ModelArts

模型封装

文档版本 01

发布日期 2024-04-03

华为云计算技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 AI 应用开发介绍

模型封装是将模型封装成一个AI应用，也称之为AI应用开发。AI应用开发是指将训练好的一个或多个模型编排开发成推理应用以满足具体业务场景下的推理需求，比如视频质量检测、交通拥堵诊断等。AI应用开发在整个AI开发流程的位置大致如图1-1所示。

图 1-1 AI 开发流程

通常为了降低开发难度、提升AI应用的性能，开发者会基于深度学习推理框架开发AI应用，例如Google开源的MediaPipe、腾讯开源的TNN等。ModelArts提供了基于华为云ModelBox推理框架的开发环境，它具备如下优点：

- 提供开箱即用的云上AI应用开发环境，预置高性能推理框架ModelBox、加速卡推理加速卡、以及TensorRT、LibTorch、MindSpore等AI引擎。
- 提供Visual Studio Code（简称VSCode）插件，方便开发者在本地连接云端环境远程开发。
- 提供应用工程模板和样例，方便开发者快速了解AI应用开发，降低开发上手难度。
- 提供脚手架命令行工具和VSCode图形化命令，方便开发者进行AI应用开发、构建、本地调试以及真实业务调试。
- 和ModelArts推理对接，开发好的AI应用可以直接发布至ModelArts的AI应用管理模块。

AI 应用开发流程

AI应用开发的流程主要包括以下几个步骤：

1. **准备开发环境**

   使用华为云账号创建云上的AI应用开发环境，并使用本地VSCode插件进行远程连接。
2. **创建并运行第一个应用**

开发环境提供云上的ModelArts VSCode插件，可以通过插件创建一个应用工程，该工程包含一个可以直接运行的应用样例。开发者可以通过查看工程结构、运行样例，初步了解应用工程组成及运行方法。

3. **应用开发、调试**

   开发者可以参考当前的工程样例开发自己的AI应用，并在当前环境中进行代码调试、包括使用环境内的测试文件调试，以及接入云上真实业务数据的调试。当进行AI应用开发时，开发者需要学习ModelBox框架的相关概念和接口，以便基于框架开发出更高性能的AI应用。

4. **将应用发布到ModelArts模型管理**

   开发好的AI应用可以发布到ModelArts推理的AI应用管理，用户可以进一步在推理控制台界面上执行生产部署等操作。

2 准备开发环境

2.1 创建访问密钥

使用华为云账号在云上创建AI应用开发环境，需要提前创建访问密钥（AK/SK），用于对请求加密签名，确保请求的机密性、完整性和请求双方身份的正确性。

本节介绍如何通过ModelArts管理控制台创建访问密钥（AK/SK）。如果您已经获取过访问密钥，可跳过此章节。

获取访问密钥

1. 登录华为云，在页面右上方单击“控制台”，进入华为云管理控制台。

   图 2-1 控制台入口

2. 在控制台右上角的账户名下方，单击“我的凭证”，进入“我的凭证”页面。

   图 2-2 我的凭证

3. 在“我的凭证”页面，选择“访问密钥>新增访问密钥”，如图2-3所示。

图 2-3 单击新增访问密钥

4. 填写该密钥的描述说明，单击“确定”。根据提示单击“立即下载”，下载密钥。

图 2-4 新增访问密钥

5. 密钥文件会直接保存到浏览器默认的下载文件夹中。打开名称为“credentials.csv”的文件，即可查看访问密钥（Access Key Id和Secret Access Key）。

2.2 安装 VS Code 软件及插件

本节介绍如何在本地安装Microsoft Visual Studio Code（简称VS Code）软件及ModelArts提供的VSCode插件工具，协助用户完成SSH远程连接Notebook。

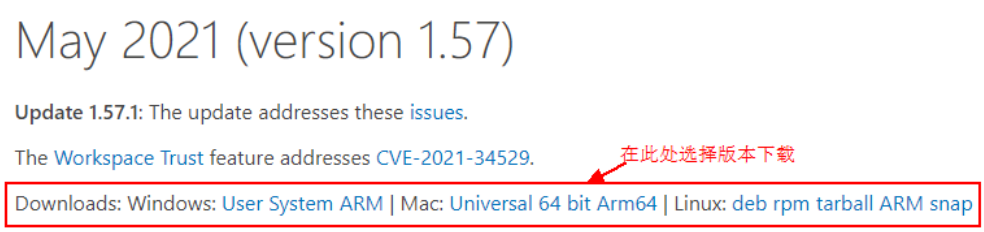
Step1 安装 VS Code

1. 下载并安装VS Code。

   Windows用户直接单击此处下载：**https://update.code.visualstudio.com/1.57.1/win32-x64-user/stable**。

   其他系统用户的下载地址: **https://code.visualstudio.com/updates/v1_57**。

图 2-5 VS Code 的下载位置



VS Code版本要求：

建议用户使用VS Code 1.57.1版本或者最新版本进行远程连接。

VS Code安装指导如下：

图 2-6 Windows 系统下 VS Code 安装指导

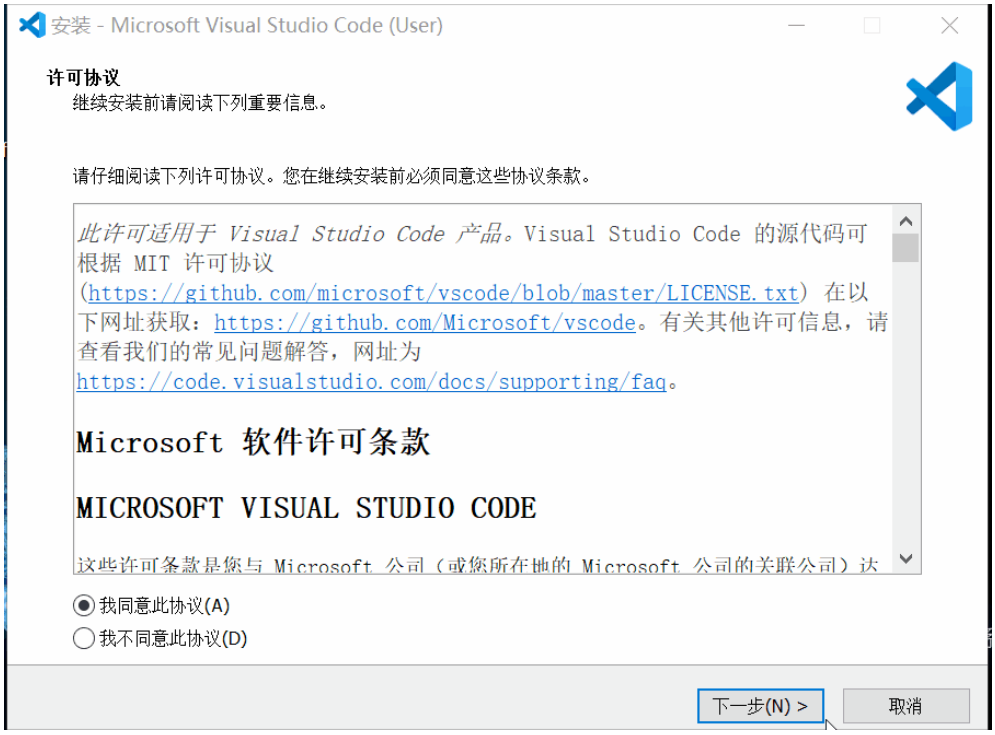
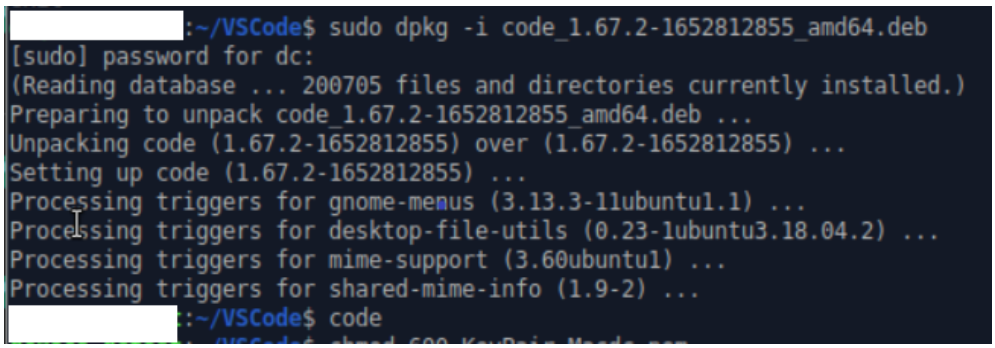


图 2-7 Linux 系统下 VS Code 安装指导



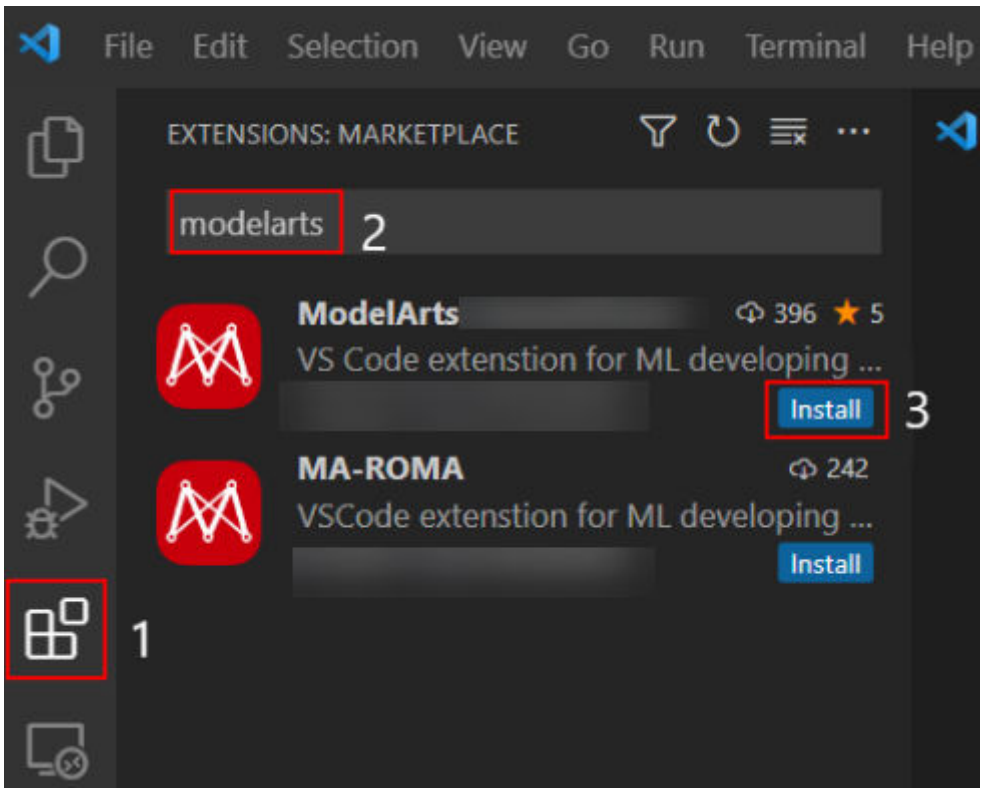
说明

Linux系统用户，需要在非root用户进行VS Code安装。

Step2 安装 VS Code 插件

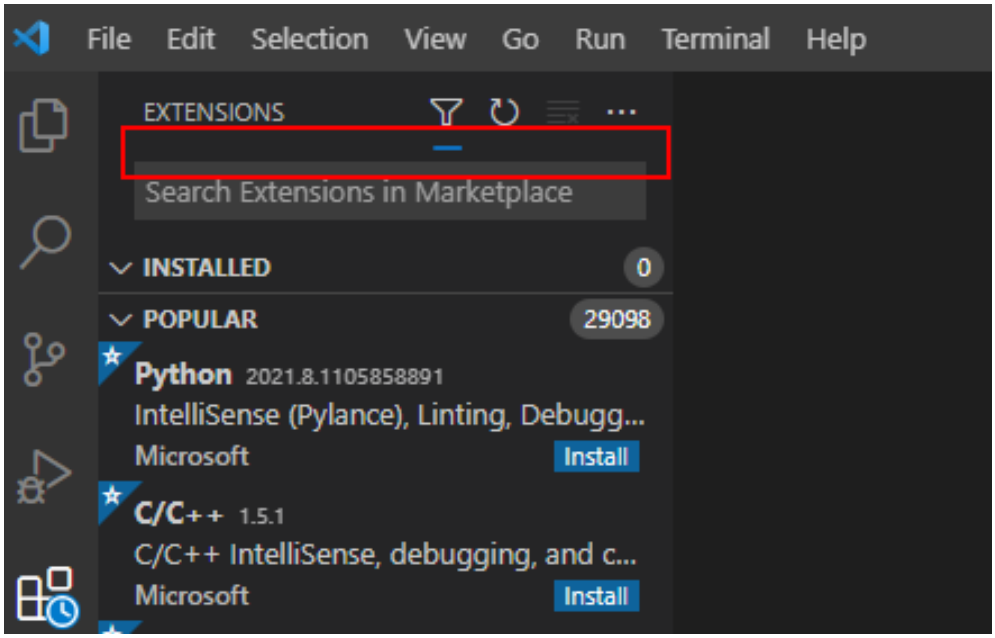
1. 在本地的VS Code开发环境中，如图2-8所示，在VS Code扩展中搜索“ModelArts-HuaweiCloud”并单击“安装”。

