:01:44 --> 81b44d980821
[8]
[9] 2021-03-03 15:01:44 Step 2/3 : MAINTAINER AUTO_TEST
[10] 2021-03-03 15:01:44 --> Using cache
[11]
[12] 2021-03-03 15:01:44 --> c1301e2a53dc
[13]
[14] 2021-03-03 15:01:44 Step 3/3 : CMD echo hello world...
[15] 2021-03-03 15:01:44 --> Using cache
[16]
[17] 2021-03-03 15:01:44 --> 106970eb9fca
[18]
[19] 2021-03-03 15:01:44 Successfully built 106970eb9fca
[20]
[21] 2021-03-03 15:01:44 Successfully tagged 100.7.36.88:5000/other/ubuntu18.04-inspur:test
[22]
[23] 2021-03-03 15:01:44 new image info: 100.7.36.88:5000/other/ubuntu18.04-inspur:test
[24] 2021-03-03 15:01:44 Dockerfile build image has been completed, and start to configure the tini component!
[25] 2021-03-03 15:01:46 the tini component has been configured, and started to push the image to the harbor repository!
[26] 2021-03-03 15:02:11 push image to the harbor repository success!
```

成功后，点击【镜像管理】，显示制作的镜像信息。

镜像管理 使用指南

All PyTorch TensorFlow Caffe MxNet PaddlePaddle Other

全部 个人 组 公共 最近使用时间(Z-A) [搜索] + 创建 导入 传输列表

- ubuntu1...** (2021-06-08)
100.2.126.15:5000/other/ubuntu18.04...
个人 | 291.5MB | 0次
test
最近使用时间: 2021-06-08 08:48:09
上传者: inspur
备注:
- Caffe caffe** (2021-05-28)
100.2.126.15:5000/caffe/caffe
公共 | 2.1GB | 44次
19.06-py2
最近使用时间: 2021-06-07 17:38:46
上传者: admin
备注:
- tensorflow tensorflow** (2021-05-28)
100.2.126.15:5000/tensorflow/tensorflow
公共 | 4.8GB | 19次
20.09-py3-cuda11
最近使用时间: 2021-06-07 17:18:14
上传者: admin
备注:
- ubuntu1...** (2021-05-28)
100.2.126.15:5000/other/ubuntu18.04...
公共 | 291.5MB | 4次
latest
最近使用时间: 2021-06-04 11:25:43
上传者: admin
备注:
- pytorch pytorch** (2021-05-28)
100.2.126.15:5000/pytorch/pytorch
公共 | 3.6GB | 6次
20.08-py3-cuda11
最近使用时间: 2021-05-31 15:46:12
上传者: admin
备注:
- tensorflow tensorflow** (2021-05-28)
100.2.126.15:5000/mlu/tensorflow
公共 | 1.1GB | 0次
1.3.0-tf1-ubuntu16.04-py3
最近使用时间: 2021-05-28 17:22:03
上传者: admin
备注:
- pytorch pytorch** (2021-05-28)
100.2.126.15:5000/mlu/pytorch
公共 | 2.7GB | 0次
0.14.1-ubuntu16.04
最近使用时间: 2021-05-28 17:16:17
上传者: admin
备注:
- torch-se...** (2021-05-28)
100.2.126.15:5000/serving/torch-server
公共 | 2.8GB | 0次
0.3.0-gpu
最近使用时间: 2021-05-28 17:02:52
上传者: admin
备注:

内部导入镜像

点击【镜像管理】->【导入】，弹出导入界面

The screenshot displays the AIStation interface for image management. At the top, there are tabs for different frameworks: All, PyTorch, TensorFlow, Caffe, MxNet, PaddlePaddle, and Other. Below these are filters for '全部' (All), '个人' (Personal), '组' (Group), and '公共' (Public). A search bar and a '镜像排序' (Image Sort) dropdown are also present. On the right, there are three buttons: '+ 创建' (Create), '导入' (Import), and '传输列表' (Transfer List). A red arrow points to the '导入' button. The main area shows a grid of image cards, each with a logo, name, date, ID, size, and usage count. The cards include 'caffe', 'ubuntu1...', 'tensorflow', 'pytorch', 'tensorflow', 'pytorch', 'torch-se...', and 'torch-se...'.

默认选择内部导入，点击文件夹图标，普通用户默认进入用户目录，选择镜像 tar、tar.gz 或者 tgz 包，内部导入只能导入 docker save 保存的镜像包。

导入

* 导入方式 内部导入 外部导入

* 选择镜像文件

* 镜像名称

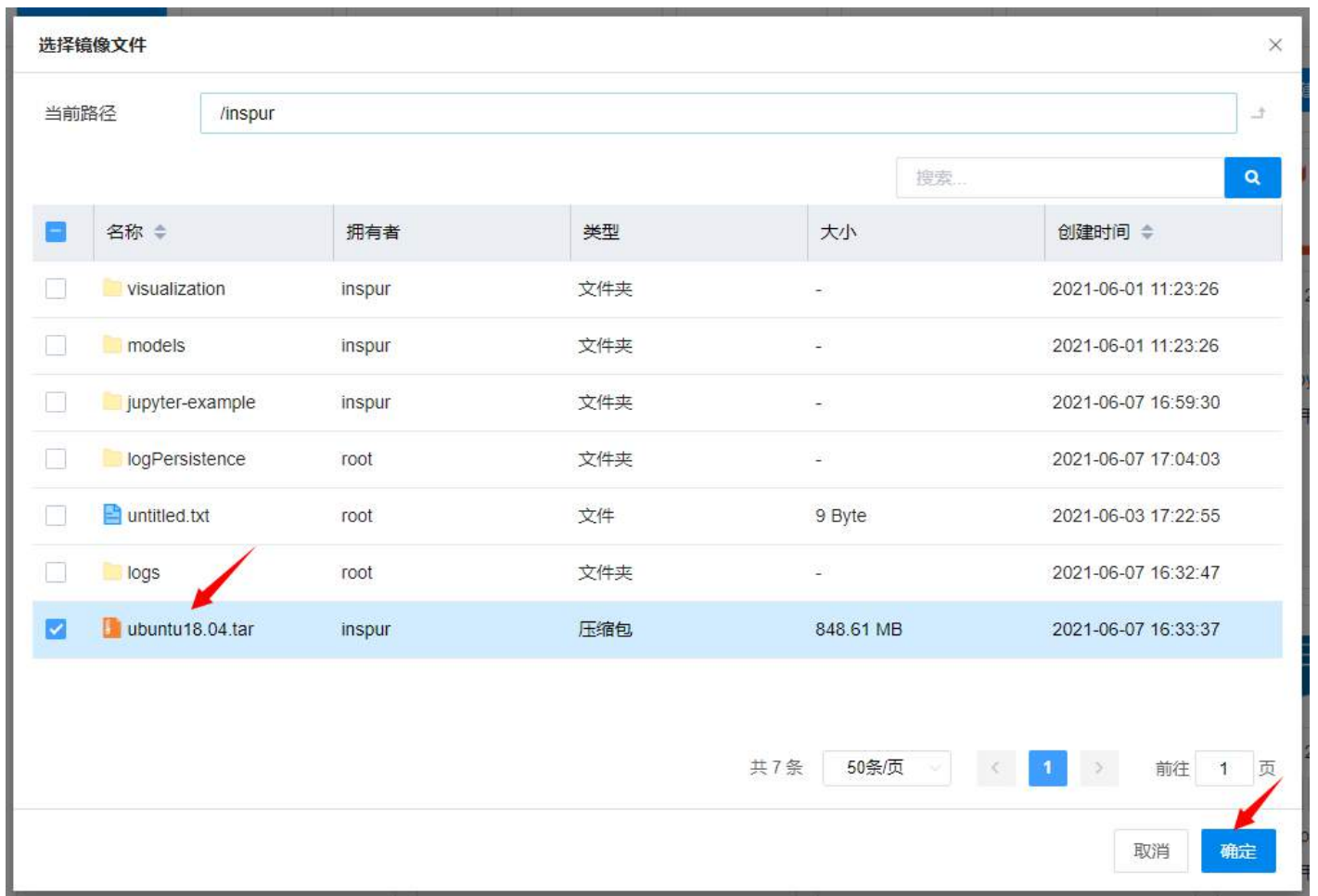
* 镜像类型 请选择

* 标签

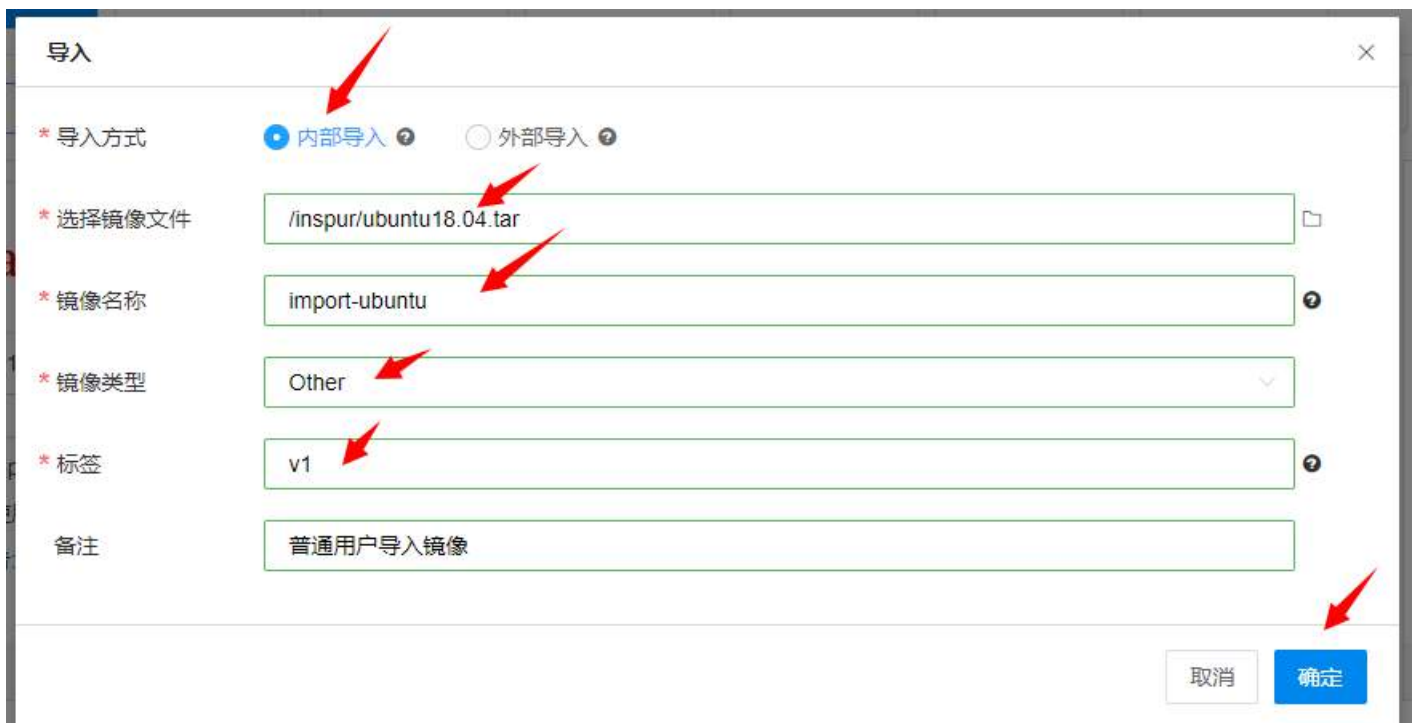
备注

取消 确定

注意：镜像名称和标签需要满足 Docker 官方规范，只能包括小写字母、数字、下划线 (_)、连接线 (-)、反斜线 (/)，且只能使用小写字母或数字开头，并且特殊字符（下划线、连接线、反斜线）不能连续使用。弹出选择镜像文件界面，选择镜像 tar 包，点击【确定】



选择 tar 包后，输入镜像名称、镜像标签或备注信息，点击【确定】，镜像名称和标签不支持大写。



注意：镜像名称和标签需要满足 Docker 官方规范，只能包括小写字母、数字、下划线 (_)、连接线 (-)、反斜线 (/)，且只能使用小写字母或数字开头，并且特殊字符（下划线、连接线、反斜线）不能连续使用。点击【传输列表】，查看导入镜像的进度信息。点击进度列表的日志文件图标，可以查看导入镜像的日志记录。如果出现异常，进度列表会显示异常原因。支持单条或者批量删除状态为成功和失败的进度记录。

传输列表

✕

删除

<input type="checkbox"/>	镜像名称	状态	异常原因	操作类型	进度	排队位置	创建时间	完成时间	操作
<input type="checkbox"/>	100.2.126....	成功	-	内部镜像...	100%	-	2021-06-0...	2021-06-0...	删除 日志
<input type="checkbox"/>	100.2.126....	成功	-	导出镜像	100%	-	2021-06-0...	2021-06-0...	删除 日志

当前选中 0 条

共 2 条

50条/页

<

1

>

前往

1

页

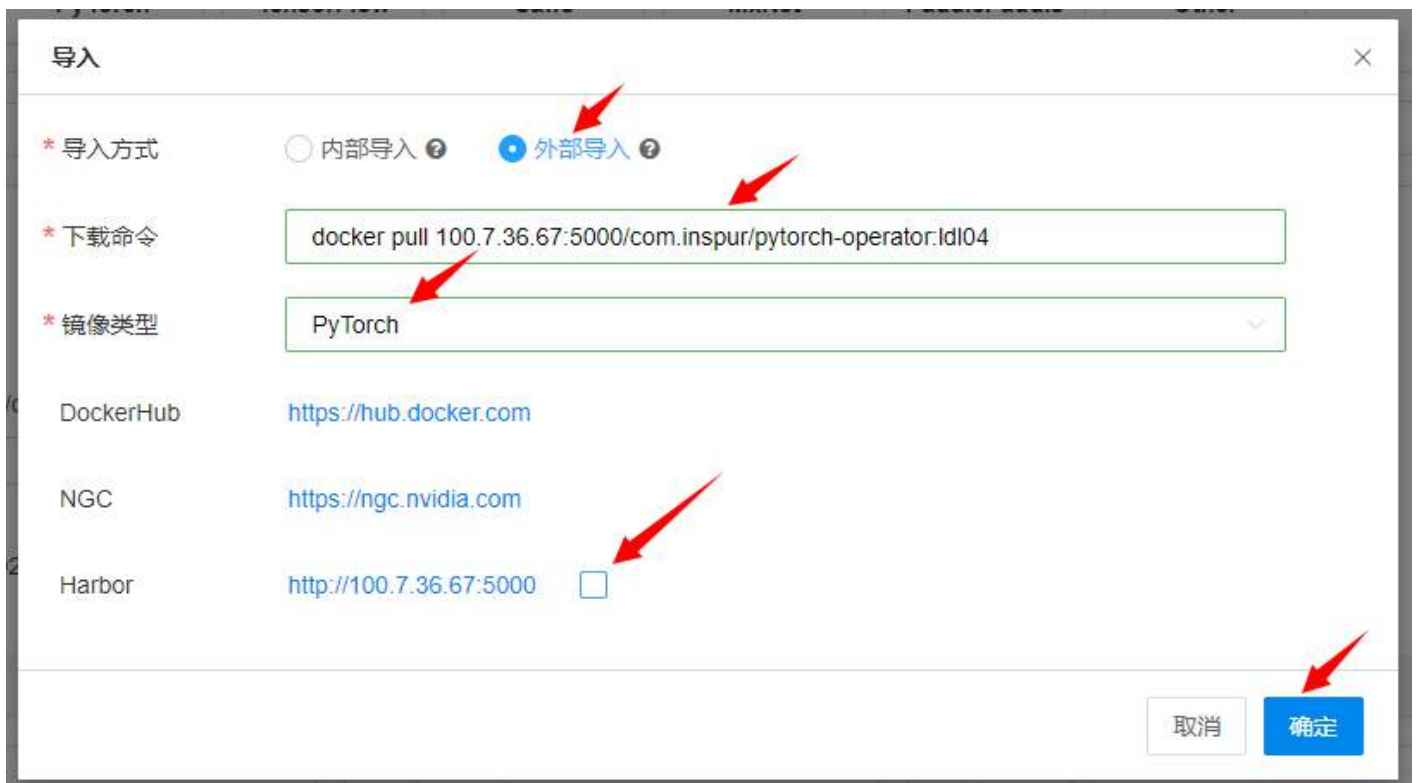
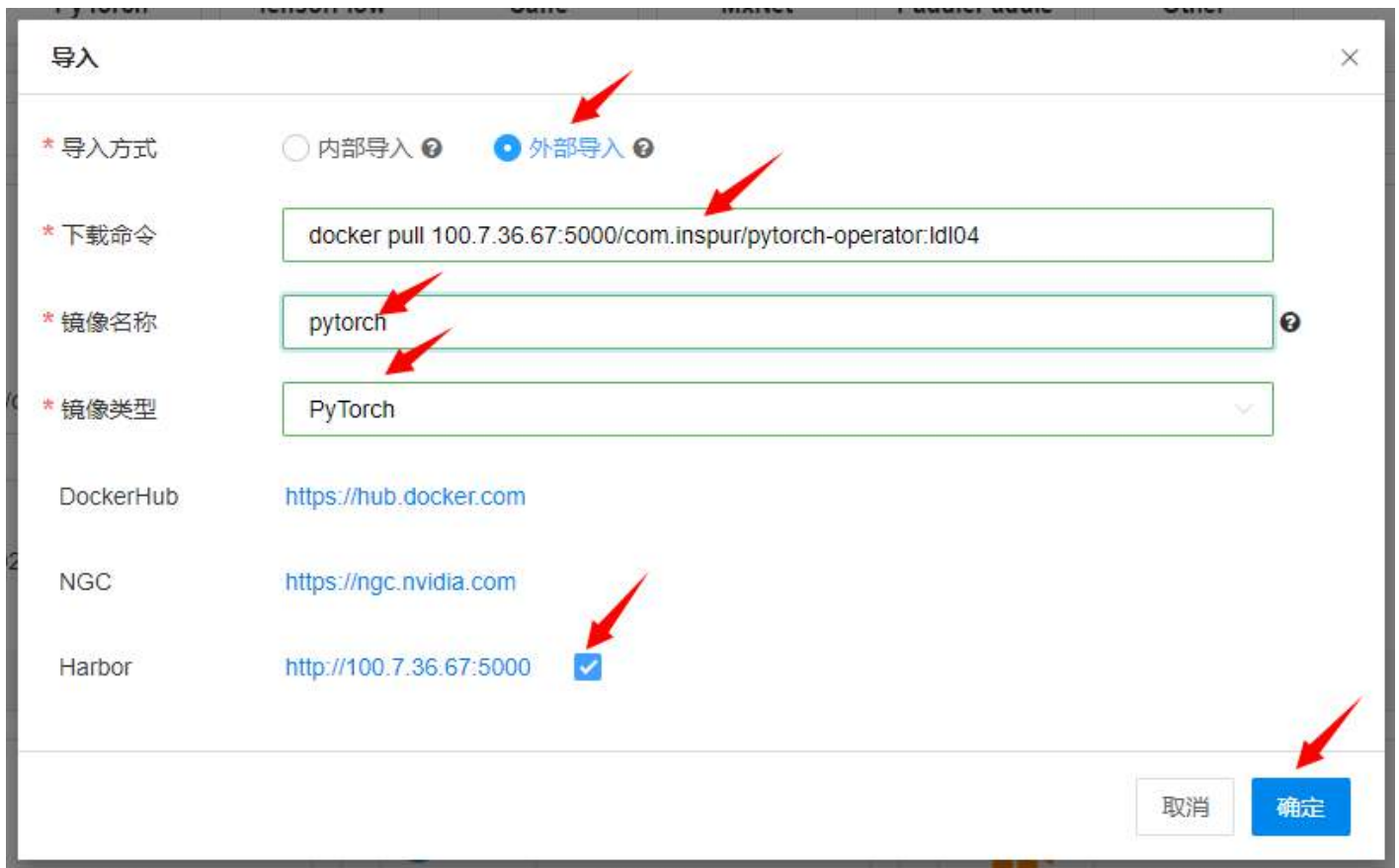
导入完成，点击【镜像管理】，显示导入的镜像

The screenshot displays the 'Image Management' (镜像管理) page in AIStation. At the top, there are tabs for different frameworks: All, PyTorch, TensorFlow, Caffe, MxNet, PaddlePaddle, and Other. Below these are filters for '全部' (All), '个人' (Personal), '组' (Group), and '公共' (Public). A dropdown menu shows '最近使用时间(Z-A)' (Sort by last used time Z-A). There are buttons for '+ 创建' (Create), '导入' (Import), and '传输列表' (Transfer List).

The main area contains a grid of image cards. Each card shows the image name, a date, a size, and usage statistics. A red arrow points to the 'import-u...' image. Other visible images include 'caffe', 'tensorflow', 'ubuntu1...', 'pytorch', 'tensorflow', 'pytorch', and 'torch-se...'. Each card also includes a '备注' (Remarks) field and a '上传者' (Uploader) field.

外部镜像导入

普通用户点击【镜像管理】，点击【导入】按钮，点击【外部导入】，【下载命令】输入 docker pull 镜像命令，可以去配置的外部 harbor 仓库、Docker Hub 或者 NGC 官方网站复制 pull 镜像命令。如果勾选配置的外部 harbor 仓库，则需要输入镜像名称，未勾选，则不需要输入镜像名称。点击【确定】，【传输列表】可以查看镜像导入进度，同时，点击日志图标，可以查看外部导入镜像的日志记录，如果导入失败，进度条会显示异常原因。支持单条或者批量删除状态为成功和失败的进度记录。



注意：外部导入，需要连接外网，拉取的镜像源最好是国内镜像源，如果是国外源，可能会有限制，导致 pull 过程缓慢，需要花费很长时间，严重时，会出现 pull 失败现象。

模型管理

AIStation 平台提供了统一的模型管理能力，集中管理在训练任务中得到的模型、用户本地开发的模型、外部平台输出的模型。模型管理提供统一的导入功能入口，可以方便把本地和外部模型导入到模型管理系统中。在模型管理中提供在线测试和离线测试两种方式进行模型的测试，测试过程提供详细的日志信息查看，测试完成后提供统一的模型发布功能，为部署模型做准备。

导入模型

在训练任务完成后，AIStation 会自动把模型文件保存到用户家目录中，能够方便地导入模型管理中：点击【模型管理】->【导入】，进入导入界面。导入主要包括四大部分：基本信息、模型文件、参数信息、评估信息。

其中名称、版本、场景、开发/训练文件为必填项，界面中标红色标记的均为必填项。其余参数，包括数据、镜像、脚本、模型参数（batch_size、learning_rate、weight_decay、momentum）、评估信息等可选填。用户通过选择用户家目录下的开发/训练文件，将该模型文件导入。

示例中用户选择的文件为本次训练任务中得到的模型文件。



导入模型的名称只接受汉字、英文字母、数字、下划线,不能以下划线开头,版本为整数。

导入模型的场景参数,默认内置了 image classification(图像分类)、object detection(物体检测)、semantic segmentation(语义分割) 和 other(其他) 四类。可通过更改 yml 配置文件,并重启服务,添加或更改。

导入模型的脚本参数只可以选择以.py 和.prototxt 结尾的脚本文件。

模型参数中 batch_size, learning_rate、weight_decay、momentum 可根据实际情况填写。若填写,必须为数字,其中 batch_size 是整数,范围为 [0-100000],其余参数支持的数据精度为小数点前最多 5 位,小数点后最多 16 位。

评估信息中的预处理脚本支持任意 linux 下的文件,召回率、准确率、精确率、f1 值可根据实际情况填写。若填写,必须为数字,参数精度为小数点前最多 5 位,小数点后最多 16 位。

The screenshot shows the 'Model Test Instance' configuration page in AIStation. It includes the following sections:

- 基本信息 (Basic Information):** Name (test), Version (2), Scenario (image classification), Dataset (Mnist_data, MNIST_caffe, MNIST_pytorch, cifar10, cifar10_caffe), Model (100.2.126.46.5000/caffe/caffe-19.06-py2), Script (zzz/test.py).
- 模型文件 (Model Files):** Development/Training File (zzz.py), Description.
- 参数信息 (Parameter Information):** batch_size (128), learning_rate (0.1), weight_decay (0.01), momentum (0.9), Other Parameters.
- 评估信息 (Evaluation Information):** Dataset (Mnist_data, MNIST_caffe, MNIST_pytorch, cifar10, cifar10_caffe), Model (100.2.126.46.5000/caffe/caffe-19.06-py2), Test Script (zzz/test.html), Accuracy (0.91), Precision (0.92), Recall (0.9), F1 Score (0.9).

Buttons for '取消' (Cancel) and '确定' (Confirm) are located at the bottom right.

点击确定后，进入传输列表。

对于模型较大的文件，传输列表用于展示操作（导入、导出、删除、发布）的进度，同一用户的同类型操作排队处理。传输列表进度删除，导入失败后二次导入等功能详见【模型传输列表】章节。

传输列表



名称	版本	操作类型	状态	操作
test	1	导入	完成	🗑️ 🔄

对于外部模型的导入（即非本平台训练产生的模型文件），若模型小于 1GB，可通过文件管理上传至用户家目录，若模型大于 1GB，可通过 xftp 登录到后台，传输至用户家目录，然后再将该外部模型导入模型管理。

共享模型

模型的共享属性有三种：私有、组、全局。

普通用户点击【模型管理】，普通用户可以共享私有和自己组内模型。

选择要共享的模型，点击右上方【共享】按钮，支持批量操作。

The screenshot shows the '模型管理' (Model Management) interface. At the top, there are search filters and a '共享' (Share) button. Below is a table of models with columns for Name, Version, Creator, Is Public, Status, Sharing, and Creation Time. The '共享' column shows '组' (Group) or '全局' (Global) for selected models. A dialog box titled '共享' (Share) is open, showing radio buttons for '组' (Group) and '全局' (Global), with '组' selected. At the bottom of the dialog are '取消' (Cancel) and '确定' (Confirm) buttons.

名称	版本	发布者	是否公开	状态	共享	创建时间	操作
Ada	4	zyh	否	未发布	组	2021-11-29 15:30:33	自 自 自
1-161	6	klz	否	未发布	全局	2021-12-09 15:03:00	自 自 自
zz	1	zyh	否	未发布	全局	2021-11-29 16:09:16	自 自 自
memememem	1	klz	否	未发布	全局	2021-12-09 15:31:14	自 自 自
1-008	2	zyh	否	未发布	全局	2021-11-29 15:30:02	自 自 自
1601	1	zyh	否	未发布	组	2021-11-29 15:44:32	自 自 自
ss	1	zyh	否	未发布	私有	2021-12-29 19:46:36	自 自 自

取消共享

取消共享操作和模型共享操作互逆。

普通用户点击【模型管理】，可以将自己共享到组或全局的模型取消共享。支持批量操作。

The screenshot shows the '模型管理' (Model Management) interface. The '取消共享' (Cancel Share) button is highlighted in the top right corner. The table below is identical to the one in the previous screenshot, showing the current sharing status of various models.

名称	版本	发布者	是否公开	状态	共享	创建时间	操作
Ada	4	zyh	否	未发布	组	2021-11-29 15:30:33	自 自 自
1-161	6	klz	否	未发布	全局	2021-12-09 15:03:00	自 自 自
zz	1	zyh	否	未发布	全局	2021-11-29 16:09:16	自 自 自
memememem	1	klz	否	未发布	全局	2021-12-09 15:31:14	自 自 自
1-008	2	zyh	否	未发布	全局	2021-11-29 15:30:02	自 自 自
1601	1	zyh	否	未发布	组	2021-11-29 15:44:32	自 自 自
ss	1	zyh	否	未发布	私有	2021-12-29 19:46:36	自 自 自

删除模型

普通用户点击【模型管理】，普通用户可以删除拥有者是自己的模型。支持批量操作。



点击确定，二次确认后，进入传输列表。

传输列表

名称	版本	操作类型	状态	操作
as	1	删除	完成	🗑️ 🔄
as	1	导入	完成	🗑️ 🔄

导出模型

普通用户在【模型管理】中，点击导出按钮，并在弹出框中自定义 tar 包名称，导出模型 tar 包至用户家目录下。



导出

* 名称

取消

确定

传输列表



名称	版本	操作类型	状态	操作
test	2	导出	完成	
test	1	删除	完成	
test	2	导入	完成	
test	1	导入	完成	

模型测试

模型测试分为离线测试和在线测试，离线测试是根据用户自定义的镜像、模型输入、模型文件发起的测试。在线测试是通过平台内置的 `servimg` 引擎发起的测试。下面分别说明两种测试方法。普通用户点击【模型管理】，点击模型列表中的“发起测试”按钮，发起模型测试任务，用户可在弹出框自定义模型测试的任务属性。

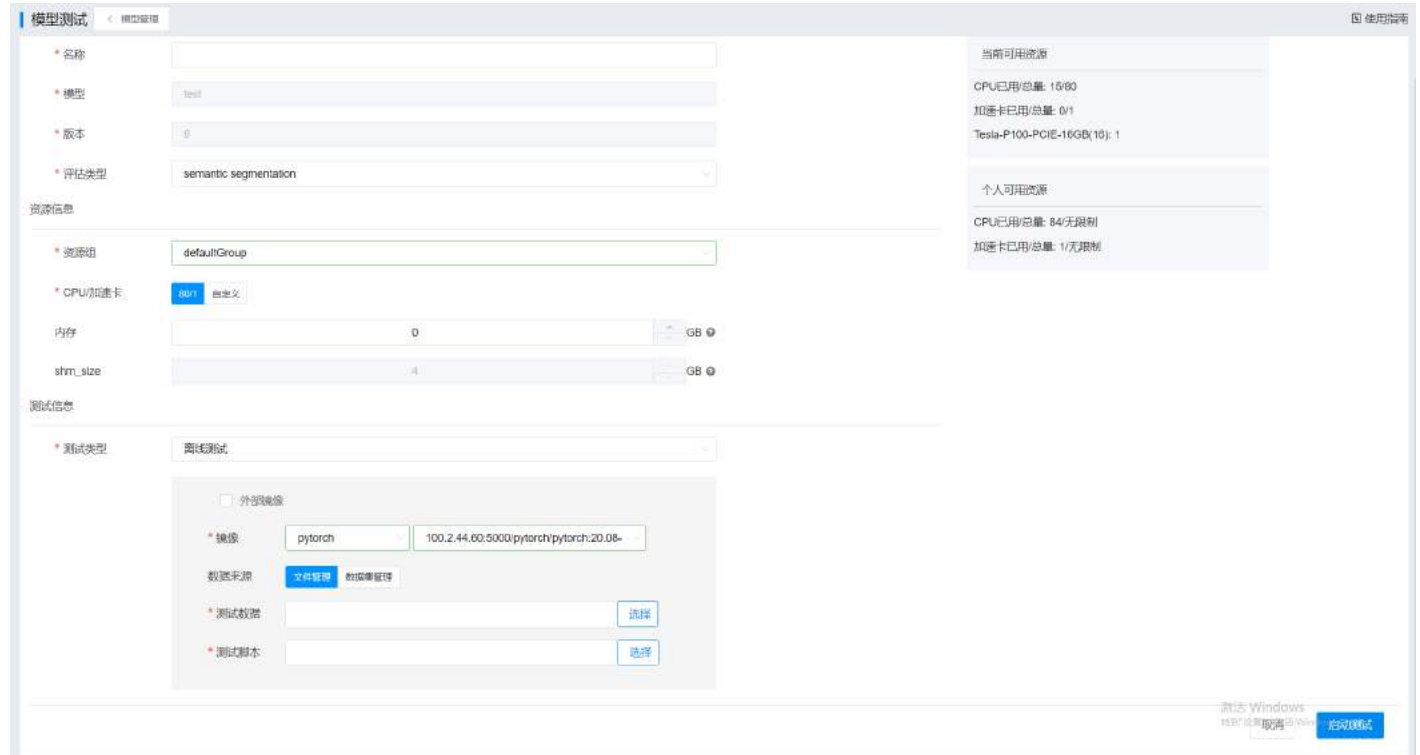
名称	版本	所有者	是否测试	状态	共享	创建时间	操作
test	6	klz	是	未发布	全局	2021-12-09 15:03:00	
mmmmmmmm	2	klz	否	未发布	私有	2021-12-08 15:28:00	
auto_workflow_wf_test_1_7419	17	klz	否	未发布	私有	2021-12-02 21:09:51	
auto_workflow_test_wf_7629	3	klz	否	未发布	私有	2021-12-08 17:13:30	
zz	7	klz	否	未发布	私有	2021-12-08 16:10:25	
aaa	666	klz	否	未发布	私有	2021-12-08 13:45:25	
testxxx	3	klz	否	未发布	组	2021-11-30 15:19:08	
auto_workflow_wf_1_1648	2	klz	否	未发布	私有	2021-12-03 18:51:51	
auto_workflow_wf_1_5008	10	klz	否	未发布	私有	2021-12-24 16:57:30	
moduName	1	klz	否	未发布	私有	2021-11-30 15:46:00	
auto_workflow_wf_1_7120	14	klz	否	未发布	私有	2021-12-21 13:53:00	
auto_workflow_wf_1_4988	2	klz	否	未发布	私有	2021-12-02 20:09:30	

离线测试

用户可以发起离线测试，用以测试模型的精确度等指标。用户在进入发起模型测试页面后，在属性“测试类型”下拉框中选择“离线测试”，然后选择相应的测试脚本以及测试数据，点击【启动测试】按钮进行模型测试任务提交。提交后的模型测试任务将会展示在模型测试实例页面。

【注意事项】：测试脚本中需要手动更改模型加载路径，示例：比如您的模型文件为 `my_mnist.h5`，你的

模型测试任务名称为 `mnist_test`, 则模型在启动容器中的路径为 `/mnist_test/my_mnist.h5`。



在模型实例页面会展示模型测试任务列表，每条测试任务后面都有“日志”按钮，点击“日志”按钮，可以实时查看模型测试任务日志。



日志

```

[18] 2021-02-25 08:54:57.074688: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x2e553a0 initialized for platform CUDA (this does not guarantee that XLA will be used). Devices:
[19] 2021-02-25 08:54:57.074771: I tensorflow/compiler/xla/service/service.cc:176] StreamExecutor device (0): Tesla V100-PCIE-32GB, Compute Capability 7.0
[20] 2021-02-25 08:54:57.083205: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1634] Found device 0 with properties:
[21] name: Tesla V100-PCIE-32GB major: 7 minor: 0 memoryClockRate(GHz): 1.38
[22] pciBusID: 0000:3b:00.0
[23] 2021-02-25 08:54:57.083297: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudart.so.11.0
[24] 2021-02-25 08:54:57.083362: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcublas.so.11
[25] 2021-02-25 08:54:57.083410: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcufft.so.10
[26] 2021-02-25 08:54:57.083457: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcurand.so.10
[27] 2021-02-25 08:54:57.083505: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcusolver.so.10
[28] 2021-02-25 08:54:57.083552: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcusparsparse.so.11
[29] 2021-02-25 08:54:57.083600: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudnn.so.8
[30] 2021-02-25 08:54:57.104206: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1762] Adding visible gpu devices: 0
[31] 2021-02-25 08:54:57.104319: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudart.so.11.0
[32] 2021-02-25 08:54:57.868121: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1175] Device interconnect StreamExecutor with strength 1 edge matrix:
[33] 2021-02-25 08:54:57.868184: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1181] 0
[34] 2021-02-25 08:54:57.868201: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1194] 0: N
[35] 2021-02-25 08:54:57.875946: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1320] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 30132 MB memory) -> physical GPU (device:
0, name: Tesla V100-PCIE-32GB, pci bus id: 0000:3b:00.0, compute capability: 7.0)
[36] WARNING:tensorflow:No training configuration found in save file: the model was "not" compiled. Compile it manually.
[37] WARNING:tensorflow:From /inspur/serving_models/tensorflow-serving/save_model_test/model_test.py:11: The name tf.train.AdamOptimizer is deprecated. Please use tf.compat.v1.train.AdamOptimizer instead.
[38]
[39] Model: "sequential"
[40]
[41] Layer (type) Output Shape Param #
[42] =====
[43] conv2d (Conv2D) (None, 24, 24, 32) 832
[44]
[45] max_pooling2d (MaxPooling2D) (None, 12, 12, 32) 0
[46]
[47] conv2d_1 (Conv2D) (None, 6, 6, 64) 100416
[48]
[49] max_pooling2d_1 (MaxPooling2D) (None, 3, 3, 64) 0
[50]
[51] flatten (Flatten) (None, 576) 0
[52]
[53] dense (Dense) (None, 576) 332352
[54]
[55] dense_1 (Dense) (None, 10) 5770
[56] =====
[57] Total params: 439,370
[58] Trainable params: 439,370
[59] Non-trainable params: 0
[60]
[61] 2021-02-25 08:54:58.799264: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcublas.so.11
[62] 2021-02-25 08:54:59.541321: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Successfully opened dynamic library libcudnn.so.8
[63] [[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]]

```

在线测试

用户使用在线模型测试发起 serving 服务，需要在发起模型测试时选择“在线测试”选项。平台会暴露出 8500、8501 的映射端口用于 grpc 和 restful 服务访问。

模型管理 > 模型测试

模型列表 | 模型测试实例

基本信息

* 名称:

* 模型: mnsit

* 版本: 33

* 评估类型: semantic segmentation

资源信息

* 资源组: defaultGroup

* CPU/加速卡: 16/1 | 32/2 | 自定义

内存: GB

shm_size: GB

测试信息

* 测试类型: 在线测试

外部镜像

* 引擎: tensorflow-serving | cpu - 100.2.126.15:5000/serving/tensorflr

模型管理 > 模型测试实例

模型列表 | 模型测试实例

测试任务

已加载任务

任务名称	状态	运行任务	等待任务	加速卡已用	CPU已用	引擎	地址	启动内容	端口	提交时间	操作
训练	运行中	1	0	1	4	tensorflow-serving	100.2.126.15:5000/serv...		8501->30504,8500->30...	2021-06-05 19:35:16	删除

共 1 页 | 列表 | 1 页

