ALIBABA CLOUD

阿里云

机器学习PAI

PAI-EAS模型在线服务

文档版本：20210528

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.概述

为实现一站式算法应用，PAI针对在线推理场景提供了在线预测服务PAI-EAS（Elastic Algorithm Service），支持基于异构硬件（CPU和GPU）的模型加载和数据请求的实时响应。

通过PAI-EAS，您可以将模型快速部署为RESTful API，再通过HTTP请求的方式调用该服务。PAI-EAS提供的弹性扩缩和蓝绿部署等功能，可以支撑您以较低的资源成本获取高并发且稳定的在线算法模型服务。同时，PAI-EAS还提供了资源组管理、版本控制及资源监控等功能，便于将模型服务应用于业务。

计费

PAI-EAS支持将模型服务部署在公共资源组或专属资源组。公共资源组中，根据每个模型服务占用的资源量计费。专属资源组中，根据资源组管理的服务器资源包年包月或按量计费。PAI-EAS的定价和计费规则请参见PAI-EAS计费说明。

地域

PAI-EAS支持的地域包括华北2（北京）、华东2（上海）、华东1（杭州）、华南1（深圳）、中国（香港）、新加坡（新加坡）、印度尼西亚（雅加达）、印度（孟买）、美国（弗吉尼亚）及德国（法兰克福）。

名词解释

名词	描述
资源组	PAI-EAS将集群资源分为不同的资源组进行隔离，创建模型服务时，您可以选择将其部署在默认的公共资源组或自己额外购买的专属资源组。
模型服务	模型文件和在线预测逻辑部署成的常驻服务。您可以对模型服务进行创建、更新、停止、启动、扩容及缩容操作。
模型文件	通过离线训练获得的离线模型。基于不同框架会得到不同格式的模型，通常与Processor一起部署，从而获得模型服务。
Processor	包含在线预测逻辑的程序包，通常与模型文件一起部署，从而获得模型服务。针对常用的PMML、TensorFlow（Saved Model）及Caffe模型，PAI-EAS提供了预置的官方Processor。
自定义processor	PAI-EAS预置Processor无法满足所有的服务部署需求，您可以通过自定义Processor，实现更灵活地服务部署。PAI-EAS支持通过C++、Java或Python开发自定义Processor。
服务实例	每个服务可以部署多个服务实例以提高能够支持的并发请求数。如果资源组中有多台服务器资源，PAI-EAS会自动将不同实例分布至不同服务器，从而更好地保障服务高可用性。
高速直连	专属资源组和您自己的VPC连通后，在VPC中，可以通过客户端直接访问服务的每个实例。

资源组管理

PAI-EAS不仅支持将模型部署至系统提供的公共资源组，而且支持用户创建并管理自己的专属资源组，详情请参见专属资源组。

模型部署方式

PAI-EAS支持的服务部署方式包括控制台上传模型、PAI-Studio一键部署、PAI-DSW部署及本地客户端部署，详情请参见服务部署。

模型服务管理

在PAI-EAS模型在线服务页面，您可以管理模型服务：

- 查看模型调用信息。
- 在线调试。
- 查看日志、监控及服务部署相关信息。
- 扩容、缩容、启动、停止及删除模型服务。

EASCMD

为了让更多环境的开发者使用便捷的模型服务部署功能，PAI-EAS支持通过EASCMD方式完成服务部署的所有操作，详情请参见命令使用说明。

相关说明

- 使用公网地址访问已部署的模型服务，需要开通阿里云API网关服务，其计费详情请参见计费概述。
- 使用公网地址访问的服务，必须在API网关中绑定自己的域名，否则每天调用服务的次数不能超过1000次。

2.计量计费

本文为您介绍PAI-EAS的计费方式。

PAI-EAS的计费详情请参见PAI-EAS计费说明。

3.开通及购买

PAI-EAS支持将模型服务部署在公共资源组或专属资源组。本文为您介绍如何开通PAI-EAS及购买专属资源组。

背景信息

公共资源组中，系统根据每个模型服务占用的资源量计费。专属资源组中，系统根据资源组管理的服务器资源包年包月或按量计费。PAI-EAS的定价和计费规则请参见PAI-EAS计费说明。

对于两种部署方式，其使用条件如下。

部署方式	计费主体	计费方式	使用条件
公共资源组	模型服务运行时长（模型服务占用公共资源的时长）	后付费（按量计费）	开通PAI-EAS
专属资源组	资源组运行时长	后付费（按量计费）	开通PAI-EAS并购买（创建）专属资源组
		预付费（包年包月）	

开通PAI-EAS

开通PAI-EAS需要开通PAI的基础功能，即**PAI（Studio、DSW、EAS）后付费**，详情请参见开通。

购买专属资源组

购买专属资源组之前，必须开通PAI-EAS，详情请参见开通PAI-EAS。

1. 进入**PAI EAS模型在线服务**页面。
   i. 登录PAI控制台。
   ii. 在左侧导航栏，选择**模型部署 > 模型在线服务（EAS）**。
2. 购买（创建）专属资源组。您可以根据实际需要，购买（创建）预付费专属资源组或后付费专属资源组：
   - 购买（创建）预付费专属资源组：
     a. 在**PAI EAS模型在线服务**页面，单击**新建资源组 > 新建预付费资源组（包年包月）**。

b. 在购买页面，配置参数。

参数	描述
商品类型	选择**PAI-EAS资源组预付费**。
地域和可用区	系统支持的地域包括： ■ **中国** 　■ **华东1（杭州）** 　■ **华东2（上海）** 　■ **华北2（北京）** 　■ **华南1（深圳）** 　■ **中国（香港）** ■ **亚太** 　■ **新加坡（新加坡）** 　■ **印度尼西亚（雅加达）** ■ **欧洲与美洲** 　■ **德国（法兰克福）** 　■ **美国（弗吉尼亚）** ■ **中东与印度**仅支持**印度（孟买）**
节点规格	系统支持的GPU和CPU规格。 **说明** 在不同地域，系统支持的**节点规格**不同。
机器数量	取值范围：1~1000。
购买时长	取值范围：1个月~11个月、1年或3年。
到期自动续费	到期后，是否自动续费。

c. 单击**立即购买**。

支付成功后，系统自动分配资源。您可以在**PAI EAS模型在线服务**页面，查看已创建的所有专属资源组。

说明 如果因为当前地域资源不足等原因，导致专属资源组创建失败，则系统自动创建退款订单，已支付的费用会原路返回。

- 购买（创建）后付费专属资源组：

a. 在**PAI EAS模型在线服务**页面，单击**新建资源组 > 新建后付费资源组（计时付费）**。

b. 在购买页面，配置参数。

参数	描述
商品类型	选择**PAI-EAS资源组后付费**。
机器资源	系统支持的GPU和CPU规格。 **说明** 在不同地域，系统支持的**机器资源**不同。
地域	系统支持的地域包括： ■ **华东1（杭州）** ■ **华东2（上海）** ■ **华北2（北京）** ■ **华南1（深圳）** ■ **中国（香港）** ■ **新加坡（新加坡）** ■ **印度尼西亚（雅加达）** ■ **印度（孟买）** ■ **德国（法兰克福）** ■ **美国（弗吉尼亚）**
机器数量	取值范围：1~100。

c. 单击**立即购买**。

支付成功后，系统自动分配资源。您可以在**PAI EAS模型在线服务**页面，查看已创建的所有专属资源组。

4.授权

如果使用RAM用户管理模型服务或专属资源组，则需要主账号对其授权。本文介绍如何为RAM用户授权PAI-EAS操作权限、财务权限及模型部署权限。

背景信息

PAI-EAS中涉及的权限点如下。

权限点类型	权限点名称	权限点内容
模型服务权限	eas:EditInstance	写权限，可以创建、更新及删除模型服务。
	eas:ListInstance	列举权限。在PAI-EAS控制台概览页面，可以列举模型服务。
	eas:ReadInstance	读权限，可以查看模型服务监控、日志、服务地址及在线调试。
	eas:OperateInstance	操作权限，可以启动模型服务、停止模型服务及切换流量。
资源组权限	eas:ListResourceGroup	列举权限。在PAI-EAS控制台概览页面，可以列举资源组。部署服务时，可以列举并使用资源组。
	eas:ReadResourceGroup	读权限，可以进入资源组详情页面，并查看资源组信息（包括服务器数量、型号及服务状态等）。
	eas:OperateResourceGroup	操作权限，可以创建（购买）或删除资源组、续费（后付费资源组）、扩容或缩容（后付费资源组）及开通或关闭资源组的VPC直连功能。