图 2-8 安装 VS Code 插件



2. 安装过程预计1~2分钟，如图2-9所示，请耐心等待。

图 2-9 安装过程



3. 安装完成后，系统右下角提示安装完成，导航左侧出现ModelArts图标和SSH远程连接图标，表示VS Code插件安装完成。

图 2-10 安装完成提示

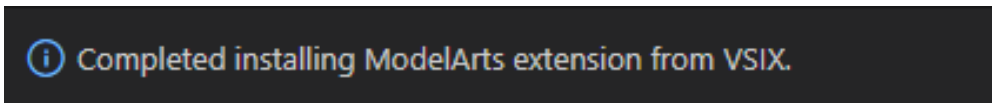
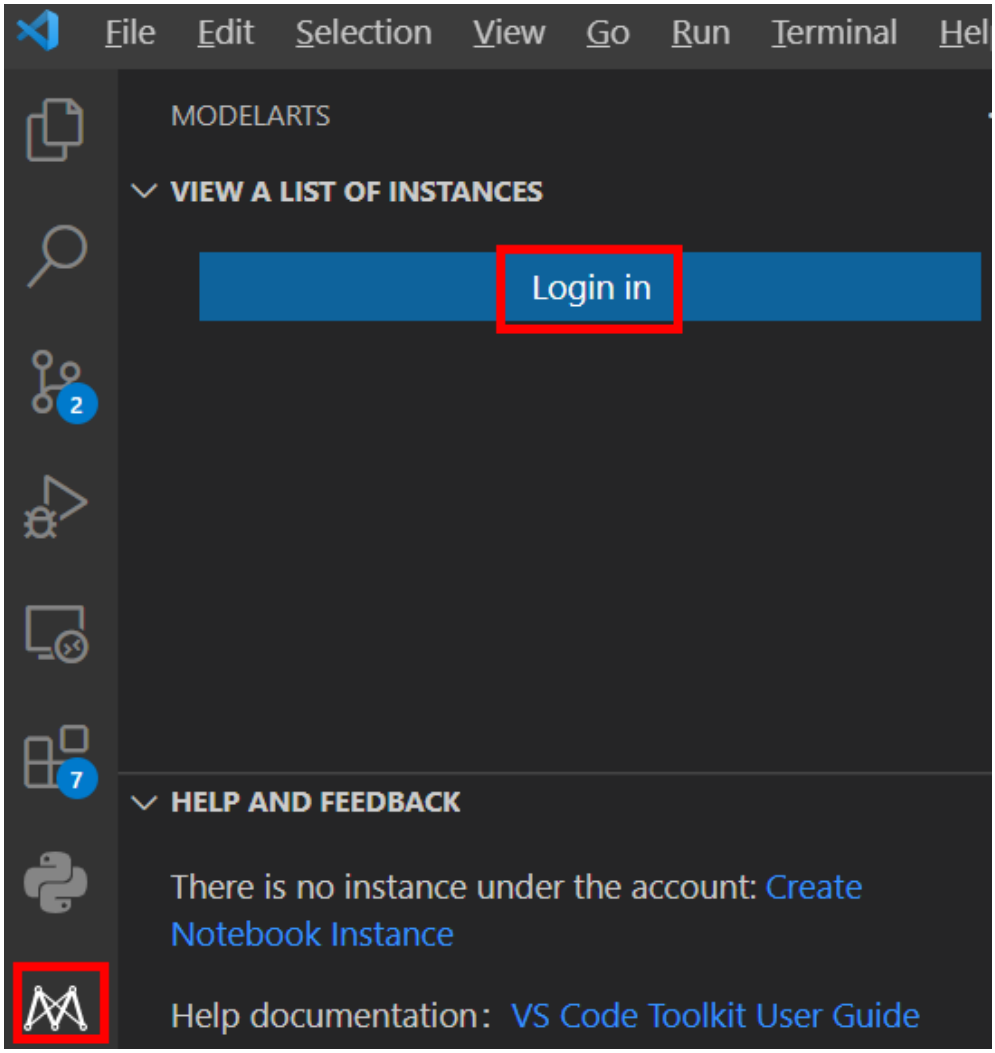
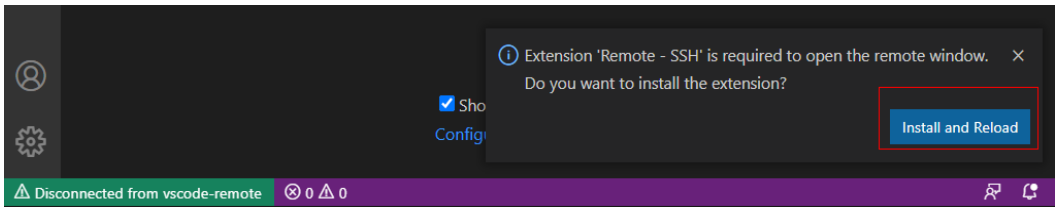


图 2-11 安装完成



当前网络不佳时SSH远程连接插件可能未安装成功，此时无需操作，在连接Notebook之后，会弹出如下图对话框，单击Install and Reload即可。

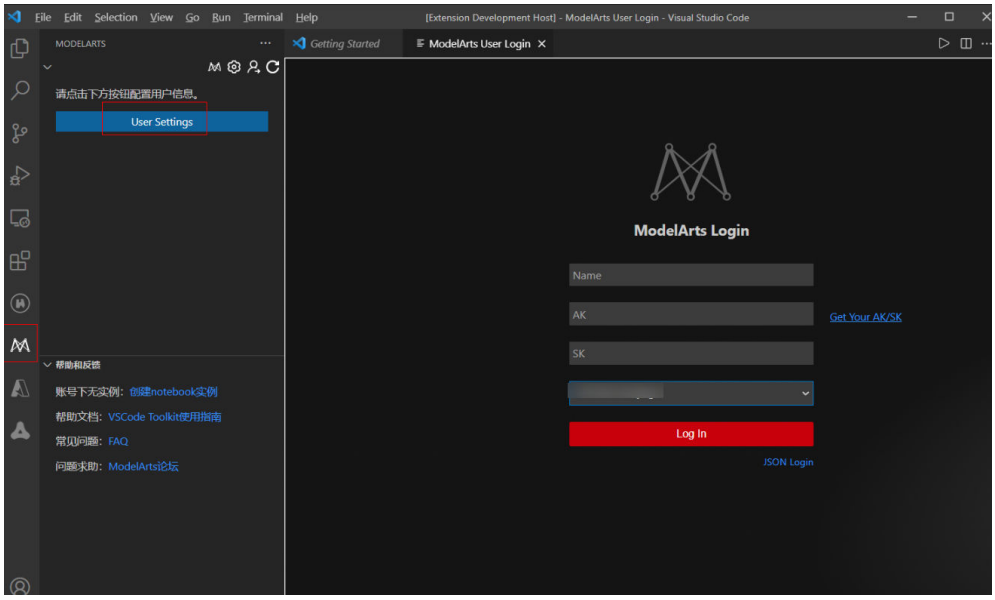
图 2-12 重新连接远程 SSH



Step4 登录 VS Code 插件

1. 在本地的VS Code开发环境中，单击ModelArts图标  ，单击“User Settings”，配置用户登录信息。

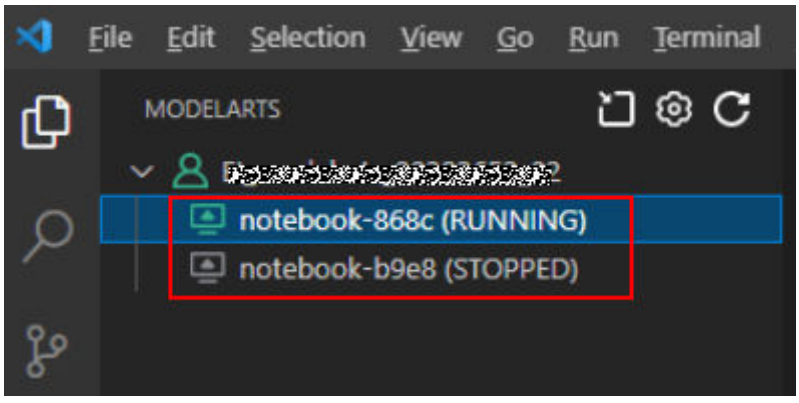
图 2-13 登录插件



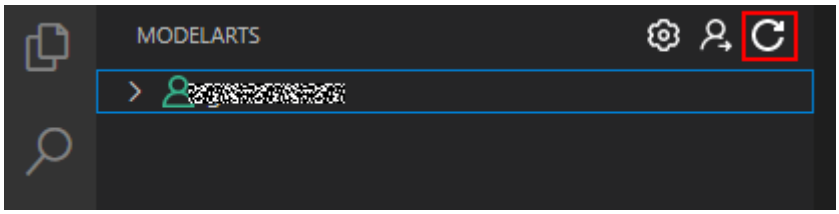
2. 输入如下用户登录信息，单击“登录”。
   - Name：自定义用户名，仅用于VS Code页面展示，不与任何华为云用户关联。
   - AK：在华为云账号中心“ 我的凭证 > 访问密钥”中创建访问密钥，获取AK（**参考链接**）。
   - SK：在华为云账号中心“我的凭证 > 访问密钥”中创建访问密钥，获取SK（**参考链接**）。
   - 选择区域：例如“上海一”。此处的区域必须和远程连接的Notebook在同一个区域，否则插件无法显示需要连接的实例列表。

   登录成功后显示Notebook实例列表。

图 2-14 登录成功



如果该账号下还没有创建过远程连接的实例，则列表显示为空。需要参考**创建云上开发环境**创建实例后，在插件刷新后即可显示实例列表。



2.3 创建云上开发环境

在开始进行AI应用开发前，您需要创建云上开发环境，即创建Notebook实例。

背景信息

- “运行中”的Notebook将一直收费，当您不需要使用时，建议停止Notebook，避免产生不必要的费用。在创建Notebook时，也可以选择开启自动停止功能，在指定时间内停止运行Notebook，避免产生不必要的费用。
- 只有处于“运行中”状态的Notebook，才可以执行打开、停止、删除操作。
- 基于ModelBox框架创建的Notebook实例，只能通过SSH远程访问。
- 一个账户最多创建10个Notebook。

创建 Notebook 实例

1. 登录ModelArts管理控制台，在左侧导航栏中选择“全局配置”，检查是否配置了访问授权。若未配置，请先配置访问授权。参考**使用委托授权**完成操作。

   图 2-15 配置授权

2. 登录ModelArts管理控制台，在左侧导航栏中选择“开发环境 > Notebook”，进入“Notebook”新版管理页面。
3. 单击“创建”，进入“创建Notebook”页面，请参见如下说明填写参数。
   a. 填写Notebook基本信息，包含名称、描述、是否自动停止，详细参数请参见**表2-1**。

   图 2-16 Notebook 基本信息

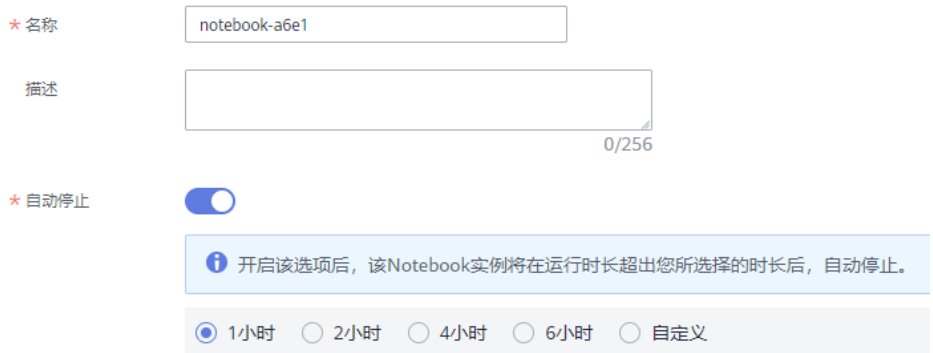
   

表 2-1 基本信息的参数描述

参数名称	说明
“名称”	Notebook的名称。只能包含数字、大小写字母、下划线和中划线，长度不能超过128位且不能为空。
“描述”	对Notebook的简要描述。
“自动停止”	默认开启，且默认值为“1小时”，表示该Notebook实例将在运行1小时之后自动停止，即1小时后停止规格资源计费。 开启自动停止功能后，可选择“1小时”、“2小时”、“4小时”、“6小时”或“自定义”几种模式。选择“自定义”模式时，可指定1~24小时范围内任意整数。

b. 填写Notebook详细参数，如工作环境、资源规格等，详细参数请参见表2-2。

图 2-17 Notebook 实例镜像选择

图 2-18 Notebook 实例的详细参数

表 2-2 Notebook 实例的详细参数说明

参数名称	说明
“镜像”	支持公共镜像和自定义镜像。 ● 公共镜像：即预置在ModelArts内部的镜像。 ● 自定义镜像：可以将基于公共镜像创建的实例保存下来，作为自定义镜像使用。 一个镜像对应支持一种AI引擎，不可以在同一个Notebook实例中切换AI引擎。 基于ModelBox框架进行AI应用开发时，基础镜像需要选择ModelBox镜像。
“资源池”	“公共资源池”无需单独购买，即开即用，按需付费，即按您的Notebook实例运行时长进行收费。 “专属资源池”需要单独购买并创建。
“类型”	不同的镜像支持的芯片类型不同，根据实际需要选择。
“规格”	根据选择的芯片类型不同，可选资源规格也不同。请根据界面实际情况和需要选择。
“存储配置”	包括“云硬盘EVS”和“弹性文件服务SFS”。请根据界面实际情况和需要选择。 ● 选择“云硬盘EVS”作为存储位置。 根据实际使用量设置磁盘规格。磁盘规格默认5GB。磁盘规格的取值范围为5GB～4096GB。 从Notebook实例创建成功开始，直至实例删除成功，磁盘每GB按照规定费用收费。 “云硬盘EVS”的存储路径挂载在/home/ma-user/work目录下。用户在Notebook实例中的所有文件读写操作都是针对该存储目录下的的内容操作，与OBS对象存储（OBS对象桶）无关。Notebook实例运行中，可以**动态挂载OBS并行文件系统**用来读取数据。 停止或重启Notebook实例时，内容会被保留，不丢失。 删除Notebook实例时，内容不保留。
“SSH远程开发”	开启此功能后，用户可以在本地开发环境中远程接入Notebook实例的开发环境。 ModelBox镜像仅支持本地SSH远程连接，需要开启“SSH远程开发”功能。
“密钥对”	开启“SSH远程开发”功能后，需要设置此参数。 可以选择已有密钥对。 也可以单击密钥对右侧的“立即创建”，跳转到数据加密控制台，在“密钥对管理 > 私有密钥对”页面，单击“创建密钥对”。 **注意** 创建好的密钥对，请下载并妥善保存，连接云上Notebook开发环境时，需要用到密钥对进行鉴权认证。