```
[1] 2021-02-25 09:15:16.305198: I tensorflow_serving/model_servers/server.cc:88] Building single TensorFlow model file config: model_name: mnist model_base_path: /mnist
[2] 2021-02-25 09:15:16.305518: I tensorflow_serving/model_servers/server_core.cc:464] Adding/updating models.
[3] 2021-02-25 09:15:16.305538: I tensorflow_serving/model_servers/server_core.cc:587] (Re-)adding model: mnist
[4] 2021-02-25 09:15:16.406130: I tensorflow_serving/core/basic_manager.cc:740] Successfully reserved resources to load servable {name: mnist version: 1}
[5] 2021-02-25 09:15:16.406170: I tensorflow_serving/core/loader_harness.cc:66] Approving load for servable version {name: mnist version: 1}
[6] 2021-02-25 09:15:16.406187: I tensorflow_serving/core/loader_harness.cc:74] Loading servable version {name: mnist version: 1}
[7] 2021-02-25 09:15:16.406249: I external/org_tensorflow/tensorflow/cc/saved_model/reader.cc:32] Reading SavedModel from: /mnist/1
[8] 2021-02-25 09:15:16.409008: I external/org_tensorflow/tensorflow/cc/saved_model/reader.cc:55] Reading meta graph with tags { serve }
[9] 2021-02-25 09:15:16.409051: I external/org_tensorflow/tensorflow/cc/saved_model/reader.cc:93] Reading SavedModel debug info (if present) from: /mnist/1
[10] 2021-02-25 09:15:16.409225: I external/org_tensorflow/tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX2 AVX512F FMA
[11] To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
[12] 2021-02-25 09:15:16.470521: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:206] Restoring SavedModel bundle.
[13] 2021-02-25 09:15:16.471809: I external/org_tensorflow/tensorflow/core/platform/profile_utils/cpu_utils.cc:112] CPU Frequency: 2200000000 Hz
[14] 2021-02-25 09:15:16.519344: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:190] Running initialization on SavedModel bundle at path: /mnist/1
[15] 2021-02-25 09:15:16.525869: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:277] SavedModel load for tags { serve }; Status: success: OK. Took 119620 microseconds.
[16] 2021-02-25 09:15:16.526718: I tensorflow_serving/servables/tensorflow/saved_model_warmup_util.cc:59] No warmup data file found at /mnist/1/assets.extra/tf_serving_warmup_requests
[17] 2021-02-25 09:15:16.527023: I tensorflow_serving/core/loader_harness.cc:87] Successfully loaded servable version {name: mnist version: 1}
[18] 2021-02-25 09:15:16.559910: I tensorflow_serving/model_servers/server.cc:371] Running gRPC ModelServer at 0.0.0.0:8500 ...
[19] [warn] getaddrinfo: address family for nodename not supported
[20] 2021-02-25 09:15:16.587186: I tensorflow_serving/model_servers/server.cc:391] Exporting HTTP/REST API at localhost:8501 ...
[21] [evhttp_server.cc : 238] NET_LOG: Entering the event loop ...
```



当日志显示 serving 已经启动时，则证明 serving 服务可用，此时可以通过平台向外暴露的端口进行服务访问。

```
[root@node1 ~]# curl http://100.7.36.88:49081/v1/models/mnist
{"model_version_status": [
  {
    "version": "1",
    "state": "AVAILABLE",
    "status": {
      "error_code": "OK",
      "error_message": ""
    }
  }
]}
```

发布模型

普通用户默认没有发布权限，需要管理员通过用户管理界面赋予该用户模型发布的权限。

用户	姓名	角色	用户组	CPU已用(核)	GPU已用(卡)	磁盘已用/总	下载状态	紧急任务	模型发布	状态	最近登录时间	操作
admin	预置系统管理员	系统管理员		-	-	-	成功	否	否	正常	2021-06-07 18:43:46	<ul style="list-style-type: none"> 开启下载 关闭下载 禁用紧急任务 禁用模型发布 开启模型发布 重置密码
inspur	igly	普通用户	default_group	4/无限制	0/无限制	470.00 MB/无限制	成功	否	否	正常	2021-06-02 17:04:06	
lsg	lsg	普通用户	default_group	0/无限制	0/无限制	0 MB/无限制	成功	否	否	正常	2021-06-02 15:41:22	
sys_user	系统用户管理专用	系统管理员		-	-	-	成功	否	否	正常	-	
xlz	xlz	普通用户	default_group	0/无限制	0/无限制	0 MB/无限制	成功	否	否	正常	2021-06-02 16:58:32	
zzz	zheng	普通用户	default_group	0/无限制	0/无限制	1.00 MB/无限制	成功	否	否	正常	2021-06-07 18:30:16	
zzz1	zheng	普通用户	zzGroup	0/无限制	0/无限制	7.00 MB/无限制	成功	否	否	正常	-	

赋权后，该普通用户需要重新登录。点击【模型管理】，选中要发布的模型，点击发布按钮。



点击确定后，二次确认，进入传输列表。

传输列表

名称	版本	操作类型	状态	操作
test	2	发布	完成	🗑️ 📄
test	2	导出	完成	🗑️ 📄
test	1	删除	完成	🗑️ 📄
test	2	导入	完成	🗑️ 📄
test	1	导入	完成	🗑️ 📄

发布成功的模型是一个 tar 包，已发布模型可以通过北向接口查询和下载。

现以 100.7.36.88 环境为例，将两个北向接口及其操作说明描述如下：

1) 已发布模型列表查询

<https://100.7.36.88:32002/istorage/v1/openapi/model/publish>

接口返回示例：

其中，modelPath 为可下载模型的路径；url 为下载模型的 http 请求。

```
"image": "",
"script": "",
"batchSize": null,
"learningRate": null,
"weightDecay": null,
"momentum": null,
"otherParam": "",
"assessmentInfo": [],
"path": "/zyh128/11.tar.gz;/zyh128/100.7.36.88_5000_other_test-admin_v18.tar",
"publishStatus": 1,
"sceneClassification": "image classification",
"modelPath": "/mnt/inspurfs/model/02601156-f847-4307-987c-6dc13859311e.tar",
"createTime": 1613705452000,
"url": "https://100.7.36.88:32002/istorage/v1/openapi/file/download?filePath=/mnt/inspurfs/model/02601156-f847-4307-987c-6dc13859311e.tar",
},
{
  "id": "250a6755-425c-4aab-98af-4dbadb696158",
  "userId": "51219d11cf5b4eb49804a31547ee0302",
  "account": "zyh128",
  "modelName": "666q",
```

2) 下载模型文件 <https://100.7.36.88:32002/istorage/v1/openapi/file/download?filePath=xxxxx>
filePath 参数传入 1) 中返回的 modelPath 即可。

操作说明：上述两个北向接口需要传入 token 参数。

下载模型文件可以通过 curl 命令在后台下载，-H 参数后传入 token 参数，-o 参数后传入下载路径和名称。

```
curl -k -H "X-auth-Token:fb4acdc497584a9caa01ccf3542299f6" https:// 100.7.36.88:32002/ istorage/ v1/
openapi/ file/ download? filePath=/mnt/inspurfs/model/02601156-f847-4307-987c-6dc13859311e.tar -o /mnt/
inspurfs/user-fs/zyh128/02601156-f847-4307-987c-6dc13859311e.tar
```

模型传输列表

进入传输列表的操作有：导入、导出、删除、发布。同一用户同一操作进行排队。

对于模型较大的文件，可以展示操作进度。对于完成、失败、排队的进度，可以点击删除按钮逐条删除。进行中的进度无法删除。

对于导入失败的进度，提供二次导入操作，且只能操作一次，二次操作作为新的进度展示。

二次导入操作按钮如下：

传输列表



名称	版本	操作类型	状态	操作
qq	2	导入	失败	
test	2	发布	完成	
test	2	导出	完成	
test	1	删除	完成	
test	2	导入	完成	
test	1	导入	完成	

模型列表

可以根据名称、状态、导入时间等查询模型列表。模型列表列包括：名称、版本、拥有者、是否测试、状态、共享、创建时间、操作。可以根据创建时间进行升序或降序排列

名称	版本	拥有者	是否测试	状态	共享	创建时间	操作
test	2	zzz	否	已发布	私有	2021-06-07 18:30:03	
qq	2	zzz	否	未发布	私有	2021-06-02 16:54:41	

名称相同，版本不一样时，会折叠展示

名称	版本	拥有者	是否测试	状态	共享	创建时间	操作
qq	1	zyh	否	未发布	私有	2021-12-20 14:30:35	
qq	2	zyh	否	未发布	私有	2021-12-20 14:30:19	

编辑模型

选择模型列表中的一个模型，点击编辑按钮，输入要更改的模型信息，点击确定



节点管理

节点管理

用户点击【节点管理】页面，可以看到所属资源组的节点列表信息，包括节点名称，状态，计算状态，IP，节点加速卡的型号，BMC 地址，CPU 核数，加速卡数，内存，网络类型，交换机名称，docker 存储，所属资源组。



节点数据

用户点击【节点管理】界面，可以看到所属资源组的节点列表信息，如果想看节点详情信息，可以点击节点名称，跳转到该节点的详情页面。在详情页点击数据标签，可查看节点详情数据信息，包括：节点名称、IP、数据路径、数据大小、使用次数、使用状态、缓存时间、最近使用时间。

节点详情 < 节点管理 使用指南

数据 镜像 挂载信息 故障详情 删除

<input type="checkbox"/>	节点名称	IP	数据路径	数据大小	使用次数	使用状态	缓存时间	最近使用时间
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	127.0MB	4	已使用	2021-12-27 10:08:05	2021-12-30 13:54:06
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	210.6MB	1	已使用	2021-12-29 17:17:41	2021-12-29 17:18:02
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	0.0MB	1	未使用	2021-12-28 18:26:13	2021-12-28 18:59:28
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	127.0MB	6	未使用	2021-12-27 10:59:38	2021-12-28 14:59:28
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	24.2GB	2	未使用	2021-12-27 16:41:57	2021-12-27 16:50:24
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	24.0GB	2	未使用	2021-12-27 16:41:57	2021-12-27 16:46:24
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	235.8MB	1	未使用	2021-12-27 16:41:38	2021-12-27 16:42:54
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	0.0MB	1	未使用	2021-12-27 16:41:37	2021-12-27 16:42:24
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	353.4MB	1	未使用	2021-12-27 16:41:37	2021-12-27 16:42:24
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	210.6MB	1	未使用	2021-12-27 16:41:37	2021-12-27 16:42:24
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	68.8MB	2	未使用	2021-12-27 10:58:34	2021-12-27 16:41:54
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	0.0MB	1	未使用	2021-12-27 16:41:37	2021-12-27 16:41:54
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/cach...	127.0MB	5	未使用	2021-12-27 11:08:18	2021-12-27 15:53:23

删除数据

用户可以选择节点内的数据进行删除，只能删除未使用状态的数据集，如果删除使用状态的数据集，则提示删除失败。如果删除未使用状态的数据集，才会删除成功。如下图:

节点详情 < 节点管理 使用指南

数据 镜像 挂载信息 故障详情

[删除](#)

<input type="checkbox"/>	节点名称	IP	数据路径	数据大小	使用次数	使用状态	缓存时间	最近使用时间
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	127.0MB	4	已使用	2021-12-27 10:08:05	2021-12-30 13:54:06
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	210.6MB	1	已使用	2021-12-29 17:17:41	2021-12-29 17:18:02
<input checked="" type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	0.0MB	1	未使用	2021-12-28 16:26:13	2021-12-28 16:59:28
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	127.0MB	6	未使用	2021-12-27 10:59:38	2021-12-28 14:59:28
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	24.2GB	2	未使用	2021-12-27 16:41:57	2021-12-27 16:50:24
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	24.0GB	2	未使用	2021-12-27 16:41:57	2021-12-27 16:46:24
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	235.8MB	1	未使用	2021-12-27 16:41:38	2021-12-27 16:42:54
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	0.0MB	1	未使用	2021-12-27 16:41:37	2021-12-27 16:42:24
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	353.4MB	1	未使用	2021-12-27 16:41:37	2021-12-27 16:42:24
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	210.6MB	1	未使用	2021-12-27 16:41:37	2021-12-27 16:42:24
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	68.8MB	2	未使用	2021-12-27 10:58:34	2021-12-27 16:41:54
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	0.0MB	1	未使用	2021-12-27 16:41:37	2021-12-27 16:41:54
<input type="checkbox"/>	node1	100.2.44.60	/mnt/node-stor/.cach...	127.0MB	5	未使用	2021-12-27 11:08:18	2021-12-27 15:53:23



节点镜像

用户点击【节点管理】界面，可以看到所属资源组的节点列表信息，如果想看节点缓存的镜像，可以点击节点名称，跳转到该节点的镜像页面，再点击镜像标签，进入到该节点的镜像列表页面，如下图所示。

节点名称	IP	镜像名称	标签	大小	是否在用	上传者	创建时间	操作
node1	100.2.44.60	100.2.44.25:5000...	latest	181.0MB	已使用	admin	2021-12-30 14:13...	删除
node1	100.2.44.60	100.2.44.25:5000...	latest	914.5MB	已使用	admin	2021-12-30 13:55...	删除
node1	100.2.44.60	100.2.44.25:5000...	latest	631.7MB	已使用	admin	2021-12-30 13:41...	删除
node1	100.2.44.60	100.2.44.25:5000...	latest	692.4MB	已使用	admin	2021-12-30 13:33...	删除
node1	100.2.44.60	100.2.44.25:5000...	latest	1.2GB	已使用	admin	2021-12-30 08:26...	删除
node1	100.2.44.60	100.2.44.25:5000...	latest	2.6GB	已使用	admin	2021-12-29 20:23...	删除
node1	100.2.44.60	100.2.44.60:5000...	jupyter	4.6GB	已使用	inspur	2021-12-28 17:01...	删除
node1	100.2.44.60	caffe/caffe	19.06-py2	4.8GB	已使用	admin	2021-12-28 17:01...	删除
node1	100.2.44.60	caffe	resize	4.6GB	已使用	admin	2021-12-28 17:01...	删除
node1	100.2.44.60	tensorflow	resize	11.9GB	未使用	admin	2021-12-28 17:00...	删除

在详情页点击镜像标签，可查看节点详情镜像信息，包括：节点名称、IP、镜像名称、标签、大小、是否在用、上传者、创建时间。其中是否在用表示当前的镜像正在被任务使用。

删除镜像

平台支持单个或批量删除镜像功能，只能删除未使用的镜像，在用的镜像不能删除。删除成功时，页面提示操作成功。

节点挂载信息

用户点击【节点管理】界面，可以看到所属资源组的节点列表信息，如果想看节点详情信息，可以点击节点名称，跳转到该节点的详情页面。在详情页点击挂载信息标签，可查看节点详情挂载信息，包括：节点名称、IP、挂载路径、文件系统类型、总容量、已使用、剩余。

节点名称	IP	挂载路径	文件系统类型	总容量	已使用	剩余
node1	100.2.44.60	/	xfs	15.44 TB	822.96 GB	14.64 TB
node1	100.2.44.60	/boot	xfs	1014.00 MB	164.64 MB	849.36 MB
node1	100.2.44.60	/boot/efi	vfat	199.79 MB	11.02 MB	188.77 MB
node1	100.2.44.60	/home	xfs	30.00 TB	70.75 GB	29.93 TB
node1	100.2.44.60	/mnt/beegfs	beegfs	599.70 GB	551.00 MB	599.18 GB

故障详情

用户点击【节点管理】界面，可以看到所属资源组的节点列表信息，如果想看节点的故障详情，可以点击节点名称，进入故障详情界面，查看节点故障信息。



站内信

站内信

该模块目前用来记录所有的系统管理员删除开发环境和训练任务的通知，并且提供实时提醒功能（刷新频率为 10s）



普通用户登录平台，点击【站内信】或者用户点击右上角的站内信图标，打开站内信页面显示当前所有未读的站内信信息列表，包括：类型（开发环境运行和删除、训练任务运行、完成、失败和删除）、内容（开发环境或训练任务的信息）、状态（未读）、到达时间。

可以统一标记为已读。标记已读后，该信息自动不显示。已读信息暂时不提供历史数据查看功能（直接执行删除操作）。

工具指南

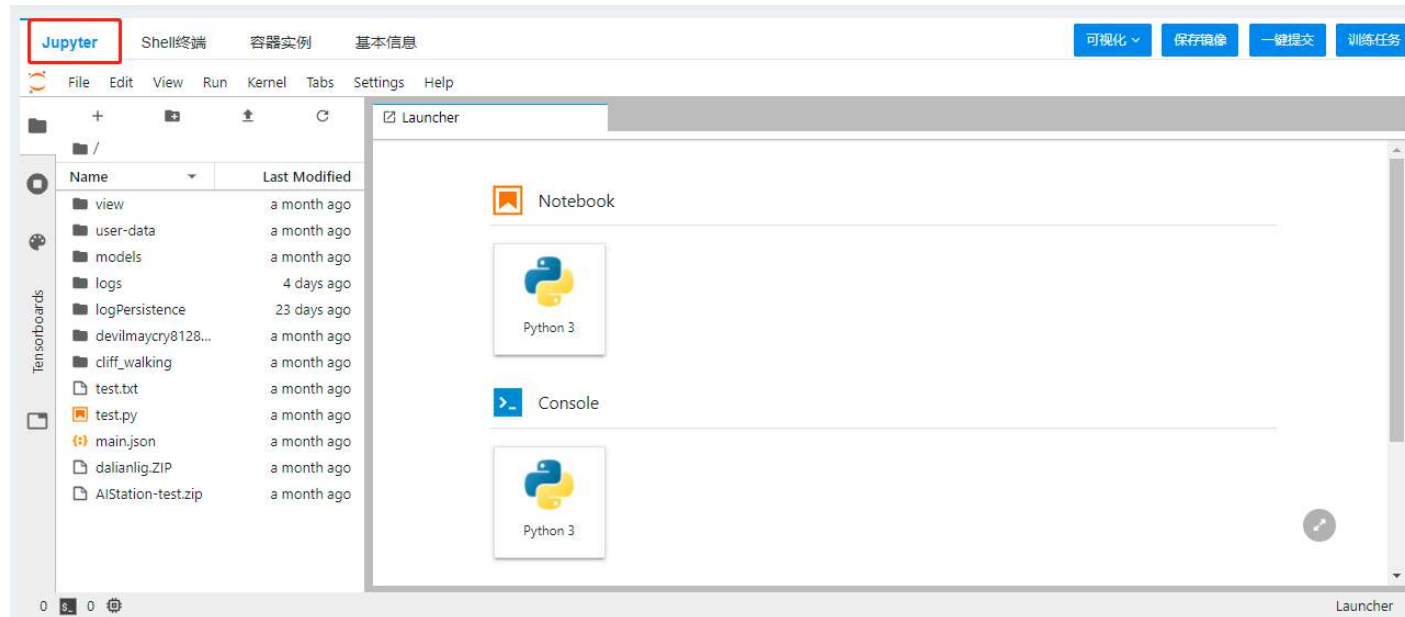
Jupyter Lab 简介及常用操作

JupyterLab 是一个交互式的开发环境，是 Jupyter Notebook 的下一代产品，可以使用它编写 Notebook、操作终端、编辑 Markdown 文本、打开交互模式、查看 csv 文件及图片等功能。可以说，JupyterLab 是

开发者们下一阶段更主流的开发环境。JupyterLab 支持更加灵活和更加强大的项目操作方式，但具有和 Jupyter Notebooks 一样的组件。

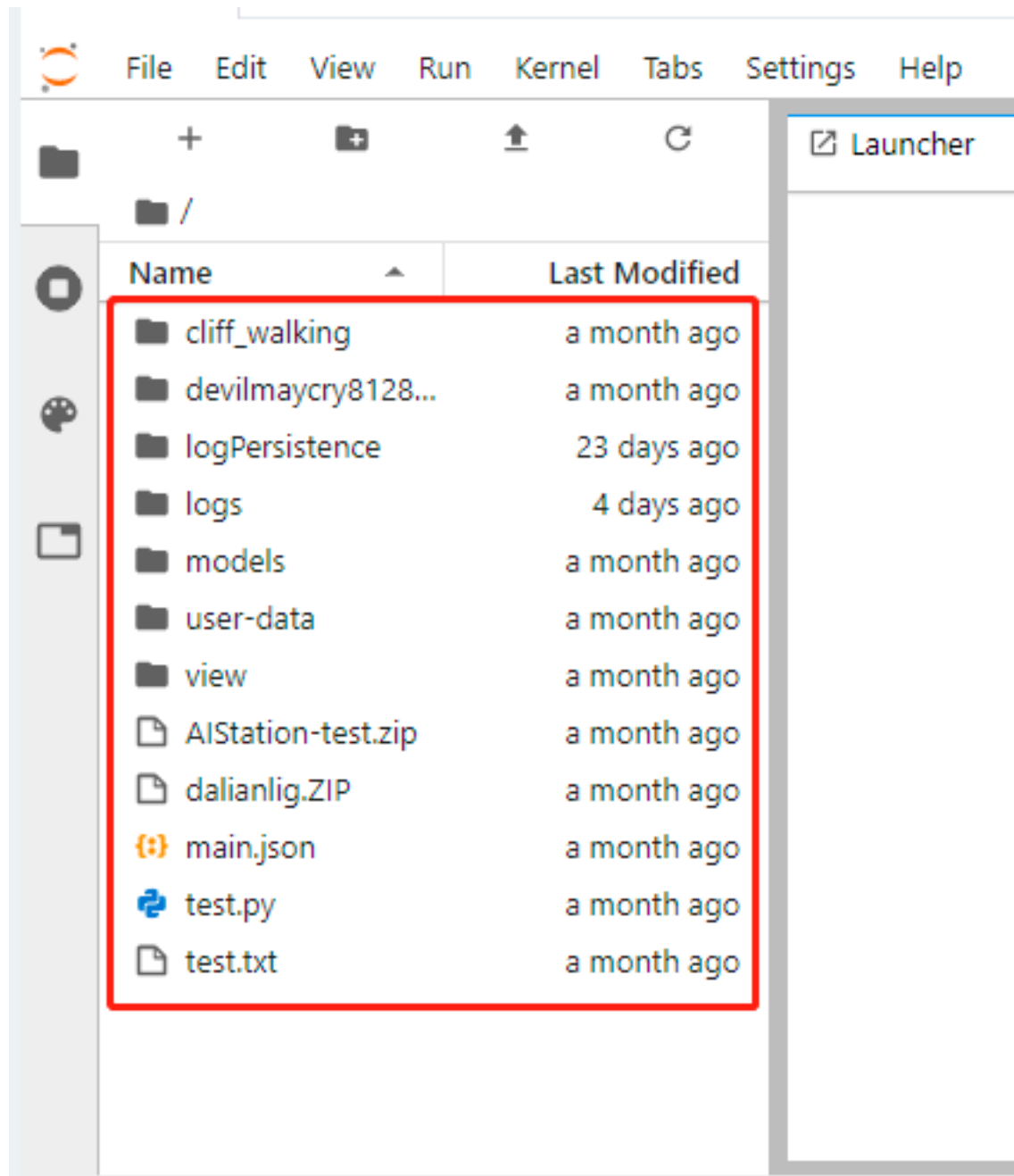
打开 Jupyter Lab

当创建的开发环境处于运行状态时，点击具体的开发环境名称，进入 JupyterLab 页面后，自动打开 Launcher 页面，如下图所示。您可以使用开源支持的所有功能，详细操作指导可参见 JupyterLab 官网文档。



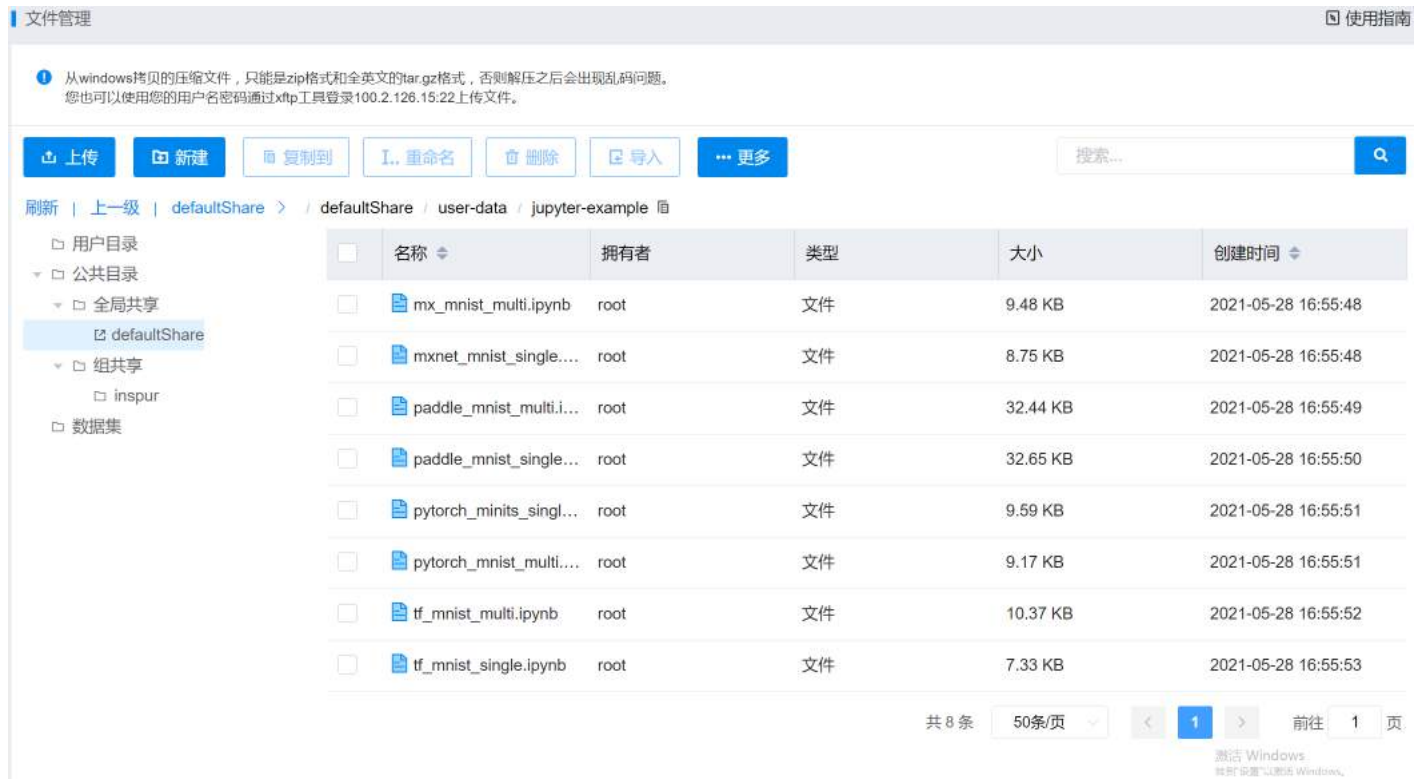
默认文件目录

左侧文件目录默认显示当前用户目录下的文件，如下图所示：

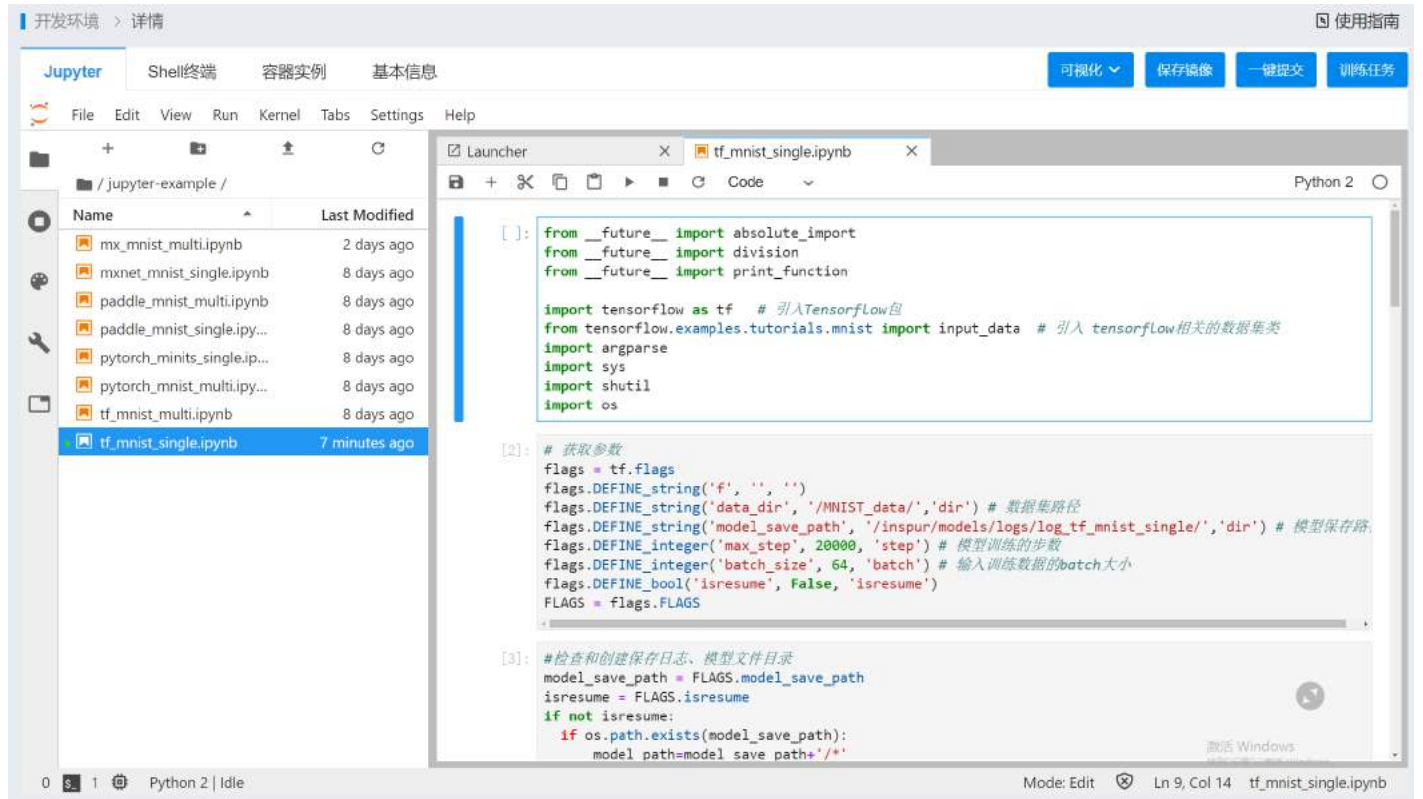


预置 Jupyter-Example 使用

平台预置的各框架相应算例的 ipynb 文件初始都保存在/defaultShare/user-data/jupyter-example 目录下，可以通过文件管理中复制到功能复制到用户目录。



当相应框架的开发环境创建完成，进入 jupyter 页面，打开相应算例，当左下角齿轮图标右侧状态为 Idle 时，可依次运行 cell 中代码，调试脚本。



Kernel 重启

在使用 JupyterLab 时，在页面左下角齿轮图标右边表示 Kernel 环境与状态，状态为 Idle 时，表示正常；当为其他状态，可点击齿轮图标进行修改。

镜像组件安装

为了适配 AIStation 平台，本文档主要介绍 openssh、JupyterLab 和 python 组件的安装。若需要安装或更新相关组件版本，请参考本文的示例。如果基础镜像 (FROM 所用镜像) 中已安装 python3，可忽略文档中 python3 的安装内容。

注意：JupyterLab 必须使用 python3 及以上版本，文档中的 Dockerfile 只支持 python3 的 JupyterLab 安装，不支持 python2。

一. Centos7 系统

1.1 Dockerfile 安装组件

1.1.1 Dockerfile 下载链接：

<https://github.com/wjyzzu/inspur-dockerfile/tree/main/base/centos> 请下载 Dockerfile 文件，此文件包含 openssh、python3 和 JupyterLab 组件安装的命令。

1.1.2 制作步骤

- (1) 根据需要，修改 FROM 中的基础镜像；建议基础镜像提前上传到 AIStation 中。
- (2) 修改好 Dockerfile，请上传到个人用户目录下。
- (3) 通过镜像管理-创建功能，制作新镜像。
- (4) 基于制作的镜像，创建开发环境测试。
- (5) 注意：上述 Dockerfile 文件默认 python 版本为 3.6.11；若需指定 Jupyterlab 版本，请按如下示例修改：`pip --no-cache-dir install jupyterlab==2.2.9`

1.2 手动安装组件

1.2.1 在线安装

- (1) 将基础镜像上传至 AIStation。
- (2) 选择基础镜像创建开发环境，参考 1.1 部分 Dockerfile 文件，执行安装命令，完成组件安装。
- (3) 保存镜像。

1.2.2 离线安装

若 AIStation 平台无法连接外网，可以参考以下方式：

- (1) 将相关软件安装包或依赖下载完成，上传到用户目录，基于基础镜像创建开发环境，在开发环境中进行组件的离线安装，最后保存镜像，通过 AIStation 平台完成镜像制作。
- (2) 直接在网络正常的环境中，在线安装组件，以 docker save 的方式保存成 tar 包，通过 AIStation-镜像管理-内部导入的功能导入到 AIStation 中。

下面主要介绍如何下载及安装组件：

1.2.2.1 安装 openssh 和 openssl

- (1) 下载安装包和依赖包到指定目录需要安装的组件：

```
openssh-7.4p1-21.el7.x86_64.rpm
```

```
openssh-clients-7.4p1-21.el7.x86_64.rpm
```

```
openssh-server-7.4p1-21.el7.x86_64.rpm
```

基于基础镜像，挂载本地目录，创建容器，下载各类组件，示例：

```
docker run -it -v /home/inspur/image_components/centos7.4:/home/inspur/image_components /centos7.4 centos:centos7.4.1708 /bin/bash
```

可以使用如下命令，查看需要 yum 安装的组件包

```
yum list | grep 包名
```

下载离线环境使用的组件：

```
#openssh
```

```
yum install -downloadonly -downloadaddir=/home/inspur/image_components/centos7.4/openssh openssh-7.4p1-21.el7.x86_64
```

```
#openssl
```

```
yum install -downloadonly -downloadaddir=/home/inspur/image_components/centos7.4/openssl openssl
```

(2) 安装 openssh 和 openssl 组件

进入安装包目录安装 openssh

```
cd /home/wjy/image_components/centos7.4/openssh
```

```
rpm -ivh .rpm -force -nodeps
```

安装 openssl

```
cd /home/wjy/image_components/centos7.4/openssl
```

```
rpm -ivh .rpm -force -nodeps
```

-nodeps 就是安装时不检查依赖关系

-force 就是强制安装

(3) openssh 配置

先新建目录，如果存在，不需要新建

```
mkdir -p /run/sshd
```

执行 ssh-keygen

```
/usr/bin/ssh-keygen -A
```

执行配置：

```
cat /etc/ssh/ssh_config | grep -v StrictHostKeyChecking > /etc/ssh/ssh_config.new &&
```

```
echo " StrictHostKeyChecking no" >> /etc/ssh/ssh_config.new &&
```

```
cat /etc/ssh/sshd_config | grep -v PermitRootLogin > /etc/ssh/sshd_config.new &&
```

```
echo "PermitRootLogin yes" >> /etc/ssh/sshd_config.new &&
```

```
mv -f /etc/ssh/ssh_config.new /etc/ssh/ssh_config &&
```

```
mv -f /etc/ssh/sshd_config.new /etc/ssh/sshd_config > /dev/null 2>&1;
```

1.2.2.2 安装 JupyterLab

安装 jupyterlab，必须安装 python3 和 pip。

(1) 安装 python3

```
yum install -downloadonly -downloadaddir=/home/wjy/image_components/centos7.4/python_depends make  
zlib zlib-devel bzip2-devel openssl-devel sqlite-devel readline-devel gdbm-devel gcc libffi-devel
```

进入到下载的依赖包目录，安装依赖包：