权限策略的释义：

- 权限策略是阿里云对RAM用户进行授权的基本单位。
- 权限策略是权限点的父概念，即一个权限策略可以包括单个或多个权限点。
- 主账号根据权限策略名称区分不同的权限策略，并为RAM用户进行权限策略授权。
- 阿里云支持系统权限策略（通用权限策略）和用户自定义权限策略（根据特定阿里云产品的具体需求，自定义策略）。PAI-EAS的权限策略属于自定义策略，需要用户自己定义。

针对不同的使用场景，需要为RAM用户授权的权限不同：

- 如果RAM用户需要购买（创建）专属资源组，则需要为其授权资源组的操作权限和财务权限，详情请参见为RAM用户授权PAI-EAS操作权限和为RAM用户授权财务权限。
- 如果RAM用户需要管理模型服务部署，则需要为其授权资源组的操作权限，并且手动将RAM用户的AccessKey绑定至数加租户系统，详情请参见为RAM用户授权PAI-EAS操作权限和为RAM用户授权模型服务部署权限。

为RAM用户授权PAI-EAS操作权限

因为PAI-EAS的权限策略属于特定产品的权限策略，所以必须先创建自定义权限策略，再将其授权给RAM用户。

1. 登录RAM控制台。
2. 创建自定义权限策略。
    i. 在左侧导航栏，选择**权限管理 > 权限策略管理**。

ii. 在**权限策略管理**页面，单击**创建权限策略**。

iii. 在**新建自定义权限策略**页面，配置参数。

参数	描述
策略名称	建议根据实际需要的权限点和业务命名策略。
备注	描述信息，便于区分各权限策略。
配置模式	单击**脚本配置**。
策略内容	参见PAI-EAS涉及的权限点，定义**策略内容**。一个权限策略可以包含单个或多个权限点。 **注意** 请根据RAM用户需要使用的权限，谨慎定义权限策略。 例如，定义RAM用户模型服务权限和资源组权限，可以配置**策略名称**为Model&ResourceGroup，并配置**策略内容**如下。阿里云账号为RAM用户授权Model&ResourceGroup策略后，RAM用户就拥有了模型服务和资源组的所有权限。 `{ "Statement": [ { "Effect": "Allow", "Action": [ "eas:ReadInstance", "eas:ListInstance", "eas:EditInstance", "eas:OperateInstance", "eas:ListResourceGroup", "eas:ReadResourceGroup", "eas:OperateResourceGroup" ], "Resource": "*" } ], "Version": "1" }`

iv. 单击**确定**。

3. 为RAM用户授权。

i. 在左侧导航栏，选择**人员管理 > 用户**。

ii. 在**用户**页面，单击待授权RAM用户**操作**列下的**添加权限**。

iii. （可选）在**添加权限**面板，配置参数。

参数	描述
授权应用范围	单击**整个云账号**。
被授权主体	系统自动填入，通常无需修改。
选择权限	a. 单击**自定义策略**。 b. 在左侧**权限策略名称**列表，单击已定义的权限策略（例如Model_R&W），该策略会显示在右侧**已选择**列表。

iv. 单击**确定**。

为RAM用户授权财务权限

如果RAM用户购买（创建）专属资源组，则不仅需要授权资源组的操作权限eas:OperateResourceGroup，而且需要授权财务权限AliyunFinanceConsoleFullAccess，否则RAM用户无法下单付款。因为财务权限为系统权限策略，所以无需自定义策略，直接进行授权即可。

1. 登录RAM控制台。
2. 在左侧导航栏，选择**人员管理 > 用户**。
3. 在**用户**页面，单击待授权RAM用户**操作**列下的**添加权限**。
4. 在**添加权限**面板，配置参数。

参数	描述
授权应用范围	单击**整个云账号**。
被授权主体	系统自动填入，通常无需修改。
选择权限	i. 单击**系统策略**。 ii. 在左侧**权限策略名称**列表，单击AliyunFinanceConsoleFullAccess权限策略，该策略会显示在右侧**已选择**列表。

5. 单击**确定**。

为RAM用户授权模型服务部署权限

如果RAM用户管理模型服务部署，不仅需要为RAM用户授权自定义权限策略，而且还需要手动将RAM用户的AccessKey绑定至数加租户系统。主账号可以先在**用户管理**中查看待授权RAM用户的AccessKey，再将该AccessKey告知RAM用户。

1. 主账号查看待授权RAM用户的AccessKey。
   i. 主账号登录RAM控制台。
   ii. 在左侧导航栏，选择**人员管理 > 用户**。
   iii. 在**用户**页面，单击待授权RAM用户**用户登录名称/显示名称**列下的用户名称。

iv. 在用户详细信息页面的**用户AccessKey**区域，查看RAM用户的AccessKey。

如果待授权RAM用户没有AccessKey，则单击**创建AccessKey**为其创建一个。

2. RAM用户在数加控制台的个人信息中，绑定自己的AccessKey，详情请参见更新个人信息。

5.EAS资源组

5.1. 概述

PAI-EAS将集群资源分为不同的资源组进行隔离。新建模型服务时，您可以选择将模型服务部署在公共资源组或自己创建的专属资源组。

公共资源组和专属资源组对比如下。

公共资源组	专属资源组	
后付费（按量付费）	预付费（包年包月）	后付费（按量付费）
• 公共计算资源 • 支持CPU	• 独享专属计算资源（资源隔离更安全） • 支持CPU • 支持GPU（P4、P100、T4及V100卡） • 支持自定义Processor • 支持VPC高速直连	

5.2. 公共资源组

公共资源组提供的资源类型为CPU，本文为您介绍如何使用公共资源组部署模型服务及公共资源组的计费方式。

使用说明

您可以通过以下方式将模型服务部署至公共资源组：

- 控制台的可视化方式：在**资源和模型**面板，选择**资源组种类**为**公共资源组**，详情请参见部署指导。
- EASCMD方式：在resource字段指定资源为CPU，详情请参见命令使用说明。

计费

模型服务一旦部署并处于**运行中**，系统就开始计费，详情请参见PAI-EAS计费说明。

> **说明** 建议及时停止无用的模型服务，以免产生不必要的费用。

停止计费

在**PAI EAS 模型在线服务**页面的服务列表区域，单击待停止服务**操作**列下的**停止**，即可停止模型服务和计费。

> **说明** 请确保被停止的服务不需要再使用，以免造成不必要的业务损失。

5.3. 专属资源组

专属资源组包括预付费资源组和后付费资源组。本文为您介绍如何管理两种类型的专属资源组，包括创建、扩容、临时扩容、续费及开通VPC高速直连。此外，两种资源组的生命周期和计费方式均不相同，您需要根据使用场景，选择合适类型的资源组。

背景信息

相比公共资源组，专属资源组具有以下优势：

- 提供预付费和后付费专属资源组，您可以根据需要灵活选择资源组类型。
- 支持通过自定义Processor的方式，部署更多类型的模型。
- 可以和您自己的VPC连通，从而进行高效无损的模型调用。