参数名称	说明
“远程访问白名单”	开启“SSH远程开发”功能后，可以设置此参数。 设置为允许远程接入访问这个Notebook的IP地址（例如本地PC的IP地址或者访问机器的外网IP地址，最多配置5个），不设置则表示无接入IP地址限制。 如果用户使用的访问机器和华为云ModelArts服务的网络有隔离，则访问机器的外网地址需要在主流搜索引擎中搜索“IP地址查询”获取，而不是使用**ipconfig**或**ifconfig/ip**命令在本地查询。  白名单IP地址如果配置错误将无法连接Notebook开发环境。 创建完Notebook后，可以在Notebook详情页中修改白名单IP地址。

4. 参数填写完成后，单击“立即创建”进行规格确认。
5. 参数确认无误后，单击“提交”，完成Notebook的创建操作。

   进入Notebook列表，正在创建中的Notebook状态为“创建中”，创建过程需要几分钟，请耐心等待。当Notebook状态变为“运行中”时，表示Notebook已创建并启动完成。
6. 在Notebook列表，单击实例名称，进入实例详情页，查看Notebook实例配置信息。

   包括Notebook实例名称、规格、状态、镜像类型、用户ID、存储路径、存储容量、Notebook地址和端口号、允许远程访问的白名单IP地址、认证密钥文件名。

图 2-19 查看 Notebook 实例详情

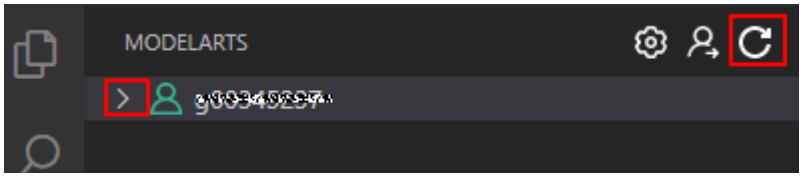
在白名单右侧单击，可以修改允许远程访问的白名单IP地址。

2.4 连接云上开发环境

本节介绍如何使用本地VSCode插件连接云上开发环境Notebook。

注意：若新创建的实例未显示，单击刷新按钮重新登录，登录成功后单击左侧箭头展开实例列表。

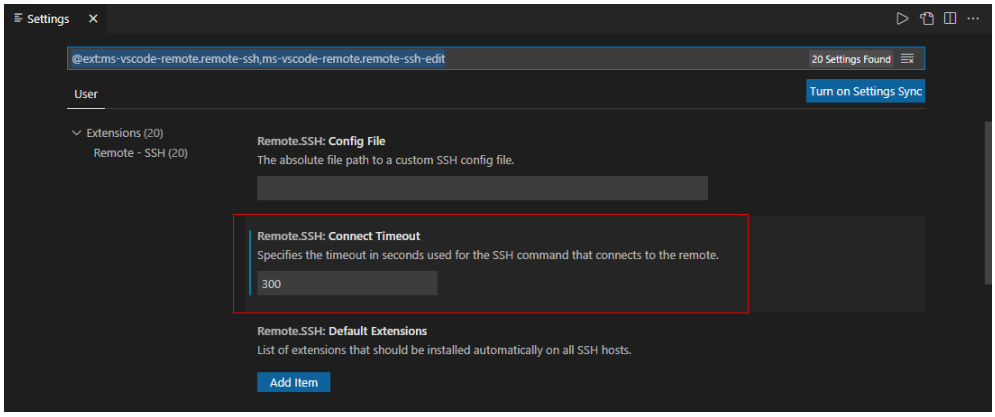
图 2-20 刷新实例



由于插件连接需要一定时间，建议将“Connect Timeout”设置为较大值。

操作方法：执行“Ctrl+Shift+P”，输入“Remote-SSH:Settings”

图 2-21 设置 Connect Timeout



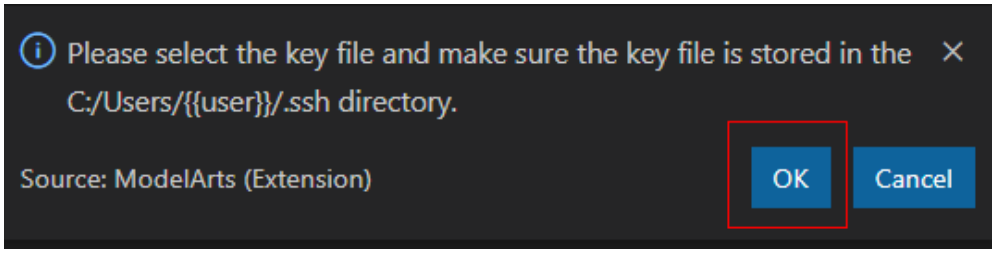
1. 单击实例名称，在VSCode插件中会显示Notebook实例详情，单击“连接”按钮进行远程连接。如果Notebook实例是停止状态，连接时VSCode插件会先启动实例再进行连接。

   图 2-22 连接 Notebook 实例

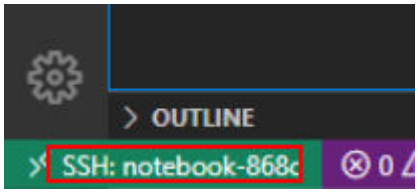
2. 第一次连接Notebook时，系统右下角会提示需要先配置密钥文件。选择本地密钥pem文件（必须放在用户家目录下的.ssh文件夹下），根据系统提示单击“OK”。

图 2-23 配置密钥文件



3. 单击“确定”后，插件自动连接远端Notebook实例。首次连接比较耗时，需要约1~2分钟，取决于本地的网络情况。VSCode环境左下角显示SSH：实例名称，如下图所示即为连接成功。

图 2-24 连接成功



3 第一个 AI 应用

3.1 创建应用工程

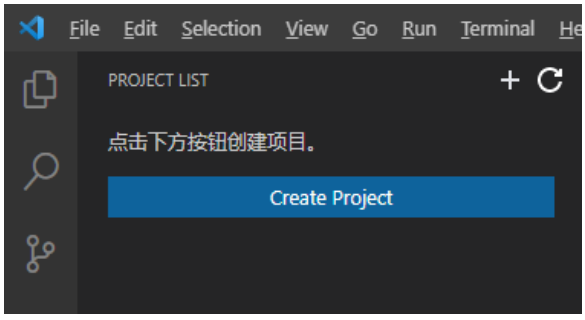
使用VS Code连接到云上开发环境后，用户就可以开发AI应用了。云上AI应用开发环境中预置了丰富的开发工具，用户可以通过左侧导航栏的ModelArts图标使用相关功能。本章节通过创建并运行两个不同类型的AI应用工程，帮助开发者快速了解应用工程的组成及运行方法。

创建应用工程

使用VS Code连接到云上开发环境后，需要先创建应用工程。

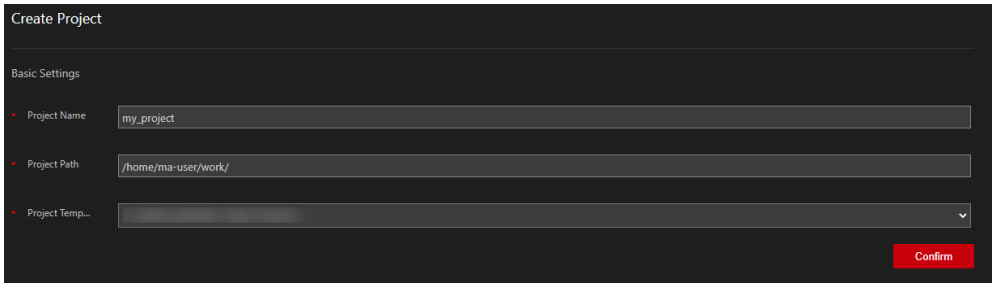
1. 单击VS Code左侧导航栏的ModelBox图标，单击“Create Project”创建工程。如果项目列表中已经有工程了，则单击工程列表中右上角的“+”创建工程。

   图 3-1 创建工程

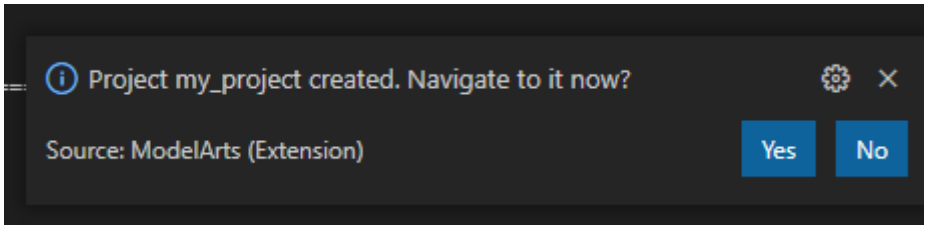
2. 目前插件中内置了行人检测（pedestrian_detection）以及空工程（empty）两个样例，输入工程名称、工程路径以及选择工程模板，单击“Confirm”创建工程。

图 3-2 新建项目



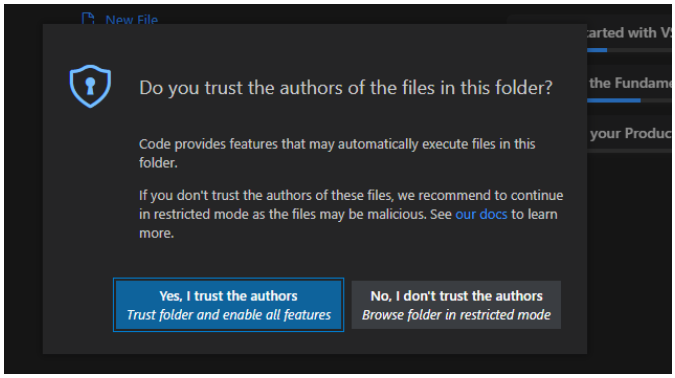
3. 创建完成后右下角会弹出信息，单击“yes”打开工程。

图 3-3 打开工程



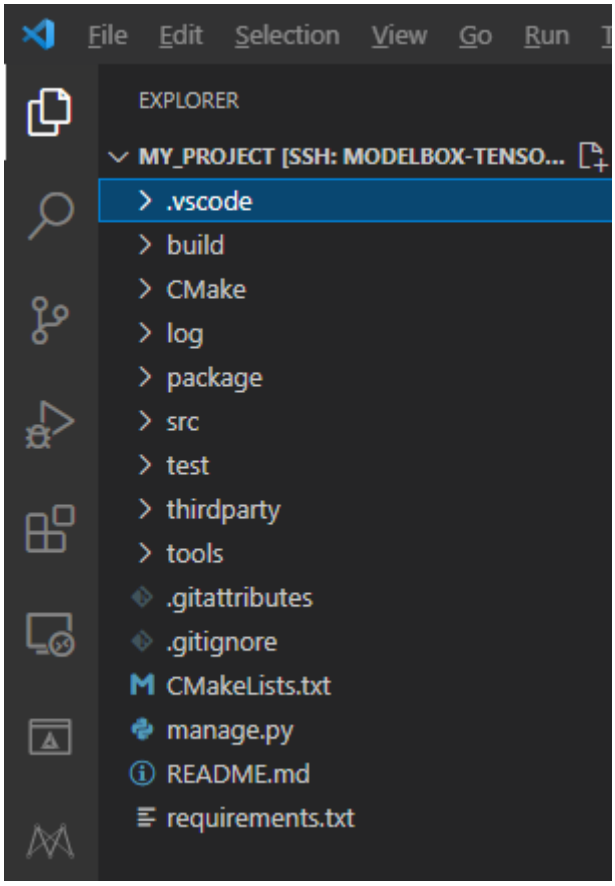
4. 工程打开后，VS Code弹出的确认信息框选择“Yes”选项。

图 3-4 确认信息



此时左侧Explorer栏可以看到创建好的工程结构。

图 3-5 工程结构



工程中的目录作用介绍如下：

- .vscode： vscode 配置文件，包含工程设置和运行的配置文件
- build：编译产物所在目录
- CMake：CMake配置文件
- log：制作镜像以及实例运行日志
- package：rpm打包配置文件
- src: 工程开发代码主目录
- test: 测试代码及测试数据
- thirdparty：工程依赖的第三方库
- tools: 工程构建依赖工具
- manage.py： 工程套件命令行入口