```
cd /home/wjy/image_components/centos7.4/python_depends
```

```
rpm -ivh *.rpm -force -nodeps
```

下载 python tgz 包，解压进入目录

```
cd /home/wjy/image_components/python
```

```
wget https://www.python.org/ftp/python/3.7.6/Python-3.7.6.tgz
tar -zxvf Python-3.7.6.tgz
cd Python-3.7.6
./configure
make && make install
# 修改配置
mv /usr/bin/python /usr/bin/python27
mv /usr/bin/pip /usr/bin/pip27
ln -s /usr/local/bin/python3 /usr/bin/python
ln -s /usr/local/bin/pip3 /usr/bin/pip
sed -i "s#/usr/bin/python#/usr/bin/python2.7#" /usr/bin/yum
sed -i "s#/usr/bin/python#/usr/bin/python2.7#" /usr/libexec/urlgrabber-ext-down
(2) centos7 安装 pip3(若没有安装 pip3 再安装)
下载 get-pip.py, 安装 pip
cd /home/wjy/image_components
wget https://bootstrap.pypa.io/get-pip.py
# 安装 pip
python get-pip.py
(3) 安装 JupyterLab
pip install jupyterlab(在线安装)
cd /home/inspur/image_components/jupyterlab
pip3 install --no-index --find-links=/home/inspur/install_packages/jupyterlab jupyterlab-1.2.3-py2.py3-none-any.whl(离线安装)
jupyter lab --ip=0.0.0.0 --no-browser --allow-root (运行测试, 是否安装成功)
(4) jupyter 配置
# 下载配置文件
wget -P /home/inspur/image_components/jupyter_configure https://raw.githubusercontent.com/Winowang/jupyter_gpu/master/jupyter_notebook_config.py && wget -P /home/inspur/image_components/jupyter_configure https://raw.githubusercontent.com/Winowang/jupyter_gpu/master/custom.js
# 拷贝配置文件
mkdir /etc/jupyter && cp -rf /home/inspur/image_components/jupyter_configure/* /etc/jupyter
```

二. Ubuntu 系统

2.1 Dockerfile 安装组件

2.1.1 Dockerfile 文件内容

Dockerfile 下载链接: <https://github.com/wjyzzu/inspur-dockerfile/tree/main/base/ubuntu>

2.1.2 制作步骤

- (1) 根据需要, 修改 FROM 中的基础镜像; 建议基础镜像提前上传到 AIStation 中。
- (2) 修改好 Dockerfile, 请上传到个人用户目录下。
- (3) 通过镜像管理-创建功能, 制作新镜像。(4) 基于制作的镜像, 创建开发环境测试。
- (5) 注意: 上述 Dockerfile 文件默认 python 版本为 3.6.11;

若需指定 jupyterlab 版本, 请按如下示例修改:

```
pip --no-cache-dir install jupyterlab==2.2.9
```

2.2 手动安装组件

2.2.1 在线安装

- (1) 将基础镜像上传至 AIStation。
- (2) 选择基础镜像创建开发环境, 参考上述 Dockerfile 文件, 执行安装命令, 完成组件安装。
- (3) 保存镜像。

2.2.2 离线安装

若 AIStation 平台无法连接外网, 可以参考以下方式:

- (1) 将相关软件安装包或依赖下载完成, 上传到用户目录, 基于基础镜像创建开发环境, 在开发环境中进行组件的离线安装, 最后保存镜像, 通过 AIStation 平台完成镜像制作。
- (2) 直接在网络正常的环境中, 在线安装组件, 以 docker save 的方式保存成 tar 包, 通过 AIStation-镜像管理-内部导入的功能导入到 AIStation 中。

下面主要介绍如何下载及安装组件:

2.2.2.1 安装 openssh 和 openssl

- (1) 依赖包存放目录 (ubuntu 离线安装包存放路径, 在线安装忽略)

/var/cache/apt/archives

(2) 查看依赖包

```
apt-get update
```

```
apt-cache depends packname
```

(3) 查看安装包版本

```
sudo apt-cache madison openssh-client
```

(4) 下载安装包和依赖包，进入一个干净的镜像中

进入 ubuntu 镜像/var/cache/apt/archives 移除不需要的文件, 然后下载所有依赖文件, 下载完后, 拷贝到自己的组件目录/home/inspur/image_components/ubuntu18.04/openssh

(根据需要修改)

```
apt-get install -d apt-cache depends openssh-server=1:7.2p2-4 | grep Depends | grep -v debconf-2.0 | cut -d: -f2 | tr -d "<>"
```

(5) 安装 openssh 和 openssl 组件

进入目录

```
cd /home/inspur/image_components/ubuntu18.04/openssh
```

进行安装

```
dpkg -i *.deb
```

(6) 修复安装依赖问题 (出问题执行)

```
apt-get -f install
```

(7)openssh 配置

先新建目录, 如果存在, 不需要新建

```
mkdir -p /run/sshd
```

执行 ssh-keygen

```
/usr/bin/ssh-keygen -A
```

执行配置:

```
cat /etc/ssh/ssh_config | grep -v StrictHostKeyChecking > /etc/ssh/ssh_config.new && echo " StrictHostKeyChecking no" » /etc/ssh/ssh_config.new && cat /etc/ssh/sshd_config | grep -v PermitRootLogin > /etc/ssh/sshd_config.new && echo "PermitRootLogin yes" » /etc/ssh/sshd_config.new && mv -f /etc/ssh/ssh_config.new /etc/ssh/ssh_config && mv -f /etc/ssh/sshd_config.new /etc/ssh/sshd_config > /dev/null 2>&1;
```

2.2.2.2 安装 jupyter

(1) 安装 python3

进入干净镜像，下载依赖包

```
apt-get install -y build-essential libncurses5-dev libgdbm-dev libnss3-dev libssl-dev libreadline-dev libffi-dev  
zlib1g-dev libsqlite3-dev
```

安装所有依赖包：

```
cd /home/wjy/image_components/ubuntu18.04/python_depends
```

```
dpkg -i *deb
```

下载 python tgz 包（如下为 3.7.6 版本），解压进入

```
cd /home/wjy/image_components/python
```

```
wget https://www.python.org/ftp/python/3.7.6/Python-3.7.6.tgz
```

```
tar -zxvf Python-3.7.6.tgz
```

```
cd Python-3.7.6
```

```
./configure
```

```
make && make install
```

```
# 修改配置 mv /usr/bin/python /usr/bin/python27
```

```
mv /usr/bin/pip /usr/bin/pip27
```

```
ln -s /usr/local/bin/python3 /usr/bin/python
```

```
ln -s /usr/local/bin/pip3 /usr/bin/pip
```

(2) 安装 pip

离线安装：

下载 get-pip.py，安装 pip

```
cd /home/wjy/image_components
```

```
wget https://bootstrap.pypa.io/get-pip.py
```

```
# 安装 pip
```

```
python get-pip.py
```

(3) 安装 jupyter

```
pip install jupyterlab(在线安装)
```

cd /home/inspur/image_components/jupyterlab (进入该目录，离线安装，如果需要最新，就联网下载最新离线包)

```
pip3 install - no-index - find-links=/home/wjy/install_packages/jupyterlab jupyterlab-1.2.3-py2.py3-none-any.whl (离线安装)
```

jupyter lab -ip=0.0.0.0 -no-browser -allow-root (运行测试, 是否安装成功)

(4)jupyter 配置

```
离线配置: # 下载配置文件 wget -P /home/inspur/image_components/jupyter_configure https://raw.githubusercontent.com/Winowang/jupyter_gpu/master/jupyter_notebook_config.py && wget -P /home/inspur/image_components/jupyter_configure https://raw.githubusercontent.com/Winowang/jupyter_gpu/master/custom.js  
# 拷贝配置文件  
mkdir /etc/jupyter && cp -rf /home/inspur/image_components/jupyter_configure/* /etc/jupyter
```

镜像制作手册

一. 基础镜像

基础镜像都需要安装 cuda、cudnn 等深度学习必备组件, 如果所制作的镜像对操作系统版本(官方镜像操作系统都是最新的)、cudnn 版本没有要求, 以及要求操作系统是 ubuntu 系统, 都可以去 NGC 官网, 下载指定 cuda 版本的基础镜像, 参考 1.1 章节。

如果镜像的操作系统版本为 centos, 或指定 cuda、cudnn 版本, NGC 官方没有该基础镜像, 则需要自制符合指定操作系统或 cuda 版本的基础镜像, 具体参考 1.2 章节。

注意: 制作镜像, 需要在联网环境下在线制作。由于与组件相关的依赖包较多, 不推荐下载安装包离线安装的方式。本文以在线制作镜像为主, 提供离线下载安装组件的命令。

由于镜像中的操作系统都是 MINI 版, 一般在线安装定制组件时, 一些共享库或者组件依赖包不会被自动安装。在实际镜像制作过程中, 可参考本文档安装所需的共享库或依赖包。若遇到难以解决的问题请联系我们。

通过 Dockerfile 制作镜像, 如果没有最终失败, 请忽略在线制作镜像时输出的红色内容, 这不影响最终制作的镜像。

本文档, 以某客户定制镜像制作作为示例, 定制要求:

操作系统: centos7.4

numpy==1.15.4

opencv-python==3.4.3.18

tensorflow-gpu==1.8.0

Keras==2.2.4

python3.6.6

cuda9.0

cuda9.0

1.1 NGC 官方基础镜像

Ubuntu 系统的 cuda 基础镜像，请在 NGC 官方网站下载，一般不需要单独制作，单独制作只针对 centos 版本。

NGC 官方基础镜像下载地址：

<https://ngc.nvidia.com/catalog/containers/nvidia:cuda/tags>

The screenshot displays the NVIDIA NGC website interface. The top section shows a grid of container images with categories like 'ALL CONTAINER TYPES', 'CONTAINERS', 'HELM CHARTS', 'MODULES', and 'RESOURCES'. A search bar contains 'Cuda: cuda'. A red arrow points to the 'CUDA' card in the grid. The bottom section shows a list of tags for '9.0-cudnn7-devel-ubi7' and '9.0-cudnn7-devel-centos7'. A red arrow points to the '9.0-cudnn7-devel-centos7' tag, and another red arrow points to the 'Pull Tag' button next to it. Below the tag list is a 'Manifest' table with columns: DIGEST, OS/ARCH, COMPRESSED SIZE, and CREATED.

DIGEST	OS/ARCH	COMPRESSED SIZE	CREATED
sha256:76dd3757446b905f39b526ce28ccc3a09a49639...	linux/amd64	1.33 GB	06/19/2018

点击 Pull Tag 获取 pull 镜像命令，下载需要的基础镜像：`docker pull nvcr.io/nvidia/cuda:9.0-cudnn7-devel-centos7`

```
[root@aimaster docker_images]# docker pull nvcr.io/nvidia/cuda:9.0-cudnn7-devel-centos7
9.0-cudnn7-devel-centos7: Pulling from nvidia/cuda
Digest: sha256:5f4fd37b0b19f5c9fd49ae26ae190b77a94a2617e6539a12990fa34192d34b7b0
Status: Image is up to date for nvcr.io/nvidia/cuda:9.0-cudnn7-devel-centos7
[root@aimaster docker_images]# docker images |grep 9.0-cudnn7-devel-centos7
nvcr.io/nvidia/cuda
s ago          2.54GB          9.0-cudnn7-devel-centos7          fd0de7bcd7b0          6 month
```

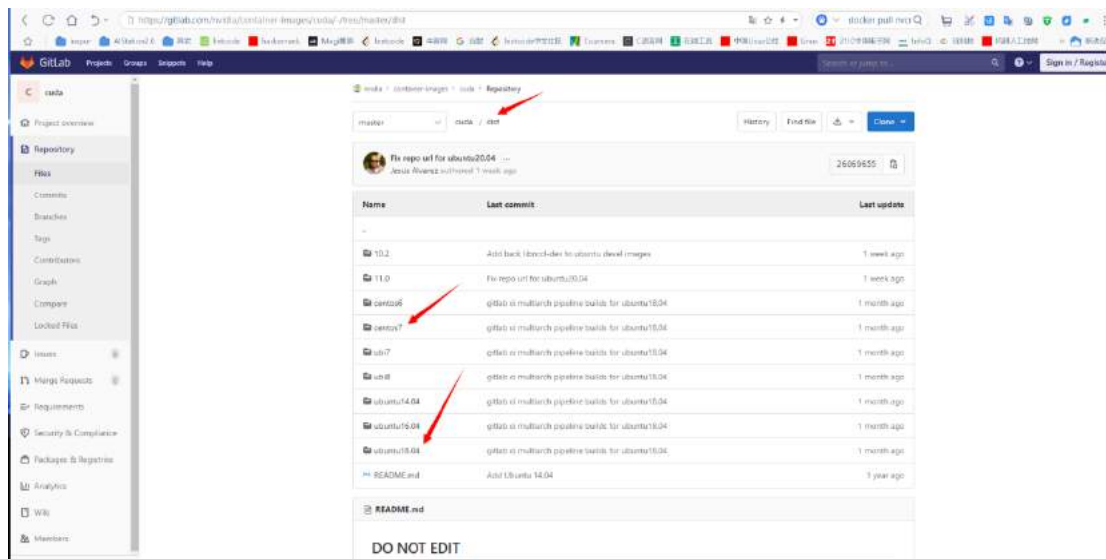
1.2 自制带 cuda 的基础镜像

由于客户定制的镜像，指定了操作系统，NGC 官方镜像没有该基础镜像，需要自制基础镜像。

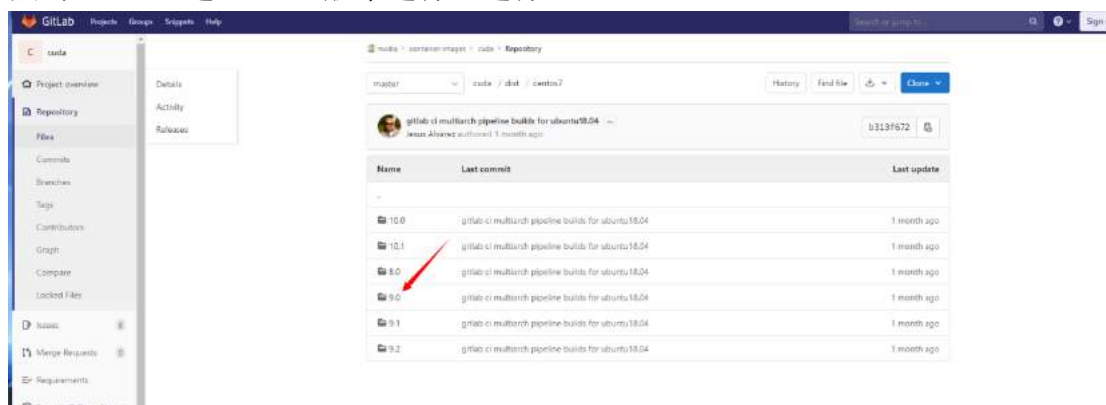
1.2.1 下载 Dockerfile

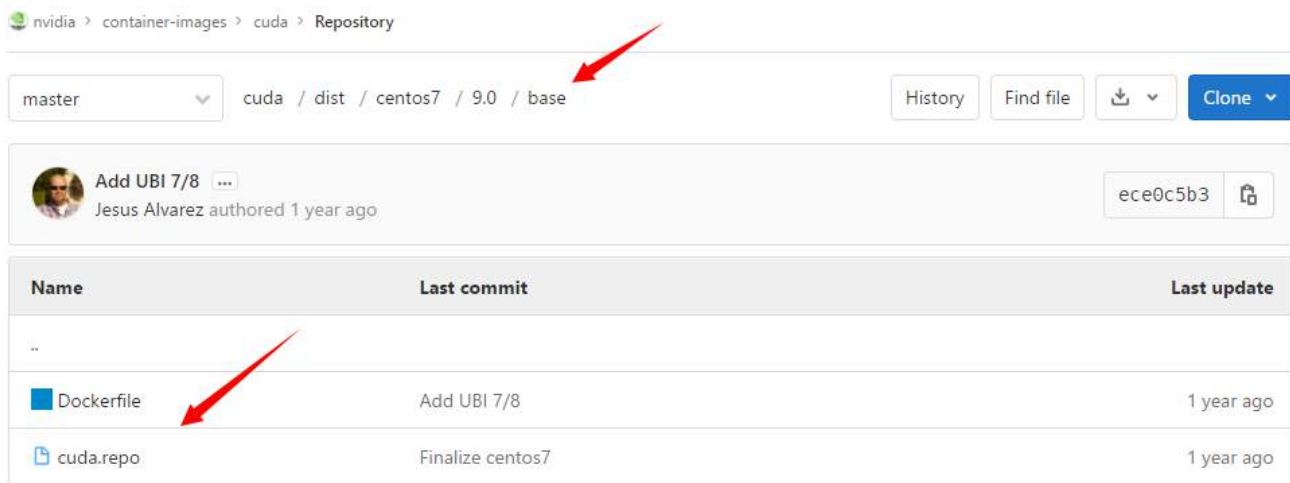
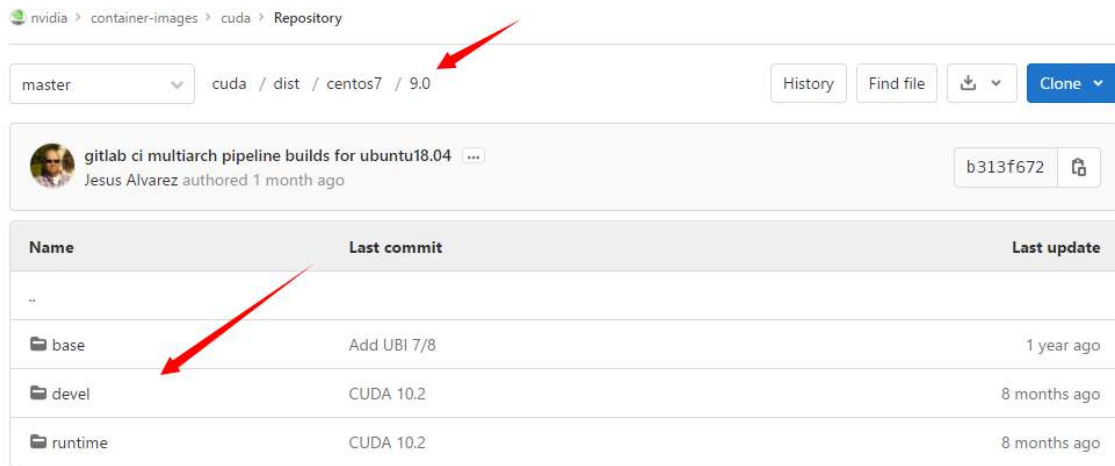
下载指定版本的 cuda 和操作系统的 Dockerfile 和 cuda.repo，然后再修改 Dockerfile。

<https://gitlab.com/nvidia/container-images/cuda/-/tree/master/dist>（需要翻墙，才能访问）

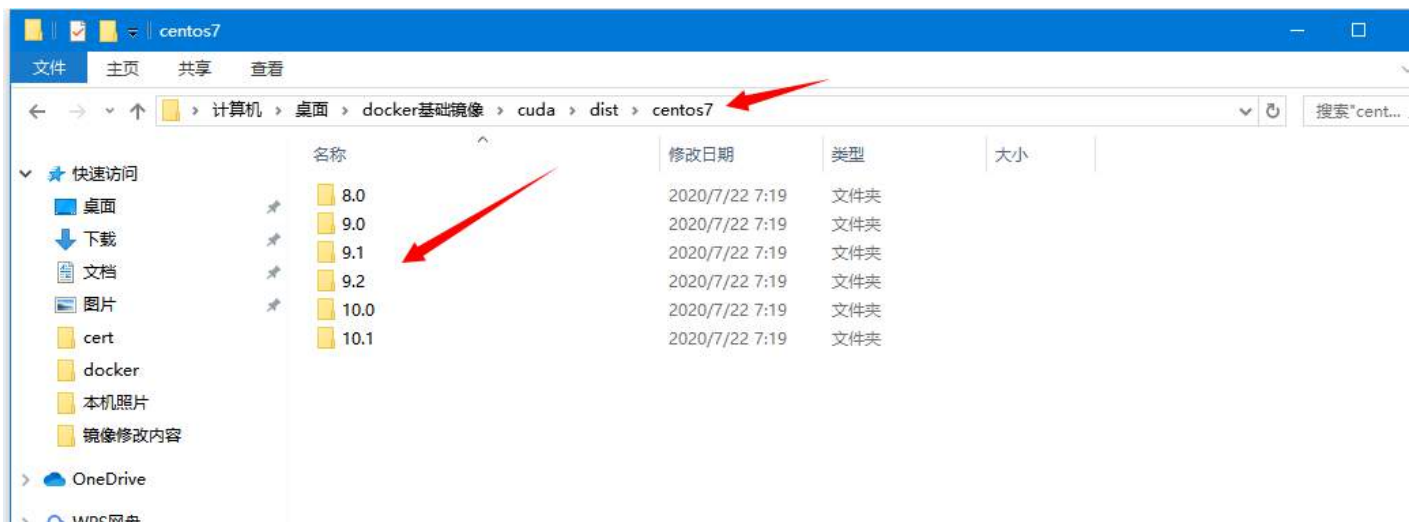


点击 centos7 进入 cuda 版本选择，选择 cuda9.0





下载后，如图所示：



gitlab cuda9.0 下面，有 base、devel、runtime 目录，我们需要将每个下面的 Dockerfile 文件，汇总成一个 Dockerfile 文件，制作一个最全的 cuda 镜像，防止镜像使用 GPU 训练时出现问题。base、devel 和 runtime 官方解释，如下：

CUDA images come in three flavors and are available through the NVIDIA public hub repository.

base: starting from CUDA 9.0, contains the bare minimum (libcudart) to deploy a pre-built CUDA application. Use this image if you want to manually select which CUDA packages you want to install. runtime: extends the base image by adding all the shared libraries from the CUDA toolkit. Use this image if you have a pre-built application using multiple CUDA libraries. devel: extends the runtime image by adding the compiler toolchain, the debugging tools, the headers and the static libraries. Use this image to compile a CUDA application from sources.

1.2.2 修改 Dockerfile

将 cuda9.0 下面的所有 Dockerfile 汇总成一个 Dockerfile，重复内容，只保留一份，汇总后的 Dockerfile 内容，如下：

Dockerfile 文件：

```
FROM centos:centos7.4.1708 # 替换定制的操作系统版本，参考 1.2.3
```

```
LABEL maintainer "NVIDIA CORPORATION cudatools@nvidia.com"
```

```
RUN NVIDIA_GPGKEY_SUM=d1be581509378368edeec8c1eb2958702feedf3bc3d17011adbf24efacce4ab5  
&& \
```

```
curl -fsSL https://developer.download.nvidia.com/compute/cuda/repos/rhel7/x86_64/7fa2af80.pub | sed '/^Ver-  
sion/d' > /etc/pki/rpm-gpg/RPM-GPG-KEY-NVIDIA && \
```

```
echo "$NVIDIA_GPGKEY_SUM /etc/pki/rpm-gpg/RPM-GPG-KEY-NVIDIA" | sha256sum -c -strict -
```

```
COPY cuda.repo /etc/yum.repos.d/cuda.repo
```

```
ENV CUDA_VERSION 9.0.176
```

```
ENV CUDA_PKG_VERSION 9-0-$CUDA_VERSION-1
```

```
RUN yum install -y \
```

```
cuda-cudart-$CUDA_PKG_VERSION && \
```

```
ln -s cuda-9.0 /usr/local/cuda && \
```

```
rm -rf /var/cache/yum/*
```

```
#nvidia-docker 1.0
```

```
LABEL com.nvidia.volumes.needed= "nvidia_driver"
LABEL com.nvidia.cuda.version= "${CUDA_VERSION}"
RUN echo "/usr/local/nvidia/lib" » /etc/ld.so.conf.d/nvidia.conf && \
echo "/usr/local/nvidia/lib64" » /etc/ld.so.conf.d/nvidia.conf
ENV PATH /usr/local/nvidia/bin:/usr/local/cuda/bin:${PATH}
ENV LD_LIBRARY_PATH /usr/local/nvidia/lib:/usr/local/nvidia/lib64
#nvidia-container-runtime
ENV NVIDIA_VISIBLE_DEVICES all
ENV NVIDIA_DRIVER_CAPABILITIES compute,utility
ENV NVIDIA_REQUIRE_CUDA "cuda>=9.0"
#runtime
RUN yum install -y \
cuda-libraries-${CUDA_PKG_VERSION} \
cuda-cublas-9-0-9.0.176.4-1 && \
rm -rf /var/cache/yum/*
#cuda (替换指定的 cuda 版本，默认的是最新的，替换参考 1.2.4)
ENV CUDNN_VERSION 7.6.0.64
LABEL com.nvidia.cudnn.version= "${CUDNN_VERSION}"
#cuDNN license: https://developer.nvidia.com/cudnn/license_agreement
RUN CUDNN_DOWNLOAD_SUM=90659ea77734b7b671afe930c9898d21a13b888998f1dd3940cc57d6b2f29b86
&& \
curl -fsSL http:// developer.download.nvidia.com/ compute/ redistrib/ cudnn/ v7.6.0/ cudnn-9.0-linux-x64-
v7.6.0.64.tgz -O && \
echo "${CUDNN_DOWNLOAD_SUM} cudnn-9.0-linux-x64-v7.6.0.64.tgz" | sha256sum -c - && \ tar -
no-same-owner -xzf cudnn-9.0-linux-x64-v7.6.0.64.tgz -C /usr/local -wildcards ' cuda/lib64/libcudnn.so.*'
&& \
rm cudnn-9.0-linux-x64-v7.6.0.64.tgz && \
ldconfig
#devel
RUN rm -rf /usr/local/cuda-9.0/include && \
```

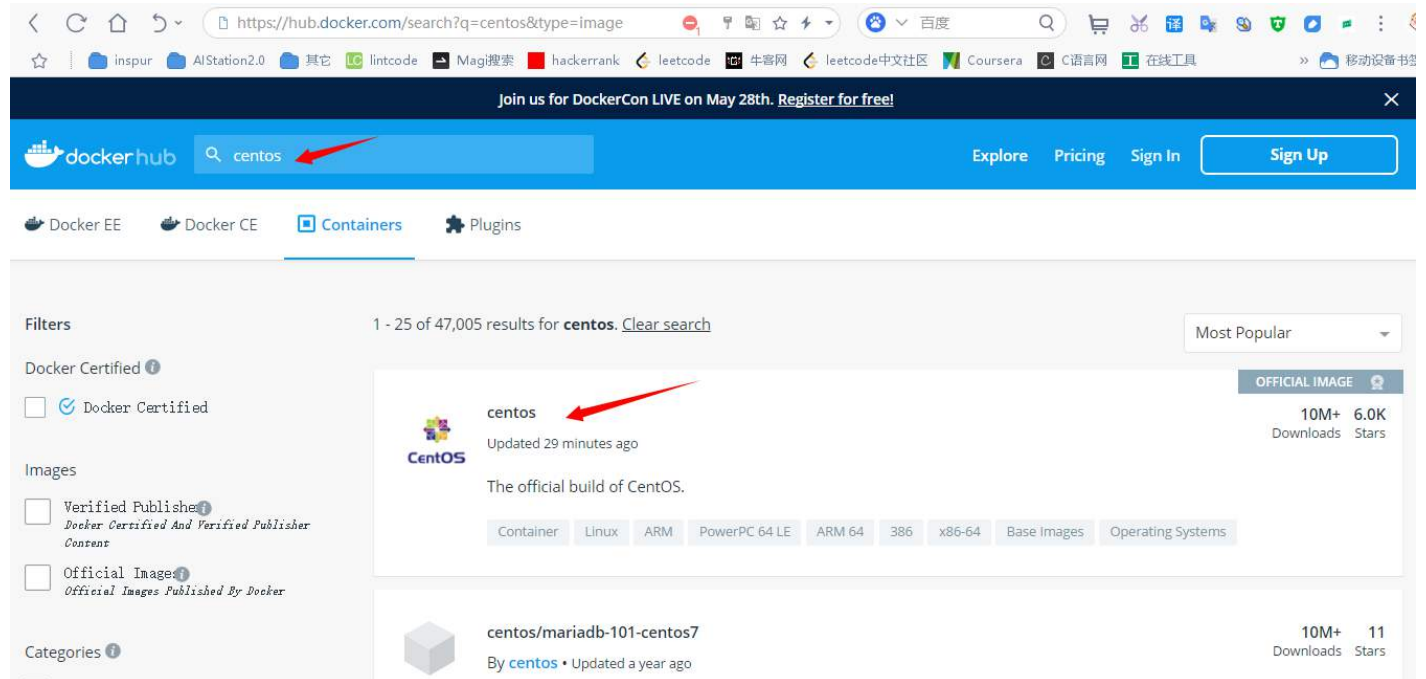
```
yum install -y \  
cuda-libraries-dev-$CUDA_PKG_VERSION \  
cuda-nvml-dev-$CUDA_PKG_VERSION \  
cuda-minimal-build-$CUDA_PKG_VERSION \  
cuda-command-line-tools-$CUDA_PKG_VERSION \  
cuda-core-9-0-9.0.176.3-1 \  
cuda-cublas-dev-9-0-9.0.176.4-1 && \  
rm -rf /var/cache/yum/*  
ENV LIBRARY_PATH /usr/local/cuda/lib64/stubs
```

1.2.3 替换 FROM 基础镜像

Dockerfile 中的 FROM 操作系统镜像，换成自己需要的操作系统，操作系统镜像下载地址和查看操作系统镜像标签和版本地址如下：

<https://hub.docker.com/>

在搜索框输入 centos, 点击第一个官方 centos, 进入寻找需要的操作系统版本，替换 FROM 部分内容。



The screenshot shows the Docker Hub search results for 'centos'. The search bar at the top contains 'centos' with a red arrow pointing to it. Below the search bar, the results are displayed. The first result is the official 'centos' image, which is highlighted with a red arrow. The image details include 'Updated 29 minutes ago', 'The official build of CentOS.', and '10M+ Downloads' and '6.0K Stars'. Below this, there is another result for 'centos/mariadb-101-centos7' with '10M+ Downloads' and '11 Stars'. The left sidebar shows filters for 'Docker Certified', 'Verified Publisher', and 'Official Image'.

https://hub.docker.com/_/centos?tab=tags&page=1&name=centos7.4

CentOS The official build of CentOS.

500M+

Container Linux ARM PowerPC 64 LE ARM 64 386 x86-64 Base Images Operating Systems

Official Image

Description Reviews **Tags**

Q centos7.4 x Sort by Latest

IMAGE **centos7.4.1708**
Last updated a year ago by doijanky

DIGEST 9f266d35e02c OS/ARCH linux/amd64 COMPRESSED SIZE 69.96 MB

docker pull centos:centos7.4.1708

1.2.4 替换 cudnn（如果没有指定版本，忽略该部分内容）

由于官方 cudnn 中的 Dockerfile 中，cudnn 都是最新的，如果需要指定版本，需要自己手动查找和修改该部分内容。

https://gitlab.com/nvidia/container-images/cuda/-/blob/centos7/9.0/

GitLab Projects Groups Snippets Help Search or jump to... Sign in / Register

nvidia > container-images > cuda > Repository

centos7 cuda / 9.0 / devel / cudnn7 / **Dockerfile** Find file Blame History Permalink

Update cuDNN to 7.6.0.64
Jesus Alvarez authored 11 months ago 3bbc0c8e

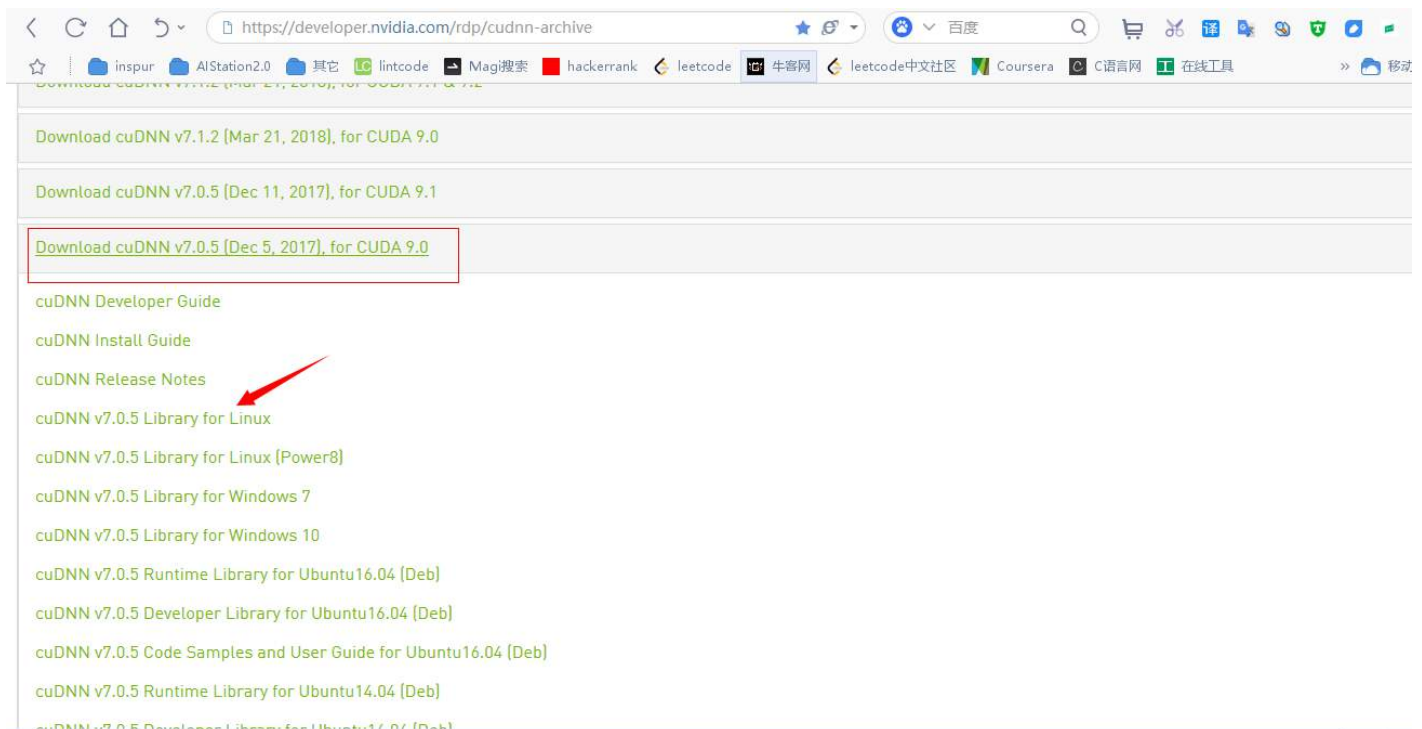
Dockerfile 706 Bytes Edit Web IDE

```

1 ARG IMAGE_NAME
2 FROM s[IMAGE_NAME]:9.0-devel-centos7
3 LABEL maintainer "NVIDIA CORPORATION <cuda@nvidia.com>"
4
5 ENV CUDNN_VERSION 7.6.0.64
6 LABEL com.nvidia.cudnn.version="${CUDNN_VERSION}"
7
8 # cuDNN license: https://developer.nvidia.com/cudnn/license_agreement
9 RUN CUDNN_DOWNLOAD_SUM=90659ea77734b7b671afe930c9898d21a13b888998f1dd3940cc57d6b2f29b86 && \
10 curl -fsSL http://developer.download.nvidia.com/compute/redis/cudnn/v7.6.0/cudnn-9.0-linux-x64-v7.6.0.64.tgz -o && \
11 echo "SCUDNN_DOWNLOAD_SUM cudnn-9.0-linux-x64-v7.6.0.64.tgz" | sha256sum -c - && \
12 tar --no-same-owner -xzf cudnn-9.0-linux-x64-v7.6.0.64.tgz -C /usr/local && \
13 rm cudnn-9.0-linux-x64-v7.6.0.64.tgz && \
14 ldconfig

```

查看自己指定的 cudnn 版本链接：<https://developer.nvidia.com/rdp/cudnn-archive>



点击下载，查看版本号：



1.2.4.1 在线 cudnn

将 #cudnn 部分修改为如下内容

```
#cudnn
ENV CUDNN_VERSION 7
LABEL com.nvidia.cudnn.version= "${CUDNN_VERSION}"
#cuDNN license: https://developer.nvidia.com/cudnn/license_agreement
RUN curl -fsSL http://developer.download.nvidia.com/compute/redis/cudnn/v7.0.5/cudnn-9.0-linux-x64-v7.tgz -O && \
tar -no-same-owner -xzf cudnn-9.0-linux-x64-v7.tgz -C /usr/local --wildcards 'cuda/lib64/libcudnn.so.*' \
&& \
rm cudnn-9.0-linux-x64-v7.tgz && \
ldconfig
```

1.2.4.2 离线 cudnn

如果上述在线下载出现问题，或者比较缓慢，可以使用离线 cudnn，先下载 cudnn tar 包，然后和 Dockerfile 文件放到一起。

然后将 Dockerfile 中的 #cudnn 修改如下内容

```
#cudnn
ENV CUDNN_VERSION 7
LABEL com.nvidia.cudnn.version= "${CUDNN_VERSION}"
#cuDNN license: https://developer.nvidia.com/cudnn/license_agreement
COPY cudnn-9.0-linux-x64-v7.tgz /home/cudnn-9.0-linux-x64-v7.tgz
RUN tar -no-same-owner -xzf /home/cudnn-9.0-linux-x64-v7.tgz -C /usr/local && \
rm /home/cudnn-9.0-linux-x64-v7.tgz && \
ldconfig
```

1.2.5 其它部分修改

#devel RUN 部分添加 `rm -rf /usr/local/cuda-9.0/include && \`，避免制作失败，根据 cuda 版本，修改该部分的 cuda 版本。

1.2.6 在线制作 cuda 基础镜像

进入 Dockerfile 目录

cd /home/wjy/Dockerfile/cuda9.0-wjy

#build Dockerfile

docker build -t centos7.4-cuda9.0-cudnn7.0-base-inspur:latest .