类别介绍

专属资源组按照付费方式分为预付费专属资源组和后付费专属资源组，简称预付费资源组和后付费资源组 。两者的使用区别如下。

类型	预付费资源组	后付费资源组
付费方式	包年包月	按量计费
资源组操作	扩容、续费及临时扩容	扩容、缩容及停用

创建资源组

1. 进入**PAI EAS模型在线服务**页面。
   i. 登录PAI控制台。
   ii. 在左侧导航栏，选择**模型部署 > 模型在线服务（EAS）**。
2. 创建资源组。您可以根据实际需要，创建预付费资源组或后付费资源组：
   - 创建预付费资源组：
     a. 在**PAI EAS模型在线服务**页面的上方，选择**新建资源组 > 新建预付费资源组（包年包月）**。

b. 在购买页面，配置参数。

参数	描述
商品类型	选择**PAI-EAS资源组预付费**。
地域和可用区	系统支持的地域包括： ■ **中国** ■ **华东1（杭州）** ■ **华东2（上海）** ■ **华北2（北京）** ■ **华南1（深圳）** ■ **中国（香港）** ■ **亚太** ■ **新加坡（新加坡）** ■ **印度尼西亚（雅加达）** ■ **欧洲与美洲** ■ **德国（法兰克福）** ■ **美国（弗吉尼亚）** ■ **中东与印度**仅支持**印度（孟买）**
节点规格	系统支持的GPU和CPU规格，详细的资源列表请参见资源类型。 **说明** 在不同地域，系统支持的**节点规格**不同。
机器数量	取值范围：1~1000。
购买时长	取值范围：1个月~11个月、1年或3年。
到期自动续费	到期后，是否自动续费。

c. 单击**立即购买**。

支付成功后，系统自动分配资源。您可以在**PAI EAS模型在线服务**页面，查看已创建的所有资源组或修改资源组名称。

说明 如果因为当前地域资源不足等原因，导致资源组创建失败，则系统自动创建退款订单，已支付的费用会原路返回。

- 创建后付费资源组：

a. 在**PAI EAS模型在线服务**页面的上方，选择**新建资源组 > 新建后付费资源组（计时收费）**。

b. 在购买页面，配置参数。

参数	描述
商品类型	选择**PAI-EAS资源组后付费**。
机器资源	系统支持的GPU和CPU规格，详细的资源列表请参见资源类型。 **说明** 在不同地域，系统支持的**机器资源**不同。
地域	系统支持的地域包括： ■ **华东1（杭州）** ■ **华东2（上海）** ■ **华北2（北京）** ■ **华南1（深圳）** ■ **中国（香港）** ■ **新加坡（新加坡）** ■ **印度尼西亚（雅加达）** ■ **印度（孟买）** ■ **德国（法兰克福）** ■ **美国（弗吉尼亚）**
机器数量	取值范围：1~100。

c. 单击**立即购买**。

支付成功后，系统自动分配资源。您可以在**PAI EAS模型在线服务**页面，查看已创建的所有资源组或修改资源组名称。

查看资源组详情

1. 在**PAI EAS模型在线服务**页面，单击右上方的**查看全部资源组**。
2. 在**资源组列表**页面，单击待查看资源组的**资源组ID/名称**。
3. 在资源组详情页面，您可以进行以下操作。

操作	描述
查看资源组ID	资源组ID是资源组的唯一标识。通过EASCMD方式部署模型服务时，需要使用该参数。 

操作	描述
编辑资源组名称	单击资源组名称后面的图标，可以编辑资源组名称，建议使用具有业务含义的名称。通过控制台可视化方式部署模型服务时，需要根据资源组名称选择使用的资源组。
查看资源组中的服务器	单击**机器列表**页签。
查看部署在当前资源组的服务列表	单击**服务列表**页签。

管理预付费资源组

- 扩容

  在资源组的有效期内，提高该资源组的服务器数量。扩容的服务器与原购买的服务器到期时间相同，即原服务器到期后，扩容的服务器也会到期，不能继续使用。具体的扩容操作如下：

  i. 在**PAI EAS 模型在线服务**页面的资源组区域，单击预付费资源组下的**扩容**。

  ii. 在**变配**页面，选择**地域和可用区**，并配置**机器数量**。

  iii. 选中**PAI-EAS预付费服务协议**复选框，其他参数使用默认值。

  iv. 单击**立即购买**。

- 续费

  增加指定资源组的使用时长。例如续费1个月，即将预付费资源组的使用时长延长1个月。具体的续费操作如下：

  i. 在**PAI EAS 模型在线服务**页面的资源组区域，单击预付费资源组下的**服务续费**。

  ii. 在**续费**页面，选择**购买时长**，并选中**PAI-EAS预付费服务协议**复选框，其他参数使用默认值。

  iii. 单击**立即购买**。

  > **说明** 预付费资源组不支持删除操作。如果您需要释放未到期的预付费资源组，请提交工单。

- VPC高速直连

  通过弹性网卡ENI（Elastic Network Interface）将预付费资源组与您在当前地域的专有网络连通，开通该功能的具体方法请参见VPC高速直连。

- 临时扩容

如果在线服务处于业务调用高峰期，则可以通过购买临时服务器以提高资源配置量，从而确保服务稳定运行。调用高峰期结束后，可以随时停止使用临时服务器。临时扩容的规则包括：

- 临时扩容的服务器采用后付费方式，即按照使用时长计费。
- 业务需求降低后，您可以随时释放临时服务器，进而系统对其停止计费。
- 如果您未执行释放操作，则该临时服务器将与资源组中的预付费服务器共享生命周期，即同时到期停止。
- 临时服务器使用期间，请保证您的账户余额充足，以免欠费导致停机。账户欠费后的停机策略请参见后付费资源组生命周期。

临时扩容的具体操作如下：

i. 在**PAI EAS 模型在线服务**页面的资源组区域，单击预付费资源组下的**临时扩容**。
ii. 在**临时机器购买**页面，配置**机器数量**，其他参数使用默认值。
iii. 单击**确认扩容**。

释放临时资源的操作如下：

i. 在资源组详情页面，单击**机器列表**页签。
ii. 在**临时机器购买**页面，配置**机器数量**，其他参数使用默认值。
iii. 单击待释放临时服务器**操作**列下的**释放**。

管理后付费资源组

- 扩容

  提高指定资源组服务器数量，具体操作如下：

  i. 在**PAI EAS 模型在线服务**页面的资源组区域，单击后付费资源组下的**扩容**。
  ii. 在**变配**页面，配置**机器数量**，选中**PAI-EAS后付费服务协议**复选框，其他参数使用默认值。
  iii. 单击**立即购买**。

- 缩容或停用

  降低指定资源组的服务器数量，如果数量降为0，则资源组停止使用。具体操作如下：

  i. 在**PAI EAS 模型在线服务**页面的资源组区域，单击后付费资源组下的**缩容/停用**。
  ii. 确认待释放的资源组无需再使用后，单击**缩容/停用**。
  iii. 在**降配**页面，配置**机器数量**，选中**PAI-EAS后付费服务协议**复选框，其他参数使用默认值。

  > **说明** 如果需要停用后付费资源组，则配置**机器数量**为0。

  iv. 单击**立即购买**。

- 删除

  对于停用的后付费资源组，如果不再使用，可以将其删除。具体操作如下：

  i. 在**PAI EAS 模型在线服务**页面的资源组区域，单击后付费资源组下的**删除**。

ii. 在**删除资源组**对话框，单击**确认删除**。

- VPC高速直连

  通过弹性网卡ENI（Elastic Network Interface）将后付费资源组与您在当前地域的专有网络连通，开通该功能的具体方法请参见VPC高速直连。

计费

预付费资源组是在创建资源组前，预先付费。后付费资源组是成功创建资源组且资源组状态为**运行中**，才开始计费，详情请参见PAI-EAS计费说明。

停止或减少后付费资源组计费的方法，请参见管理后付费资源组的缩容或停用部分。

RAM用户操作权限及授权管理

如果RAM用户创建（购买）、查看及管理资源组，则需要主账号对其授权。具体授权方式请参见授权。

RAM用户创建（购买）资源组，还需要额外增加该用户的财务权限，否则无法下单付款，详情请参见为RAM用户授权财务权限。

预付费资源组生命周期

对于预付费资源组，您可以选择到期不续费、到期自动续费或到期手动续费。如果您需要继续使用PAI服务，建议提前续费或选择自动续费。请确保您绑定续费的账户余额充足，如果账户余额不足以支付预付服务费，则无法进行续费。如果资源组到期未续费或续费不成功，则资源组将被停止使用。如果在自到期之日起的15个自然日（15X24=360小时）内续费，即可恢复使用该资源组。如果自到期之日起的15个自然日（15X24=360小时）内仍未续费，则实际欠费满15个自然日后，系统终止该资源组的使用，同时删除该资源组和全部相关数据。