该工程包含已经配置好的CMakeList文件，开发者在src目录下进行AI应用开发后，不修改或少量修改CMakeList即可直接编译运行，开发者也可以将该工程目录直接上传至自己的git代码仓库。

如果工程中根目录中有requirements.txt文件，意味着该工程需要安装第三方python依赖，单击菜单栏中的“Terminal”->“New Terminal”新建一个终端窗口，输入以下命令即可安装。这个路径中的文件在实例重启之后会恢复，建议每次重启之后都重新执行下面的命令。

```
pip install -r requirements.txt  --target /home/ma-user/.local/lib/python3.7/site-packages --upgrade --force-reinstall
```


3.2 应用样例 1：视频车辆检测

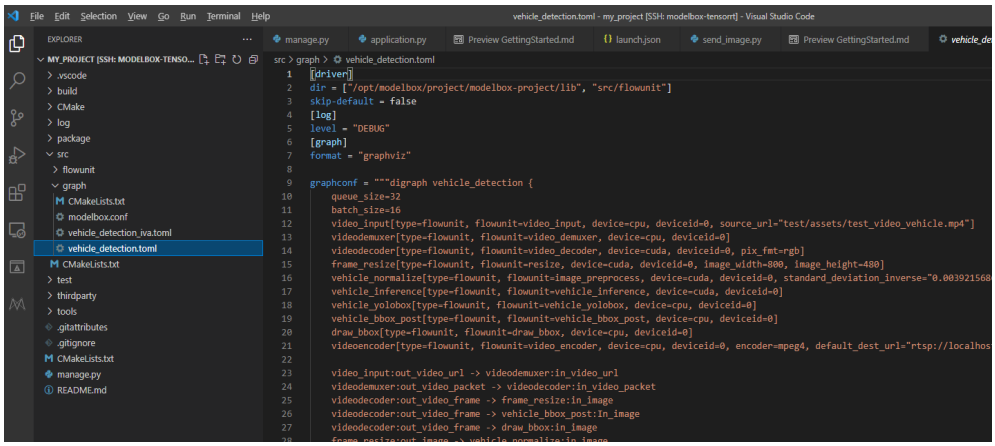
3.2.1 样例介绍

在创建工程时选择“vehicle_detection”工程模板将会创建视频车辆检测样例工程，该样例是一个视频类的ModelBox应用样例，通过读取视频，经过目标检测的模型得到视频中车辆的位置。该样例中“src”目录下预置了视频车辆检测的具体实现，它包含“flowunit”和“graph”两个目录，分别代表功能单元和图。

在基于ModelBox框架的AI应用开发中，AI应用是由图构成的。其中，功能单元可以理解为一段相对独立的数据处理功能，所有功能单元编排连线后构成图，代表一个AI应用。图在ModelBox中的呈现形式为toml文件，在该样例中位于如下路径：

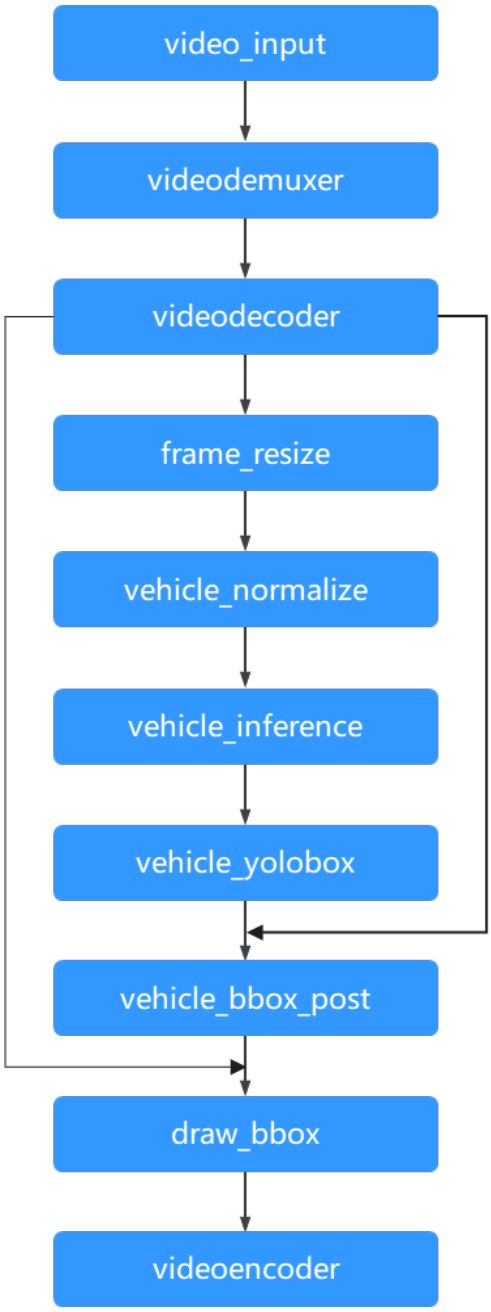
```
src/graph/vehicle_detection.toml
```

图 3-6 AI 应用样例



该toml文件代表的数据处理流程如下：

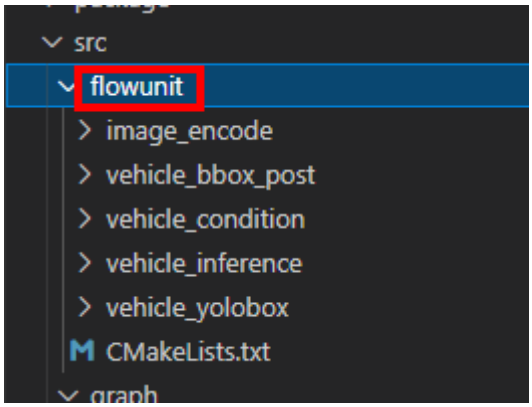
图 3-7 数据处理流程



该图从上到下代表了对输入视频的每一步数据处理。“video_input”是功能单元接收的输入视频流，往下分别经过“videodemuxer”和“videodecoder”功能单元，“videodecoder”功能单元输出“image”，“image”经过前处理，包含“resize”、“normalize”之后，送给模型（“vehicle_inference”是一个yolov3模型），模型将推理得到的bbox结果传入后续的后处理功能单元进行处理（vehicle_bbox_post），可得到最终的bbox框，将bbox框和“videodecoder”出来的“image”一同送入“draw_bbox”中，将绘制完bbox的image传入“videoencoder”，即得到带有检测框的视频流输出。

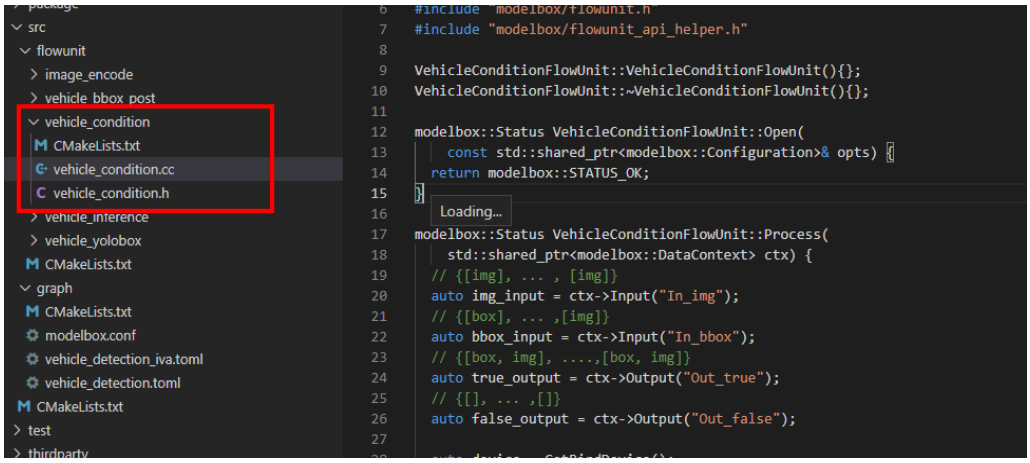
流程图中的各个功能单元的实现，一部分是直接使用ModelBox框架预置的功能单元，预置功能单元位于“/usr/local/lib64/”目录下，以二进制so文件的形式提供；另一部分是自己开发的功能单元，位于“src/flowunit”目录下。

图 3-8 功能单元



每一个功能单元中即为该单元模块的实现代码，功能单元有三种类型，分别为“cpp”、“python”和“infer”，即c++功能单元、python功能单元以及推理功能单元。图3-9为一个c++功能单元的样例，Process函数为数据处理主函数。

图 3-9 C++功能单元样例



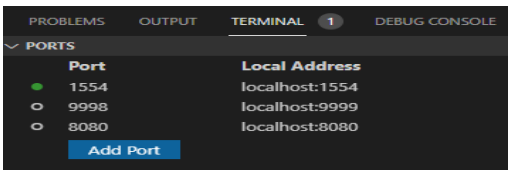
3.2.2 运行应用样例

1. 运行前准备

   首先下载并本地安装PotPlayer播放器，用于播放AI应用运行结果视频流。

   在VSCode中配置端口转发， 操作方法：执行“Ctrl+Shift+P”，输入“Forward a Port”，输入“1554”端口。

   图 3-10 配置端口转发

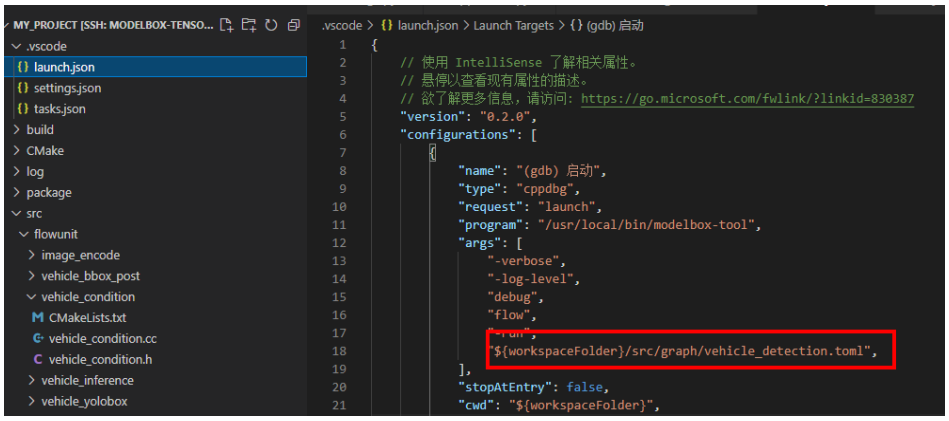
   

2. 编译运行AI应用

   a. 在“.vscode/task.json”中配置了工程运行时需要执行的任务。

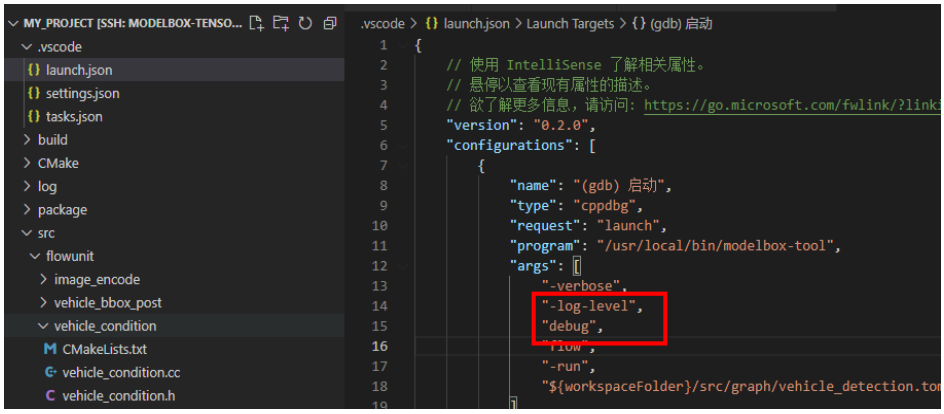
b. 在“.vscode/launch.json”配置运行的图路径，这里预置为样例的图。

图 3-11 配置图路径



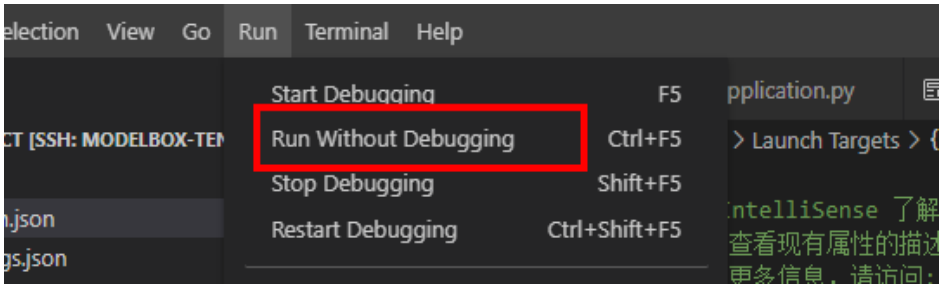
c. 在“.vscode/launch.json”配置日志级别，默认为“debug”。