```
[root@aimaster cuda9.0-wjy]# docker build -t test-dockerfile:latest .
Sending build context to Docker daemon  5.12kB
Step 1/21 : FROM centos:centos7.4.1708
--> 9f266d35e02c
Step 2/21 : LABEL maintainer "NVIDIA CORPORATION <cuda@nvidia.com>"
--> Running in c134676424ee
--> 85b439e11145
Removing intermediate container c134676424ee
Step 3/21 : RUN NVIDIA_GPGKEY_SUM=d1be581509378368edeec8c1eb2958702feedf3bc3d17011adb2f24efacce4ab5 && curl -fsSL https://developer.download.nvidia.com/compute/cuda/repos/rhel7/x86_64/7fa2af80.pub | sed '/^Version/d' > /etc/pki/rpm-gpg/RPM-GPG-KEY-NVIDIA && echo "SNVIDIA_GPGKEY_SUM /etc/pki/rpm-gpg/RPM-GPG-KEY-NVIDIA" | sha256sum -c --strict -
--> Running in a276ada0a53e
/etc/pki/rpm-gpg/RPM-GPG-KEY-NVIDIA: OK
--> 07bee1779867
Removing intermediate container a276ada0a53e
Step 4/21 : COPY cuda.repo /etc/yum.repos.d/cuda.repo
--> 504233f78c2c
Removing intermediate container c9e100a3c0d0
Step 5/21 : ENV CUDA_VERSION 9.0.176
--> Running in fa82ff62b578
--> 08fc080824a2
Removing intermediate container fa82ff62b578
Step 6/21 : ENV CUDA_PKG_VERSION 9-0-SCUDA_VERSION-1
--> Running in 3b4f3c33e9d8
--> fa3c60act05d
Removing intermediate container 3b4f3c33e9d8
Step 7/21 : RUN yum install -y cuda-cudart-SCUDA_PKG_VERSION && ln -s cuda-9.0 /usr/local/cuda && rm -rf /var/cache/yum/*
--> Running in dcde980e258c
Loaded plugins: fastestmirror, ovl
Determining fastest mirrors
 * base: mirrors.tuna.tsinghua.edu.cn
 * extras: mirrors.bfsu.edu.cn
 * updates: mirror.bit.edu.cn
Resolving Dependencies
--> Running transaction check
--> Package cuda-cudart-9-0.x86_64 0:9.0.176-1 will be installed
--> Processing Dependency: cuda-license-9-0 for package: cuda-cudart-9-0-9.0.176-1.x86_64
--> Running transaction check
--> Package cuda-license-9-0.x86_64 0:9.0.176-1 will be installed
--> Processing Dependency: cuda-license-9-0 for package: cuda-cudart-9-0-9.0.176-1.x86_64
--> Running transaction check
--> Package cuda-license-9-0.x86_64 0:9.0.176-1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                Arch      Version      Repository    Size
=====
Installing:
 cuda-cudart-9-0        x86_64    9.0.176-1    cuda          143 k
Installing for dependencies:
 cuda-license-9-0      x86_64    9.0.176-1    cuda           28 k
=====
Transaction Summary
-----
Install 1 Package (+1 Dependent package)

Total download size: 171 k
Installed size: 549 k
Downloading packages:
warning: /var/cache/yum/x86_64/7/cuda/packages/cuda-cudart-9-0-9.0.176-1.x86_64.rpm: Header V3 RSA/SHA512 Signature, key ID 7fa2af80: NOKEY
Public key for cuda-cudart-9-0-9.0.176-1.x86_64.rpm is not installed
-----
Total                78 kB/s | 171 kB  00:02
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-NVIDIA
Importing GPG key 0x7FA2AF80:
  Userid       : "cuda@nvidia.com"
  Fingerprint: ae09 1e4b bd22 3a84 e2cc fce3 160f 4b3d 7fa2 af80
  From         : /etc/pki/rpm-gpg/RPM-GPG-KEY-NVIDIA
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : cuda-license-9-0-9.0.176-1.x86_64                1/2
*** LICENSE AGREEMENT ***
By using this software you agree to fully comply with the terms and
conditions of the EULA (End User License Agreement). The EULA is located
at /usr/local/cuda-9.0/doc/EULA.txt. The EULA can also be found at
http://docs.nvidia.com/cuda/eula/index.html. If you do not agree to the
terms and conditions of the EULA, do not use the software.
```

```
glibc-headers      x86_64 2.17-307.el7.1      base      689 k
kernel-headers    x86_64 3.10.0-1127.8.2.el7    updates  6.9 M
libgcc             x86_64 4.8.5-39.el7           base      158 k
libm               x86_64 3.0.1-3.el7            base      51 k
libstdc++-devel   x86_64 4.8.5-39.el7           base      1.5 M
mpfr               x86_64 3.1.1-4.el7            base      203 k
Updating for dependencies:
glibc              x86_64 2.17-307.el7.1         base      3.6 M
glibc-common       x86_64 2.17-307.el7.1         base      11 M
libgcc             x86_64 4.8.5-39.el7           base      102 k
libstdc++          x86_64 4.8.5-39.el7           base      305 k

Transaction Summary
-----
Install 6 Packages (+20 Dependent packages)
Upgrade ( 4 Dependent packages)

Total download size: 539 M
Downloading packages:
Delta RPMs disabled because /usr/bin/applydeltarpm not installed.
warning: /var/tmp/rpm-tmp/4477/base/packages/cpp-4.8.5-39.el7.x86_64.rpm: Header V3 RSA/SHA256 Signature, key ID 54a80eb6: NOKEY
Public key for cpp-4.8.5-39.el7.x86_64.rpm is not installed
http://developer.download.nvidia.com/compute/cuda/repos/rhel7/x86_64/cuda-nvml-dev-9-0-9.0.176-1.x86_64.rpm: [Errno 12] Timeout on http://developer.download.nvidia.com/compute/cuda/repos/rhel7/x86_64/cuda-nvml-dev-9-0-9.0.176-1.x86_64.rpm: (28, 'Operation too slow. Less than 1000 bytes/sec transferred the last 30 seconds')
Trying other mirrors.
Public key for kernel-headers-3.10.0-1127.8.2.el7.x86_64.rpm is not installed
-----
Total                    558 KB/s | 530 MB  16:11
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
Importing GPG key 0x4A80EB6:
Userid : "CentOS-7 Key (CentOS 7 Official Signing Key) <security@centos.org>"
Fingerprint: 6341 ab27 53d7 3a70 a7c2 7b11 3456 a8a7 f4ad 0eb5
Package : centos-release-7-4.1708.el7.centos.x86_64 (@CentOS)
From    : /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Updating : libgcc-4.8.5-39.el7.x86_64                1/34
  Updating : glibc-2.17-307.el7.1.x86_64              2/34
warning: /etc/ncsswitch.conf created as /etc/ncsswitch.conf.rpmnew
  Updating : glibc-common-2.17-307.el7.1.x86_64       3/34
```

报红内容，忽略，不影响安装，如果介意，修改 cuda.repo 中的 gpgcheck=0，同时在 Dockerfile 最后一个 RUN 中添加 rpm -import /etc/pki/rpm-gpg/RPM-GPG-KEY-CentOS-7 && \，解决该问题。

```
cuda-core-9-0.x86_64 0:9.0.176.3-1
cuda-cublas-dev-9-0.x86_64 0:9.0.176.4-1
cuda-libraries-dev-9-0.x86_64 0:9.0.176-1
cuda-minimal-build-9-0.x86_64 0:9.0.176-1
cuda-nvml-dev-9-0.x86_64 0:9.0.176-1

Dependency Installed:
cpp.x86_64 0:4.8.5-39.el7
cuda-cudart-dev-9-0.x86_64 0:9.0.176-1
cuda-cufft-dev-9-0.x86_64 0:9.0.176-1
cuda-curand-dev-9-0.x86_64 0:9.0.176-1
cuda-cusolver-dev-9-0.x86_64 0:9.0.176-1
cuda-cusparse-dev-9-0.x86_64 0:9.0.176-1
cuda-driver-dev-9-0.x86_64 0:9.0.176-1
cuda-misc-headers-9-0.x86_64 0:9.0.176-1
cuda-npp-dev-9-0.x86_64 0:9.0.176-1
cuda-nvgraph-dev-9-0.x86_64 0:9.0.176-1
cuda-nvrtc-dev-9-0.x86_64 0:9.0.176-1
gcc.x86_64 0:4.8.5-39.el7
gcc-c++.x86_64 0:4.8.5-39.el7
glibc-devel.x86_64 0:2.17-307.el7.1
glibc-headers.x86_64 0:2.17-307.el7.1
kernel-headers.x86_64 0:3.10.0-1127.8.2.el7
libgomp.x86_64 0:4.8.5-39.el7
libmpc.x86_64 0:1.0.1-3.el7
libstdc++-devel.x86_64 0:4.8.5-39.el7
mpfr.x86_64 0:3.1.1-4.el7

Dependency Updated:
glibc.x86_64 0:2.17-307.el7.1      glibc-common.x86_64 0:2.17-307.el7.1
libgcc.x86_64 0:4.8.5-39.el7      libstdc++.x86_64 0:4.8.5-39.el7

Complete!
---> 21e5deb9a2da
Removing intermediate container 236cea4822a8
Step 21/21 : ENV LIBRARY_PATH /usr/local/cuda/lib64/stubs
---> Running in 3a2f95595f62
---> cfcbe827a91b
Removing intermediate container 3a2f95595f62
Successfully built cfcbe827a91b
Successfully tagged test-dockerfile:latest
[root@aimaster cuda9.0-wjy]#

Removing intermediate container 3a2f95595f62
Successfully built cfcbe827a91b
Successfully tagged test-dockerfile:latest
[root@aimaster cuda9.0-wjy]# docker run -it test-dockerfile:latest bash
[root@72f7ec4de780 /]# nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2017 NVIDIA Corporation
Built on Fri Sep  1 21:08:03 CDT 2017
Cuda compilation tools, release 9.0, V9.0.176
[root@72f7ec4de780 /]#
```

1.2.7 离线下载 RUN 安装的 cuda 组件（如果在线制作顺利，忽略该部分）

1.2.7.1 离线下载 cuda 组件

Dockerfile 在线制作过程中，如果网络环境不太好，可能导致在线制作时间特别长或者制作失败。可以离线 Dockerfile RUN 中 yum 在线安装的组件，然后，修改 Dockerfile, 制作好后，再在制作的镜像中，离线安装这些组件。

在 linux 下新建一个目录，用来下载离线 cuda 组件：

```
[root@aimaster components]# pwd
/home/wjy/Dockerfile/components
[root@aimaster components]#
```

在终端执行如下命令：

```
CUDA_VERSION=9.0.176
```

```
CUDA_PKG_VERSION=9-0-CUDA_VERSION-1yuminstall--downloadonly--downloaddir =
/home/wjy/Dockerfile/componentscuda - cudart-CUDA_PKG_VERSION cuda-libraries-
CUDA_PKG_VERSIONcuda-cublas-9-0-9.0.176.4-1cuda-libraries-dev-CUDA_PKG_VERSION
cuda-nvml-dev-CUDA_PKG_VERSIONcuda - minimal - build-CUDA_PKG_VERSION cuda-
command-line-tools-$CUDA_PKG_VERSION cuda-core-9-0-9.0.176.3-1 cuda-cublas-dev-9-0-9.0.176.4-1
```

```
total 1234440
-rw-r--r--. 1 root root 32023497 Sep 23 2017 cuda-command-line-tools-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 22626151 May 24 2018 cuda-core-9-0-9.0.176.3-1.x86_64.rpm
-rw-r--r--. 1 root root 73073867 Jul 14 2018 cuda-cublas-9-0-9.0.176.4-1.x86_64.rpm
-rw-r--r--. 1 root root 82335574 Jul 14 2018 cuda-cublas-dev-9-0-9.0.176.4-1.x86_64.rpm
-rw-r--r--. 1 root root 146671 Sep 23 2017 cuda-cudart-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 1217125 Sep 23 2017 cuda-cudart-dev-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 102698085 Sep 23 2017 cuda-cufft-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 95468945 Sep 23 2017 cuda-cufft-dev-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 43847481 Sep 23 2017 cuda-curand-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 67872020 Sep 23 2017 cuda-curand-dev-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 39824385 Sep 23 2017 cuda-cusolver-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 11333919 Sep 23 2017 cuda-cusolver-dev-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 37460915 Sep 23 2017 cuda-cusparses-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 75737774 Sep 23 2017 cuda-cusparses-dev-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 18821 Sep 23 2017 cuda-driver-dev-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 6190 Sep 23 2017 cuda-libraries-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 6318 Sep 23 2017 cuda-libraries-dev-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 28715 Sep 23 2017 cuda-license-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 6010 Sep 23 2017 cuda-minimal-build-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 1193604 Sep 23 2017 cuda-misc-headers-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 95391511 Sep 23 2017 cuda-npp-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 95851387 Sep 23 2017 cuda-npp-dev-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 9251545 Sep 23 2017 cuda-nvgraph-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 9299496 Sep 23 2017 cuda-nvgraph-dev-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 64133 Sep 23 2017 cuda-nvml-dev-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 8962438 Sep 23 2017 cuda-nvrtc-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 14767 Sep 23 2017 cuda-nvrtc-dev-9-0-9.0.176-1.x86_64.rpm
-rw-r--r--. 1 root root 348817823 Nov 17 2017 cudnn-9.0-linux-x64-v7.tgz
-rw-r--r--. 1 root root 7534972 May 13 11:55 gcc-c++-4.8.5-36.el7.x86_64.rpm
-rw-r--r--. 1 root root 311728 May 13 11:55 libstdc++-4.8.5-36.el7.x86_64.rpm
-rw-r--r--. 1 root root 1580028 May 13 11:55 libstdc++-devel-4.8.5-36.el7.x86_64.rpm
[root@node2 offline-cuda]#
```

1.2.7.2 修改 Dockerfile

```
FROM centos:centos7.4.1708
LABEL maintainer "NVIDIA CORPORATION cudatools@nvidia.com"
RUN NVIDIA_GPGKEY_SUM=d1be581509378368edeec8c1eb2958702feedf3bc3d17011adbf24efacce4ab5
&& \
curl -fsSL https://developer.download.nvidia.com/compute/cuda/repos/rhel7/x86_64/7fa2af80.pub | sed '/^Ver-
sion/d' > /etc/pki/rpm-gpg/ RPM-GPG-KEY-NVIDIA && \
echo "$NVIDIA_GPGKEY_SUM /etc/pki/rpm-gpg/RPM-GPG-KEY-NVIDIA" | sha256sum -c --strict -
COPY cuda.repo /etc/yum.repos.d/cuda.repo
ENV CUDA_VERSION 9.0.176
ENV CUDA_PKG_VERSION 9-0-$CUDA_VERSION-1
RUN yum install -y \
cuda-cudart-$CUDA_PKG_VERSION && \
ln -s cuda-9.0 /usr/local/cuda && \
rm -rf /var/cache/yum/*
#nvidia-docker 1.0
LABEL com.nvidia.volumes.needed= "nvidia_driver"
LABEL com.nvidia.cuda.version= "${CUDA_VERSION}"
RUN echo "/usr/local/nvidia/lib" » /etc/ld.so.conf.d/nvidia.conf && \
echo "/usr/local/nvidia/lib64" » /etc/ld.so.conf.d/nvidia.conf
ENV PATH /usr/local/nvidia/bin:/usr/local/cuda/bin:${PATH}
ENV LD_LIBRARY_PATH /usr/local/nvidia/lib:/usr/local/nvidia/lib64
#nvidia-container-runtime
ENV NVIDIA_VISIBLE_DEVICES all
ENV NVIDIA_DRIVER_CAPABILITIES compute,utility
ENV NVIDIA_REQUIRE_CUDA "cuda>=9.0"
#runtime
#RUN yum install -y \
#cuda-libraries-$CUDA_PKG_VERSION \
#cuda-cublas-9-0-9.0.176.4-1 && \
```



```
rm -rf /var/cache/yum/*

#cuda
ENV CUDNN_VERSION 7
LABEL com.nvidia.cudnn.version= "${CUDNN_VERSION}"

#online cuda
#cuDNN license: https://developer.nvidia.com/cudnn/license_agreement
#RUN curl -fsSL http://developer.download.nvidia.com/compute/redist/cudnn/v7.0.5/cudnn-9.0-linux-x64-
v7.tgz -O && \
#tar -no-same-owner -xzf cudnn-9.0-linux-x64-v7.tgz -C /usr/local -wildcards ' cuda/lib64/libcudnn.so.*'
&& \
#rm cudnn-9.0-linux-x64-v7.tgz && \
#ldconfig

#offline cudnn
#cuDNN license: https://developer.nvidia.com/cudnn/license_agreement
COPY cudnn-9.0-linux-x64-v7.tgz /home/cudnn-9.0-linux-x64-v7.tgz
RUN tar -no-same-owner -xzf /home/cudnn-9.0-linux-x64-v7.tgz -C /usr/local && \
rm /home/cudnn-9.0-linux-x64-v7.tgz && \
ldconfig

#devel
#RUN rm -rf /usr/local/cuda-9.0/include && \
#yum install -y \
#cuda-libraries-dev-$CUDA_PKG_VERSION \
#cuda-nvml-dev-$CUDA_PKG_VERSION \
#cuda-minimal-build-$CUDA_PKG_VERSION \
#cuda-command-line-tools-$CUDA_PKG_VERSION \
#cuda-core-9-0-9.0.176.3-1 \
#cuda-cublas-dev-9-0-9.0.176.4-1 && \
rm -rf /var/cache/yum/*

ENV LIBRARY_PATH /usr/local/cuda/lib64/stubs
```

1.2.7.3 进入 build 后的基础镜像中，离线安装下载的 cuda 组件

```
cd /home/wjy/Dockerfile/cuda9.0-wjy
```

```
#build 镜像
```

```
docker build -t centos7.4-cuda9.0-cudnn7.0-base-inspur:latest .
```

```
# 挂载 cuda 组件目录，进入镜像，安装 cuda 组件
```

```
docker run -it -v /home/wjy:/home/wjy centos7.4-cuda9.0-cudnn7.0-base-inspur:latest bash
```

```
# 在镜像中，进入 cuda 组件目录
```

```
cd /home/wjy/components
```

```
# 安装组件
```

```
rpm -ivh *.rpm -force -nodeps
```

```
# 安装成功后，保存镜像
```

```
docker commit 容器 ID centos7.4-cuda9.0-cudnn7.0-base-inspur:latest
```

1.2.8 验证基础镜像

在装有 GPU 的环境，运行如下命令验证：

```
docker run -it --runtime=nvidia -e NVIDIA_VISIBLE_DEVICES=1 centos7.4-cuda9.0-cudnn7.0-base-inspur:latest bash
```

```
[root@node1 ~]# docker images |grep base
centos7.4-cuda9.0-cudnn7.0-base-inspur          latest          4e9c3dd52489   3 days ago     2.87GB
10.151.11.65:5000/other/centos7.4-cuda9.0-cudnn7.0-python3.6.6-base-inspur latest          4067c14f9b91   5 days ago     4.52GB
10.151.11.65:5000/aistation/ibase-service      sky            5b66ecb17d09   10 days ago    566MB
10.151.11.51:5000/aistation/ibase-service      latest         0abf290d85d8   2 weeks ago    566MB
10.151.11.65:5000/aistation/ibase-service      latest         0abf290d85d8   2 weeks ago    566MB
aistation/ibase-service                       latest         0abf290d85d8   2 weeks ago    566MB
[root@node1 ~]# docker run -it --runtime=nvidia -e NVIDIA_VISIBLE_DEVICES=1 centos7.4-cuda9.0-cudnn7.0-base-inspur:latest bash
[root@2657ec2f1c26 /]# nvidia-smi
Mon Jun  1 02:47:05 2020
+-----+
| NVIDIA-SMI 418.67      Driver Version: 418.67      CUDA Version: 10.1
+-----+
| GPU   Name           Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
+-----+-----+
|    0   Tesla P100-PCIE...    Off | 00000000:86:00:0 Off |             0
| N/A   35C    P0      26W / 250W |  0MiB / 16280MiB |           0%      Default |
+-----+-----+
+-----+
| Processes:
| GPU   PID     Type    Process name
| Memory Usage
+-----+-----+
| No running processes found
+-----+
[root@2657ec2f1c26 /]# nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2017 NVIDIA Corporation
Built on Fri Sep  1 21:08:03 CDT 2017
Cuda compilation tools, release 9.0, V9.0.176
[root@2657ec2f1c26 /]#
```

二. 安装组件

安装组件，可以参照文档中的 Dockerfile 示例，编写安装定制组件的 Dockerfile，将 FROM 换成上面制作的基础镜像，然后在 RUN 中执行安装定制组件命令，Dockerfile RUN 安装命令比较简单，不再详述。

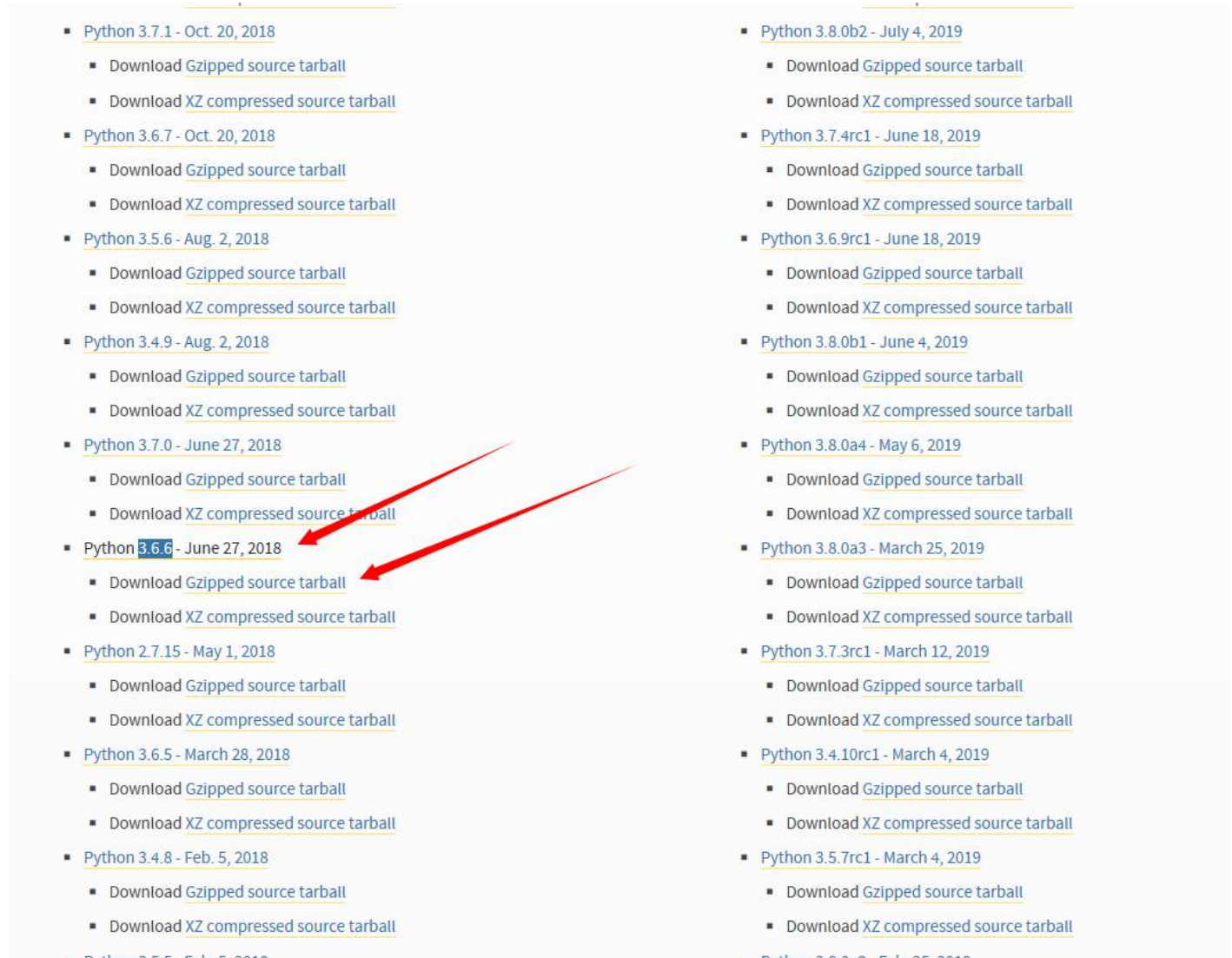
Dockerfile 下载地址：

<https://github.com/wjyzzu/inspur-dockerfile/tree/main/customization/centos7.4>

2.1 安装 python3

python 离线包下载地址:

<https://www.python.org/downloads/source/>



The screenshot displays a list of Python versions available for download. Each version entry includes the version number and release date, followed by two download options: 'Download Gzipped source tarball' and 'Download XZ compressed source tarball'. Two red arrows are drawn over the image, pointing to the entry for Python 3.6.6 (released June 27, 2018) and specifically to the 'Download Gzipped source tarball' link for that version.

- Python 3.7.1 - Oct. 20, 2018
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.6.7 - Oct. 20, 2018
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.5.6 - Aug. 2, 2018
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.4.9 - Aug. 2, 2018
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.7.0 - June 27, 2018
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python **3.6.6** - June 27, 2018
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 2.7.15 - May 1, 2018
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.6.5 - March 28, 2018
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.4.8 - Feb. 5, 2018
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.5.5 - Feb. 5, 2018
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.8.0b2 - July 4, 2019
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.7.4rc1 - June 18, 2019
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.6.9rc1 - June 18, 2019
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.8.0b1 - June 4, 2019
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.8.0a4 - May 6, 2019
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.8.0a3 - March 25, 2019
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.7.3rc1 - March 12, 2019
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.4.10rc1 - March 4, 2019
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.5.7rc1 - March 4, 2019
 - Download Gzipped source tarball
 - Download XZ compressed source tarball
- Python 3.8.0a2 - Feb. 25, 2019
 - Download Gzipped source tarball
 - Download XZ compressed source tarball

2.1.1 在本地服务器下载 python3.6.6 安装包:

```
cd /home/wjy
```

```
wget https://www.python.org/ftp/python/3.6.6/Python-3.6.6.tgz
```

2.1.2 挂载/home/wjy 目录到容器：

```
docker run -i -t -v /home/wjy:/home/wjy centos7.4-cuda9.0-cudnn7.0-base-inspur:latest /bin/bash
```

2.1.3 安装 python3.6

```
docker exec -i -t 容器 ID /bin/bash # 进入基础镜像容器
```

```
yum -y install make zlib zlib-devel bzip2-devel openssl-devel sqlite-devel readline-devel gdbm-devel gcc libffi-devel
```

```
tar -zxvf /home/wjy/Python-3.6.6.tgz -C /home/wjy
```

```
cd /home/wjy/Python-3.6.6
```

```
./configure
```

```
make && make install
```

```
mv /usr/bin/python /usr/bin/python27
```

```
mv /usr/bin/pip /usr/bin/pip27
```

```
ln -s /usr/local/bin/python3 /usr/bin/python
```

```
ln -s /usr/local/bin/pip3 /usr/bin/pip
```

2.1.4 修改配置文件

可使用如下快捷命令：

```
sed -i "s#/usr/bin/python#/usr/bin/python2.7#" /usr/bin/yum
```

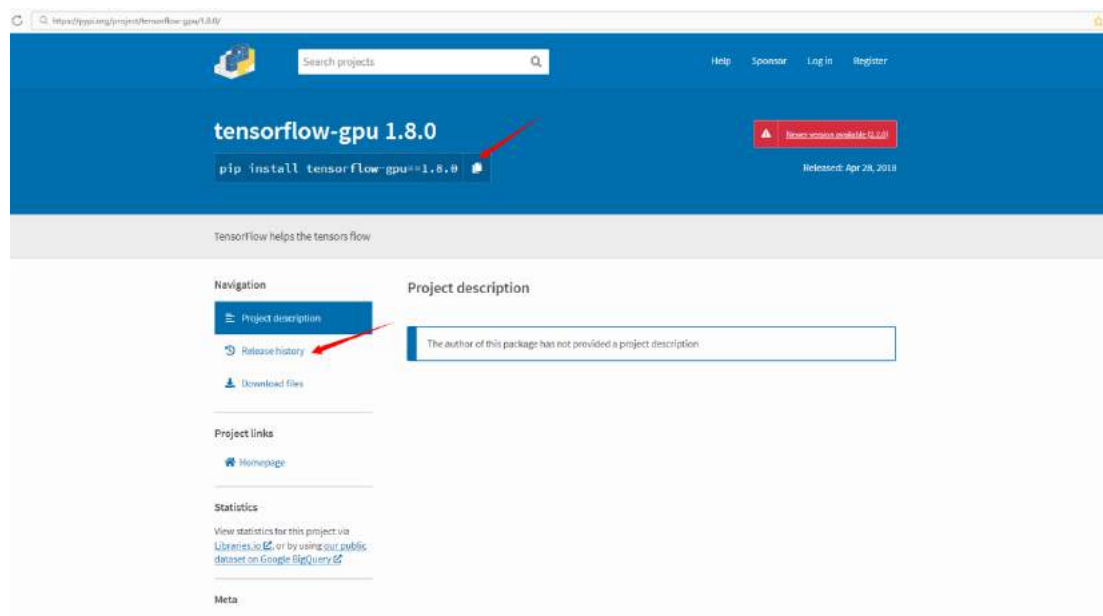
```
sed -i "s#/usr/bin/python#/usr/bin/python2.7#" /usr/libexec/urlgrabber-ext-down
```

```
source /etc/profile
```

```
python -V
```

2.2 安装 tensorflow

tensorflow 版本号查看和下载地址：<https://pypi.org/project/tensorflow-gpu/1.8.0/>



直接粘贴下载命令，在镜像中进行在线安装，或者查看历史版本，找到自己要安装的版本。

2.2.1 安装

在线安装如果总是失败或者网速过慢，可以使用 `pip download` 下载离线包，然后再进行安装。

在线安装：

```
pip install tensorflow-gpu==1.8.0 -user -default-timeout=10000 -ignore-installed -upgrade
```

离线下载：

```
pip download tensorflow-gpu==1.8.0 -d /home/wjy/tian-tan-user-image/tensorflow-gpu1.8.0
```

离线加在线安装：

```
pip install -find-links=/home/wjy/tian-tan-user-image/tensorflow-gpu1.8.0 tensorflow_gpu-1.8.0-cp36-cp36m-manylinux1_x86_64.whl
```

完全离线安装：

```
pip install -node-index -find-links=/home/wjy/tian-tan-user-image/tensorflow-gpu1.8.0tensorflow_gpu-1.8.0-cp36-cp36m-manylinux1_x86_64.whl
```

2.2.2 查看版本

```
pip list |grep tensorflow
```

2.3 安装 opencv

opencv 查看版本和下载地址: <https://pypi.org/project/opencv-python/3.4.3.18/>

2.3.1 安装

在线安装:

```
pip install opencv-python==3.4.3.18 -user -default-timeout=10000 -ignore-installed -upgrade
```

离线下载:

```
pip download opencv-python==3.4.3.18 -d /home/wjy/tian-tan-user-image/opencv3.4.3
```

离线安装:

```
pip install - find-links=/home/wjy/tian-tan-user-image/opencv3.4.3 opencv_python-3.4.3.18-cp27-cp27mu-manylinux1_x86_64.whl
```

2.3.2 查看版本

```
pip list |grep opencv*
```

2.3.3 安装 opencv 共享库

由于使用 pip 安装 opencv 后,并不会自动安装 opencv 共享库,需要手动安装,不然 python import cv2 时会报错。共享库: libSM.so.6, libXrender.so.1, libXext.so.6

```
yum whatprovides libSM.so.6 libXrender.so.1 libXext.so.6
```

```
[root@5d7aa9c589a9 /]# yum whatprovides libSM.so.6 libXrender.so.1 libXext.so.6
Loaded plugins: fastestmirror, ovl
Loading mirror speeds from cached hostfile
 * base: mirrors.bfsu.edu.cn
 * extras: mirrors.neusoft.edu.cn
 * updates: mirrors.neusoft.edu.cn
libSM-1.2.2-2.el7.i686 : X.Org X11 SM runtime library
Repo                : base
Matched from:
Provides            : libSM.so.6

libXrender-0.9.10-1.el7.i686 : X.Org X11 libXrender runtime library
Repo                : base
Matched from:
Provides            : libXrender.so.1

libXext-1.3.3-3.el7.i686 : X.Org X11 libXext runtime library
Repo                : base
Matched from:
Provides            : libXext.so.6
```

在线安装:

```
yum install -y libSM libXrender libXext --setopt=protected_multilib=false
```

离线安装:

```
yum install --downloadonly --downloadaddir=/home/wjy/opencv-share libSM-1.2.2-2.el7.x86_64 libXrender-0.9.10-1.el7.x86_64 libXext-1.3.3-3.el7.x86_64
```

安装

```
cd /home/wjy/opencv-share
```

```
rpm -ivh *.rpm --force --nodeps
```

2.4 安装 Keras

Keras 查看版本和下载地址: <https://pypi.org/project/Keras/2.2.4/>

2.4.1 安装

在线安装:

```
pip install Keras==2.2.4 --user --default-timeout=10000 --ignore-installed --upgrade
```

离线下载:

```
pip download Keras==2.2.4 -d /home/wjy/tian-tan-user-image/keras2.2.4
```

离线安装:

```
pip install --no-index --find-links=/home/wjy/tian-tan-user-image/keras2.2.4 Keras-2.2.4-py2.py3-none-any.whl
```

2.4.2 查看版本

```
pip list |grep Keras*
```

2.5 安装 numpy

numpy 查看版本和下载地址: <https://pypi.org/project/numpy/1.15.4/>

2.5.1 安装

在线安装:

```
pip install numpy==1.15.4 --user --default-timeout=10000
```

离线下载:

```
pip download numpy==1.15.4 -d /home/wjy/tian-tan-user-image/numpy1.15.4
```

离线安装:

```
pip install --no-index --find-links=/home/wjy/tian-tan-user-image/numpy1.15.4 numpy-1.15.4-cp36-cp36m-manylinux1_x86_64.whl
```

2.5.2 查看版本

三. 配置适配平台的组件

按照《镜像组件安装手册》安装 openssh、openssl、jupyter 组件; 如果使用 Dockerfile 文件制作镜像, 可以将《镜像组件安装手册》里面的 Dockerfile 内容合并到该文档的 Dockerfile 文件中, 然后使用 Dockerfile 进行在线制作。

可视化功能

在开发环境和任务管理模块 AIStation 提供主流框架模型可视化的功能。下面主要介绍可视化功能如何使用:

任务管理可视化

在任务管理模块，单机/分布式类型的任务有可视化功能。

Caffe 框架支持 Netscope 网络结构可视化以及 caffe 训练过程可视化；

Tensorflow、Mxnet、Pytorch 框架是基于 TensorBoard 可视化工具实现可视化功能；PaddlePaddle 框架是基于的 VisualDL 工具。

在任务训练过程中查看或者任务训练完成后，通过点击可视化按钮，进入可视化页面。下面通过选择完成的任务介绍可视化功能。

Caffe 可视化

caffe 网络结构可视化

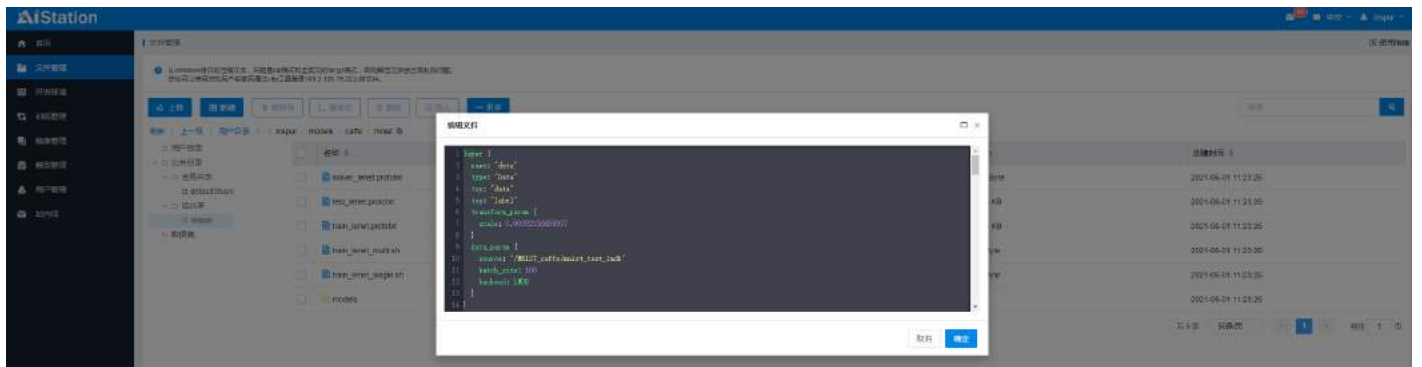
可视化任务—创建

点击可视化按钮，打开 Netscope 标签页，开始创建 Caffe 可视化任务。



Netscope

进入文件管理系统，打开 Caffe 的网络结构文件（train_lenet.prototxt），复制全文；在左侧输入框内粘贴全文，键入 shift+enter，得到网络结构。

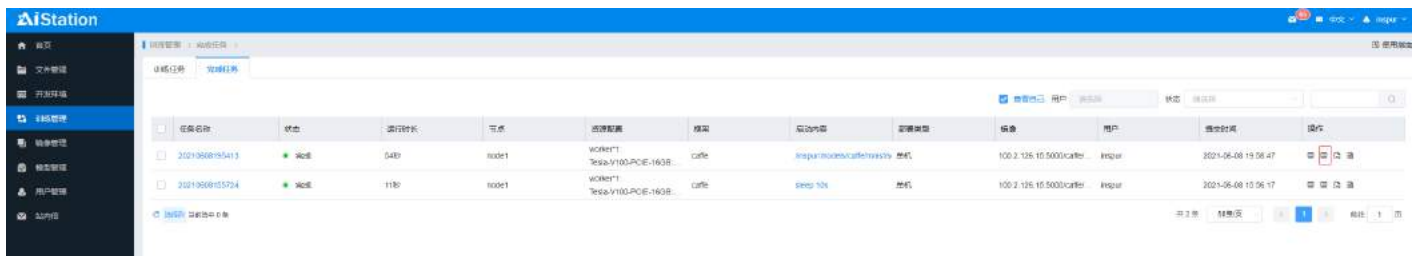




caffe 训练过程可视化

可视化任务—创建

点击训练过程可视化按钮，开始创建 Caffe 训练过程可视化。



可视化

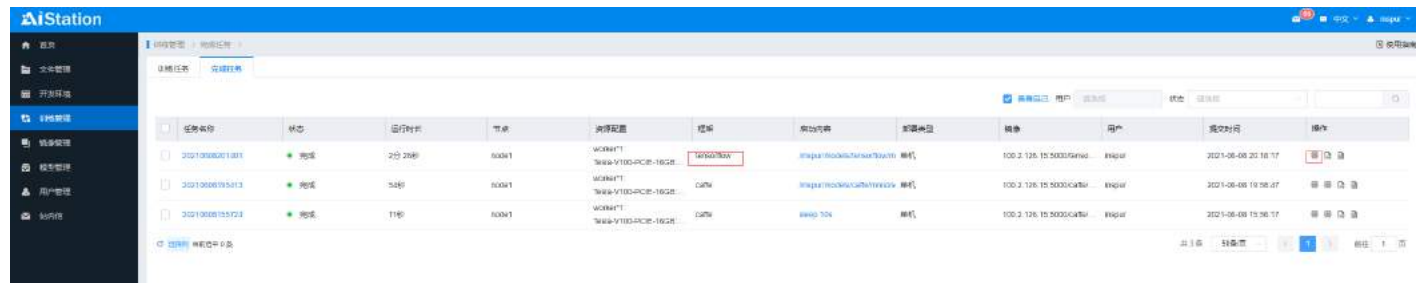
弹出训练过程可视化窗口，查看训练时间，损失率，学习率。



TensorFlow 可视化

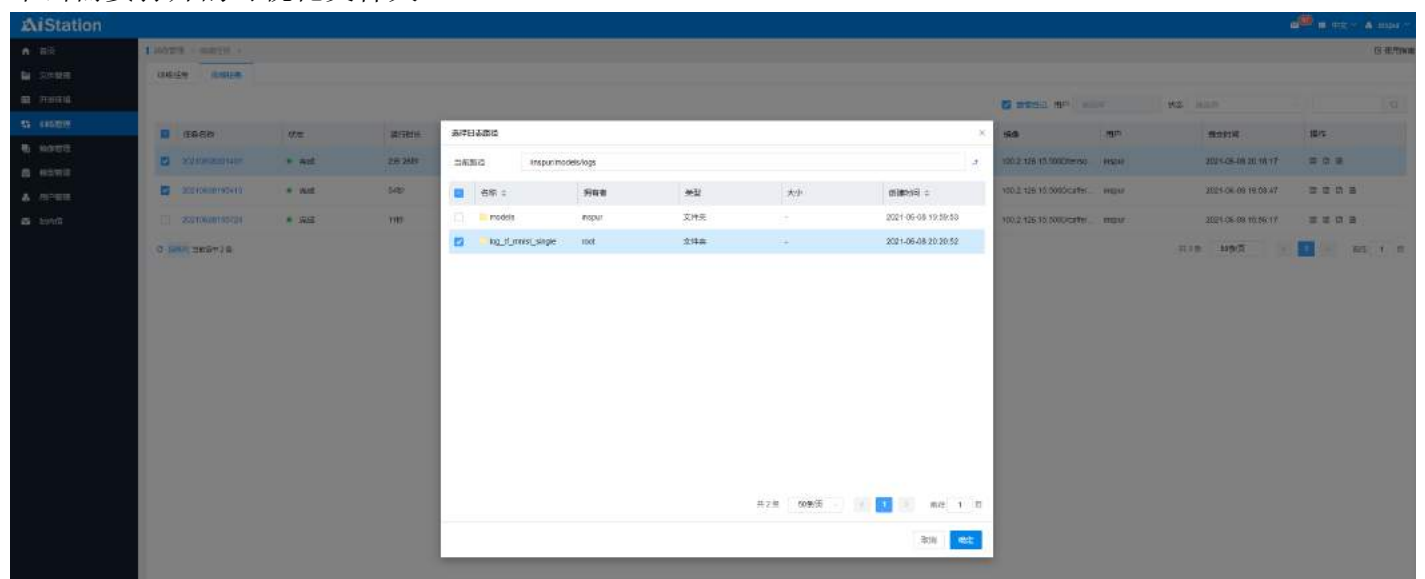
可视化任务—创建

点击可视化按钮，开始创建 TensorFlow 可视化任务。



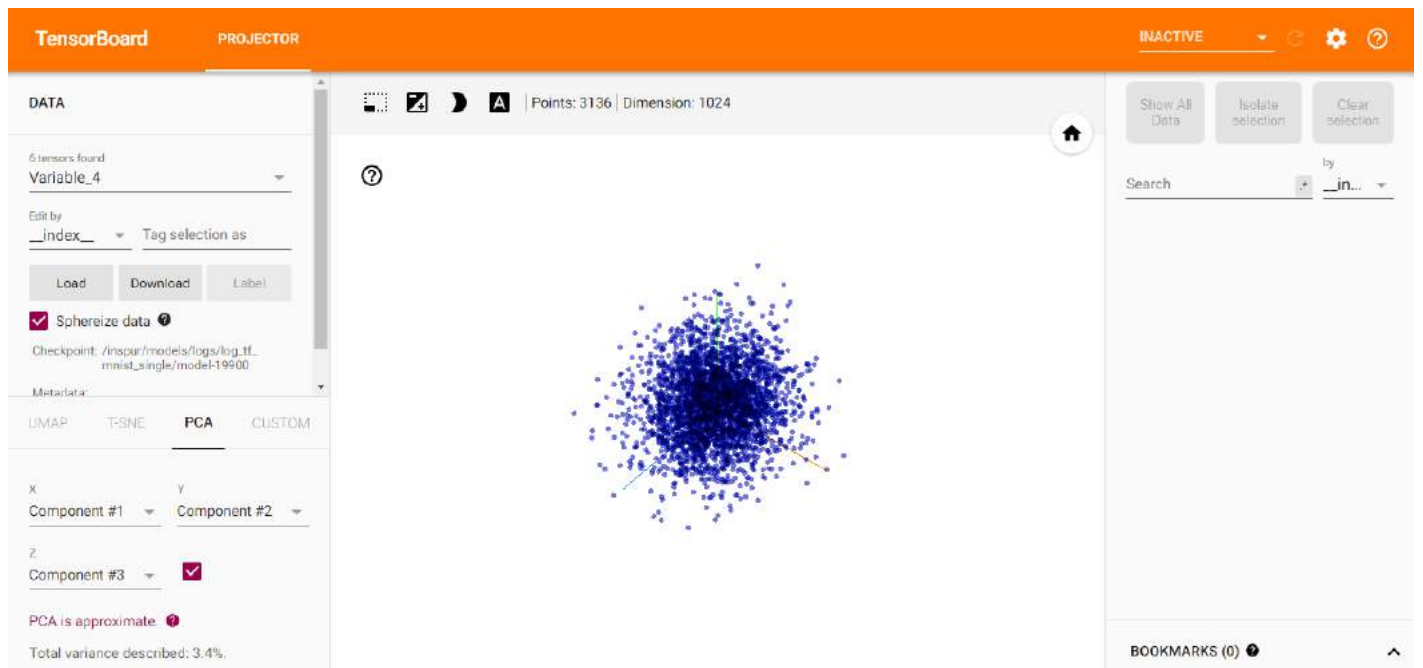
选择启动文件夹

单击需要打开的可视化文件夹。



TensorBoard

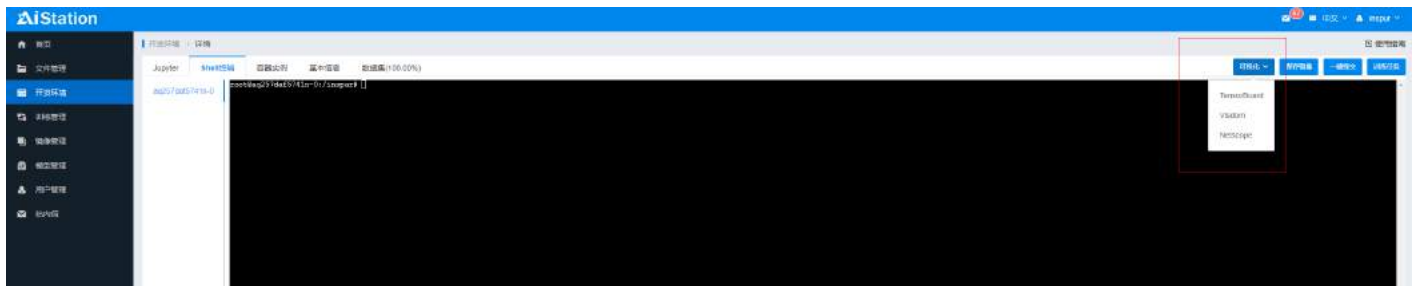
打开 TensorBoard 标签页，展示 TensorFlow 可视化信息。



其它框架的任务可视化可参考“TensorFlow 可视化”章节

开发环境可视化

在开发环境模块，当创建的开发环境运行时，您可以选择相应的可视化工具，如下图：



在开发环境，AIStation 默认提供这三种可视化工具，使用 Tensorboard 工具时，请选择正确的日志文件；Netscope 可视化工具的使用与任务管理一样，参考上述内容。

Visdom 工具是针对 Pytorch 框架任务的可视化，当使用此可视化工具时，请确认所创建的开发环境是基于 Pytorch 框架镜像创建的，具体使用方式请参考如下介绍：

1. 示例 demo.py 文件地址：‘/{user-name}/ visualization/visdom/demo.py’；
2. 创建基于 Pytorch 框架镜像的开发环境；
3. 打开 Visdom 可视化工具；
4. 修改 demo.py 文件中 hostname 和端口号与 Visdom 可视化工具所打开网页一致；
5. 在开发环境 shell 终端中执行 python demo.py 命令，运行 demo 示例，即可在 Visdom 打开的页面查看此 demo 的可视化。