后付费资源组生命周期

后付费资源组是按量计费，一旦创建，就开始计费。系统根据资源组的使用时长，按照计费周期生成账单并扣费。如果无需继续使用该资源组，建议及时停用资源组以结束计费。使用后付费资源组期间，请确保账户可用余额大于或等于上一个计费周期的账单金额，以便阿里云成功扣费，否则您将处于欠费状态。

阿里云提供延停权益，即当按量付费的资源发生欠费后，提供一定额度或时长继续使用云服务的权益。延停期间正常计费。延停的权益额度不是欠费总额的上限。您延停的额度或时长根据您在阿里云的历史消费等因素，每个月自动计算并更新。更多信息，请参见延期免停权益。

如果自欠费之日起15个自然日（15X24=360小时）内充值并结清账单，则可以恢复使用资源组，相关数据仍然得以保留。如果自欠费之日起15个自然日（15X24=360小时）内未结清账单，则实际欠费满15个自然日后，系统终止该后付费资源组的使用，同时删除该资源组及全部相关数据。

资源类型

PAI-EAS支持使用14种类型的资源创建资源组（各地域有差异），各资源的具体配置如下。

机器型号	GPU配置	CPU配置
ecs.c5.6xlarge（仅预付费）	无	24核+48 GB
ecs.g5.6xlarge（仅预付费）	无	24核+96 GB
ecs.g6.4xlarge（仅后付费）	无	16核+64 GB
ecs.g6.6xlarge（仅后付费）	无	24核+96 GB
ecs.gn5i-c4g1.xlarge	1 Nvidia Tesla P4卡	4核+16 GB
ecs.gn5i-c8g1.2xlarge	1 Nvidia Tesla P4卡	8核+32 GB
ecs.gn6i-c4g1.xlarge	1 Tesla T4卡	4核+15 GB
ecs.gn6i-c8g1.2xlarge	1 Tesla T4卡	8核+31 GB
ecs.gn6i-c16g1.4xlarge	1 Tesla T4卡	16核+62 GB
ecs.gn6i-c24g1.6xlarge	1 Tesla T4卡	24核+93 GB
ecs.gn5-c4g1.xlarge	1 NVIDIA P100卡	4核+30 GB
ecs.gn5-c8g1.2xlarge	1 NVIDIA P100卡	8核+60 GB
ecs.gn5-c28g1.7xlarge	1 NVIDIA P100卡	28核+112 GB
ecs.gn6v-c8g1.2xlarge	1 NVIDIA V100卡	8核+32 GB

5.4. VPC高速直连

在专属资源组中，PAI-EAS支持VPC高速直连功能。您购买专属资源组后，PAI-EAS会使用指定规格的服务器资源，构成该资源组，且该专属资源组归属于PAI-EAS的VPC环境。默认情况下，您不能在自己的VPC环境中直接访问该专属资源组，必须手动开通VPC高速直连功能。系统会将您VPC环境中的弹性网卡挂载至PAI-EAS服务器，从而实现两个VPC网络互通。

原理

实现VPC高速直连的原理包括网络连通和服务发现：

- 网络连通

  为PAI-EAS授权后，系统会在您指定的交换机（vSwitch）和安全组（SecurityGroup）中创建弹性网卡（免费）。因为弹性网卡会占用交换机中的网段地址，所以请确保交换机的剩余网段IP充裕。对于专属资源组中的实例，PAI-EAS会为每个实例创建一张弹性网卡，并将其与该实例绑定，从而实现在您的VPC中访问PAI-EAS VPC中的实例。

- 服务发现

在专属资源组中部署服务后，系统会根据您申请的资源数量，为该服务创建对应的PAI-EAS实例（区别于专属资源组实例，可以理解为一个进程）。系统为每个PAI-EAS实例在专属资源组节点自动分配一个端口，您可以通过专属资源组节点挂载的弹性网卡IP和该端口号，直接访问服务进程。PAI-EAS提供服务发现机制，您可以定期查询或更新服务所对应的IP:PORT列表，详情请参见VPC高速直连调用。

优势

开通VPC高速直连后，无需通过网关访问服务，避免了四层SLB和七层网络转发，您可以在VPC中直接访问PAI-EAS实例。同时，PAI-EAS预置的RPC实现了HTTP相关协议栈，对于高QPS（Queries Per Second）的大流量服务（例如图像服务），可以大幅度提高访问性能、降低访问延时。

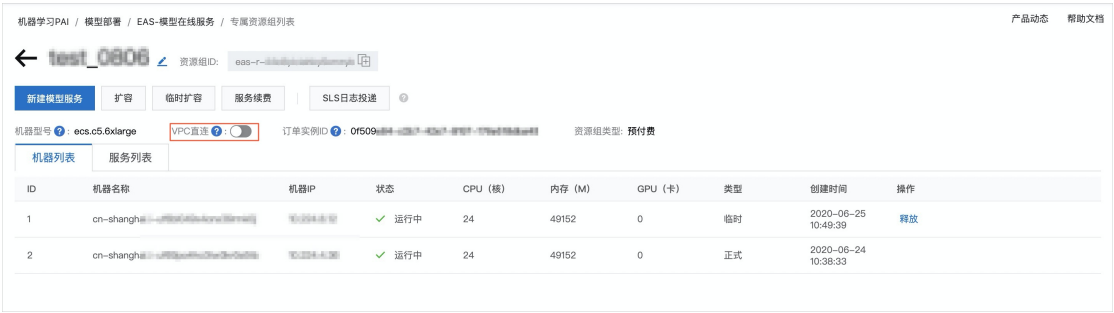
劣势

相比于网关模式，VPC高速直连模式为了提升性能，绕过了四层SLB和七层网关转发，从而舍弃了服务端负载均衡和容错功能。您需要在客户端实现相关的负载均衡和重试算法，导致服务测试和调试的难度增加。PAI-EAS将提供相关的配置SDK，帮助您完成客户端访问。

开通VPC高速直连

开通VPC高速直连，需要为PAI-EAS授权相关权限（参见授权），并指定待连通VPC环境中的交换机（vSwitch）和安全组（SecurityGroup）。网络连通后，该VPC环境中的ECS服务器即可通过创建的弹性网卡访问PAI-EAS专属资源组内的服务器。

1. 进入**PAI EAS模型在线服务**页面。
   i. 登录PAI控制台。
   ii. 在左侧导航栏，选择**模型部署 > 模型在线服务（EAS）**。
2. 在**PAI EAS模型在线服务**页面，单击右上方的**查看全部资源组**。
3. 在**资源组列表**页面，单击待查看资源组的**资源组ID/名称**。
4. 在资源组详情页面，开通VPC高速直连。
   i. 打开**VPC直连**开关。



   ii. 在**开通VPC直连通道**对话框，选择**VPC**、**交换机**及**安全组名称**。
   iii. 单击**确认**。

调用服务

如果专属资源组开通了VPC高速直连，则可以通过VPC高速直连访问部署在该专属资源组的模型服务，详情请参见VPC高速直连调用。

6.模型服务部署及管理

6.1. 服务部署

获得训练好的模型后，您可以使用PAI-EAS快速将其部署为RESTful API，实现模型在线服务。针对不同方式获取的训练模型，PAI-EAS支持不同的部署方式，且支持为已部署的服务增加服务版本。本文介绍如何使用多种方式进行服务部署及增加服务版本。

前提条件

已获得训练好的模型。

背景信息

PAI-EAS支持以下多种部署方式，以满足您部署通过各种方式获取的训练模型的需求：

- 控制台上传部署

  如果训练好的模型存储在本地、OSS或公网地址，您可以在**PAI EAS模型在线服务**页面上传该模型，将其部署为在线服务。

- PAI-Studio一键部署

  如果您使用PAI-Studio训练模型，可以在实验画布的上方选择**部署 > 模型在线部署**，将训练完成的模型一键部署为在线服务。

- 本地客户端部署

  通过客户端工具EASCMD，您可以在自己服务器上对模型服务进行管理，包括创建、查看、删除及修改服务。

- PAI-DSW部署

  由于PAI-DSW已预置EASCMD客户端，因此使用PAI-DSW训练的模型可以直接部署为在线服务。

对于已经部署的服务，PAI-EAS支持通过以下方式对其增加服务版本：

- 通过控制台增加服务版本，详情请参见增加已有服务版本。
- 通过本地客户端工具的 `modify` 命令，增加服务版本，详情请参见增加服务版本的命令。

控制台上传部署

在**PAI EAS模型在线服务**页面，可以直接上传训练完成的模型，并将其部署为在线模型服务。

1. 进入**PAI EAS模型在线服务**页面。
   i. 登录PAI控制台。
   ii. 在左侧导航栏，选择**模型部署 > 模型在线服务（EAS）**。
2. 在**PAI EAS模型在线服务**页面，单击**模型上传部署**。
3. 在**资源和模型**面板，配置参数。