图 3-12 设置日志级别



d. 单击“Run>Run Without Debugging”或者使用快捷键Ctrl+F5运行程序。

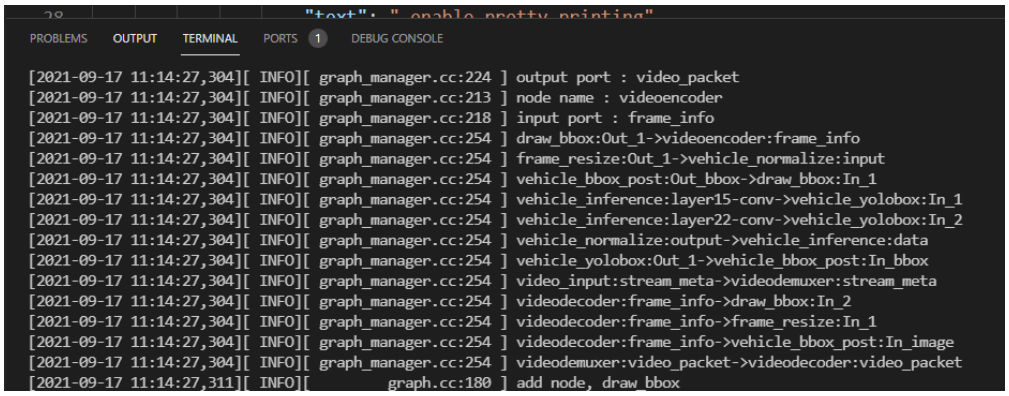
图 3-13 运行程序



3. 查看运行结果

命令执行后，会自动进行工程的编译构建和运行。单击“Terminal”控制台，可以查看实时运行日志。

图 3-14 查看运行日志

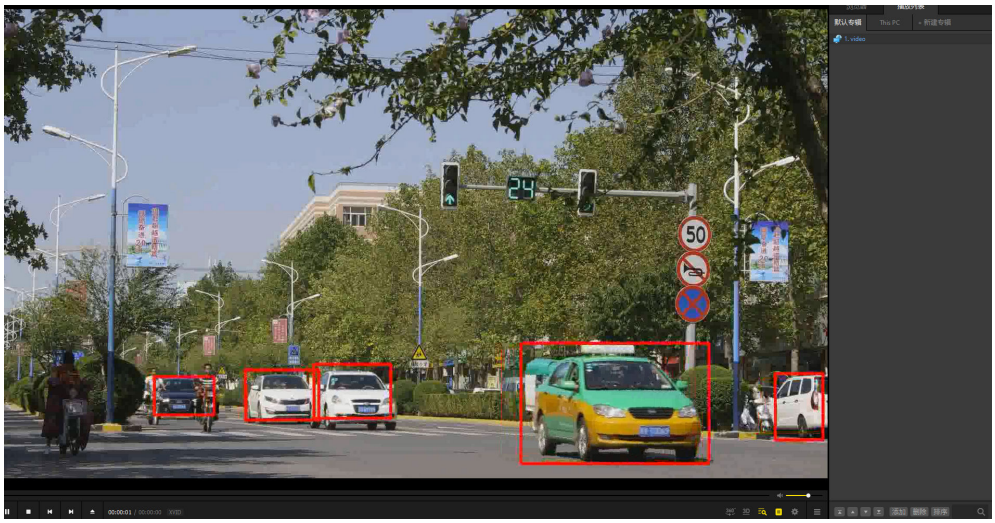


运行时，modelbox框架会先加载和解析图，然后运行视频流推理，运行视频推理时日志会很快速的打印，此时在浏览器中输入如下地址可打开rtsp流播放。

```
rtsp://localhost:1554/video
```

图 3-15 打开 rtsp 流播放视频

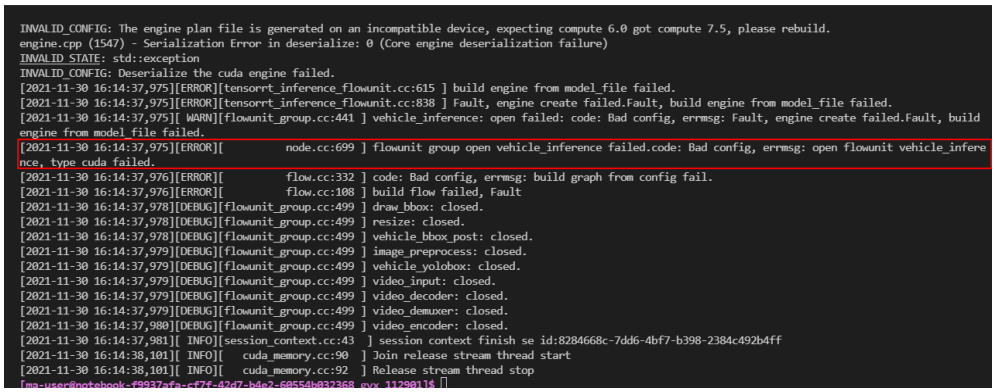
图 3-16 视频播放推理结果



如果无视频播输出可能是播放早了或者已经播放完了，可以再运行一次图，待控制台开始快速输出大量推理日志后，在浏览器输入地址查看结果。

如果运行图时看到下面的日志，代表当前样例使用的模型和开发环境带的加速卡类型不符，则会报如下错误：

图 3-17 模型依赖的硬件和开发环境硬件不匹配时的报错

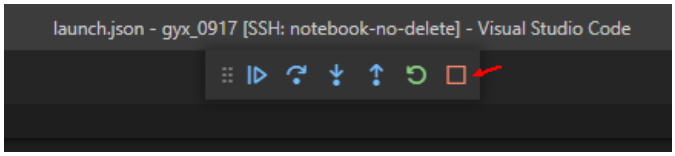


此时需要重新将TensorRT模型转换成当前开发环境的模型，或者重新创建和模型对应的开发环境。

4. 结束运行

   在VSCode上方单击停止按钮，或者在Terminal命令行窗口执行“Ctrl + c”，即可结束运行。

图 3-18 结束运行



3.3 应用样例：图片行人检测

3.3.1 样例介绍

在创建工程时选择“pedestrian_detection”工程模板将会创建图片行人检测样例工程，该样例是一个图片类的ModelBox应用样例，通过读取图片，经过目标检测的模型得到图片中行人的位置。

图片行人检测工程中根目录有requirements.txt文件，意味着该工程需要安装第三方python依赖，单击菜单栏中的“Terminal”->“New Terminal”新建一个终端窗口，输入以下命令即可安装。由于这个路径中的文件在实例重启之后会恢复，建议每次重启之后都重新执行下面的命令。

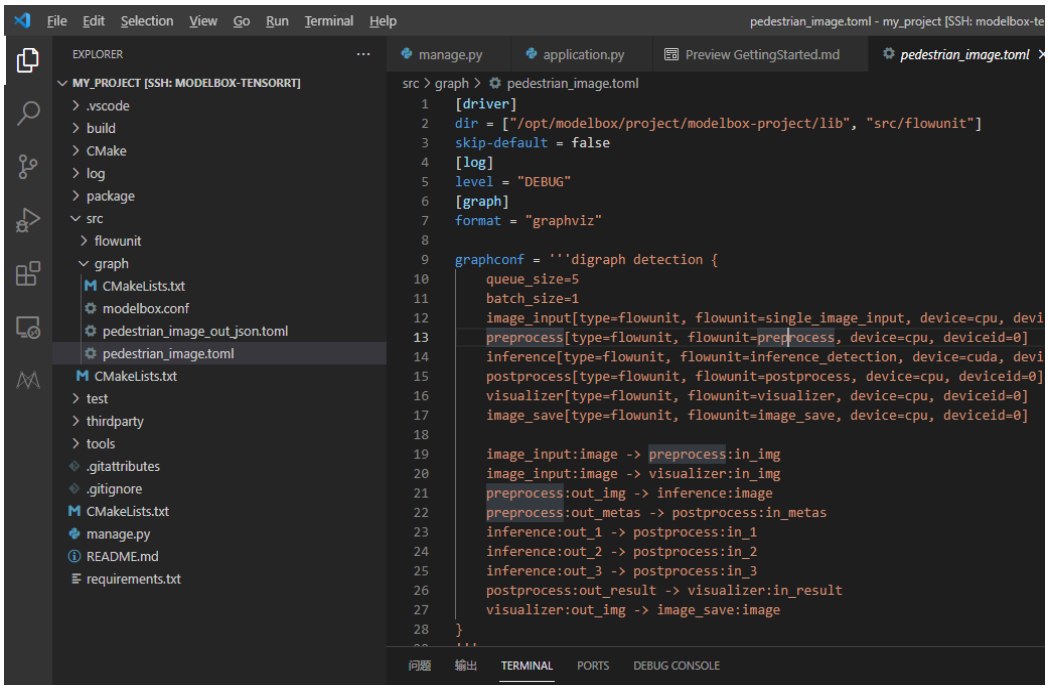
```
pip install -r requirements.txt  --target /home/ma-user/.local/lib/python3.7/site-packages --upgrade --force-reinstall
```

该样例中“src”目录下预置了图片行人检测的具体实现，它包含“flowunit”和“graph”两个目录，分别代表功能单元和图。

在该样例中toml位于如下路径：

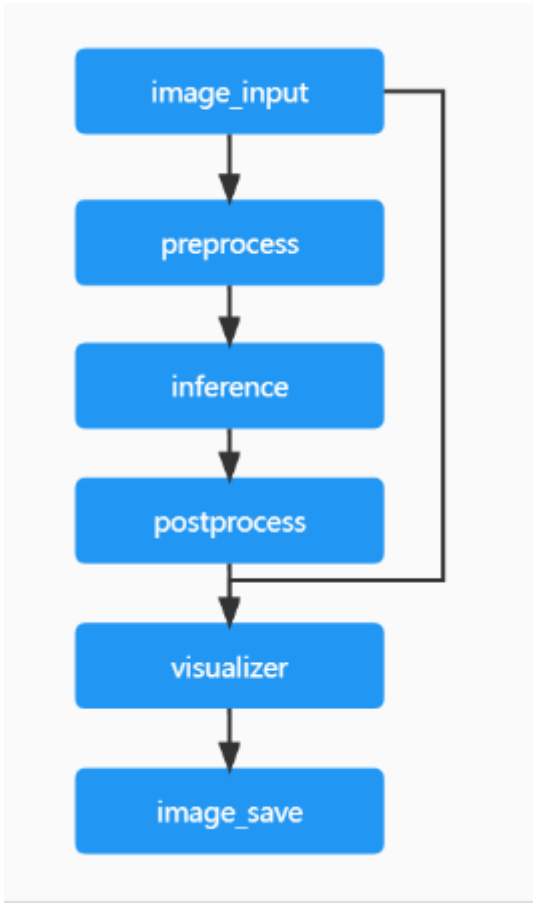
```
src/graph/pedestrian_image.toml
```

图 3-19 AI 应用样例



该toml文件代表的数据处理流程如下：

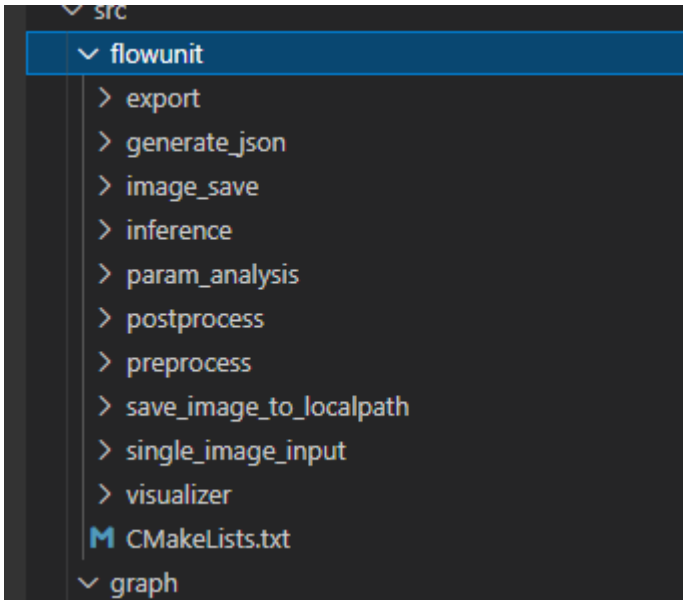
图 3-20 数据处理流程



该图从上到下代表了对输入图片的每一步数据处理。“image_input”是功能单元接收的输入图片，往下分别经过“preprocess”和“inference”功能单元，“preprocess”功能单元对图片进行预处理，预处理的输出送给模型（“inference”是一个yolov3模型），模型将推理得到的bbox结果传入后续的后处理功能单元进行处理（“postprocess”），可得到最终的bbox框，将bbox框和“image_input”出来的图片一同送入“visualizer”中，将绘制完bbox的image传入“image_save”，即可将带有检测结果的图片保存到本地。

流程图中的各个功能单元的实现，一部分是直接使用ModelBox框架预置的功能单元，预置功能单元位于“/usr/local/lib64/”目录下，以二进制so文件的形式提供；另一部分是自己开发的功能单元，位于“src/flowunit”目录下。

图 3-21 功能单元



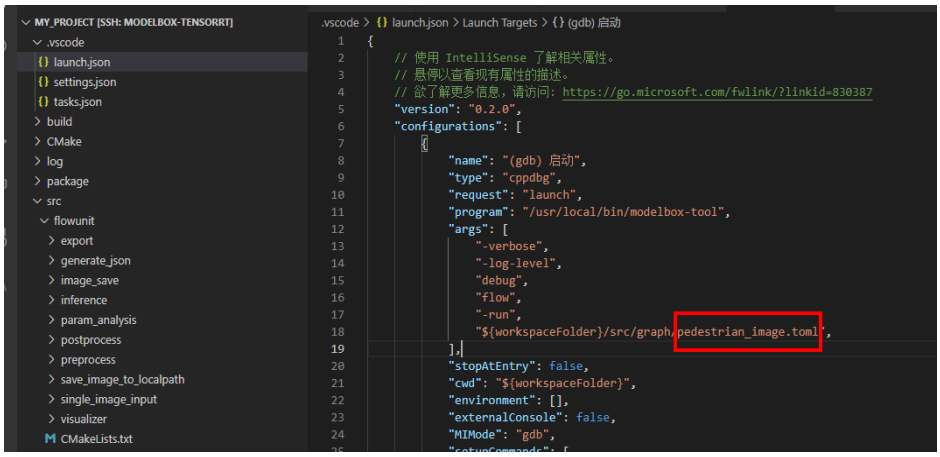
3.3.2 运行应用样例

本样例中图片行人检测样例包含了两种运行模式，一种是直接读取本地图片，图片的检测结果将会保存到当前工程目录；另外一种是通过建立http服务的方式获取用户的请求，用户发送图片请求之后将会返回该图片中行人检测的结果。