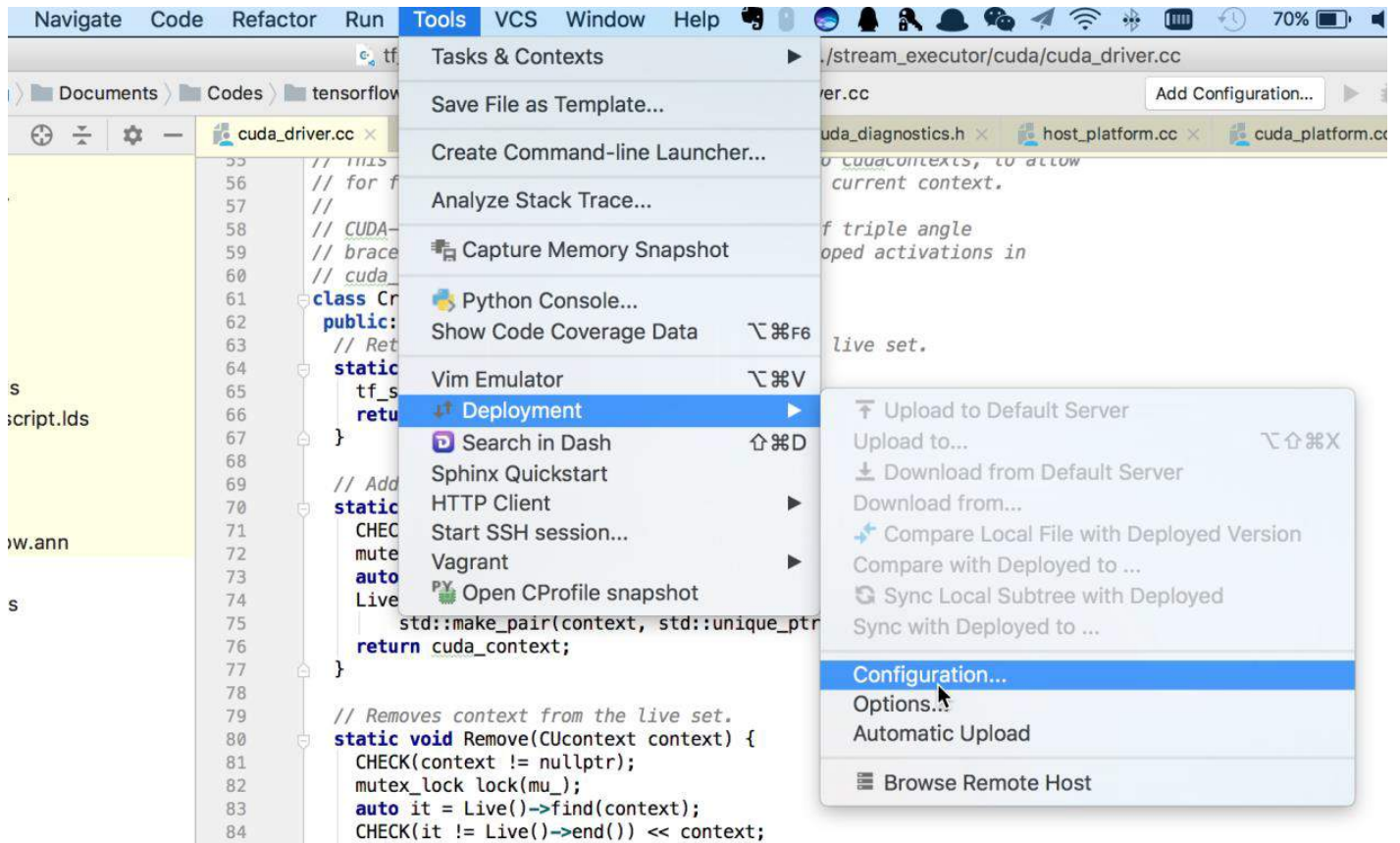
PyCharm 对接 AIStation 开发环境

以下示例所用 PyCharm 版本：PyCharm 版本 2018.3 (Professional Edition)

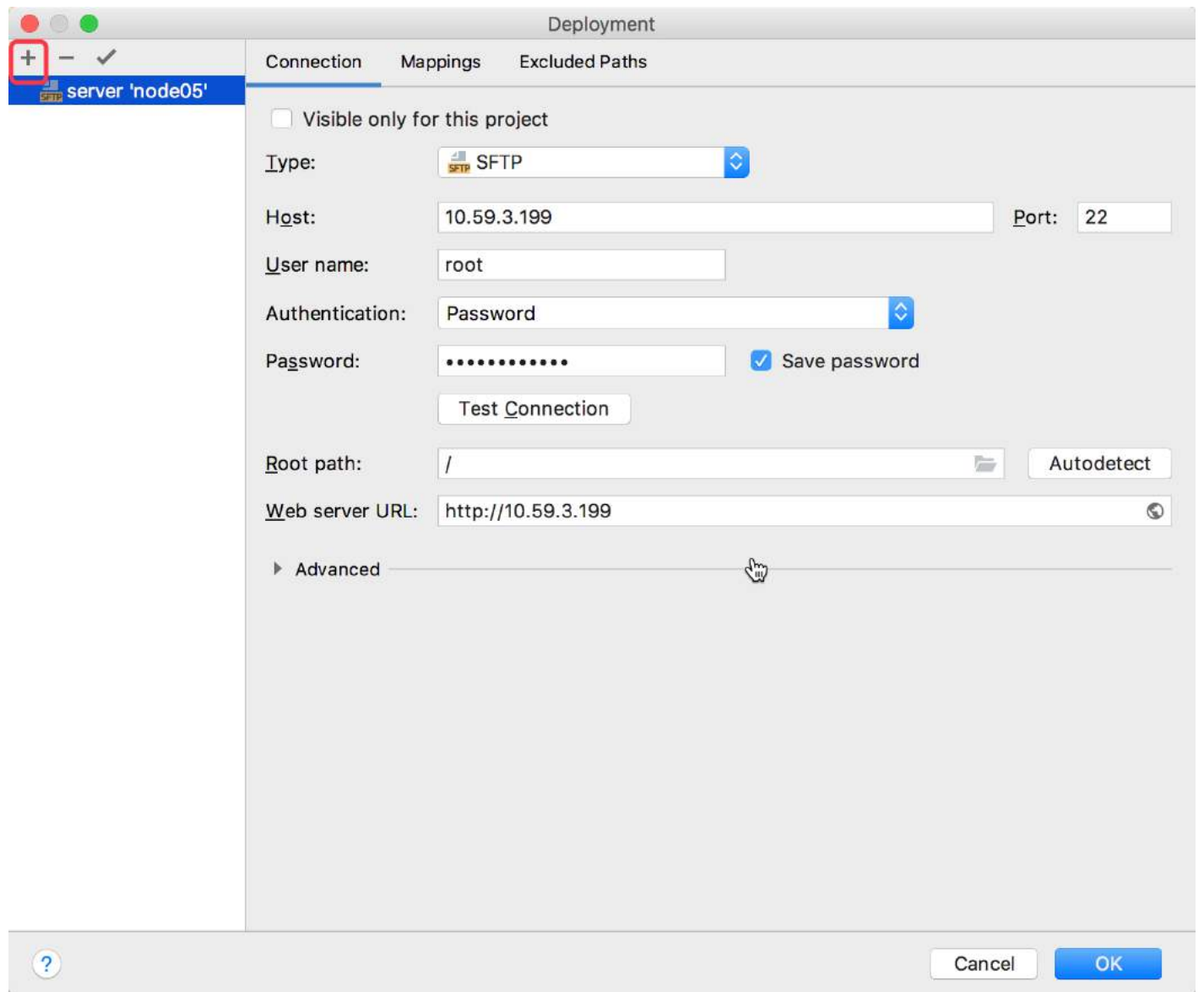
AIStation 开发环境 ssh 服务运行正常，对应 ip: 10.151.11.53；端口：54963；密码：939549。

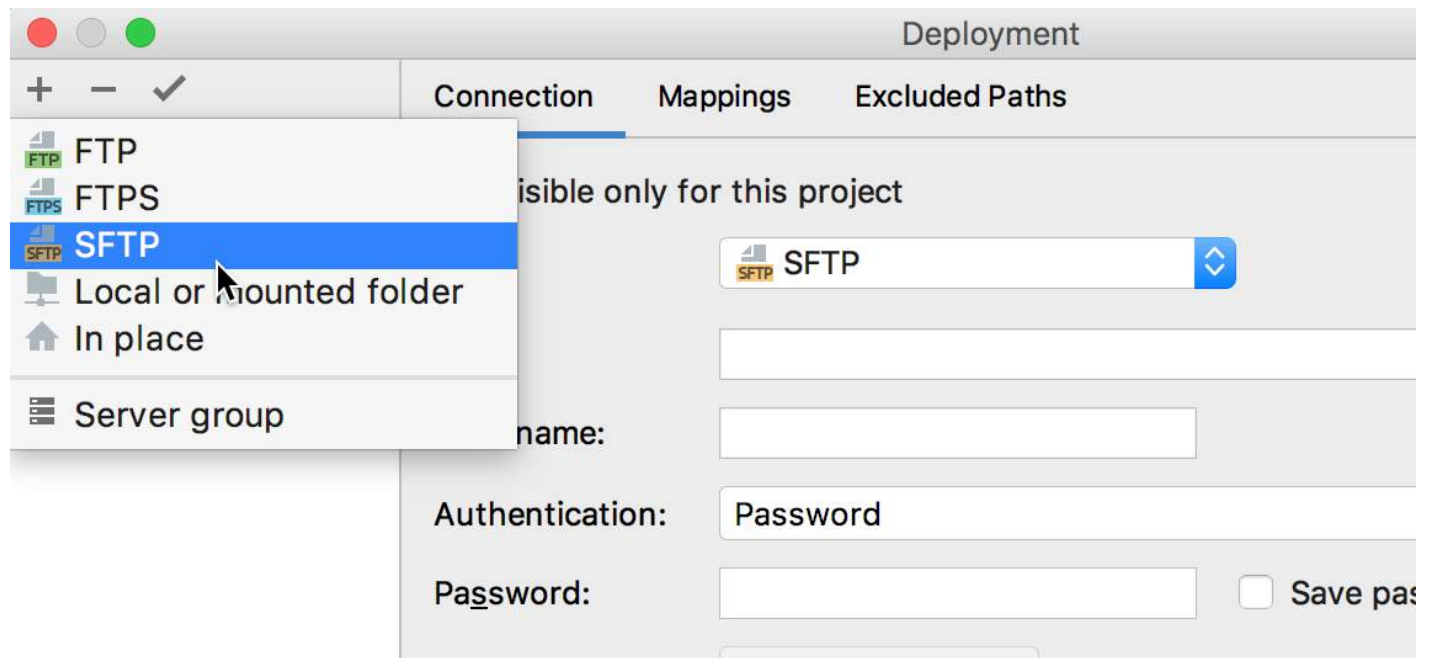
1. Pycharm 配置

打开 pycharm，选择 Tools -> Deployment -> Configuration...

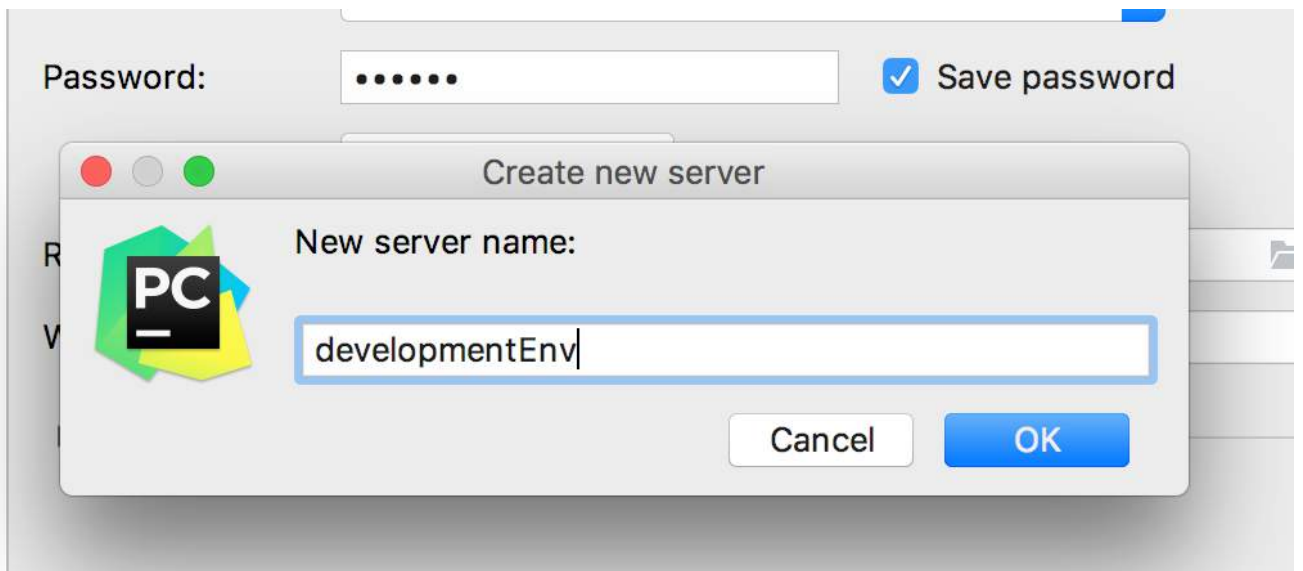


点击左上角加号，选择 SFTP

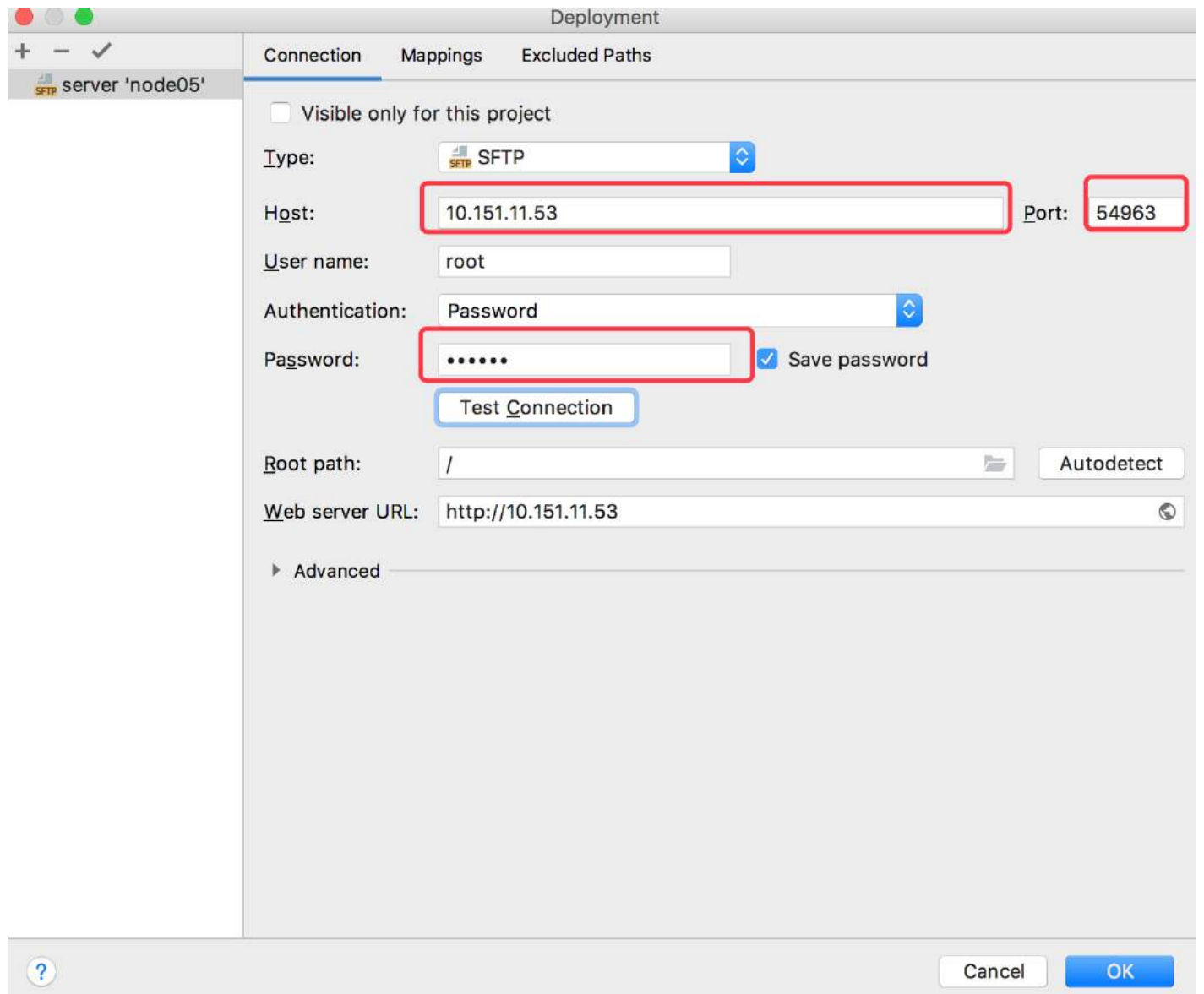




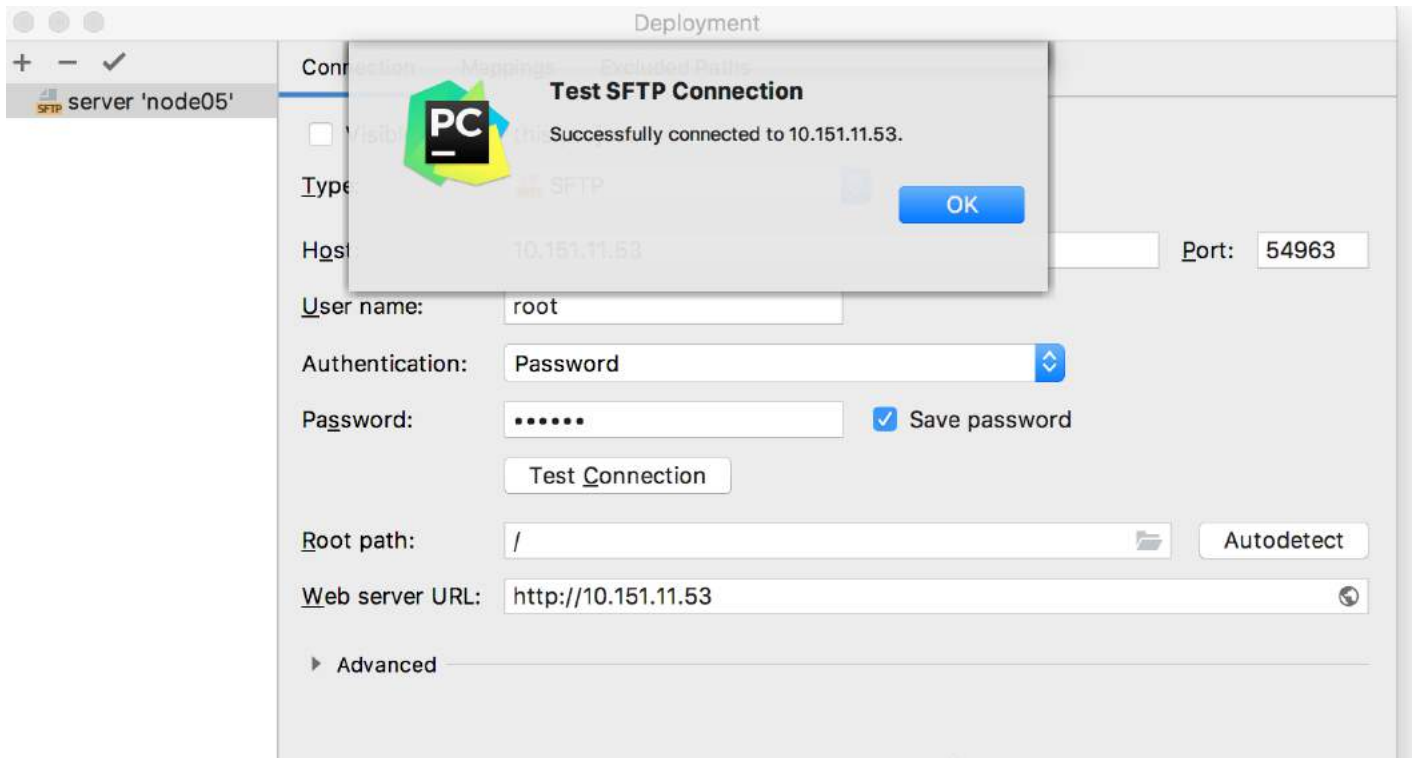
输入配置名称:



在右侧的 Connection 页面中，Type 选择为 SFTP，对应 Host 和 Post 输入在第 2 步中得到的 ip、端口。
在 Password 处输入在第 2 步中得到的密码。



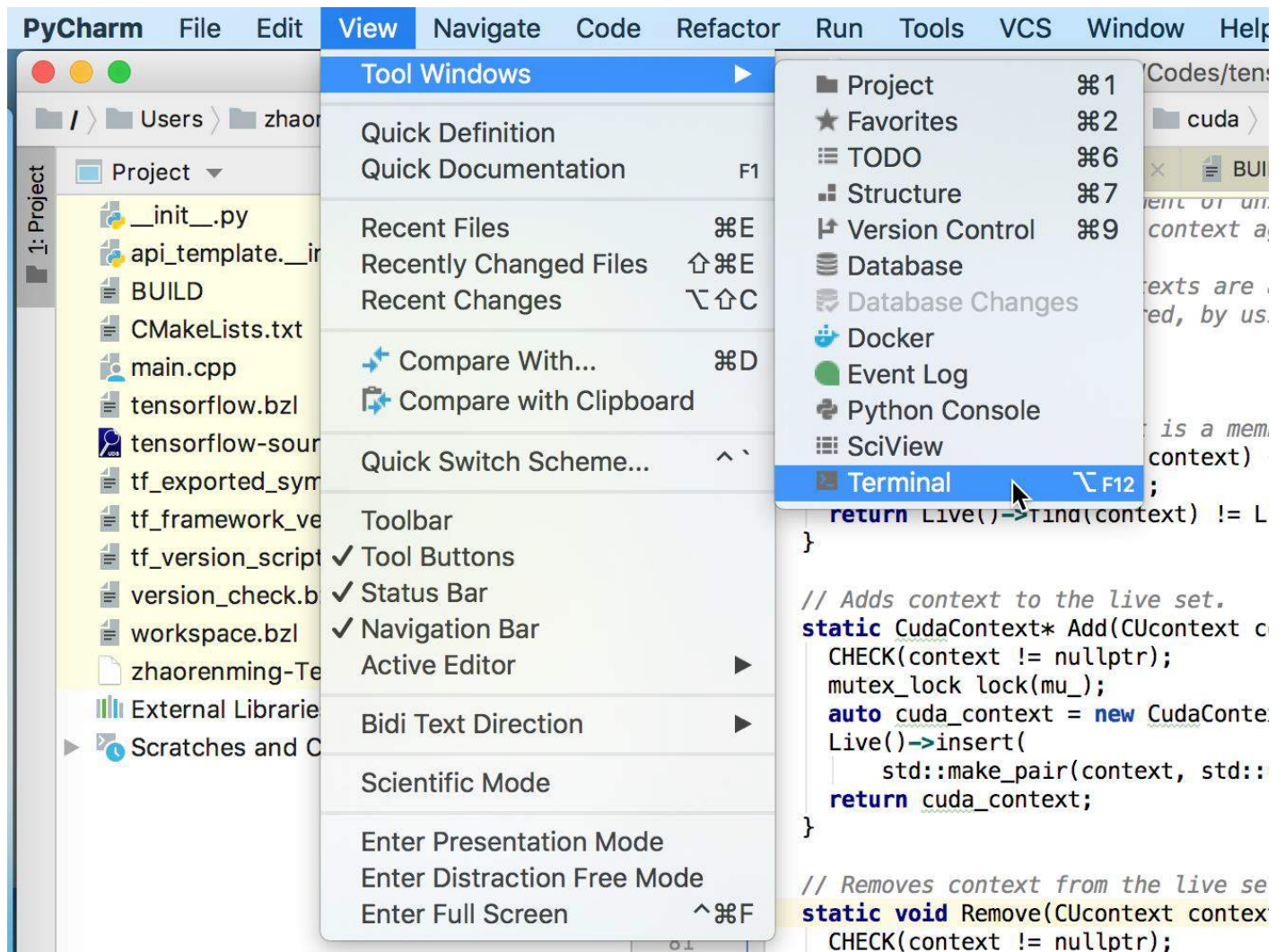
点击“Test Connection”按钮，测试连接。显示连接成功。



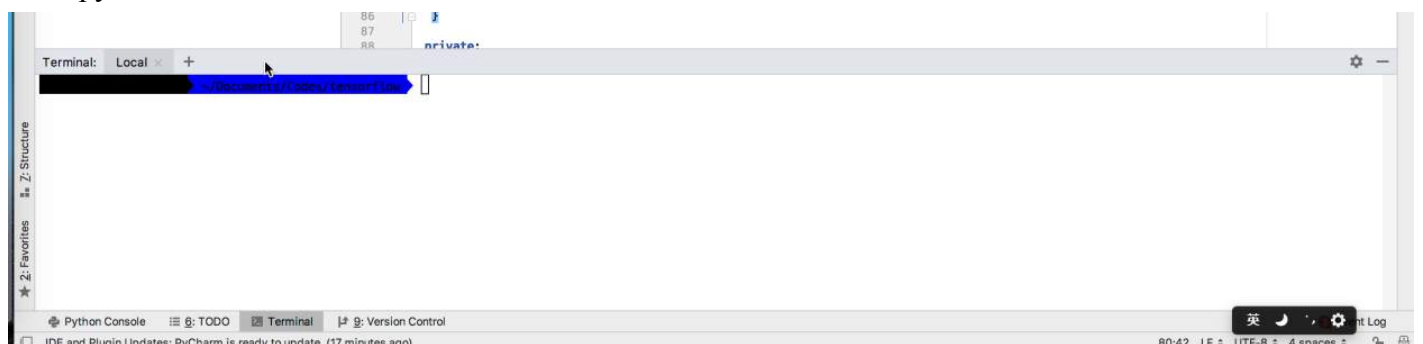
此时连接配置完成。

2. 通过 SSH 连接到开发环境

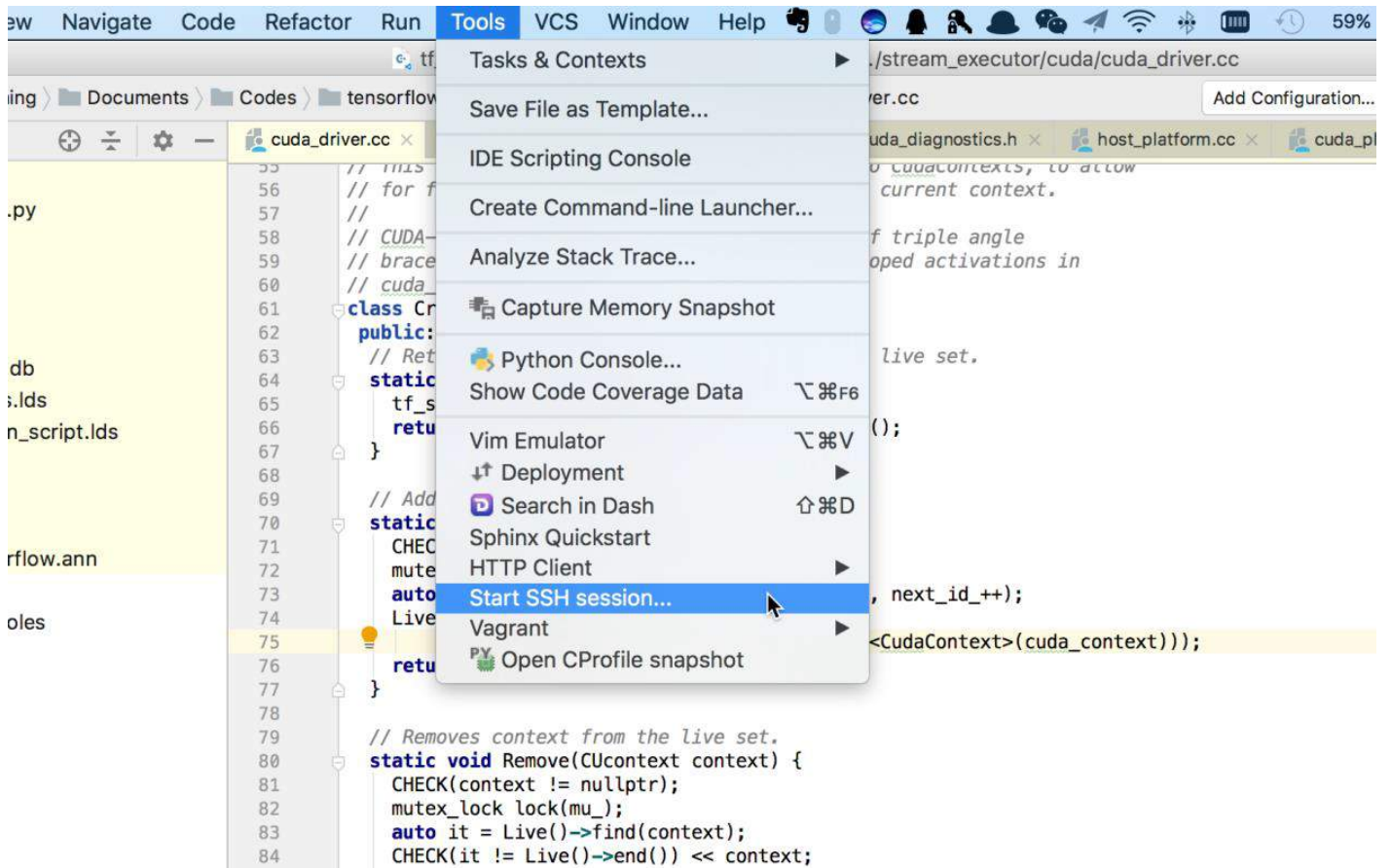
打开 pycharm，选择 View->Tool Windows->Terminal。



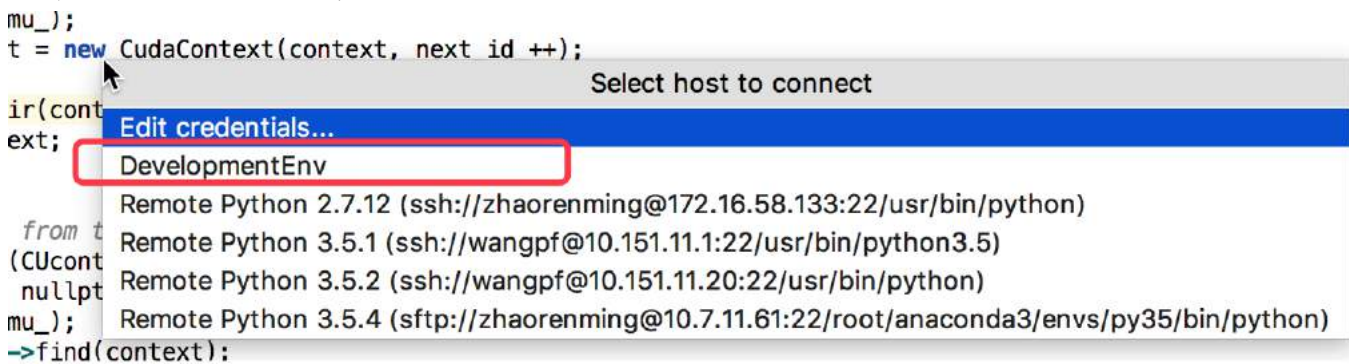
此时 pycharm 中出现命令行窗口。



选择菜单 Tools->Start SSH session...



在弹出的窗口中选择刚才的配置：

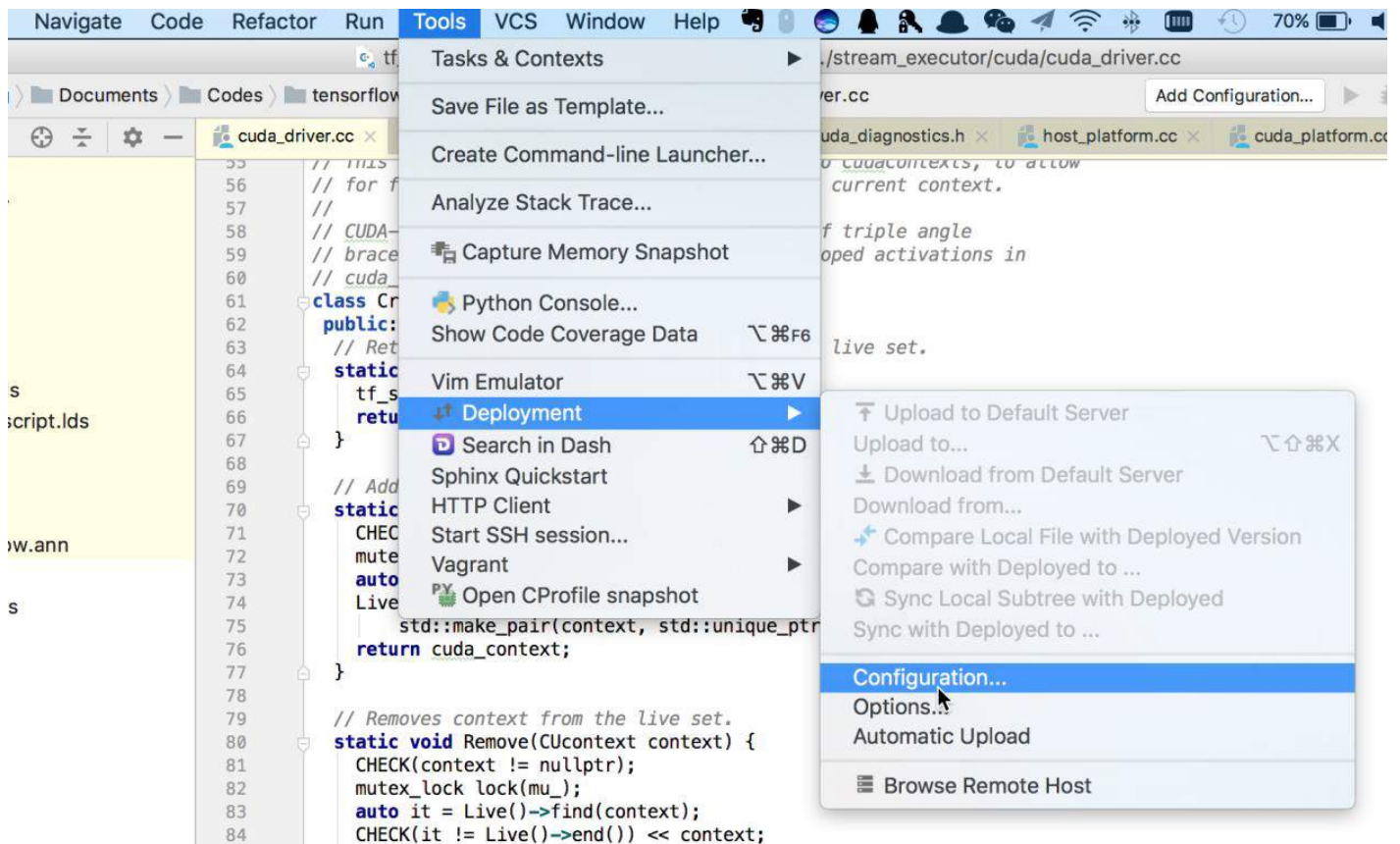


查看刚才的命令行窗口，已经成功连接至 AIStation 开发环境所在的容器。

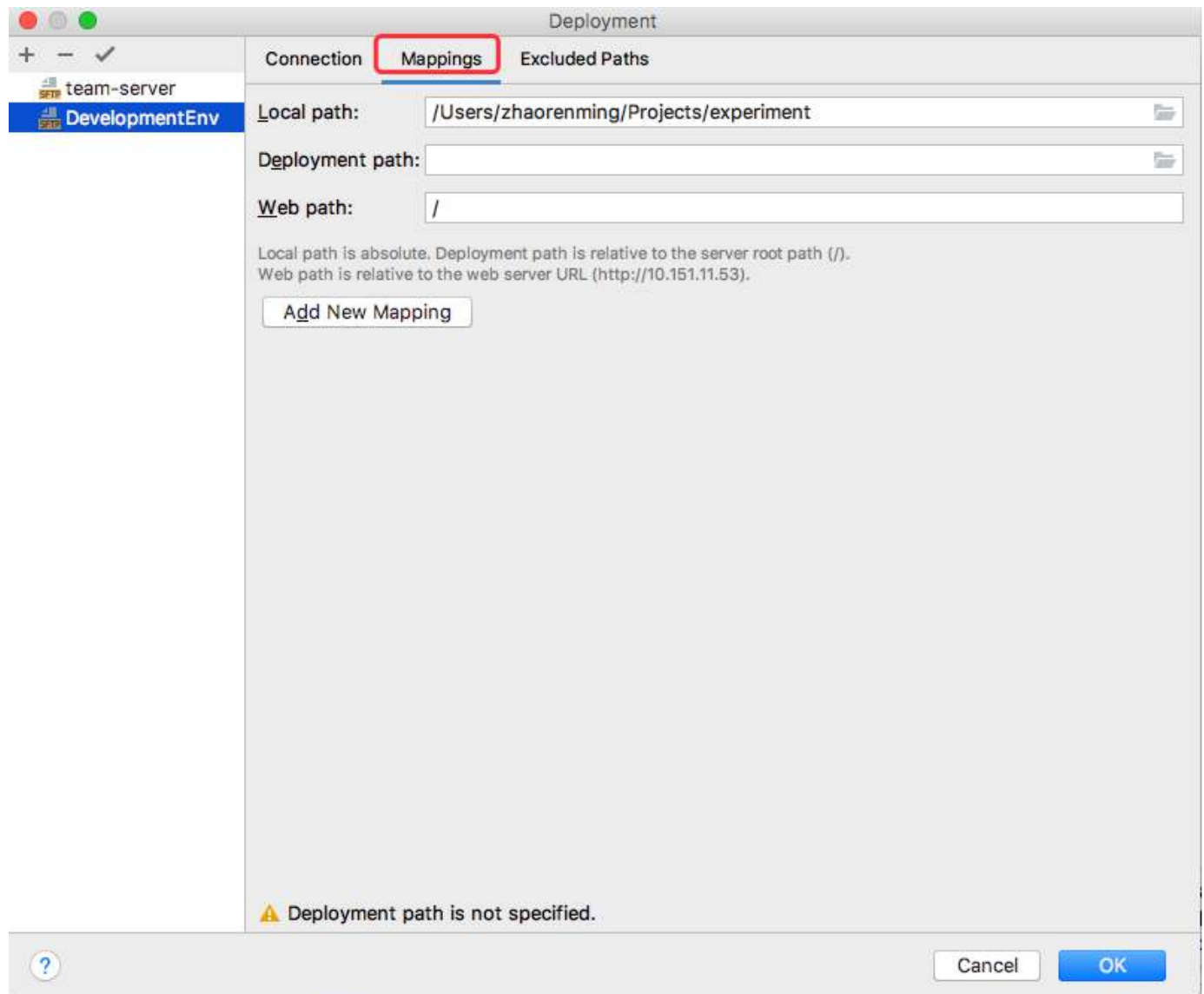


3. 配置 Remote Debug

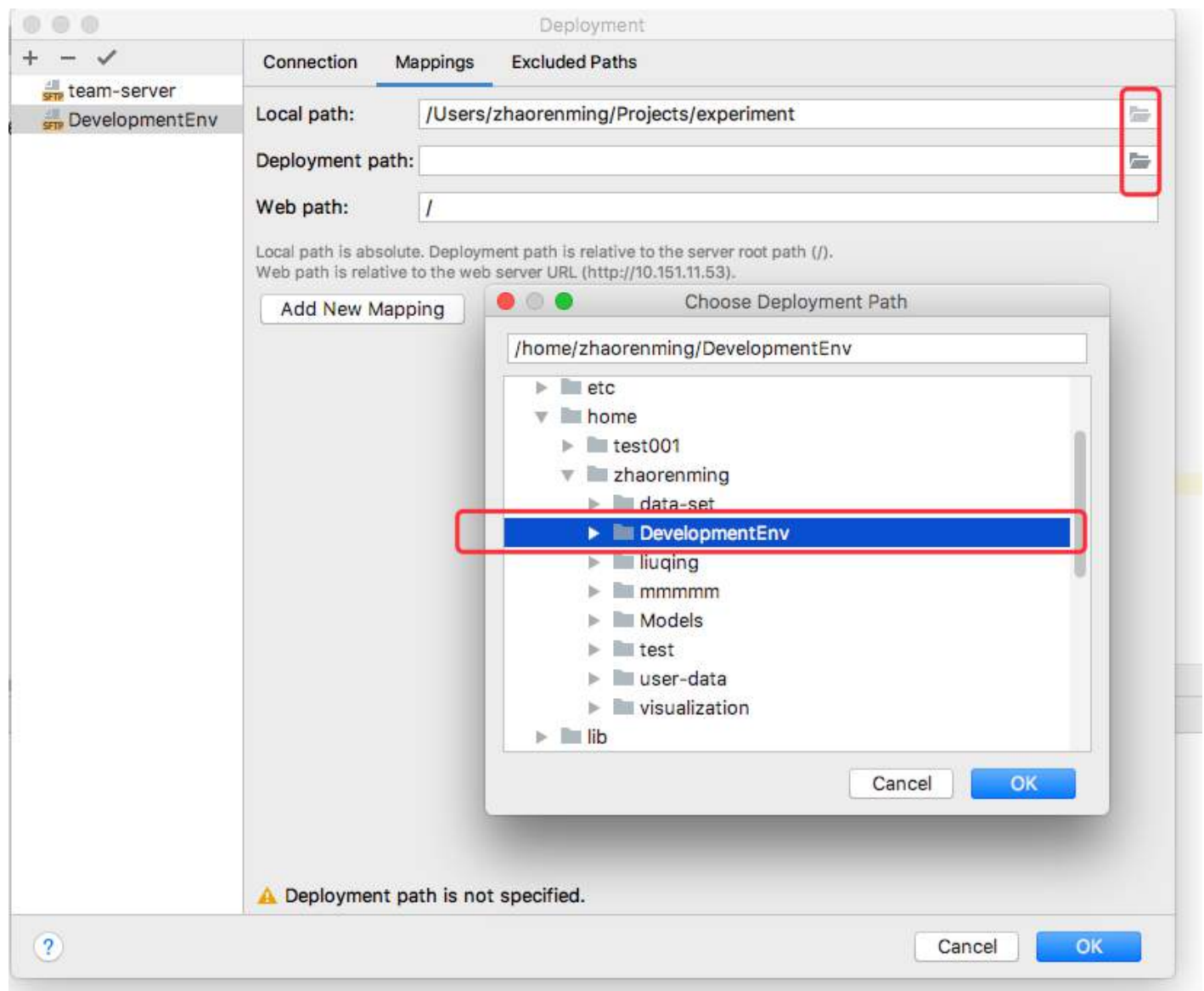
打开 pycharm, 选择 Tools -> Deployment -> Configuration...



选择之前配置的 DevelopmentEnv 的 Server。点击 Mappings。



点击”文件夹”图标，分别选择本地和远端的路径。

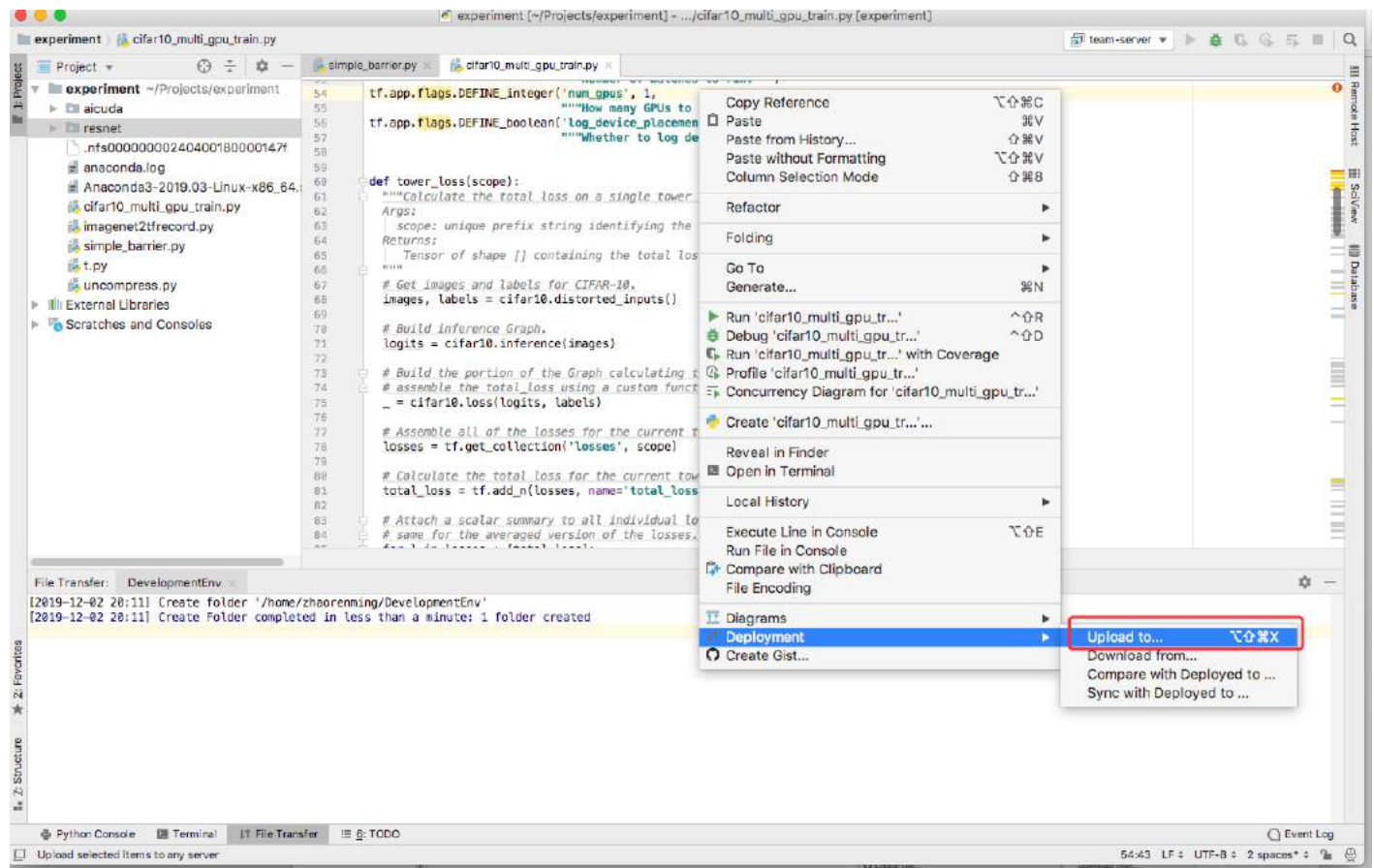


Local path 设置为本地代码所在的路径。

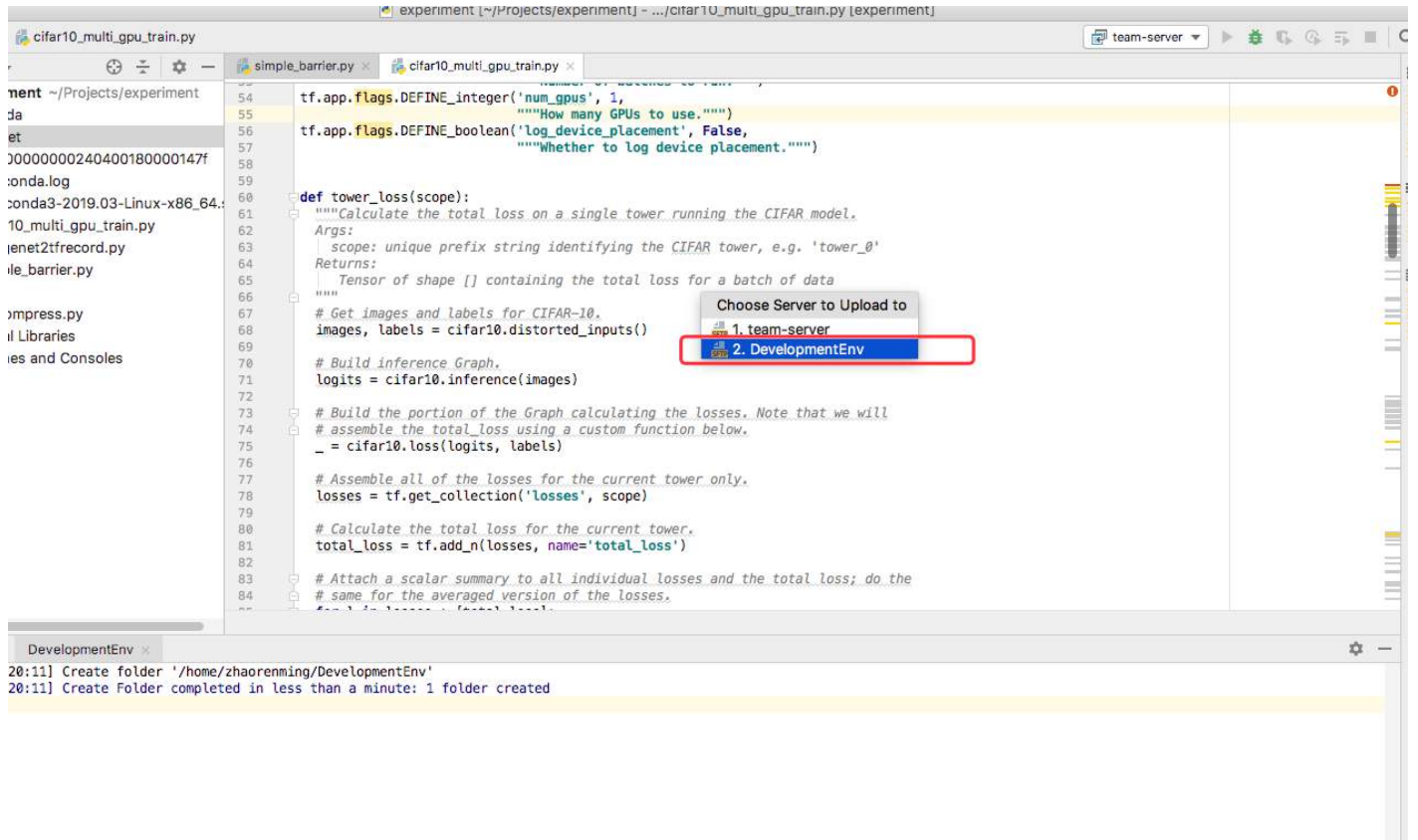
Deployment path 设置为 AIStation 开发环境中，代码所在的路径。

如图中所设置。

在 pycharm 对应的代码文件或工程文件夹点击右键，选择 Deployment->Upload to...



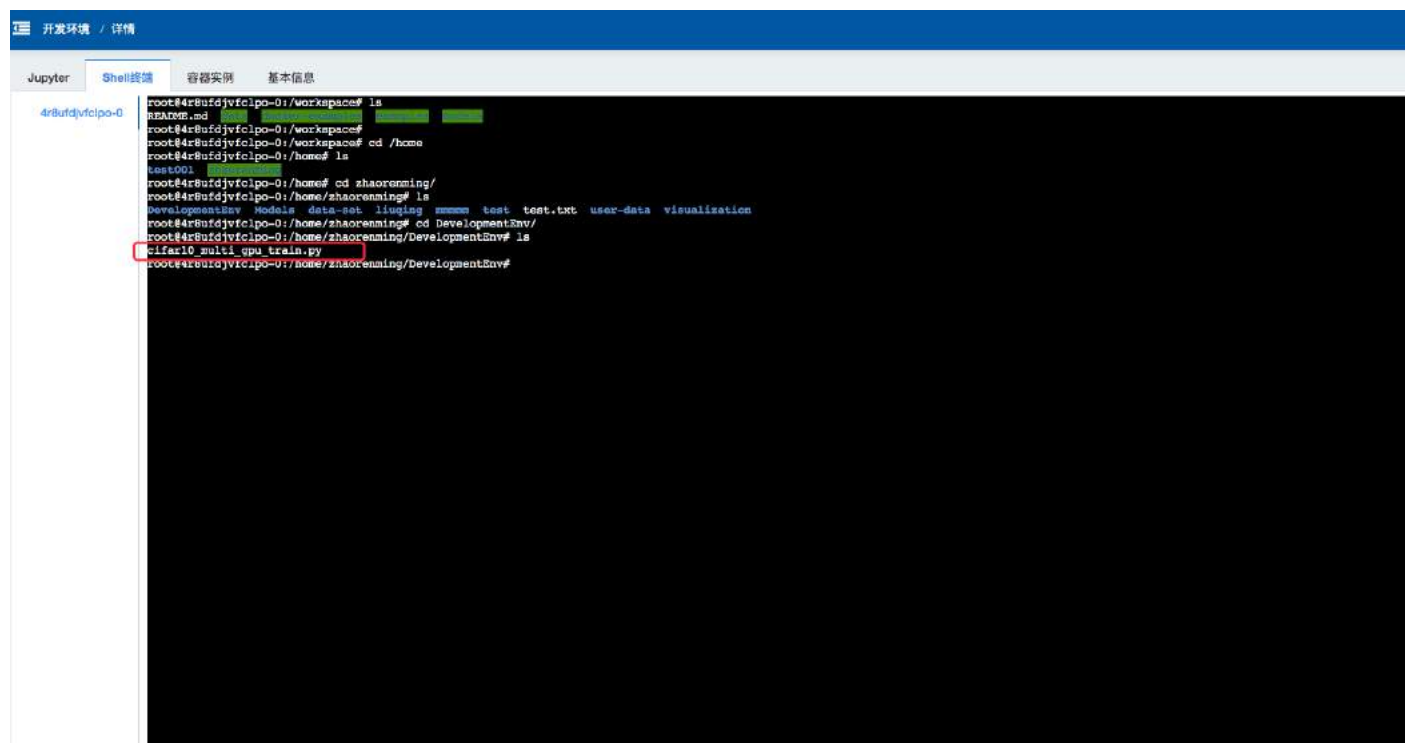
在弹出的对话框中选择，之前创建的 Server 配置” DevelopmentEnv”。



出现以下提示，代表本地的代码文件已经上传至 AIStation 开发环境对应的目录中。



查看开发环境：



4. 开发环境配置相关 debug 包

在安装了 pycharm 的操作系统中可以找到 Debug 包。

MacOS 中对应路径:

```
/Applications/PyCharm.app/Contents/debug-eggs/pycharm-debug-py3k.egg
```

```
/Applications/PyCharm.app/Contents/debug-eggs/pycharm-debug.egg
```

Windows 中对应路径:

Pycharm 安装目录下 /debug-eggs/pycharm-debug-py3k.egg

Pycharm 安装目录下 /debug-eggs/pycharm-debug.egg

pycharm-debug.egg 和 pycharm-debug-py3k.egg 分别对应本地 python 解释器为 python2 和 python3 的情况。

例如需要调试 Python3 的程序, 那么将 pycharm-debug-py3k.egg 拷贝至开发环境中 (通过 scp 命令或 xftp、FileZilla 等工具可以上传)。

```
scp -P 46313 pycharm-debug-py3k.egg root@10.151.11.53:/home/zhaorenming/DevelopmentEnv
```

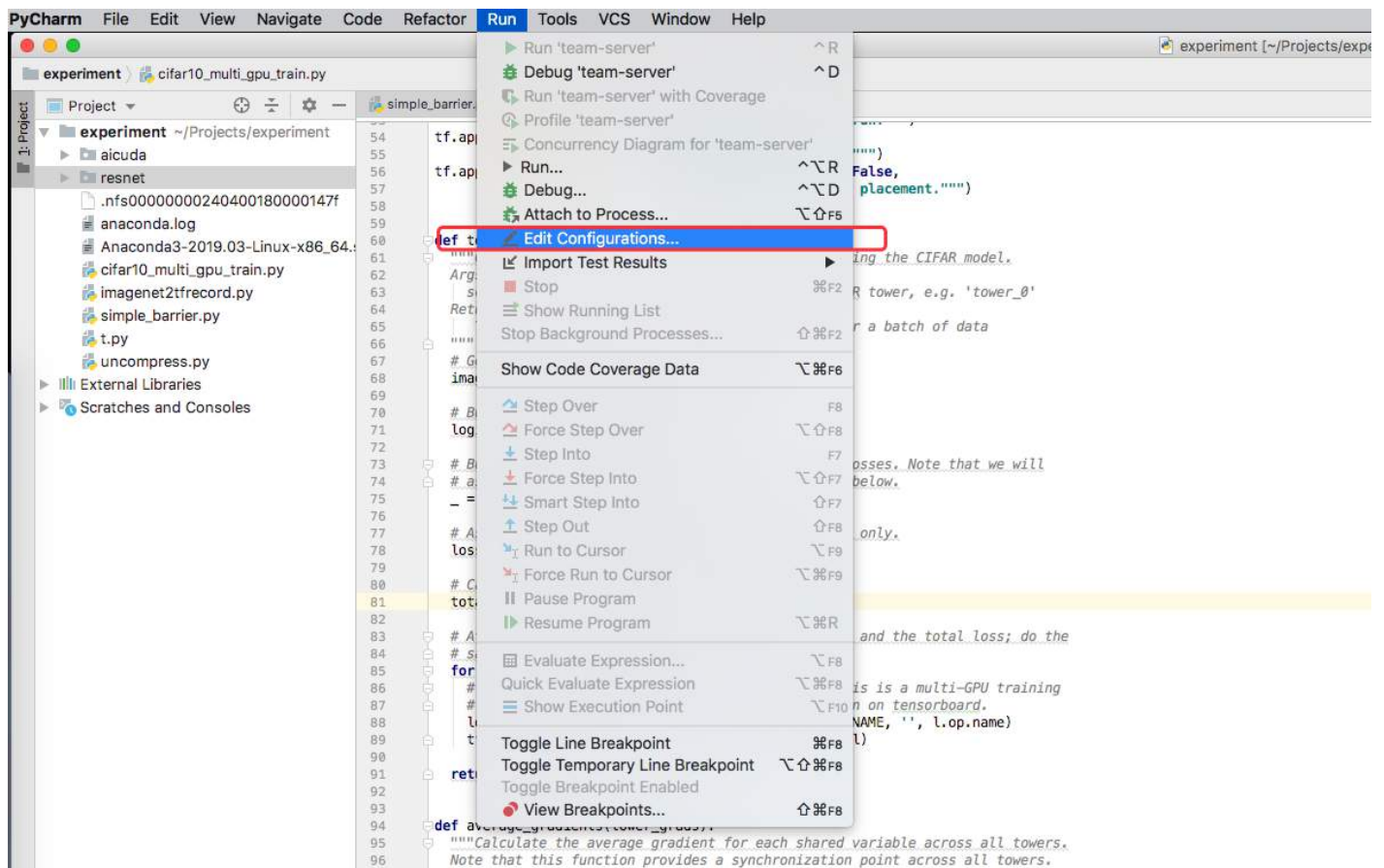


在开发环境中，运行 `easy_install pycharm-debug-py3k.egg` 进行安装。

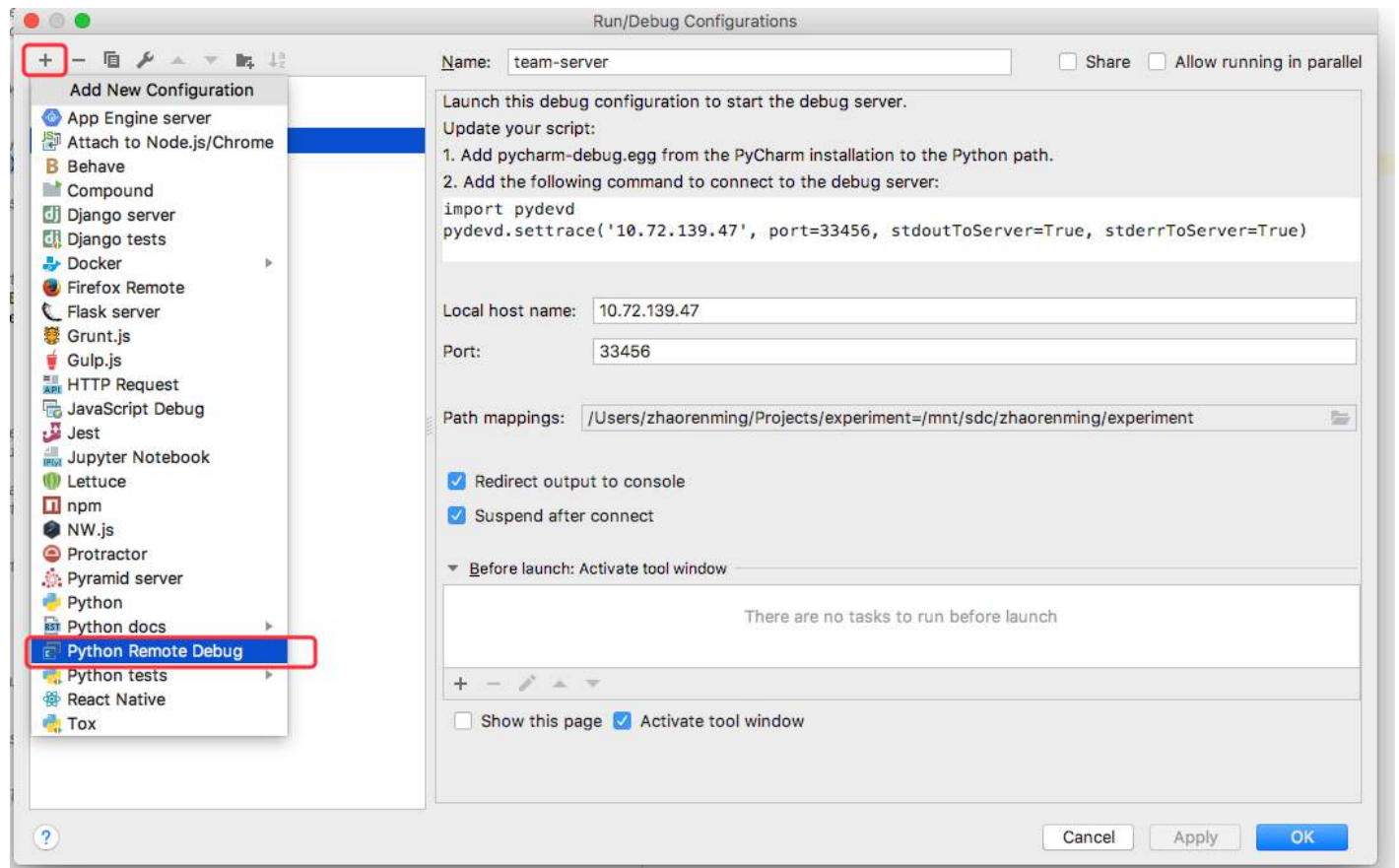
运行 `pip install pydevd`

5. 配置 Remote debug

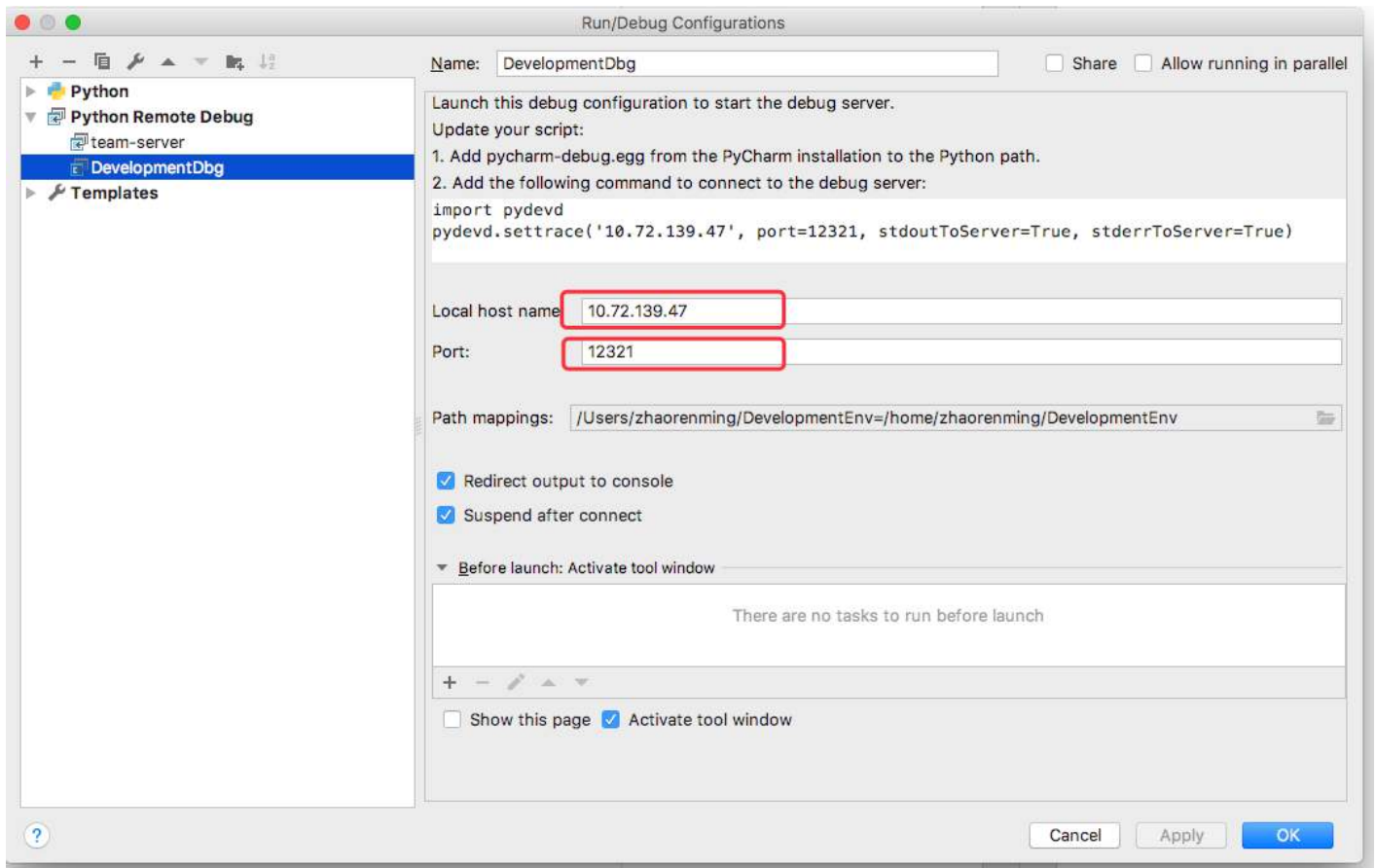
点击 Run->Edit Configurations ...



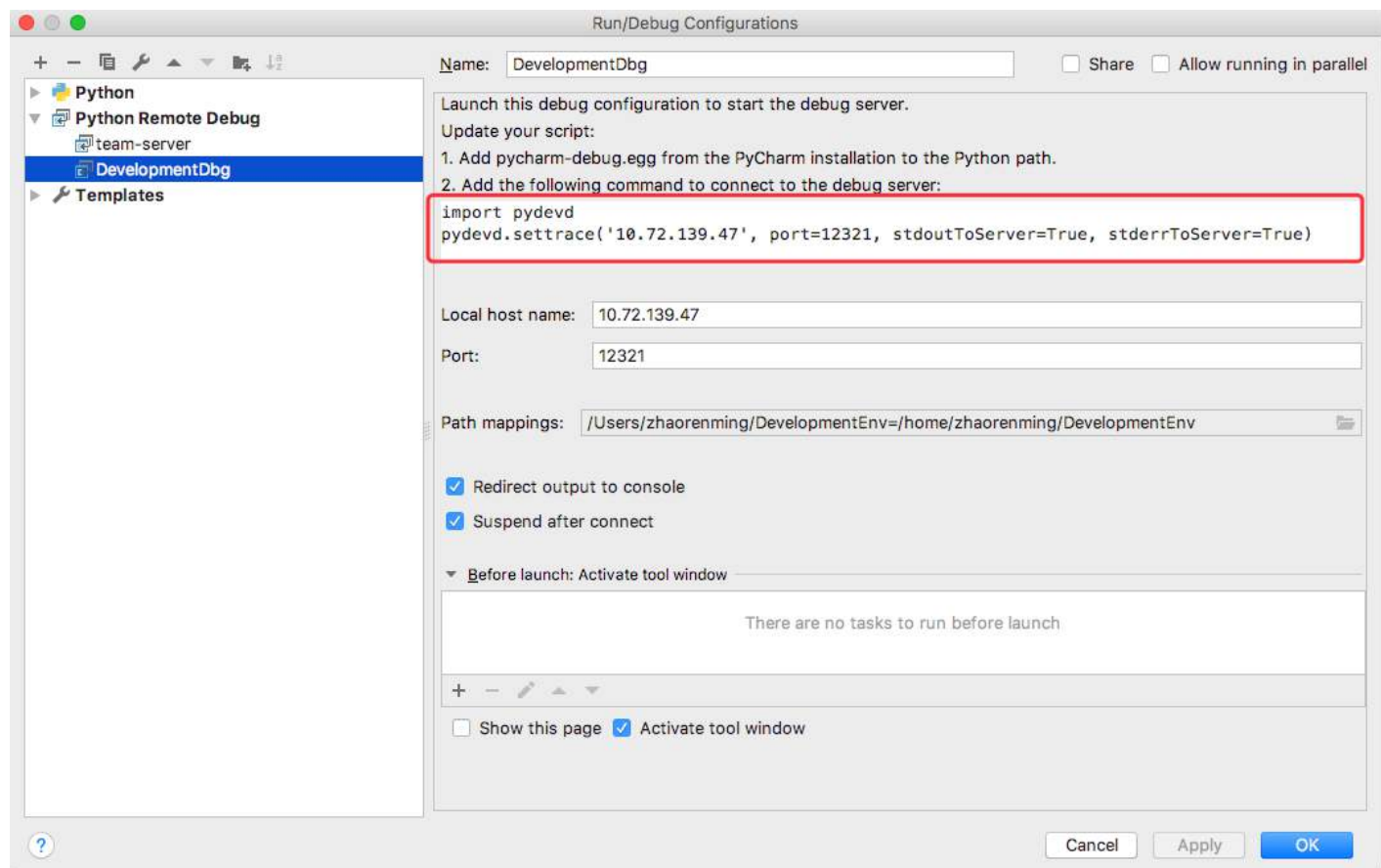
在弹出的对话框中点击左上角”+”号。选择”Python Remote Debug”。



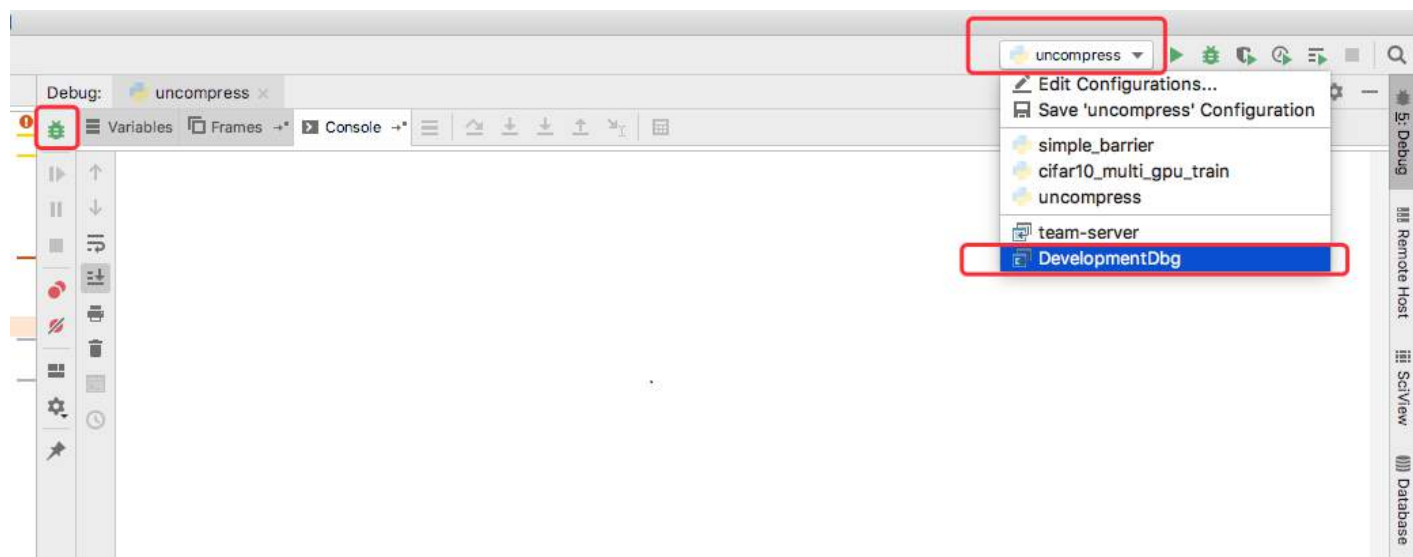
将这个 Configuration 命名为 DevelopmentDbg, 在 Local host name 处填写本地的 ip 地址, 在 port 处填写一个本地未被占用的端口。



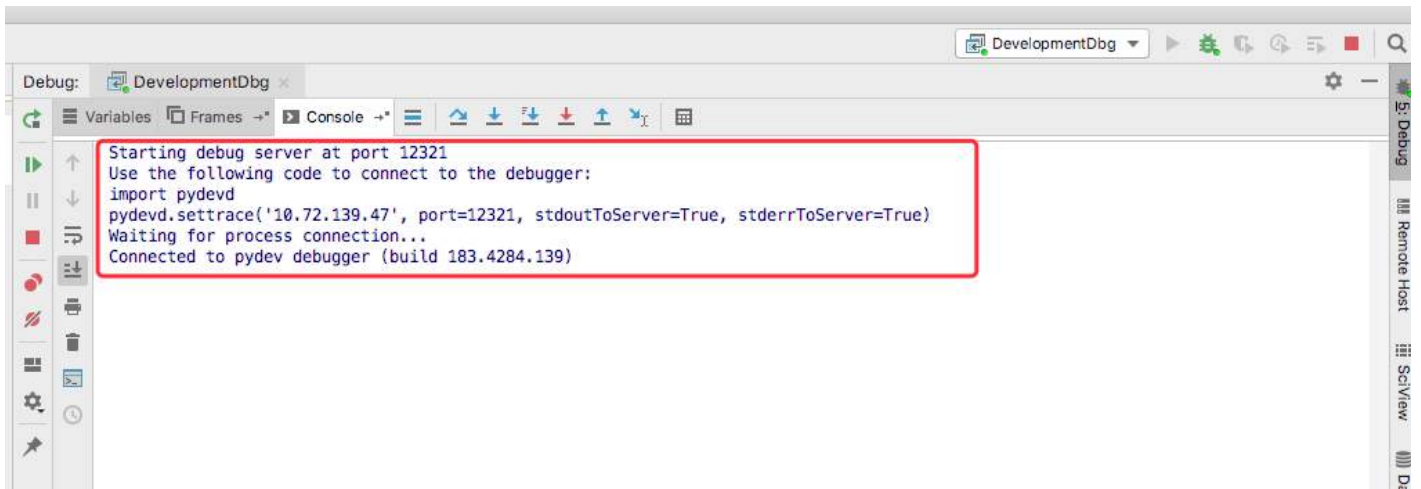
之后复制红框中的两句代码，加入到程序中。



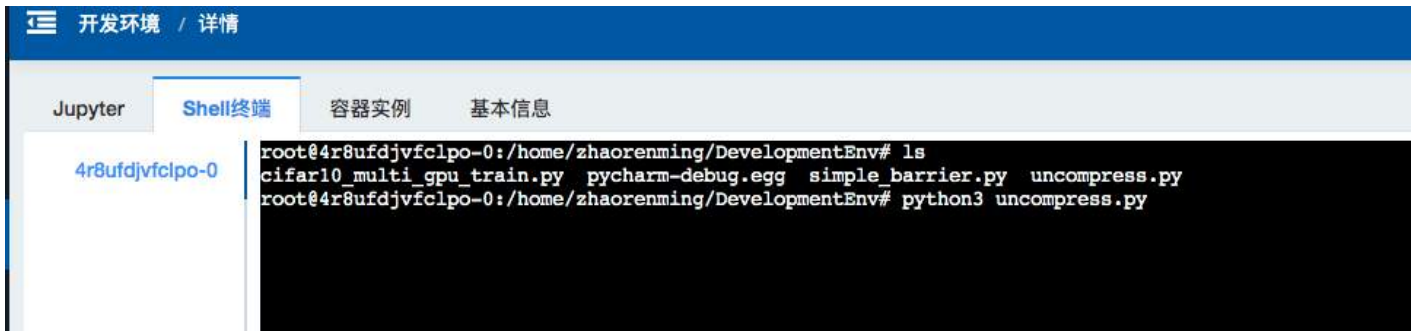
点击红框所示的下拉菜单，选择刚刚创建的 DevelopmentDbg。之后点击绿色的 debug 图标按钮。



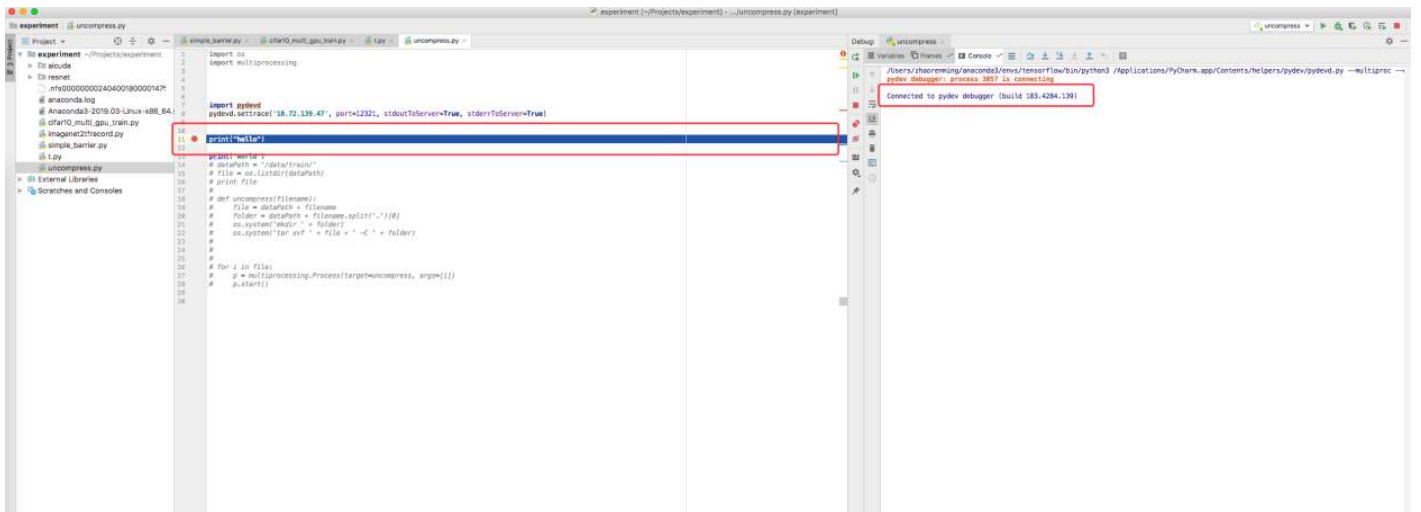
当控制台出现” Waiting for process connection...Connected to pydev debugger” 时，进入到开发环境进行操作。



进入开发环境，运行 `python3 unzip.py`。



此时 pycharm 中显示如下，程序在第一个断点处停止，可以开始远程调试。



VSCode 对接 AIStation 开发环境

若要在 VSCode 中远程编辑调试开发环境中的文件，可利用 VSCode 的两个插件实现：Remote-ssh 或者 Remote VSCode。

下面示例所用 VSCode 版本：1.53.2

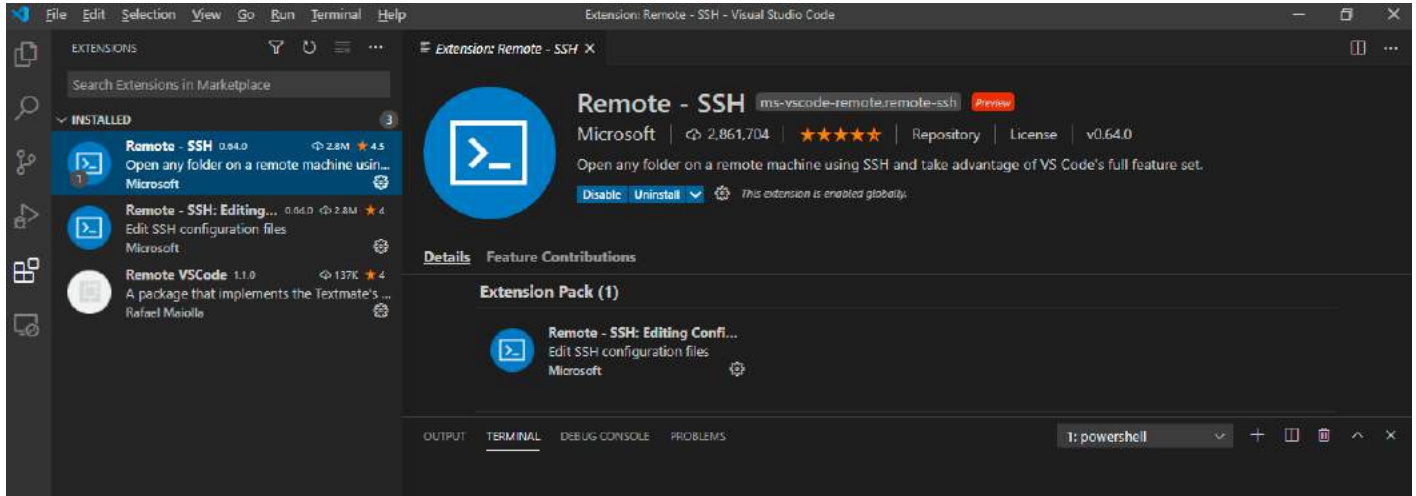
Windows 系统：win10（已自带安装 ssh）

开发环境 ssh 服务运行正常。

Remote-ssh 插件

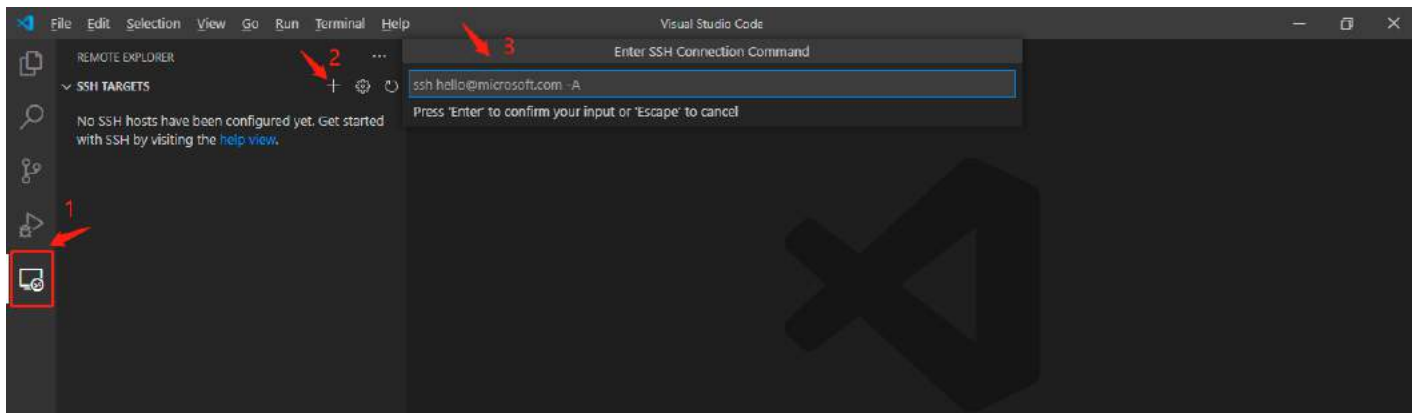
1. 下载安装插件

如下图所示：



2. 点击 Remote Explorer, 添加远程环境

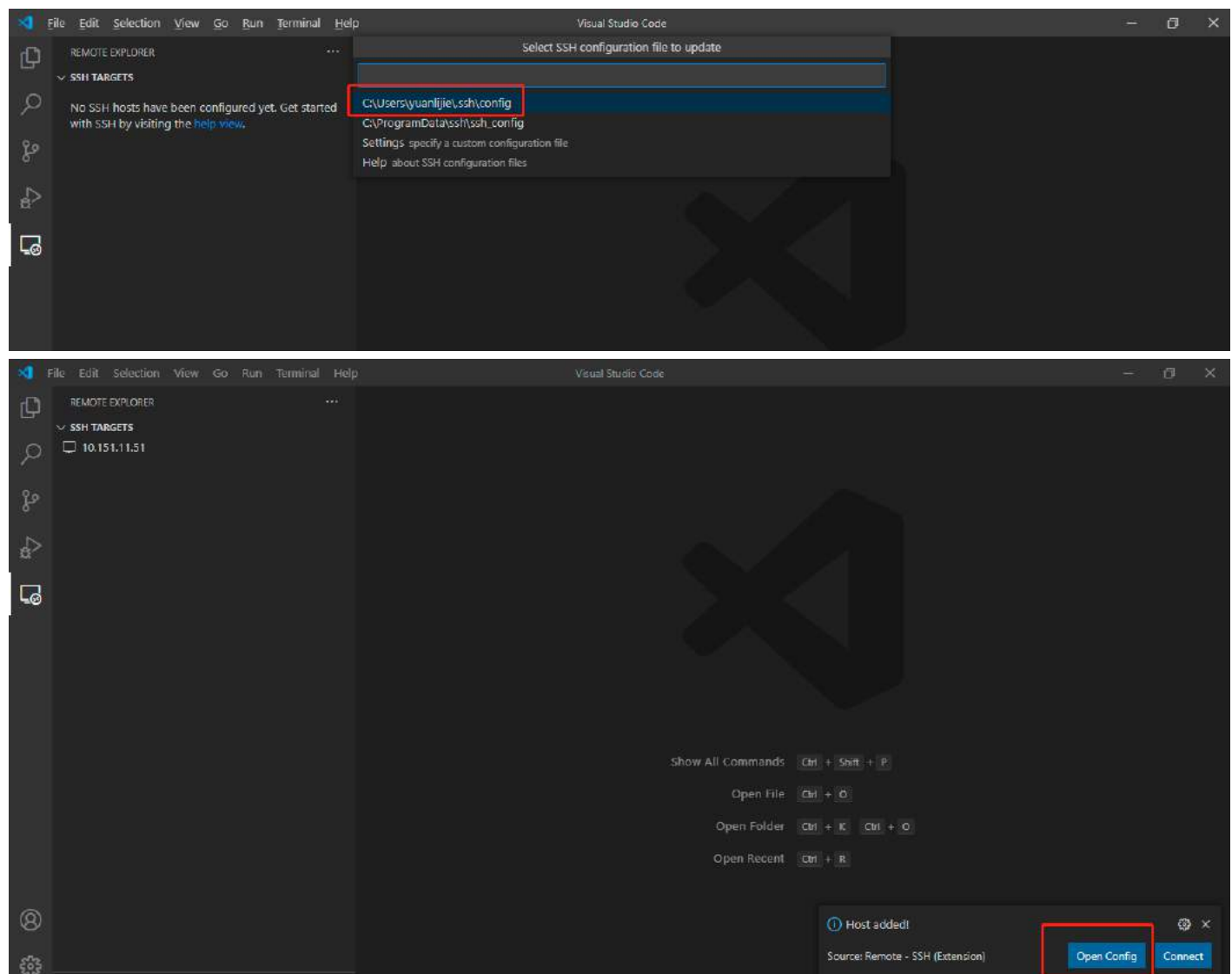
如下图所示：



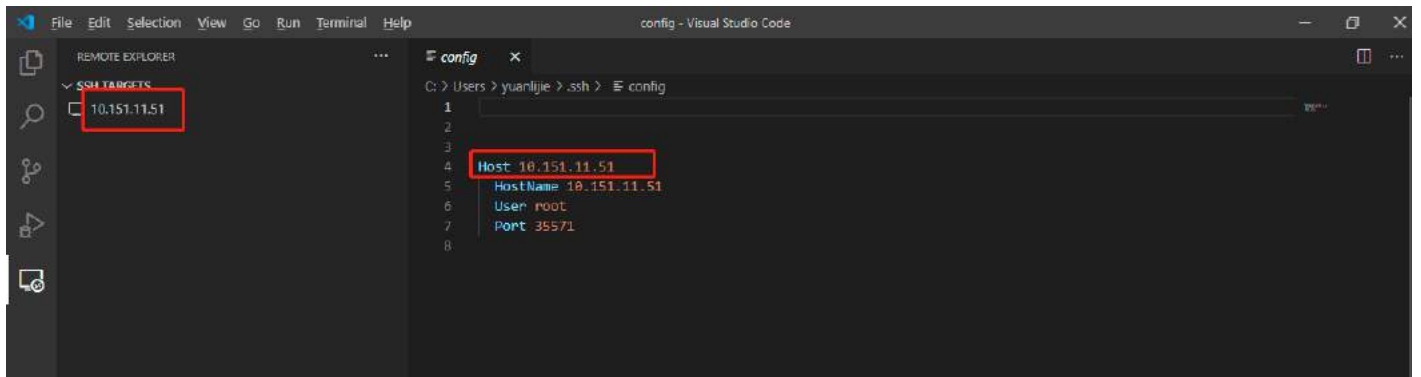
将开发环境 ssh 连接命令复制到上图所示“3”所处位置；



确认输入正确，输入“Enter”之后，如下图所示，一般选择个人用户目录下的配置文件：

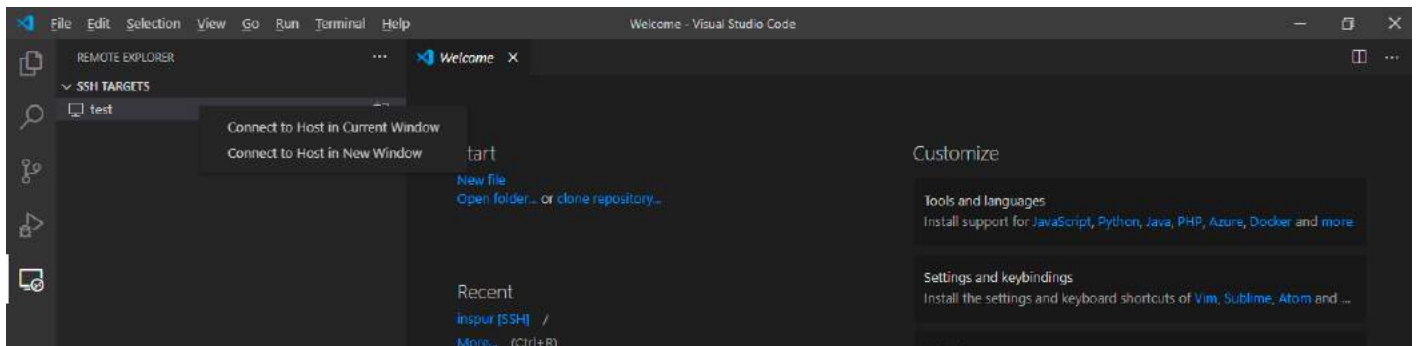


建议先打开配置文件，修改此环境名称，保存配置文件并刷新。

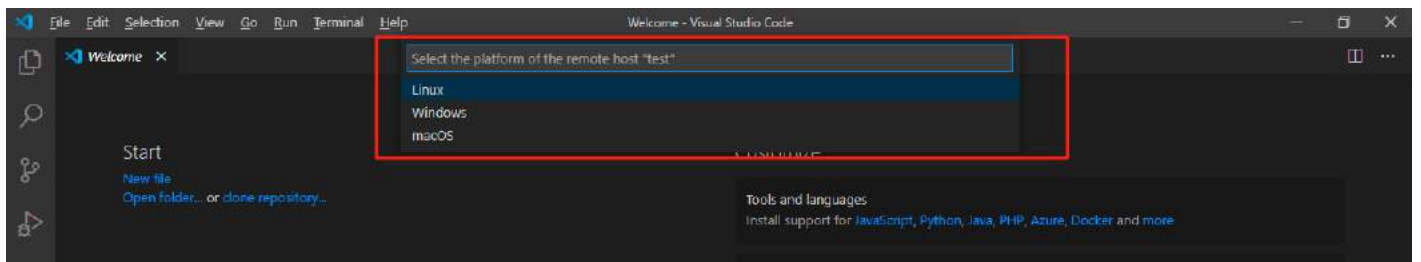


3. 连接到远程环境

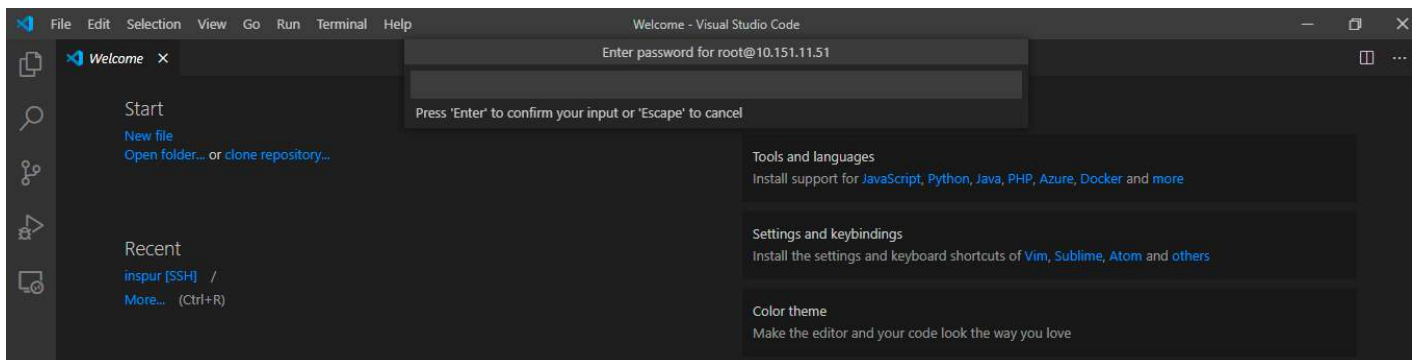
点击远程环境名称，连接远程环境：



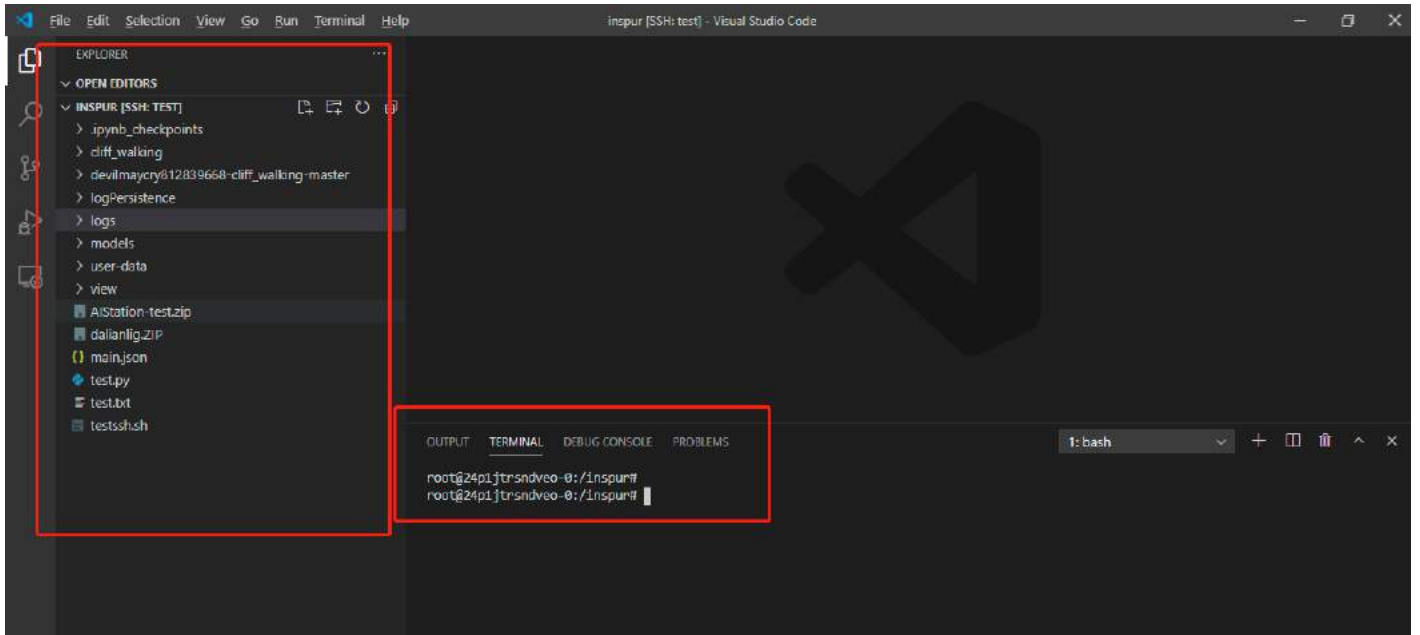
选择远程环境的系统类型：



输入远程环境的密码，等待远程环境自动安装相应依赖：

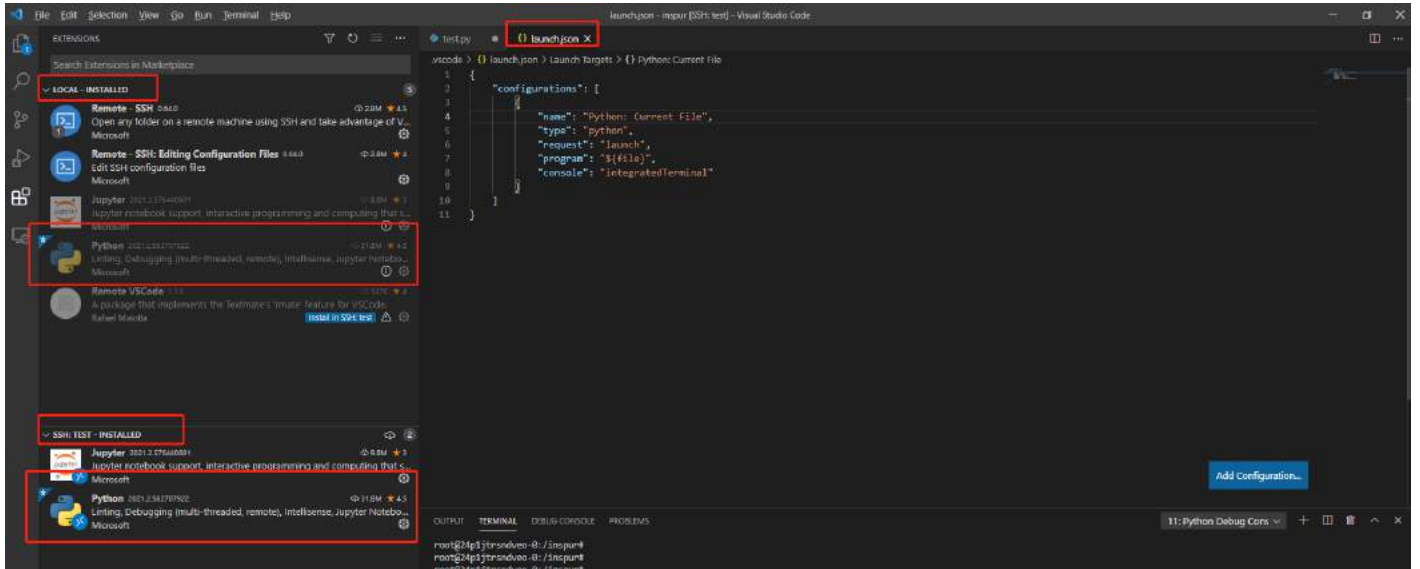


依赖安装完成后，远程环境连接成功：



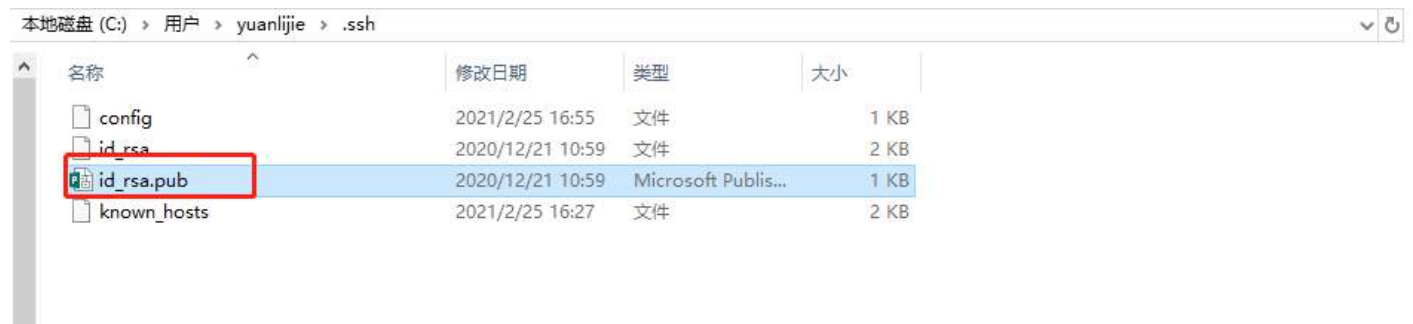
4. 远程调试代码

下面以运行 python 文件为例，其他语言请自行查阅资料。本地和开发环境都需要安装相应的插件并完成相应配置：



5. 免密配置

为解决每次输入密码的麻烦，可将本地电脑个人用户目录.ssh 文件下公钥放到远程环境中。若本地服务器.ssh 目录没有此文件，可用 ssh-keygen 命令生成。

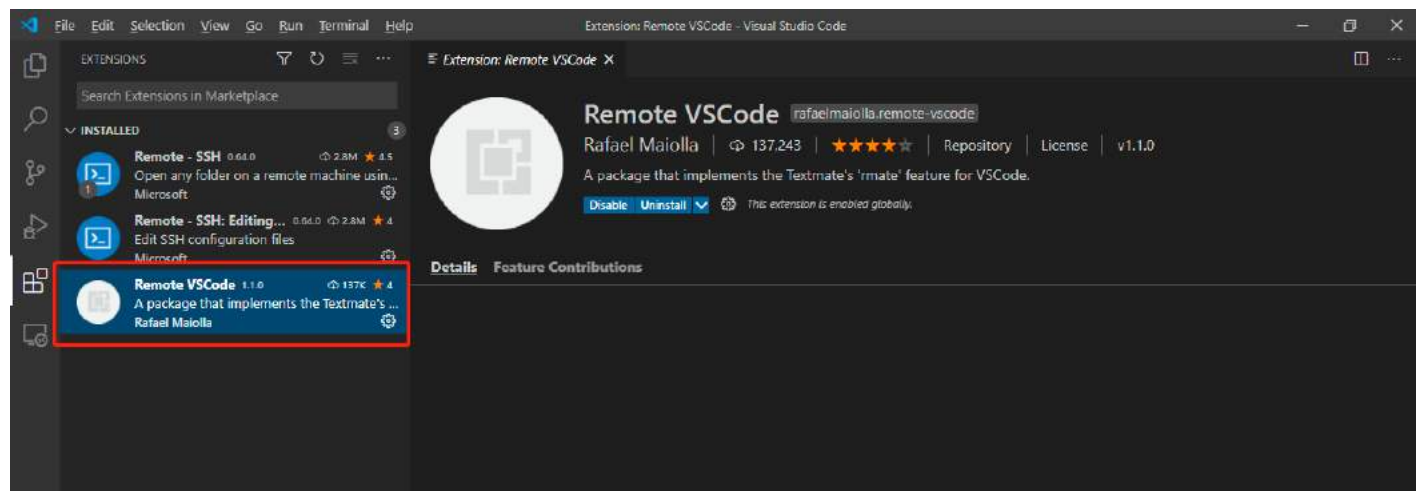


将 id_rsa.pub 复制到开发环境中，在开发环境中执行 `cat id_rsa.pub >> /root/.ssh/authorized_keys` 即可。

Rmate 插件

1. 下载安装插件

如下图所示：



开发环境执行下面命令下载安装插件：

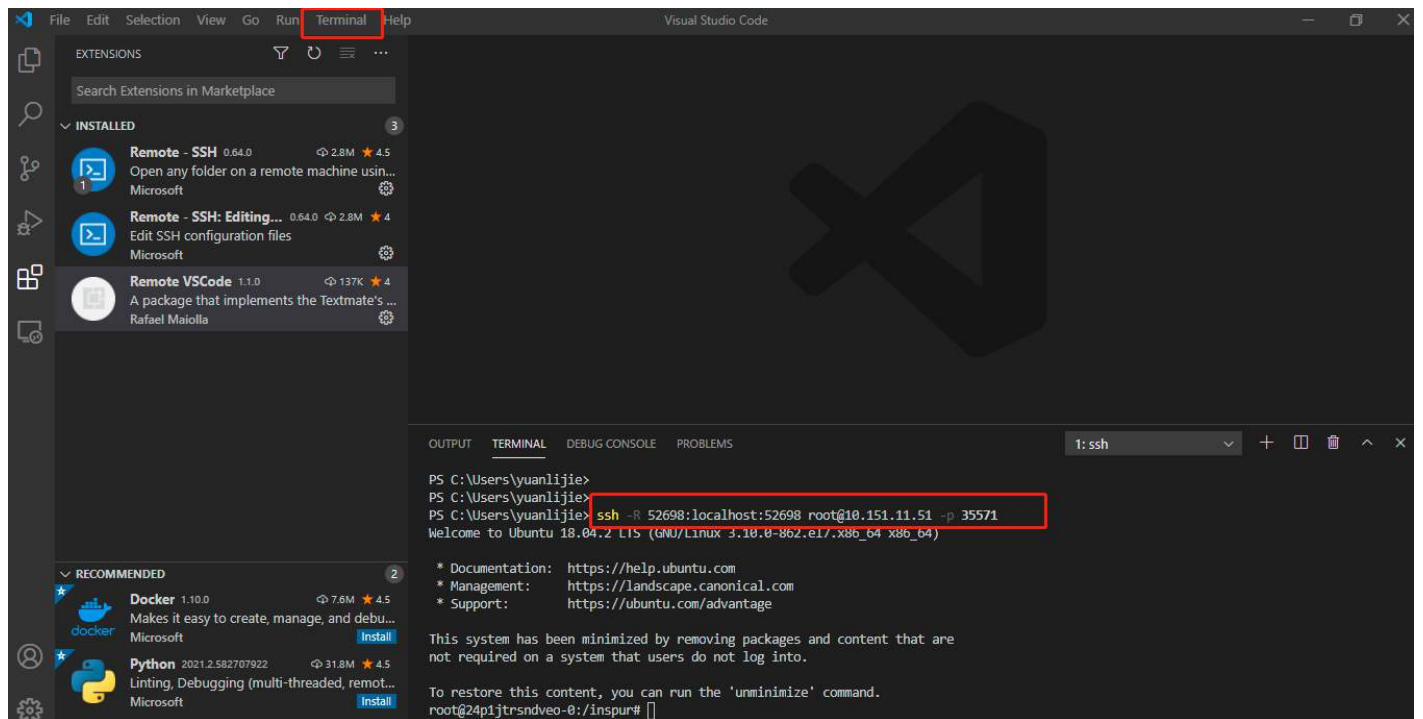
```
wget -O /usr/local/bin/rmate https://raw.githubusercontent.com/aurora/rmate/master/rmate
```