区域	参数	描述
资源组	**资源组种类**	支持使用公共资源组或已购买（创建）的专属资源组部署模型服务。

区域	参数	描述
模型	Processor种类	**资源组种类**为公共资源组时，支持除Blade以外的预置官方Processor（详情请参见预置Processor使用说明），不支持自定义Processor。 **资源组种类**为专属资源组时，支持所有的预置官方Processor（详情请参见预置Processor使用说明）和自定义Processor。
	资源种类	仅**资源组种类**为**公共资源组**时，该参数生效。
	Processor语言	仅**Processor种类**为**自定义processor**时，该参数生效。支持cpp、java及python。
	Processor包	仅**Processor种类**为**自定义processor**时，该参数生效。您可以通过以下任何一种方式配置该参数： ○ 在**Processor包**后面的文本框，输入可以公开访问的URL地址。 ○ 单击**Processor包**后的**上传本地文件**，并根据提示上传已下载的Processor文件。 系统会将文件上传至当前地域的官方OSS路径，并自动配置**Processor包**。 **说明** 通过本地上传的方式，可以使系统在模型部署时，快速加载Processor。
	Processor主文件	仅**Processor种类**为**自定义processor**时，该参数生效。自定义Processor包的主文件。
	模型文件	您可以通过以下任何一种方式配置该参数： ○ **本地上传**： a. 单击**本地上传**。 b. 单击**上传本地文件**，并根据提示上传本地模型文件。 ○ **OSS文件导入**。 单击**OSS文件导入**，并下方列表中选择模型文件所在的OSS路径。 ○ **公网下载地址**。 单击**公网下载地址**，并在下方文本框中输入可以公开访问的URL地址。

4. 单击**下一步**。
5. 在**部署详情及配置确认**面板，配置参数。
   i. 单击**新建服务**。PAI-EAS支持以下两种在线部署方式：
   - **新建服务**：表示新部署一个服务，本文使用该方式部署模型。
   - **新增蓝绿部署**：为已有的服务新增一个关联服务，可以配置两个服务的流量分配。详细的操作方法请参见新增蓝绿部署。

ii. 输入**自定义模型名称**，并配置相关参数。

参数		描述
模型部署占用资源	**实例数**	建议配置多个服务实例，以避免单点部署带来的风险。
	Quota	仅**资源组种类**为公共资源组时，该参数生效。1 Quota等于1核加4 GB内存，**Quota**取值范围为1~100。
	核数	仅**资源组种类**为专属资源组时，该参数生效。
	内存数（M）	仅**资源组种类**为专属资源组时，该参数生效。

> **说明**
> - 单实例内的CPU、GPU及内存需要位于同一台机器上。如果资源不够，则会导致部署失败。
> - 对于高稳定性要求的正式服务，建议使用包含多台机器的资源组，并部署多个服务实例。

iii. 单击**部署**。

PAI-Studio一键部署

PAI-Studio是PAI经典的可视化建模平台，训练完成的模型可以通过画布上方的**部署 > 模型在线部署**，一键部署至PAI-EAS，详情请参见基于实验模板快速构建实验。

PAI-Studio中可以一键部署至PAI-EAS的算法包括GBDT二分类、线性支持向量机、逻辑回归二分类、逻辑回归多分类、随机森林、KMeans、线性回归、GBDT回归（因为GBDT回归算法不支持INT类型数据输入，所以部署前请确保GBDT算法的输入为DOUBLE类型）及TensorFlow等。

本地客户端部署

通过客户端工具EASCMD，您可以在自己服务器上对模型服务进行管理，包括创建、查看、删除及修改服务。EASCMD客户端的下载及认证方式请参见下载并认证客户端，具体命令的使用方法详情请参见命令使用说明。

> **说明** 部署模型时，需要使用AccessKey授权，您可以登录用户信息管理控制台，查看AccessKey。

PAI-DSW部署

PAI-DSW是PAI针对深度学习推出的交互式云端开发环境，提供高性能GPU卡和开放的交互式编程环境。因为PAI-DSW已预置EASCMD客户端工具，所以PAI-DSW训练完成的模型可以直接部署至PAI-EAS，详细的命令使用方式请参见命令使用说明。

> **说明** 部署模型时，需要使用AccessKey授权，您可以登录用户信息管理控制台，查看AccessKey。

增加已有服务版本

1. 进入**PAI EAS模型在线服务**页面。

i. 登录PAI控制台。

ii. 在左侧导航栏，选择**模型部署 > 模型在线服务（EAS）**。

2. 在**PAI EAS 模型在线服务**页面，单击目标服务**操作**列下的**增加版本**。
3. 在**部署详情及配置确认**面板，配置参数。

参数	描述
选择已部署模型服务	系统自动带入，无需手动修改。
Processor种类	支持所有的预置官方Processor，详情请参见预置Processor使用说明。
模型文件	您可以通过以下任何一种方式配置该参数： ○ **本地上传**： a. 单击**本地上传**。 b. 单击**上传本地文件**，并根据提示上传本地模型文件。 ○ **OSS文件导入**。 单击**OSS文件导入**，并在下方列表中选择模型文件所在的OSS路径。 ○ **公网下载地址**。 单击**公网下载地址**，并在下方文本框中输入可以公开访问的URL地址。

4. 单击**部署**。
5. 部署完成后，从**PAI EAS模型在线服务**页面的**当前版本**列下的列表，选择需要切换的模型版本。

6.2. 客户端工具

6.2.1. 下载并认证客户端

您可以通过EASCMD客户端管理PAI-EAS服务。本文介绍如何下载客户端并对其进行用户认证。

操作步骤

1. 下载EASCMD客户端。各版本的客户端下载地址如下：
   - Linux 32版本
   - Linux 64版本
   - Mac 64版本

- Windows 64版本

2. （可选）在命令行中，将下载的客户端文件修改为可执行文件，示例如下。如果您下载的Windows 64版本，则可以跳过该步骤。

```
chmod +x <eascmd64>
```

其中<eascmd64>表示下载的客户端文件名，需要根据实际情况修改。

3. 使用阿里云账号的AccessKey进行身份认证。不同版本的认证命令存在差异：
   - 如果您下载的Windows 64版本，则在Windows的命令行中执行以下命令。

```
eascmdwin64.exe config -i <yourAccessKeyID> -k <yourAccessKeySecret> [-e Endpoint]
```

   - 如果您下载的其他版本，则在对应系统的命令行中执行以下命令。

```
./<eascmd64> config -i <yourAccessKeyID> -k <yourAccessKeySecret> [-e Endpoint]
```

需要根据实际情况替换以下参数。

参数	描述
<eascmd64>	根据系统环境下载的客户端文件名。
<yourAccessKeyID>	阿里云账号的AccessKey ID。
<yourAccessKeySecret>	阿里云账号的AccessKey Secret。
Endpoint	默认的PAI-EAS服务地域为华东2（上海），如果需要将模型部署至其它地域，可以使用-e参数指定地域对应的Endpoint（取值请参见地域与Endpoint的对应关系）。例如客户端文件名称为*eascmd64*，可以使用如下命令，将地域指定为华北2（北京）。 `./eascmd64 config -i <yourAccessKeyID> -k <yourAccessKeySecret> -e pai-eas.cn-beijing.aliyuncs.com`