运行行人检测应用（读取本地图片）

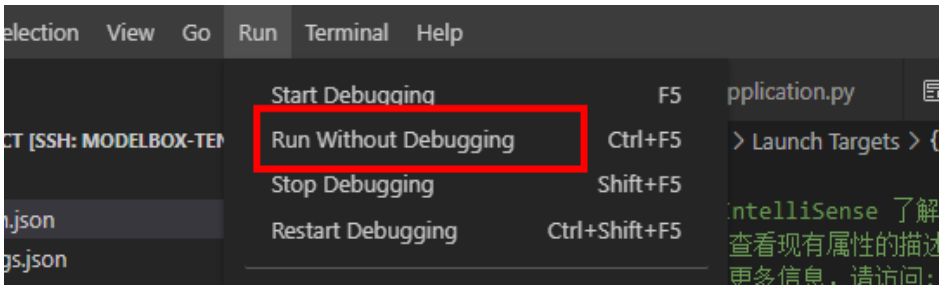
1. 编译运行AI应用
   a. 在“.vscode/launch.json”中配置运行的图路径，这里默认运行的图为“pedestrian_image.toml”。

图 3-22 配置工程名称



b. 单击“Run>Run Without Debugging”或者使用快捷键Ctrl+F5运行程序。

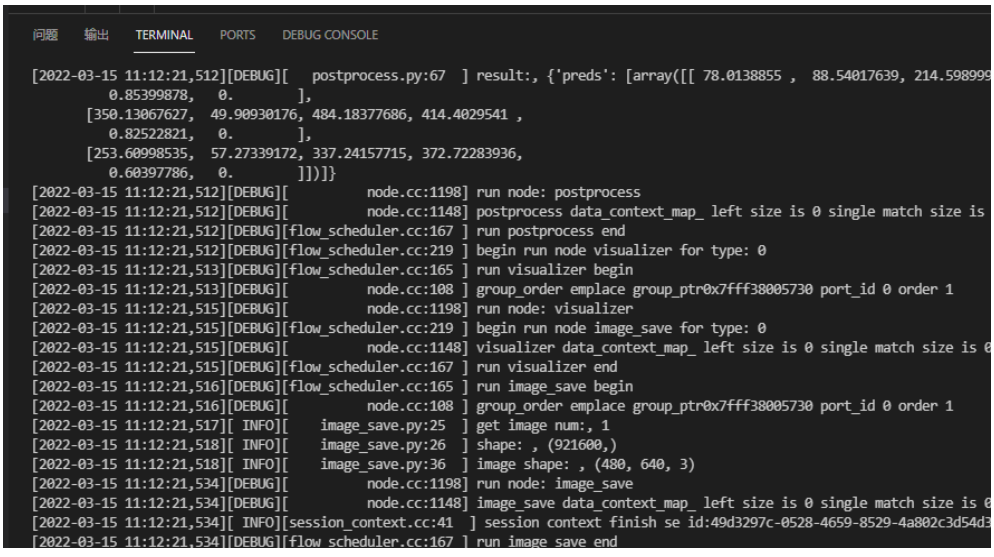
图 3-23 运行程序



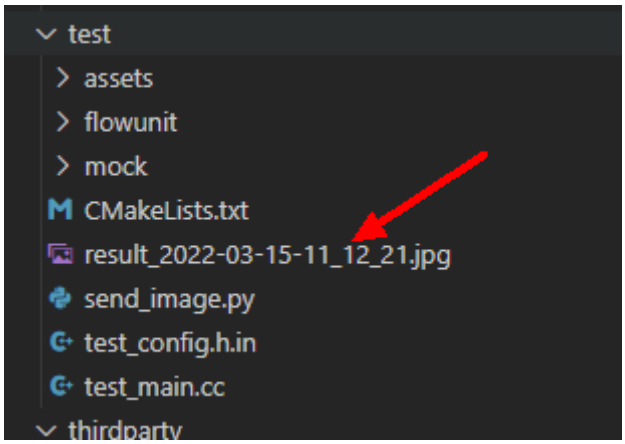
2. 查看运行结果

命令执行后，会自动进行工程的编译构建和运行。单击“Terminal”控制台，可以查看实时运行日志。

图 3-24 查看运行日志



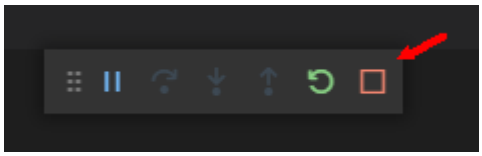
运行结束后，将会在test目录下生成一张检测结果图片。



3. 结束运行

   在VSCode上方单击停止按钮，或者在Terminal命令行窗口执行“Ctrl + c”，即可结束运行，。

   图 3-25 结束运行

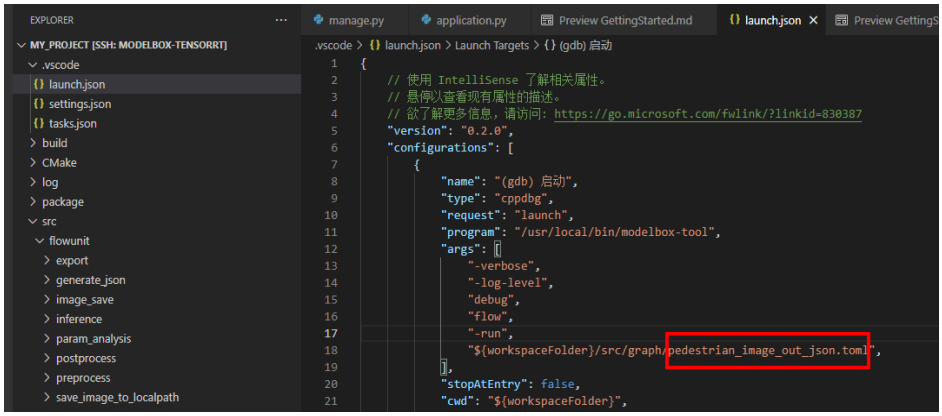


运行行人检测应用（本地模拟远程请求）

1. 编译运行AI应用

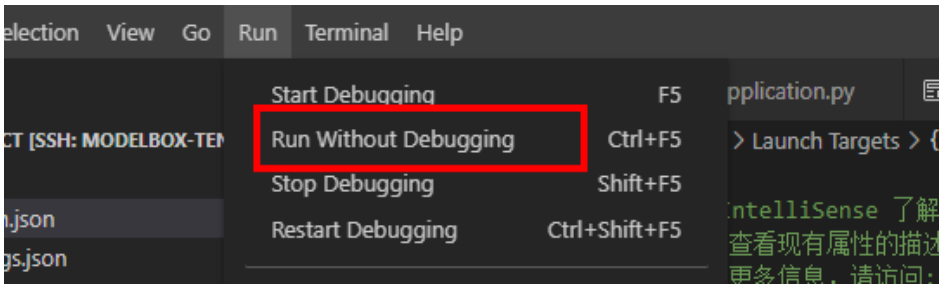
   a. 在“.vscode/launch.json”中配置运行的图路径，这里需要修改运行的图为“pedestrian_image_out_json.toml”。

   图 3-26 配置工程名称



   b. 单击“Run>Run Without Debugging”或者使用快捷键Ctrl+F5运行程序。

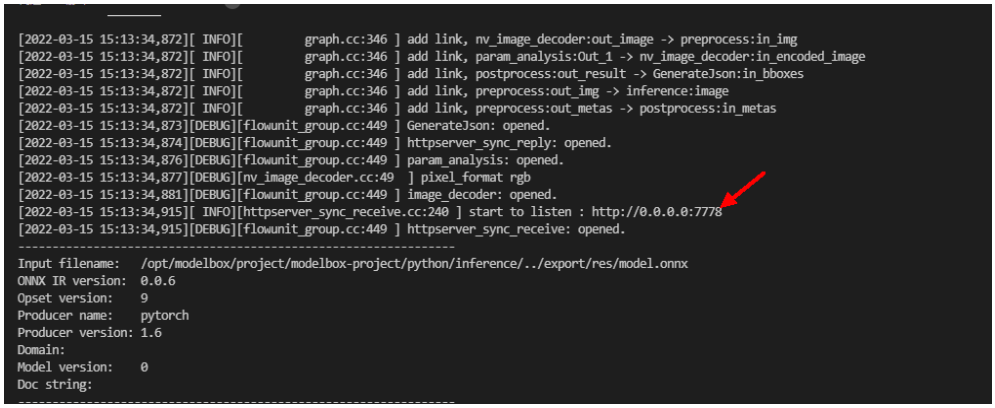
图 3-27 运行程序

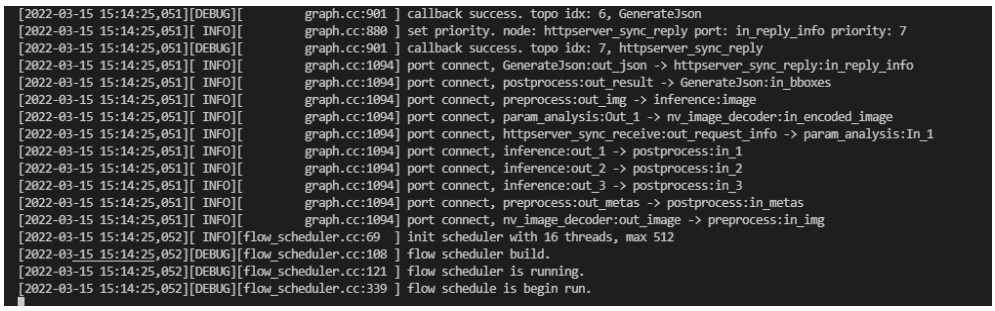


2. 查看运行结果

   命令执行后，会自动进行工程的编译构建和运行。单击“Terminal”控制台，可以查看实时运行日志。

图 3-28 查看运行日志





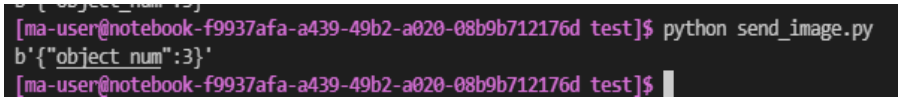
   运行结束后，程序会在这里阻塞，等待用户的请求。

3. 模拟请求

   单击菜单栏“Terminal>New Terminal”新建终端窗口，进入test目录，执行以下命令发起请求：

   ```
   python send_image.py
   ```

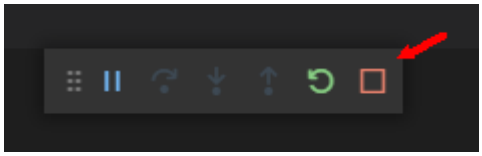
   此时会收到应用返回的行人检测的数量信息：



4. 结束运行

   在VSCode上方单击停止按钮，或者在Terminal命令行窗口执行“Ctrl + C”，即可结束运行。

图 3-29 结束运行



4 AI 应用开发和调试

4.1 ModelBox 基本概念

运行完第一个AI应用后，开发者已经对应用工程和运行方法有了初步的了解。如果您仅想了解开发到部署的端到端流程，可直接跳过此章节，进入**应用发布**。如果您需要开发自己的AI应用，则需要详细了解本章的内容。

当进行AI应用开发时，开发者可以使用当前的样例工程结构直接开发自己的业务逻辑，并在当前环境中进行代码调试。当进行AI应用开发时，开发者首先需要学习ModelBox框架的相关概念和接口，以便基于框架开发出高性能的AI应用。

本章介绍ModelBox的基本概念，帮助开发者为后续的AI应用开发做准备。

ModelBox 解决的问题

在AI应用开发时，开发者需要将训练完成模型和应用逻辑一起组成AI应用，然后上线发布成为服务。在整个过程中，开发者需要面临复杂的应用编程问题，例如：

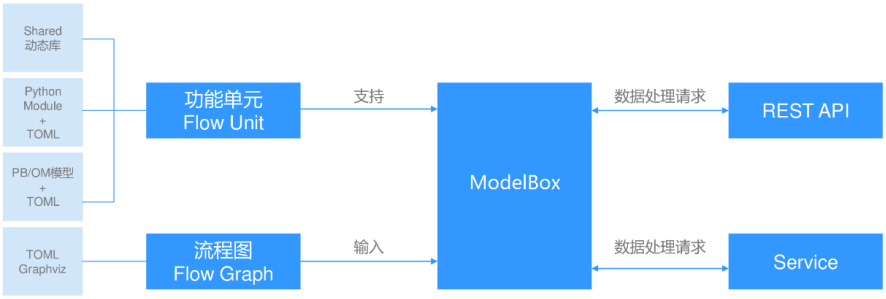
- 加速卡等复杂的API使用
- 多线程并发互斥
- 多种开发语言的配合
- 应用性能，质量不满足要求
- 服务化上线复杂