```
chmod a+x /usr/local/bin/rmate
```

2. 连接到远程环境

新建一个终端，输入如下命令：

```
ssh -R 52698:localhost:52698 username@x.x.x.x -p xxxx 其中，-p 后面为容器的端口号
```



3. 打开文件

在终端执行 `mate xxx.py` 就可以在本本地编辑修改远程环境中的代码。

预置算例-开发环境使用

注意：

1. 创建任意框架的开发环境，进入 shell 终端后，默认目录为当前用户目录；平台自带的算例存放在/defaultShare/user-data 目录下，可以通过文件管理中复制功能复制到用户目录。



以下以 `inspur` 用户为例，所用的资源组是通用标签默认资源组（可根据实际情况选择）；

在 `/inspur/models` 目录下，包含所有框架的训练脚本信息；

可以通过 `README` 获取如何运行作业脚本和注意事项；

2. 可以通过 `aistation_DL_train.sh` 脚本提交相应框架的作业脚本；

如提交 caffe 单机单卡作业：./aistation_DL_train.sh caffe mnist 1

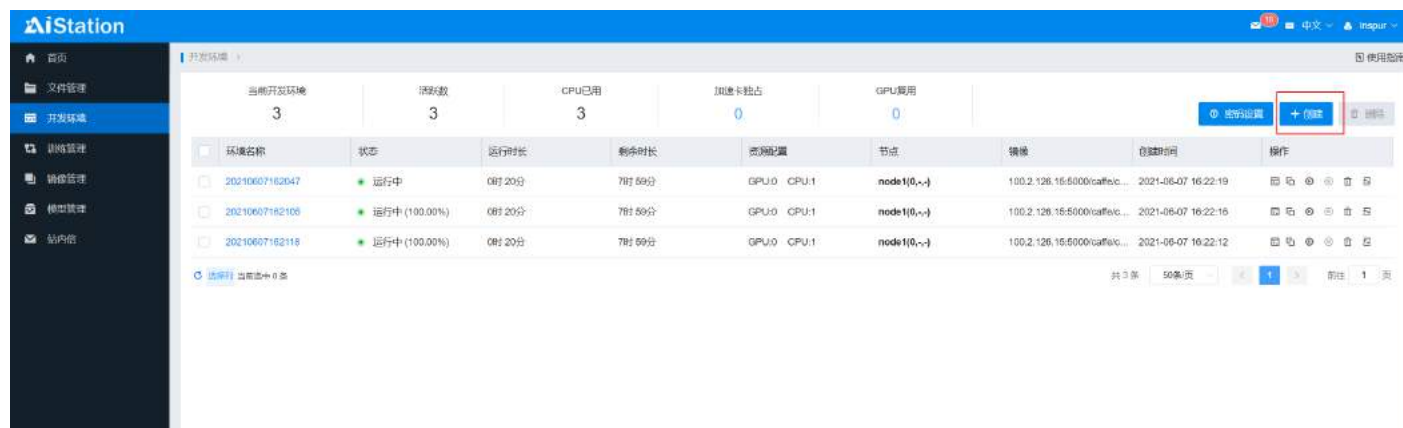
如提交 caffe 单机多卡作业：./aistation_DL_train.sh caffe mnist 2 （创建开发环境时，需要选择多加速卡资源）

3. 可以到具体框架脚本目录下，通过运行命令提交作业脚本（下文详细说明各框架单卡运行作业脚本）
由于所有作业脚本中路径以 inspur 为默认用户；如果是其他用户，首先，通过运行./user_path_change.sh user_name 修改作业脚本中的路径信息。

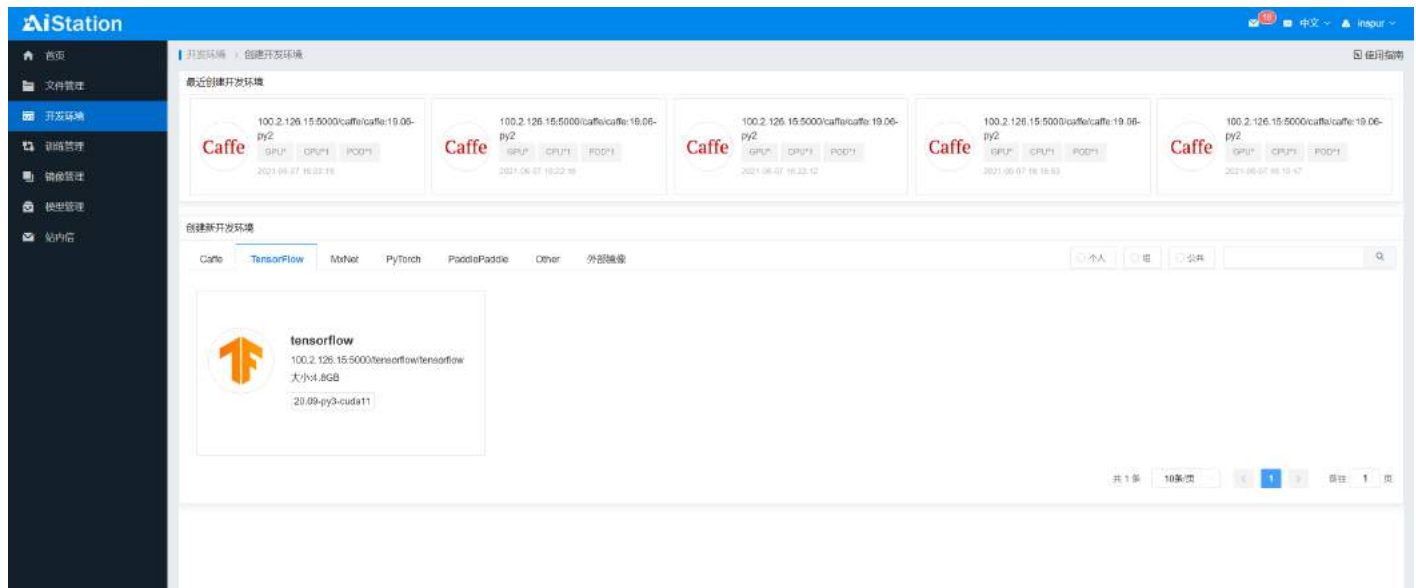
创建开发环境

创建开发环境步骤如下：

1. 点击右上角创建按钮

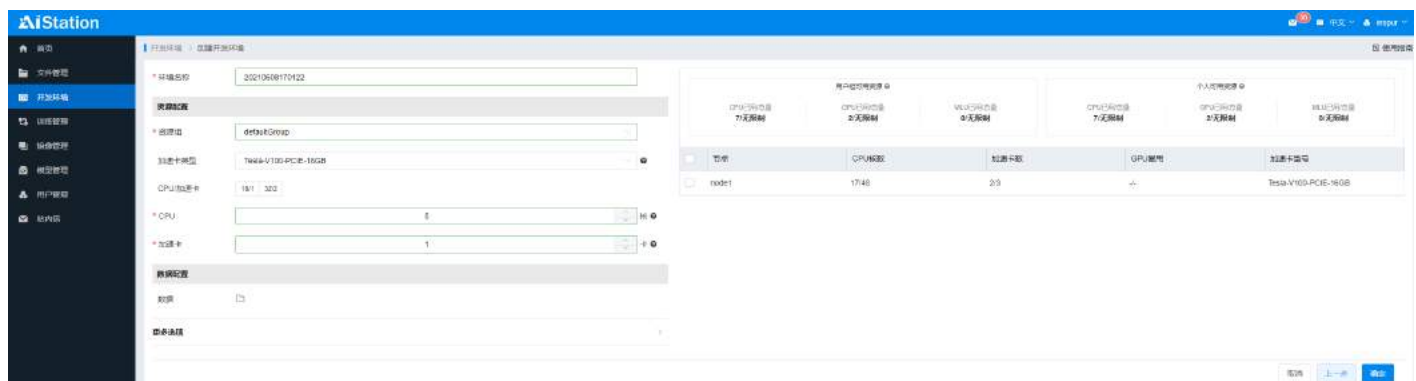


2. 选择相应框架的镜像



注意：创建开发环境，也可以选择外部相应镜像，但需要保证拉取成功。

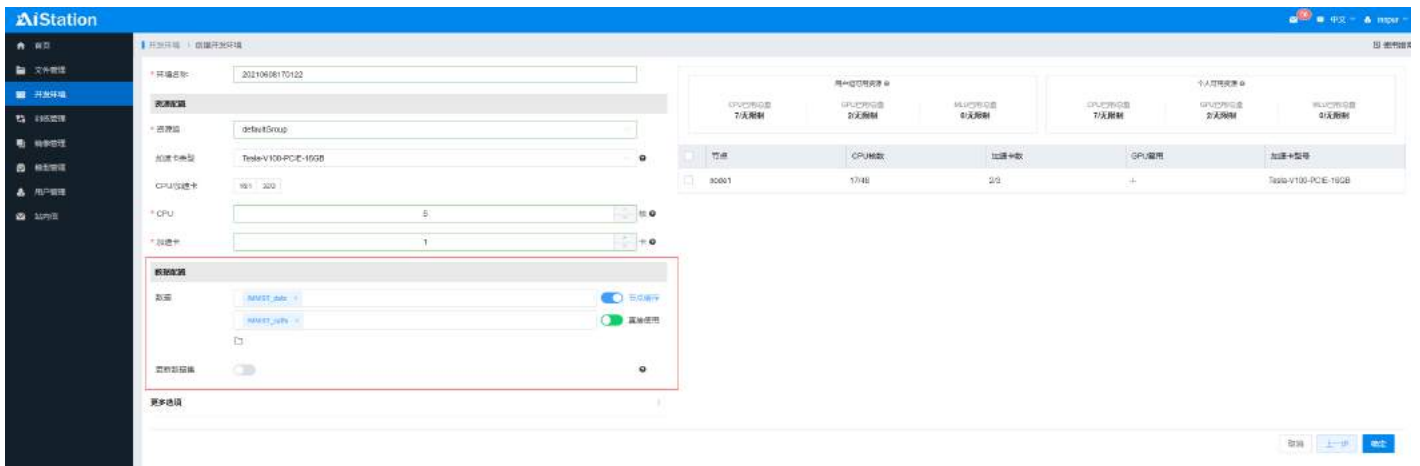
3. 开发环境参数说明



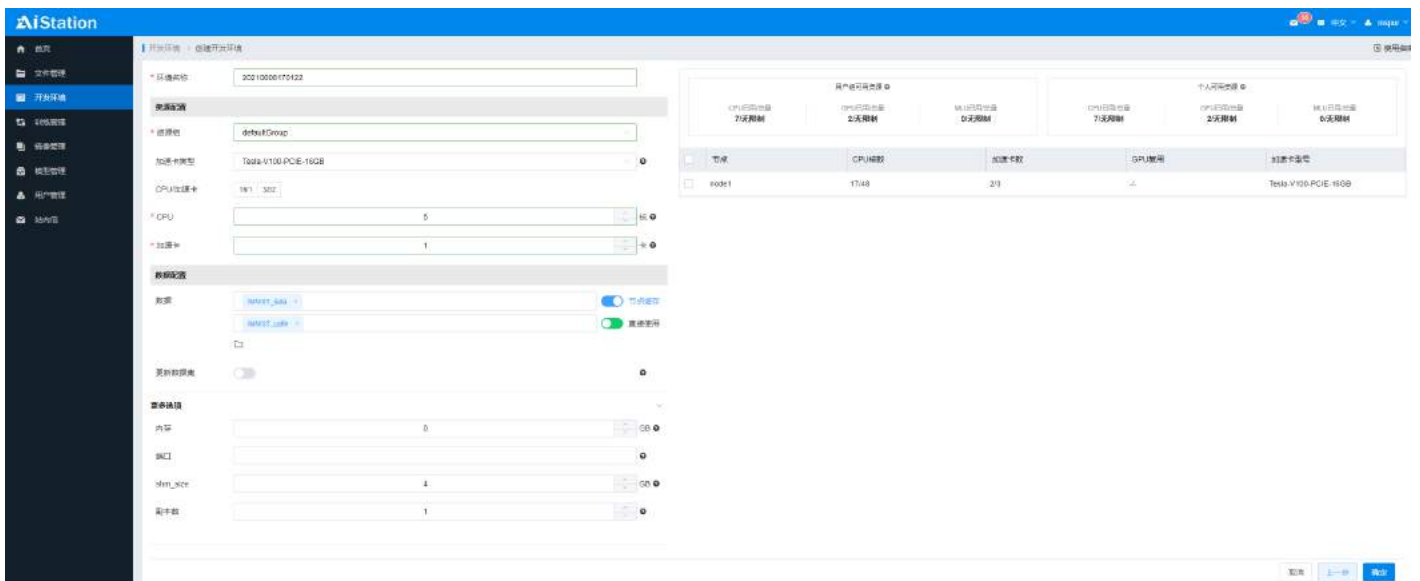
- 创建开发环境时，自动根据当前的日期和时间进行开发环境的名称的命名。用户可以修改开发环境名称。
- 用户可以选择开发环境要创建到的资源组。默认会选择到默认资源组。
- 右侧显示当前用户所属用户组和个人可用资源以及当前所选资源组下各节点可用资源情况，若选择具体的节点，开发环境将创建在选择的节点上，否则将由系统内部调度机制选择节点。
- 加速卡类型选项列出所选择资源组中所包含的加速卡类型。选择加速卡的类型之后，开发环境会

调度到使用对应加速卡的物理机。

- 用户可以配置开发环境所使用的加速卡个数和 CPU 的核数。
- 点击数据集路径右侧的文件夹图标，可以选择要挂载的数据集路径。在弹出的路径选择窗口中选择要挂载的数据集，可以选择多个需要的数据集目录（如下图）。在创建开发环境时，会将选择的路径挂载到环境中。根据是否更新数据集的开关，可以选择是否会在本地已有缓存数据集的情况下，将数据集更新的部分缓存至本地。使用方式为节点缓存是指将先将数据集缓存至本地，然后挂载到环境中；直接使用是指此数据集直接挂载到开发环境中使用。



点击更多选项，可以选择开发环境的更多设置。如下图：



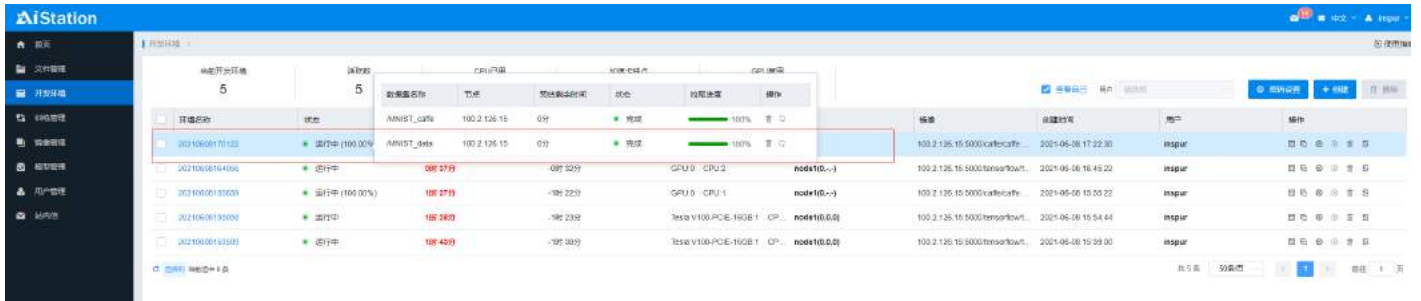
内存：大小默认为 0，表明不限制内存。用户可以自行设置。

端口：用户可自行设置所用的端口号。

shm_size: share memory size。可以自定义 shm_size 大小（需系统管理员开发权限），默认为 4GB。

副本数：一次创建多个开发环境。

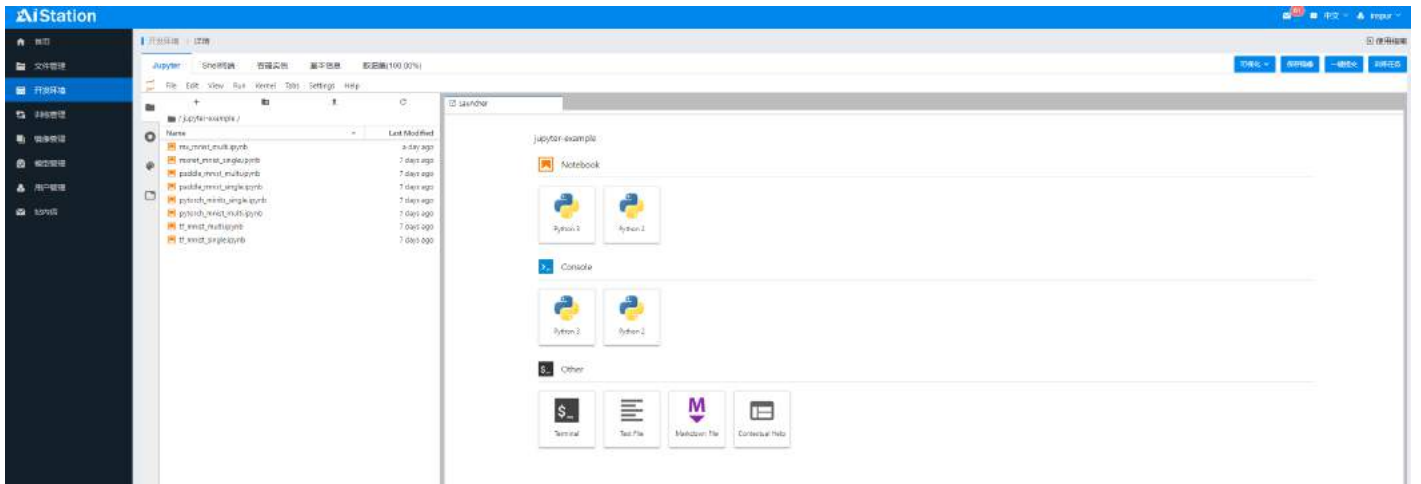
在开发环境创建过程中，数据集在同步下载，会显示下载进度。



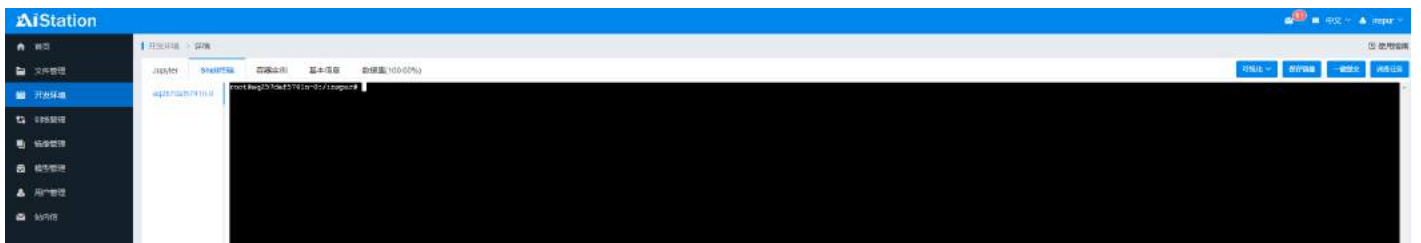
等开发环境创建完成后，点击开发环境名称，进入开发环境页面。

4. 开发环境五个标签页：

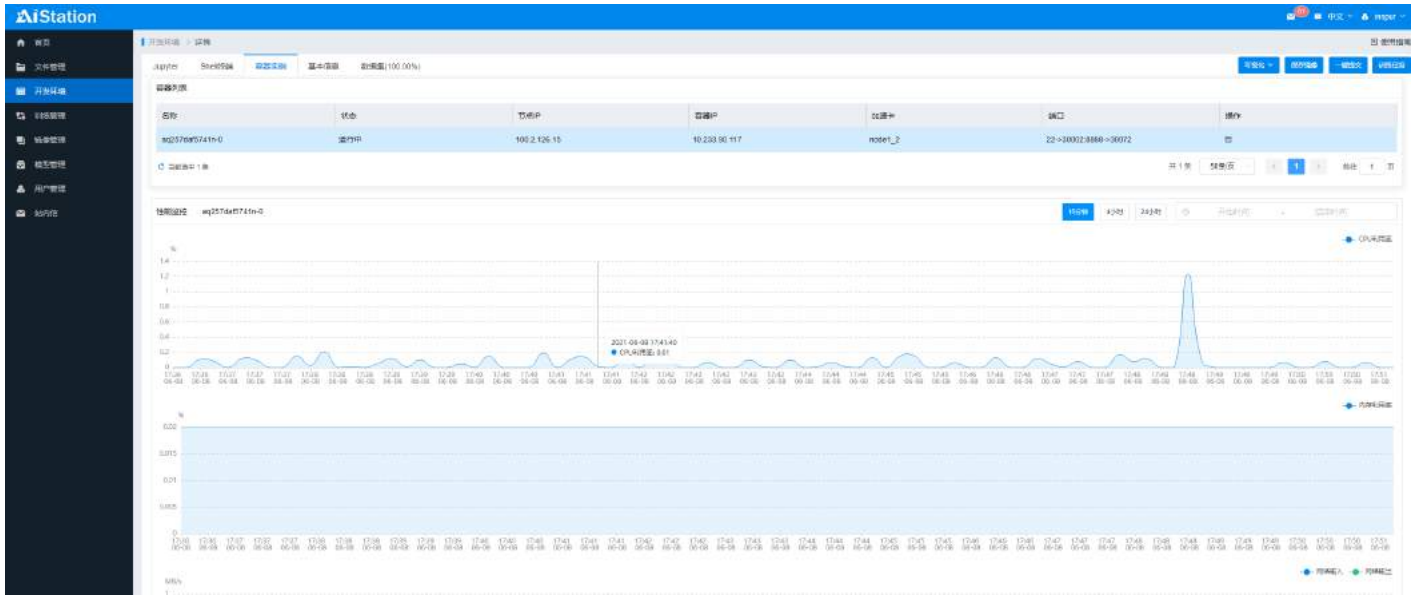
1). Jupyter: 用户可通过 Jupyter 工具进行代码的编写、调试等工作。



2). Shell 终端：用户进到所创建的开发环境终端中，进行模型的开发与调试。



3). 容器实例：展示开发环境具体的容器信息以及其资源监控信息（CPU、加速卡、内存、磁盘和网络）。



4). 基本信息：展示开发环境的名称、创建时间、所用镜像等基本信息。



5). 数据集：如果数据集过大，等开发创建完成之后，也可点击数据集页面查看数据集下载详情。



开发环境模型调试

参考上述创建开发环境的步骤，选择对应框架的镜像，例如 Caffe 的镜像在 Caffe 标签页下；按需设置相应的参数，其中数据集路径需要选择训练脚本对应的数据集，其他参数可按需设置。

Caffe 框架

镜像：Caffe 的镜像在 Caffe 标签页下；

数据集路径：MNIST_caffe；

进入 Shell 终端，切换到 caffe/mnist 目录：

```
cd /inspur/models/caffe/mnist
```

启动命令:

```
caffe train -solver=solver_lenet.prototxt -gpu all
```

注意: Caffe 框架不支持开启 GPU 共享的资源组。

Tensorflow 框架

镜像: TensorFlow 的镜像在 TensorFlow 标签页下;

数据集路径: MNIST_data;

进入 Shell 终端, 切换到 tensorflow/mnist 目录:

```
cd /inspur/models/tensorflow/mnist
```

启动命令:

```
python tf_mnist_single.py
```

Mxnet 框架

镜像: MxNet 的镜像在 MxNet 标签页下;

数据集路径: MNIST_data;

进入 Shell 终端, 切换到 mxnet/mnist 目录:

```
cd /inspur/models/mxnet
```

启动命令:

```
python mx_mnist_single.py
```

Pytorch 框架

镜像: Pytorch 的镜像在 Pytorch 标签页下;

数据集路径: MNIST_pytorch;

进入 Shell 终端, 切换到进入 pytorch/mnist 目录:

```
cd /inspur/models/pytorch/mnist
```

启动命令:

```
python pytorch_mnist_single.py
```

PaddlePaddle 框架

镜像：PaddlePaddle 的镜像在 PaddlePaddle 标签页下：

数据集路径：MNIST_data；

进入 Shell 终端，切换到 paddle 目录：

```
cd /inspur/models/paddle
```

启动命令：

```
python paddle_mnist_single.py
```

Other 框架

通过相应的镜像创建环境，脚本及数据集需要与镜像一致。

远程连接开发环境

如下图所示，查看当前开发环境的连接命令。打开一个 Shell 软件，复制命令就可以在 Shell 中登录本开发环境。



如果是本地 xshell 直接登录的前提下需要把复制中的 -p 参数去掉，比如 ssh root@100.2.126.15 30100 如果是本地 xshell 通过已有的系统中登录，直接复制就可以连接，比如 ssh root@100.2.126.15 -p 30100 其中密码用户可以按需设置成随机或者自定义密码。

加速卡类型：选择资源组内相应的加速卡类型。

CPU/加速卡：选择 worker 节点的 CPU/加速卡资源配置方案，当配额方案是“自定义”时，会弹出加速卡和 CPU 窗口，可以自定义设置资源配置方案。

py 脚本：点击窗口后第一个按钮，弹出“选择启动文件”窗口，通过历史访问或者个人数据选择相应训练脚本；或者点击命令模式使用命令行模式。脚本参数：在“脚本参数”输入框可以输入 python 脚本所需要的参数。

执行目录：当选择脚本模式时，根据脚本需要，请指定相应的执行目录（一般为脚本所在目录）。

数据集：点击窗口后按钮可以弹出“选择数据集窗口”，选定后点击确定。勾选下方“更新数据集”选项可以在进行训练前更新所选定的数据集。

更新数据集说明：勾选后，平台自动会对缓存的数据集进行识别，如果部分数据集文件发生变化，平台会实现增量更新；如果缓存中没有数据集，会全量下载数据集。如果缓存中的数据集正在使用，则不能进行更新操作。

当选择数据集之后，使用方式为节点缓存是指先将数据集缓存至本地，然后在训练任务环境中挂载使用；直接使用是指此数据集目录直接挂载到训练任务环境中使用。

点击“更多配置”可以显示以下信息选项：

内存：配置训练任务 worker 节点所需要的内存，当设置为 0 时表示无限制（需要小于 worker 所在主机目前，剩余内存量）。

日志路径：可视化输出路径，点击窗口后的按钮，选择相应路径后点击确定。

目录挂载：如果设置了共享目录，可以选择要挂载的公共文件夹。

shm_size：可以自定义 shm_size 大小（需系统管理员开发权限），默认为 4GB。

部署类型：根据所用框架及训练任务，可选择不同的部署类型（单机、分布式、MPI）。

Worker 个数：根据不同的部署类型，显示或者设置不同的 Worker 个数。

3. 创建任务成功

任务状态进入“排队中”；当资源分配完成后，任务状态进入“运行中”，单击任务名称，可查看训练任务的输出日志；训练完成后任务自动进入“完成任务”内，训练成功时任务状态为“完成”，训练失败时任务状态为“失败”。

注意：以下以 inspur 用户为例，在/{用户名}/models 目录下，包含所有框架的训练脚本信息；

单机任务创建

Caffe 单机任务

1. 点击创建按钮，开始创建 Caffe 单机任务。

2. 填写任务信息

-镜像：选择 Caffe 镜像；

-数据集：/MNIST_caffe；

-启动文件：Caffe 单机训练脚本：/inspur/models/caffe/mnist/solver_lenet.prototxt

或者使用命令行模式：

```
caffe train -solver=/inspur/models/caffe/mnist/solver_lenet.prototxt -gpu=all
```

其他参数按需设置。

Tensorflow 单机任务

1. 点击创建按钮，开始创建 Tensorflow 单机任务。

2. 填写任务信息

-镜像：选择 Tensorflow 镜像；

-数据集：/MNIST_data；

-启动文件：Tensorflow 单机训练脚本，/inspur/models/tensorflow/mnist/tf_mnist_single.py

或者使用命令行模式：

```
python /inspur/models/tensorflow/mnist/tf_mnist_single.py
```

其他参数按需选择。

Mxnet 单机任务

1. 点击创建按钮，开始创建 Mxnet 单机任务。

2. 填写任务信息

-镜像：选择 Mxnet 镜像；

-数据集：/MNIST_data；

-启动文件：Mxnet 单机训练脚本，/inspur/models/mxnet/mx_mnist_single.py

或者使用命令行模式：

```
python /inspur/models/mxnet/mx_mnist_single.py
```

其他参数按需选择。

Pytorch 单机任务

1. 点击创建按钮，开始创建 Pytorch 单机任务。

2. 填写任务信息

-镜像：选择 Pytorch 镜像；

-数据集：/MNIST_pytorch；

-启动文件：Pytorch 单机训练脚本，/inspur/models/pytorch/mnist/pytorch_mnist_single.py

或者使用命令行模式：

```
python /inspur/models/pytorch/mnist/pytorch_mnist_single.py
```

其他参数按需选择。

PaddlePaddle 单机任务

1. 点击创建按钮，开始创建 PaddlePaddle 单机任务。

2. 填写任务信息

-镜像：选择 PaddlePaddle 镜像；

-数据集：/MNIST_data；

启动文件：Paddlepaddle 单机训练脚本，/inspur/models/paddle/paddle_mnist_single.py

或者使用命令行模式：

```
python /inspur/models/paddle/paddle_mnist_single.py
```

其他参数按需选择。

分布式任务创建

Tensorflow 分布式任务

1. 点击创建按钮，开始创建 Tensorflow 分布式任务。

2. 填写任务信息

-镜像：选择 Tensorflow 镜像；

-数据集：/MNIST_data；

-启动文件：Tensorflow 分布式训练脚本，/inspur/models/tensorflow/mnist/tf_mnist_dist.py

或者 Benchmark 分布式测试脚本：/inspur/models/tensorflow/benchmarks-cnn_tf_v1.15_compatible/scripts/

tf_cnn_benchmarks/benchmark_cnn_distributed_test_runner.py;

-部署类型：PS/Worker；PS 个数：1；Worker 个数：2；

其他参数按需设置。

Mxnet 分布式任务

1. 点击创建按钮，开始创建 Mxnet 分布式任务。

2. 填写任务信息

-镜像：选择 Mxnet 镜像；

-数据集：/MNIST_data；

-启动文件：Mxnet 分布式训练脚本，/inspur/models/mxnet/mx_mnist_dist/mx_mnist_dist.py（注意：若使用 GPU，脚本中 84 行，gpus 需修改为 gpus: '0'）；

-部署类型：Server/Worker；Server 个数：1；Worker 个数：2；

其他参数按需设置。

Pytorch 分布式任务

1. 点击创建按钮，开始创建 Pytorch 分布式任务。

2. 填写任务信息

-镜像：选择 Pytorch 镜像；

-数据集：/MNIST_pytorch；

-启动文件：Pytorch 分布式训练脚本，/inspur/models/pytorch/mnist/pytorch_mnist_dist.py

-部署类型：Master/Worker；Master 个数：1，Worker 个数：2；

其他参数按需设置。

MPI 任务创建

Caffe MPI 任务

1. 点击创建按钮，开始创建 Caffe MPI 任务。

2. 填写任务信息

-镜像：选择 Caffe 镜像；

-数据集：/MNIST_caffe；

-部署类型：MPI；Worker 个数：2；

-启动文件：Caffe MPI 训练脚本， /inspur/models/caffe/mnist/solver_lenet.prototxt

或者使用命令行模式：

```
mpirun -allow-run-as-root -np 2 caffe train -solver=/inspur/models/caffe/mnist/solver_lenet.prototxt -gpu all
```

其他参数按需选择。

Tensorflow MPI 任务

1. 点击创建按钮，开始创建 Tensorflow MPI 任务。

2. 填写任务信息

-镜像：选择 Tensorflow 镜像；

-数据集： /MNIST_data；

-部署类型： MPI； Worker 个数： 2；

-此镜像只能使用命令行模式：

```
mpirun -oversubscribe -allow-run-as-root -np 2 -mca pml ob1 python /inspur/models/horovod/tensorflow_mnist.py -data_dir=/MNIST_data
```

其他参数按需设置。

Mxnet MPI 任务

1. 点击创建按钮，开始创建 Mxnet MPI 任务。

2. 填写任务信息

-镜像：选择 Mxnet 镜像；

-数据集： /MNIST_data；

-此镜像只能使用命令行模式：

```
mpirun -oversubscribe -allow-run-as-root -np 2 -mca pml ob1 python /inspur/models/horovod/mxnet_mnist.py -data_dir=/MNIST_data
```

-部署类型： MPI； Worker 个数： 2；

其他参数按需设置。

Pytorch MPI 任务

1. 点击创建按钮，开始创建 Pytorch MPI 任务。

2. 填写任务信息

-镜像：选择 Pytorch 镜像；

-数据集：/MNIST_pytorch；

-部署类型：MPI； Worker 个数： 2；

-启动文件：Pytorch MPI 训练脚本， /inspur/models/horovod/pytorch_mnist.py

或者使用命令行模式：

```
mpirun -np 2 -allow-run-as-root python /inspur/models/horovod/pytorch_mnist.py
```