地域与Endpoint的对应关系

地域	Endpoint
华东2（上海）	pai-eas.cn-shanghai.aliyuncs.com
华北2（北京）	pai-eas.cn-beijing.aliyuncs.com
华东2（上海）（从PAI-DSW内部署）	pai-eas-share.cn-shanghai.aliyuncs.com
华北2（北京）（从PAI-DSW内部署）	pai-eas-share.cn-beijing.aliyuncs.com
华北2（北京）（政务云，从PAI-DSW内部署）	pai-eas.cn-north-2-gov-1.aliyuncs.com
华东1（杭州）	pai-eas.cn-hangzhou.aliyuncs.com
华南1（深圳）	pai-eas.cn-shenzhen.aliyuncs.com
中国（香港）	pai-eas.cn-hongkong.aliyuncs.com

地域	Endpoint
新加坡（新加坡）	pai-eas.ap-southeast-1.aliyuncs.com
印度（孟买）	pai-eas.ap-south-1.aliyuncs.com
印度尼西亚（雅加达）	pai-eas.ap-southeast-5.aliyuncs.com
德国（法兰克福）	pai-eas.eu-central-1.aliyuncs.com
美国（弗吉尼亚）	pai-eas.us-east-1.aliyuncs.com

4. 认证成功后，系统输出如下类似结果。

```
Configuration saved to: /Users/test/.eas/config
```

6.2.2. 命令使用说明

您可以使用eascmd管理PAI-EAS服务。本文为您介绍如何使用eascmd客户端上传文件、创建服务、修改服务配置信息、切换服务版本、删除服务、查看服务列表、查看服务详细信息、查看服务进程及配置资源组网络的相关命令。

操作命令合集

使用eascmd命令行工具管理服务，相关的操作命令如下。

类型	功能	操作入口
服务相关	• 上传文件 • 创建服务 • 修改配置 • 修改服务配置 • 增加服务版本 • 删除服务 • 蓝绿发布 • 切换版本 • 查看服务列表 • 查看服务信息 • 查看服务进程	支持以下两种方式使用eascmd命令行工具： • 自行下载eascmd客户端，详细请参见下载并认证客户端。 • 在PAI-DSW的Terminal中，PAI-DSW已经内置了eascmd命令行工具。
资源组相关	• 查看资源组列表 • 查看资源组详情 • 查看资源组实例列表 • 配置资源组网络	

上传文件

- 功能

PAI-EAS为每位用户提供了OSS仓库，通过eascmd的 `upload` 命令，您可以直接上传模型或Processor，并获取上传后的OSS地址。

- 格式

```
eascmd upload <filename> [--inner]
```

- 参数
  - <filename>：待上传的文件名称。
  - [--inner]：如果在PAI-DSW内执行上传命令，则需要增加该参数。
- 示例

  在PAI-DSW内，将打包好的SavedModel模型文件*savedmodel_example/savedmodel_example.tar.gz*上传至OSS。

```
eascmd upload savedmodel_example/savedmodel_example.tar.gz --inner
```

  系统输出如下类似结果。

```
[OK] oss endpoint:  [http://oss-cn-shanghai.aliyuncs.com]
[OK] oss target path: [oss://eas-model-shanghai/182848887922****/savedmodel_example/savedmodel_example.tar.gz]
Succeed: Total num: 1, size: 33,013. OK num: 1(upload 1 files).
```

  其中 `oss://eas-model-shanghai/182848887922****/savedmodel_example/savedmodel_example.tar.gz` 为存储模型的OSS地址，可以用于服务部署。

创建服务

- 功能

  通过 `create` 命令创建服务。创建服务时，需要提供资源（模型或Processor）的HTTP或OSS地址，您可以将资源上传至OSS，并获取上传后的OSS地址。

- 命令

```
eascmd create <service_desc_json>
```

- 参数

  service_desc_json表示描述服务相关信息（模型存储位置及资源规格等）的JSON文件，该文件的示例如下。

```
{
 "name": "mnist_saved_model_example",
 "generate_token": "true",
 "model_path": "http://eas-data.oss-cn-shanghai.aliyuncs.com/models%2Fmnist_saved_model.tar.gz",
 "processor": "tensorflow_cpu_1.12",
 "metadata": {
  "instance": 1,
  "cpu": 1
 }
}
```

  服务相关信息JSON文件内的参数解释如下表所示。

参数	是否必选	描述
name	是	服务名称，必须在同一地域内唯一。
token	否	表示访问鉴权的Token字符串。如果未指定，则系统自动生成。
model_path	是	model_path与processor_path分别为模型和Processor的输入数据源地址，支持以下格式的地址： ○ HTTP地址：所需文件必须为TAR.GZ、TAR、BZ2或ZIP等压缩包。 ○ OSS地址：地址链接可以是具体文件路径或文件夹路径。同时，还需要提供参数oss_endpoint，示例如下。 `"model_path":"oss://wowei-beijing-tiyan/alink/", "oss_endpoint":"oss-cn-beijing.aliyuncs.com",` ○ 本地路径：如果使用 `test` 命令进行本地调试，则可以使用本地路径。
oss_endpoint	否	OSS的Endpoint，例如oss-cn-beijing.aliyuncs.com其他取值请参见访问域名和数据中心。如果model_path使用OSS地址，则必须指定该参数。
model_entry	否	表示模型的入口文件，可以包含任意文件。如果未指定，则使用model_path中的文件名。主文件路径会传递给Processor中的Load()函数。
model_config	否	表示模型的配置，支持任意文本。该参数值会传递给Processor中LoadWithConfig()函数的第二个参数。
processor	否	如果使用官方提供的预置Processor，则直接在此指定Processor Code即可。Processor在eascmd中使用的Code请参见预置Processor使用说明。 如果使用自定义Processor，则无需配置该参数，只需要配置processor_path、processor_entry、processor_mainclass及processor_type参数。
processor_path	否	processor相关的文件包路径，可以参见model_path参数的描述信息。
processor_entry	否	processor的主文件。例如*libprocessor.so*或*app.py*，其中包含了预测所需的 `initialize()` 函数和 `process()` 函数的实现。 当processor_type为cpp或python时，必须指定该参数。
processor_mainclass	否	processor的主文件，JAR包中的mainclass。例如com.aliyun.TestProcessor。 当processor_type为java时，必须指定该参数。

参数	是否必选	描述
processor_type	否	processor实现的语言，取值范围如下： ○ cpp ○ java ○ python
metadata	是	服务的Meta信息，详细参数请参见metadata参数解释。

metadata参数解释

参数	是否必选	描述	
一般参数	workers	否	每个Instance中用于并发处理请求的线程数，默认值为5。
	instance	是	服务启动的Instance数量。
	cpu	否	每个Instance需要的CPU数量。
	gpu	否	每个Instance需要的GPU数量。
	resource	否	资源组ID，配置策略如下： ○ 如果服务部署在公共资源组，则可以忽略该参数，此时服务进行按量付费。 ○ 如果服务部署在专属资源组，则配置该参数为资源组ID。例如eas-r-6dbzve8ip0xnzte5rp。
高级参数（慎重调整）	rpc.batching	否	是否开启Server端Batching，用于GPU模型加速。取值范围如下： ○ false：默认值，关闭Server端Batching。 ○ true：开启Server端Batching。
	rpc.keepalive	否	单个请求的最长处理时间。如果请求处理时长超过该值，则服务端返回408超时并关闭连接。默认值为5000，单位为ms。
	rpc.io_threads	否	每个Instance用于处理网络IO的线程数量，默认值为4。
	rpc.max_batch_size	否	每个Batch的最大Size，默认值为16。仅rpc.batching取值为true时，该参数生效。
	rpc.max_batch_timeout	否	每个Batch的最大Timeout，默认值为50 ms。仅rpc.batching取值为true时，该参数生效。
	rpc.max_queue_size	否	队列大小，默认值为64。队列满时，服务端返回450并关闭连接。为保证服务端不会压力过载，队列可以提前通知客户端向其他Instance进行重试。对于RT较长的服务队列，可以适当减小队列长度，以避免请求在队列中堆积导致大量请求超时。
	rpc.worker_threads	否	每个Instance中用于并发处理请求的线程数，与参数workers含义相同，默认值为5。