ModelBox的目标就是解决AI开发者在开发AI应用时的编程复杂度，降低AI应用的开发难度，将复杂的数据处理、并发互斥、多设备协同、组件复用、数据通信等部分交由ModelBox处理，开发者主要聚焦业务逻辑本身，而不是软件细节，在提高AI应用开发的效率同时，保证软件的性能、可靠性、安全性等属性。

ModelBox 的核心概念

开发者在使用ModelBox前，需要关注的基本核心概念包括：功能单元、流程图、接收数据处理请求和ModelBox执行引擎。

图 4-1 ModelBox 核心概念



- **功能单元**

  ModelBox将流程图中的顶点称为功能单元(FlowUnit)。功能单元是应用的基本组成部分，也是ModelBox的执行单元。在ModelBox中，内置了大量的基础功能单元，开发者可以将这些功能单元直接集成到应用流程图中，这也是基于流程图开发的一大好处。除内置功能单元外，ModelBox支持功能单元的自定义开发，支持的功能单元形式多样，如C/C++动态库、Python脚本、模型+Toml配置文件等。

- **流程图**

  ModelBox中用流程图(Graph)来表达应用逻辑。采用有向图的方式，将应用的执行逻辑表达为顶点和边，其中顶点表示了应用的某个数据处理逻辑单元，边则表示了逻辑单元之间的数据传递关系。在ModelBox中，针对流程图的开发，既可以使用文本方式直接编辑，也可以使用可视化的编辑器进行编辑。对于流程图的表述，ModelBox默认采用**Graphviz**进行解释，即图的表述需要满足Graphviz的语法要求。

- **接收数据处理请求**

  应用流程图构建完毕后，需要数据处理请求才能触发应用运行。ModelBox提供两种数据处理请求接收的方式：在flowunit中，通过在加载时调用API产生数据处理的请求，因为产生的请求是固定的，所以一般用于调试场景；标准使用方式是使用ModelBox提供的服务插件机制，在插件中接收外部请求，并调用任务相关的API来完成数据处理的请求。ModelBox提供了默认的服务插件可用于参考。

- **ModelBox**

  在应用构建完成后，结合ModelBox的框架才能形成完整可运行的应用。ModelBox作为应用入口，首先进行功能单元的扫描加载、应用流程图读取构建，然后接收数据处理请求，数据触发ModelBox中的执行引擎对功能单元进行调度，最终完成请求的数据处理任务。

更多ModelBox框架的相关知识，请访问**https://modelbox-ai.com**。

4.2 开发 AI 应用

准备模型

AI应用开发的前序步骤是模型训练，因此开始进行AI应用开发时，您已经有训练优化好的可使用的模型了。当前ModelBox 镜像支持的模型类型有TensorRT和PyTorch模型，具体的版本号在您创建开发环境选择镜像时镜像名称中可以看到。

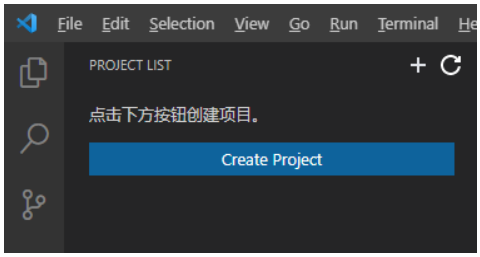
如果您的模型不是TensorRT和PyTorch模型，需要重新训练成这两种引擎的模型或者将模型转换后再继续，推荐使用TensorRT模型。TensorRT 7.1的模型转换请参考官方**指导和样例**。

ModelArts训练平台的训练模型输出通常存放在OBS桶中，您可以将模型先从OBS下载到本地，然后再将本地模型文件拖拽到工程中对应的目录下，等待VSCode下方的上传进度执行完即可。

创建空模板

用户可以基于空模板进行实例应用的开发。

图 4-2 创建工程



输入工程名称、工程路径以及选择工程模板，这里选择“empty”模板。单击“Confirm”开始创建。

开发流程图

流程图(Graph)是应用逻辑的表述，ModelBox将根据流程图构建应用的处理逻辑，因此在AI应用开发中，流程图的开发是首要进行的，流程图开发完毕后，才能明确需要开发的功能单元。图文件为graph目录下的toml文件。

流程图的格式如下，关于图的编写语法可以查看**Graphviz DOT**的指导。

```
[driver]
dir = ["build/drivers"]  #指定功能单元等驱动加载路径，可以指定多个，逗号分隔，编译后的功能单元位于build/drivers
[log]
level="INFO"
[flow]
desc = "example"
[graph]
format = "graphviz"      #流程图的格式，目前仅支持graphviz
graphconf = '''digraph example {
        node [shape=Mrecord]
        queue_size = 32
        batch_size = 1
        # 在此处编写自己的流程图
      }'''
```

开发功能单元

功能单元的类型有三种：cpp/python/infer，分别对应c++开发的功能单元、python开发的功能单元、以及推理功能单元。

- **新建推理功能单元**

  右键工程列表中的当前工程，单击“Create Flowunit”，输入流单元名称，流单元类型选择“infer”。流单元目录使用默认的路径。

  图 4-3 新建功能单元

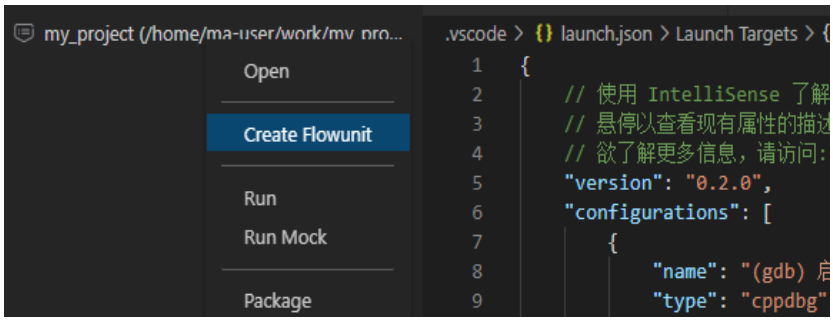
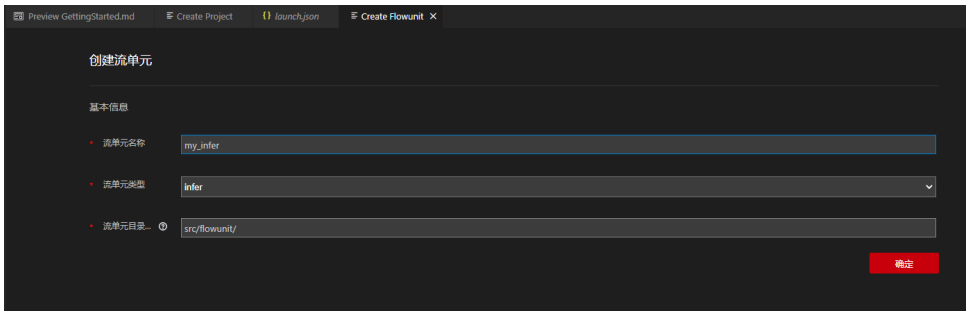


  图 4-4 新建功能单元

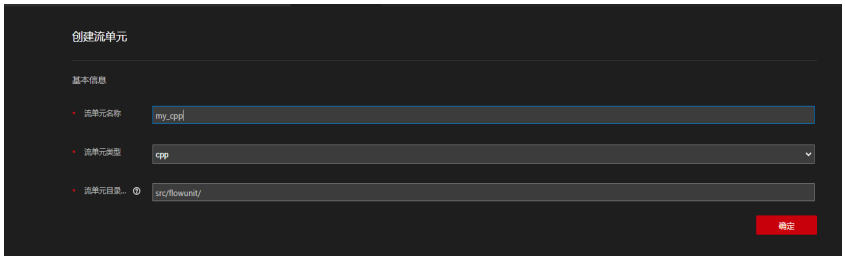


  在开发推理功能单元时，只需要提供独立的toml配置文件，指定推理功能单元的基本属性即可，其目录结构为：

```
[some-flowunit]
|---[some-flowunit].toml  // 配置文件
|---[model].engine   // 模型文件
```

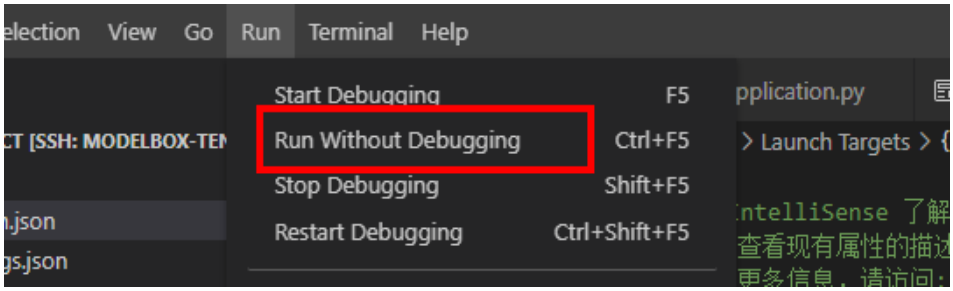
- **cpp/python功能单元**

  这里以cpp功能单元为例，右键工程列表中的当前工程，单击“Create Flowunit”，输入流单元名称，流单元类型选择“cpp”。流单元目录使用默认的路径：



  编排好toml图后，单击“Run>Run Without Debugging”或者使用快捷键Ctrl+F5运行程序。：

图 4-5 运行程序



更多开发指导和ModelBox接口信息，请请参考ModelBox文档：**开发ModelBox应用**。

4.3 调试 AI 应用

4.3.1 调试方式介绍

开发环境提供了两种AI应用调试方式：

- **开发环境内调试**
  这种方式是指在开发环境容器中直接运行AI应用。
- **部署到推理调试**
  这种方式是指将开发好的AI应用构建成应用镜像并部署到推理环境，通过在推理环境运行服务并查看日志的方式，确保开发好的AI应用可以正常部署。

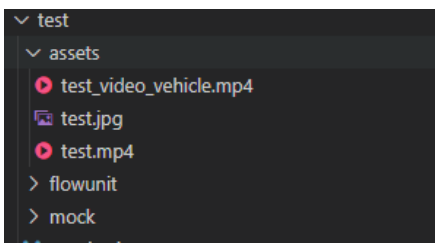
4.3.2 开发环境内调试

在开发环境容器内调试是指开发好的AI应用直接在开发环境中运行和调试。

本地输入输出调试

在开发环境中通常使用本地的测试输入作为AI应用输入，模板中的测试输入数据存放在/test/assets目录下。

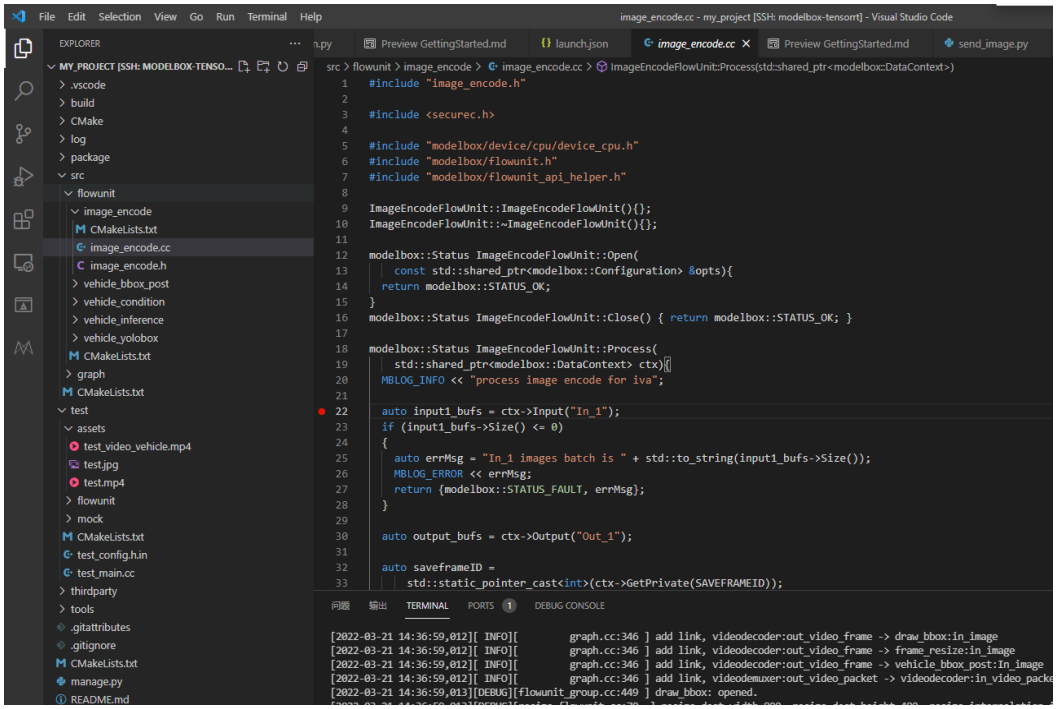
图 4-6 本地测试数据



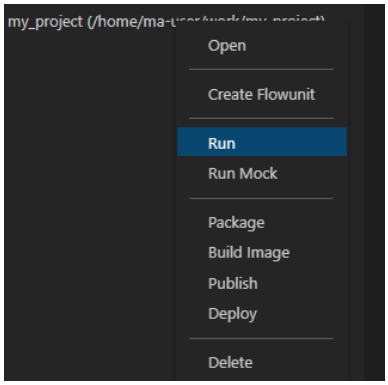
代码调试使用对应语言的调试方法即可，c++使用gdb，python使用pdb。GDB调试时，需要先配置.vscode目录下的编译配置文件tasks.json和调试配置文件launch.json，具体配置可参考上一章节中样例AI应用的配置说明。

配置好后，直接按F5即可进入调试模式。更多调试使用可参考官网**VSCode调试指南**。

图 4-7 使用 VS Code 调试代码

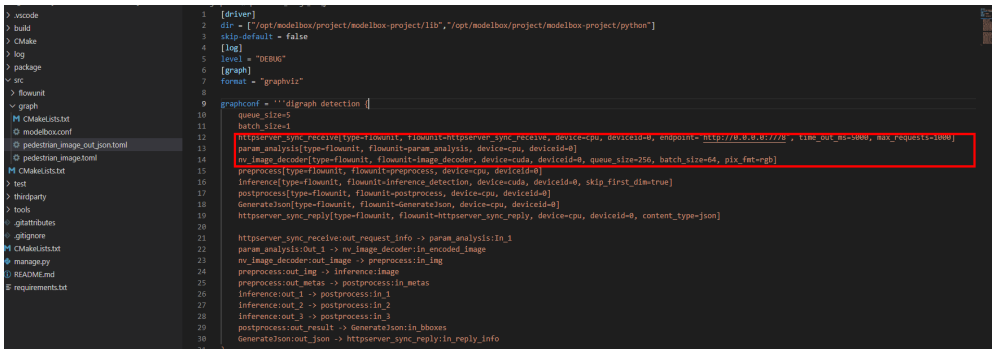


图编写完成后，鼠标右键工程列表中当前工程，单击“Run”，选择相应的toml图文件，单击运行该图，通过控制台可以查看运行日志。



本地图片请求调试

本地图片请求调试适用于同步服务的调试，该种调试方法通过解析http请求获取图片。本地图片请求调试方法可以参考图片行人检测pedestrian_detection模板中的pedestrian_image_out_json.toml文件。



该toml文件图片中框出来的三行中，httpserver_sync_receive定义一个http服务器流单元，用于接收用户的请求，param_analysis定义了一个参数解析的流单元，用于解析用户发送的http请求，这个流单元是用户自定义的流单元，开发者可以参考src/flowunit中的该流单元的实现自定义实现。image_decoder流单元定义了一个图片解码的流单元。通过以上三个流单元可以实现对用户http请求的解析。

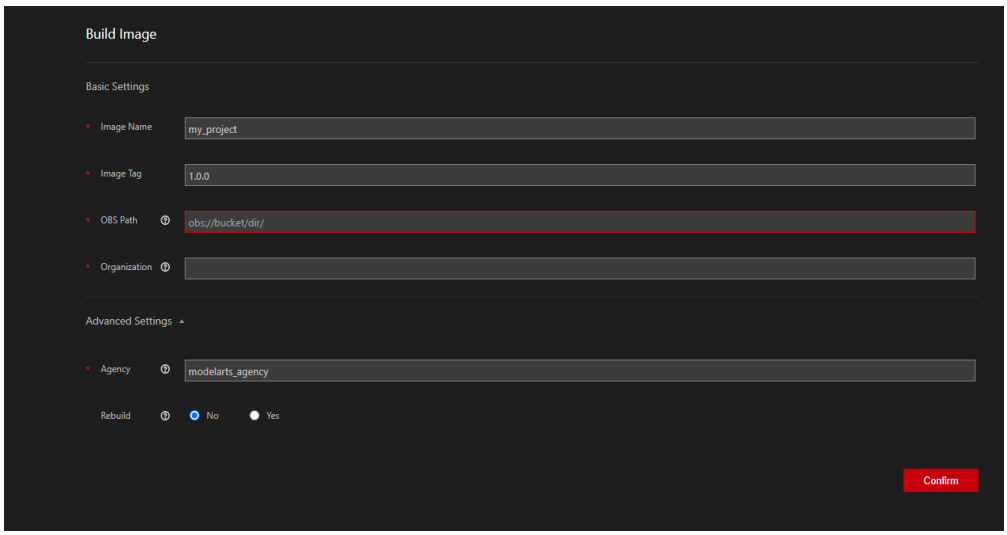
4.3.3 部署到推理调试

在开发环境容器内调试完成后，开发者可以将自己开发好的AI应用和ModelBox runtime镜像打包成新的运行镜像，并发布到ModelArts推理服务，直接测试部署的服务并查看日志，以确保开发好的AI应用可以在ModelArts推理平台正常运行。具体调试步骤如下：

1. 构建镜像

   鼠标右键工程列表中当前工程，单击“Build Image”，输入镜像名称、镜像版本、OBS路径（用于保存当前工程的文件）、SWR组织，展开高级设置，可以设置代理以及是否重新编译打包当前工程。单击“Confirm”开始制作镜像，镜像制作的日志将保存在当前工程log/docker-build.log中。

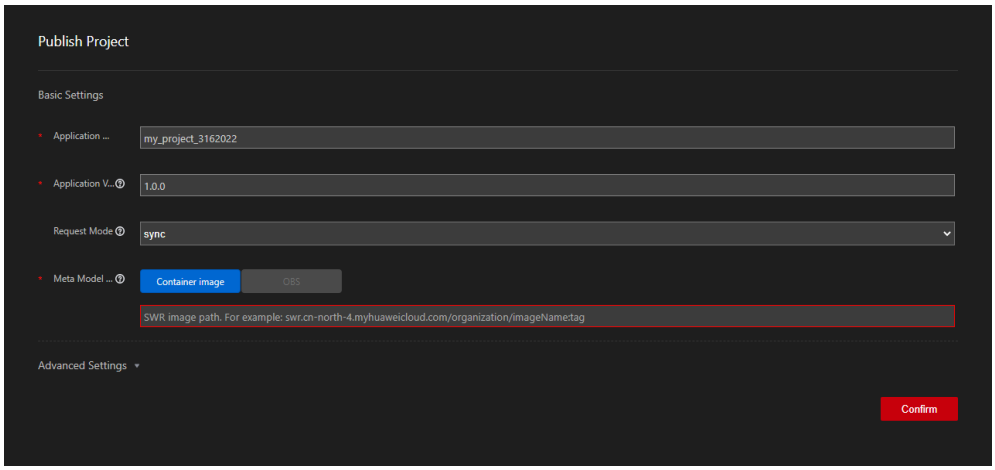
   图 4-8 构建镜像

2. 发布AI应用

   鼠标右键工程列表中当前工程，单击“Public”，输入应用名称、应用版本、请求方式以及镜像地址。单击高级选项可以配置更加丰富的选项。

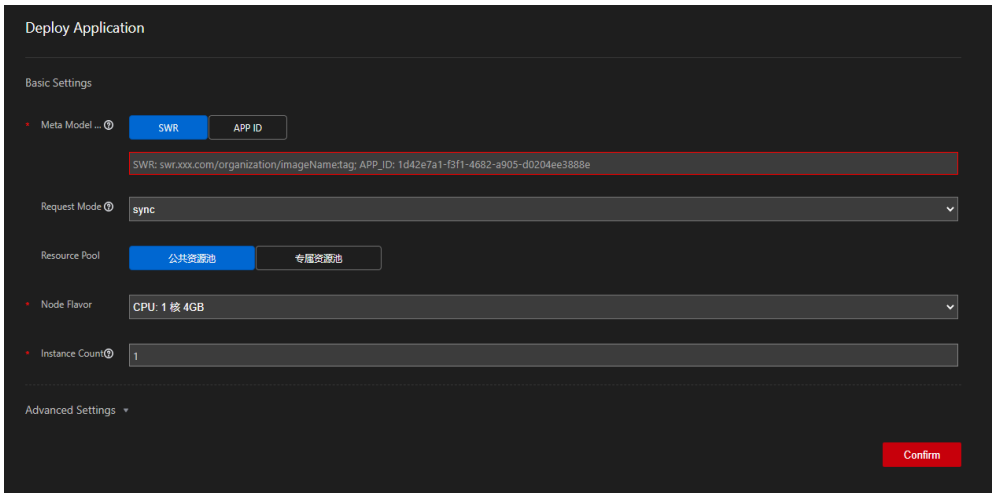
图 4-9 发布 AI 应用



3. 部署在线服务

   鼠标右键工程列表中当前工程，单击“Deploy”部署在线服务。选择元模型来源、请求模式、资源池、资源规格以及实例数。单击“Confirm”将会执行该作业。单击高级选项可以配置更加丰富的选项。

图 4-10 部署在线服务



   执行成功后，服务ID会打印在控制台日志。

4. 查看服务日志

   如果在上一步部署在线服务的步骤中，开启了日志收集功能，可以通过如下命令查看服务运行的日志。 service_id 指定服务ID，start_time指定日志搜索起始时间，end_time指定日志搜索结束时间。单击菜单中的Terminal> New Terminal新建终端窗口，输入如下命令查看该服务日志：

   ```
   python manage.py list-logs --service_id xxx --start_time '2021-12-10 14:39:30' --end_time '2021-12-10 14:59:30'
   ```

   执行成功后，将在控制台打印从start_time到end_time LTS服务收集的服务日志信息。

5 将 AI 应用发布到 ModelArts 模型管理

AI应用开发并调试完成后，开发者可以将AI应用服务到ModelArts AI应用管理模块，然后在ModelArts的推理平台进行应用的生产部署。发布AI应用包含了应用打包、构建镜像和发布这三个步骤。

- **打包AI应用**
- **构建镜像**
- **发布和部署AI应用**

打包 AI 应用

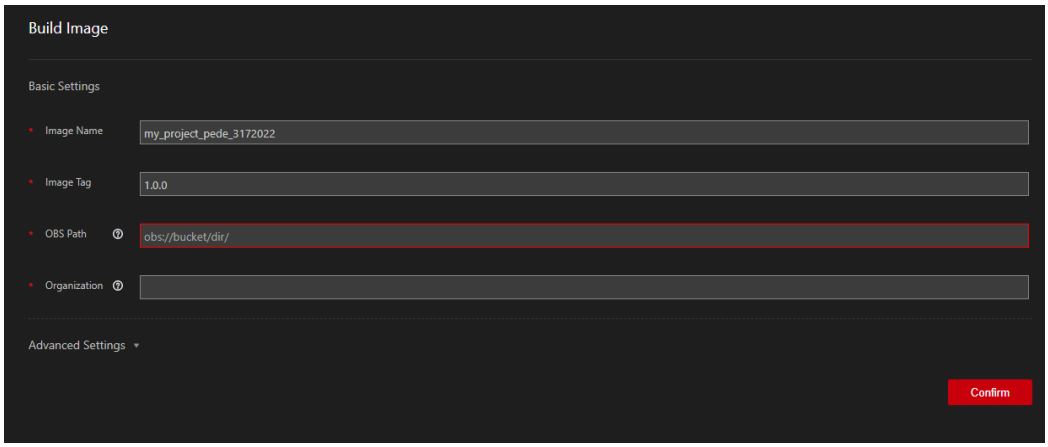
鼠标右键工程列表中当前工程，单击“Package”将会打包当前工程，生成该工程的rpm包。

构建镜像

将AI应用打好的rpm包和基础镜像打成新的应用镜像，用于部署。生产部署时，Dockerfile中配置的基础镜像需要选择runtime镜像。

鼠标右键工程列表中的当前工程，单击“Build Image”，输入镜像名称、镜像版本、OBS地址。

图 5-1 构建镜像



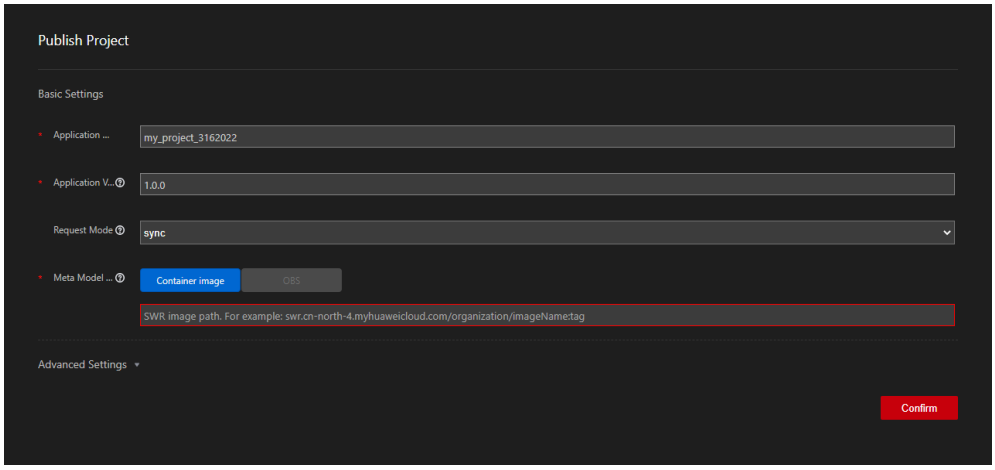
镜像构建首次执行大约耗时5~8min，执行成功后镜像将被推送到华为云SWR服务的组织中，SWR地址会打印在terminal控制台。命令执行结束后，镜像构建的日志会在下载至工程log目录下docker-build.log文件中，可通过日志查看构建过程信息。

发布和部署 AI 应用

1. 发布AI应用

   鼠标右键工程列表中当前工程，单击“Public”，输入应用名称、应用版本、请求方式以及镜像地址。单击高级选项可以配置更加丰富的选项。

   图 5-2 发布 AI 应用

2. 部署AI应用

   AI应用发布成功后，进入“部署上线>在线服务”模块进行部署操作，部署时选择发布的AI应用进行部署即可。控制台上提供了比开发环境的命令行更为丰富的部署选项，例如服务流量限制、服务自动停止等，用户可以根据需要进行配置，具体可参考**部署为在线服务**。