- 示例（假设描述服务相关信息的JSON文件为*pmml.json*）

```
eascmd create pmml.json
```

系统输出如下类似结果。

```
[RequestId]: 1651567F-8F8D-4A2B-933D-F8D3E2DD****
+------------------+-----------------------------------------------------------------------------+
| Intranet Endpoint | http://pai-eas-vpc.cn-shanghai.aliyuncs.com/api/predict/savedmodel_exanple |
|          Token | YjQxZDYzZTBiZTZjMzQ5ZmE0MzczZjIxMGZiNzZmMDBkY2VjMDg4****                |
+------------------+-----------------------------------------------------------------------------+
[OK] Creating api gateway
[OK] Building image [registry-vpc.cn-shanghai.aliyuncs.com/eas/savedmodel_exanple_cn-shanghai:v0.0.1
-20190224001315]
[OK] Pushing image [registry-vpc.cn-shanghai.aliyuncs.com/eas/savedmodel_exanple_cn-shanghai:v0.0.1
-20190224001315]
[OK] Waiting [Total: 1, Pending: 1, Running: 0]
[OK] Waiting [Total: 1, Pending: 1, Running: 0]
[OK] Service is running
```

修改配置

- 功能

  对于Instnace和CPU等metadata信息，可以直接使用 `modify` 命令的 `-D` 参数进行修改。

- 命令

  ```
  eascmd modify <service_name> -Dmetadata.<attr_name>=<attr_value>
  ```

  支持同时配置多个参数，详情请参见示例。

- 参数
  - <service_name>：服务名称。
  - <attr_name>：参数名称。
  - <attr_value>：参数取值。

- 示例

  将Instance数量配置为10，且每个Instance中的Quota数量为5（5核+20 GB）。

  ```
  eascmd modify service_test -Dmetadata.instance=10 -Dmetadata.cpu=5
  ```

  扩缩容时，可以只修改服务的metadata.instance参数。如果指定的Instance数量大于服务当前的Instance数量，则系统启动新的Instance，以达到要求的Instance数量，原有实例的运行不受影响。如果指定的Instance数量小于当前Instance数量，则系统停止部分Instance，以达到要求的Instance数量，其他实例的运行不受影响。

  > **说明** 只修改Instance的更新操作与全量更新不同，前者不会触发服务的滚动更新。

修改服务配置

- 功能

  通过 `modify` 命令可以对已部署的服务进行配置修改。

- 命令

```
eascmd modify <service_name> -s <service_desc_json>
```

- 参数
  - <service_name>：服务名称。
  - <service_desc_json>：服务描述文件。

> **说明** 在服务描述文件中，仅写需要修改的参数即可，其他不必要参数不写。如果写了模型文件信息及processor信息，则会被认定为增加一个新的服务版本。

增加服务版本

- 功能

  通过 `modify` 命令可以对已部署的服务增加服务版本。

- 命令

```
eascmd modify <service_name> -s <service_desc_json>
```

- 参数
  - <service_name>：服务名称。
  - <service_desc_json>：服务描述文件。

> **说明** 需要在服务描述文件中指定模型文件信息及processor信息。

停止服务

- 功能

  通过 `stop` 命令可以停止一个运行中的服务。

- 命令

```
eascmd stop <service_name>
```

- 参数

  <service_name>表示待停止的服务名称。

启动服务

- 功能

  通过 `start` 命令可以重新启动一个已停止的服务。

- 命令

```
eascmd start <service_name>
```

- 参数

  <service_name>表示待启动的服务名称。

删除服务

- 功能

通过 `delete` 命令可以删除服务，但是只能删除当前地域的服务。

- 命令

```
eascmd delete <service_name>
```

- 参数

  <service_name>表示待删除的服务名称。

- 示例

  假设服务名称为savedmodel_exanple，删除该服务的步骤如下：

  i. 执行删除服务的命令。

  ```
  eascmd delete savedmodel_exanple
  ```

  系统输出如下类似结果。

  ```
  Are you sure to delete the service [savedmodel_exanple] in [cn-shanghai]? [Y/n]
  ```

  ii. 输入*Y*。系统输出如下类似结果。

  ```
  [RequestId]: 1651567F-8F8D-4A2B-933D-F8D3E2DD****
  [OK] Service [savedmodel_exanple] in region [cn-shanghai] is terminating
  [OK] Service is terminating
  [OK] Service is terminating
  [OK] Service was deleted successfully
  ```

蓝绿发布

- 功能

  通过 `create -r` 命令可以对一个已存在的服务创建一个关联服务，再使用 `release` 命令根据需求随时切换流量比例，从而进行蓝绿发布。新服务的信息描述JSON文件中的name必须与旧服务同名，其它字段根据需求自由配置。系统会自动在旧服务名基础上增加随机后缀，从而得到新服务名。

  如果删除新服务，则流量会全部切换至旧服务。如果删除旧服务，则全部流量切换至新服务。蓝绿发布之前，最原始服务的Endpoint会成为后续发布迭代的流量入口，无论后续在这个基础上进行多少次蓝绿发布迭代，该入口的Endpoint始终保持不变（例如下面示例中的{domain}/api/predict/savedmodel_example），您无须修改客户端调用代码。

  > **说明** 蓝绿发布不适用于网络直连访问的方式。

- 命令

  i. 创建关联服务

  ```
  eascmd create <service_desc_json> -r
  ```

  ii. 对蓝绿服务进行切流。

  ```
  eascmd release <service_name> -w <weight>
  ```

- 参数
  - <service_desc_json>：服务信息描述的JSON文件。
  - <service_name>：创建的新服务名称。

- <weight>：新服务承载的流量百分比。

- 示例（假设服务信息描述文件为*pmml.json*）

  i. 创建关联服务

  ```
  eascmd create pmml.json -r
  ```

  系统输出以下类似信息。

  ```
  [RequestId]: 1651567F-8F8D-4A2B-933D-F8D3E2DD****
  +------------------+----------------------------------------------------------------------------+
  | Intranet Endpoint | http://xxx.cn-shanghai.pai-eas.aliyuncs.com/api/predict/savedmodel_example_
  9c16a222 |
  |          Token | YjQxZDYzZTBiZTZjMzQ5ZmE0MzczZjIxMGZiNzZmMDBkY2VjMDg4****              |
  +------------------+----------------------------------------------------------------------------+
  [OK] Building image [registry-vpc.cn-shanghai.aliyuncs.com/eas/savedmodel_exanple_9c16a222_cn-
  shanghai:v0.0.1-20190224001315]
  [OK] Pushing image [registry-vpc.cn-shanghai.aliyuncs.com/eas/savedmodel_exanple_9c16a222_cn-
  shanghai:v0.0.1-20190224001315]
  [OK] Waiting [Total: 1, Pending: 1, Running: 0]
  [OK] Waiting [Total: 1, Pending: 1, Running: 0]
  [OK] Service is running
  ```

  上述输出表示创建了一个名为savedmodel_example_9c16a222的服务，且两个服务分别有独立的流量入口，可以被单独调用。您对新服务可以进行独立测试，不会影响已有服务的线上运行，测试完成后，可以对服务进行切流操作。

  ii. 对蓝绿服务进行流量切换。

  ```
  eascmd release savedmodel_example_9c16a222 -w 20
  ```

  上述命令表示切换20%的流量到新服务savedmodel_example_9c16a222，其余80%的流量在旧服务savedmodel_example上。此时，新服务的独立访问Endpoint（{domain}/api/predict/savedmodel_example_9c16a222）关闭，旧服务Endpoint（{domain}/api/predict/savedmodel_example）流量的20%会进入新服务，80%进入旧服务。

  系统输出如下类似结果。

  ```
  Confirmed to release this service at weight [20%]? [Y/n]
  ```

  iii. 输入*Y*，并单击Enter键，系统输出如下类似结果。

  ```
  [RequestId]: 9258EEDE-6F99-4C3B-841B-B6E9774F****
  [OK] Service [savedmodel_example_9c16a222] is weighted to 20% now
  ```

切换版本

- 功能

  您可以先通过 `desc` 命令查看服务的最新版本和当前版本，再通过 `version` 命令切换服务至最新版本之前的任意版本。

- 命令

  ```
  eascmd version <service_name> <version_id>
  ```

- 参数
  - <service_name>：服务名称。
  - <version_id>：待切换服务的版本ID。

查看服务列表

- 功能

  使用 `list` （或缩写 `ls` ）命令可以查看当前用户已部署的服务列表。

- 命令

```
eascmd ls
```

- 参数

  无

- 示例

```
eascmd ls
```

  系统输出如下类似结果。

```
[RequestId]: 83945D4E-ED3E-4D35-A989-831E36BB****
+--------------------------+-------------+----------+--------------------+--------------------+---------+--------+--------------------
------------------+
|     SERVICENAME     | REGION   |INSTANCE|    CREATETIME    |    UPDATETIME    |STATUS |WEIGHT|
SERVICEPATH          |
+--------------------------+-------------+----------+--------------------+--------------------+---------+--------+--------------------
------------------+
| mnist_saved_model_example | cn-shanghai |      1 | 2019-02-21 16:35:41 | 2019-02-21 16:35:41 | Running |
0 | /api/predict/mnist_saved_model_example |
+--------------------------+-------------+----------+--------------------+--------------------+---------+--------+--------------------
------------------+
```

查看服务信息

- 功能

  通过 `desc` 命令可以查看已部署服务的详情信息。

- 命令

```
eascmd desc <service_name>
```

- 参数

  <service_name>表示服务名称。

- 示例

```
eascmd desc mnist_saved_model_example
```

  系统输出如下类似结果。

```
$ eascmd desc mnist_saved_model_example
+--------------------+-----------------------------------------------------------------------------------------------------------------------------------------+
|         Status | Running                                                                                                                                |
|      ServiceName | mnist_saved_model_example                                                                                                                |
|         Region | cn-shanghai                                                                                                                             |
|       CreateTime | 2019-02-21 16:35:41                                                                                                                   |
|       UpdateTime | 2019-02-21 16:35:41                                                                                                                    |
|      AccessToken |                                                                                                                                        |
|      PrivateToken | ZWNjMTNkNDExMmExNjZkYTM4YWQ5YTY0YmFjNjk3YWYzZTRjM2Y2****
|
|     TotalInstance | 1                                                                                                                                  |
|    RunningInstance | 1                                                                                                                                 |
|    PendingInstance | 0                                                                                                                                 |
|           CPU | 1                                                                                                                                       |
|           GPU | 0                                                                                                                                       |
|         Memory | 1000M                                                                                                                                  |
|         Image | registry-vpc.cn-shanghai.aliyuncs.com/eas/mnist_saved_model_example_cn-shanghai:v0.
0.1-20190221163541      |
|         Weight | 0                                                                                                                                     |
|     LatestVersion | 1                                                                                                                                  |
|     CurrentVersion | 1                                                                                                                                 |
|        Message | Service start successfully                                                                                                             |
|     APIGatewayUrl | 1c3b37ea83c047efa0dc6df0cacb****-cn-shanghai.alicloudapi.com/EAPI_18284888792
2****_mnist_saved_model_example |
|    APIGatewayAppKey | 2564****                                                                                                                        |
| APIGatewayAppSecret | 12562a7b8858bbba2c2e9c4517ff****                                                                                                       |
|    IntranetEndpoint | http://pai-eas-vpc.cn-shanghai.aliyuncs.com/api/predict/mnist_saved_model_exam
ple                 |
|     ServiceConfig | {                                                                                                                                 |
|             |  "generate_token": "false",                                                                                                          |
|             |  "metadata": {                                                                                                                       |
|             |    "cpu": 1,                                                                                                                         |
|             |    "instance": 1,                                                                                                                    |
|             |    "region": "cn-shanghai"                                                                                                           |
|             |  },                                                                                                                                  |
|             |  "model_path":                                                                                                                       |
|             | "http://eas-data.oss-cn-shanghai.aliyuncs.com/models%2Fmnist_saved_model.tar.gz",
|
|             |  "name":                                                                                                                             |
|             | "mnist_saved_model_example",                                                                                                         |
|             |  "processor":                                                                                                                        |
|             | "tensorflow_cpu"                                                                                                                     |
|             | }                                                                                                                                    |
+--------------------+-----------------------------------------------------------------------------------------------------------------------------------------+
```

查看服务进程

- 功能

  通过 showworkers(w) （或缩写 w ）命令可以查看服务正在运行的进程状态。

- 命令

```
eascmd w <service_name>
```

- 参数

  <service_name>表示服务名称。

- 示例

```
eascmd w mnist_saved_model_example
```

  系统输出如下类似结果。

```
[RequestId]: 4E905404-E617-4BD8-85D6-EC5C6A0D****
+--------------+----------------+---------------------+----------+---------+-------+--------+
| INNERIP  |  HOSTIP  |  STARTAT  | RESTARTS | STATUS | READY | REASON |
+--------------+----------------+---------------------+----------+---------+-------+--------+
| 172.24.XX.XX | 192.168.XX.XXX | 2019-02-21 16:35:58 |  0 | Running | [1/1] |  |
+--------------+----------------+---------------------+----------+---------+-------+--------+
```

查看资源组列表

- 功能

  通过 `resource list` （或缩写 `resource ls` ）命令可以查看当前账户下的资源组列表。

- 命令

```
eascmd resource ls
```

- 参数

  无

- 示例

```
eascmd resource ls
```

  系统输出如下类似结果。

```
+-------------------------+-------------+---------------+----------+----------+-----------------+--------------------+---------------+
|  RESOURCENAME  | CLUSTERID | INSTANCECOUNT | GPUCOUNT | CPUCOUNT |  OWNERUID  |  CREATETIME  |  STATUS  |
+-------------------------+-------------+---------------+----------+----------+-----------------+--------------------+---------------+
| eas-r-lzo32vrdbtukr7te3i | cn-shanghai |  1 |  0 |  16 | 182848887922**** | 2020-03-18 13:09:24 | ResourceReady |
+-------------------------+-------------+---------------+----------+----------+-----------------+--------------------+---------------+
```

查看资源组详情

- 功能

  通过 `resource desc` 命令可以查看某个资源组的详细信息。

- 命令

```
eascmd resource desc <resource_id>
```

- 参数

  <resource_id>表示待查看的资源组ID，即 `resource list(ls)` 命令返回结果中的RESOURCENAME字段。

- 示例

```
eascmd -c ~/.eas/shanghai2.conf resource desc eas-r-lzo32vrdbtukr7te3i
```

系统输出如下类似结果。

```
+---------+-----------------+-------------------------------------------------------------+
|  Basic | ResourceName   | eas-r-lzo32vrdbtukr7te3i                    |
|       | Region        | cn-shanghai                                 |
|       | CpuCount      | 16                                          |
|       | GpuCount      | 0                                           |
|       | instanceCount  | 1                                          |
|       | CreateTime     | 2020-03-18 13:09:24                        |
|       | LastStatus     | ResourceReady                              |
|       | Message        | Resource is ready                          |
|       | RoleArn        | acs:ram::xxx:role/AliyunPAIAccessingENIRole |
| Network | VpcId         | vpc-uf6s9pv47nu03srne****                   |
|       | VSwitchId      | vsw-uf6voq53e893k56ws****                  |
|       | SecurityGroupId | sg-uf6c5twkfar8l06c****                   |
|       | DestinationCIDR |                                           |
|       | AuxVSwitchList  | []                                        |
+---------+-----------------+-------------------------------------------------------------+
```

查看资源组实例列表

- 功能

  通过 `resource list_instance` （或缩写为 `resource li` ）命令可以查看某个资源组的实例列表及每个实例的资源使用情况。

- 命令

  ```
  eascmd resource list_instance <resource_id>
  ```

- 参数

  <resource_id>表示待查看的资源组ID，即 `resource list(ls)` 命令返回结果中的RESOURCENAME字段。

- 示例

  ```
  eascmd resource li eas-r-lzo32vrdbtukr7te3i
  ```

  系统输出如下类似结果。

  ```
  +-----------------------------------+--------------+--------+---------------+---------------+------------------+--------------------+
  ----------------+------------+
  |        INSTANCENAME         | INSTANCEIP | STATUS | TOTAL/USED CPU | TOTAL/USED GPU | TOTAL/USED M
  EMORY |    CREATETIME     | INSTANCETYPE | CHARGETYPE |
  +-----------------------------------+--------------+--------+---------------+---------------+------------------+--------------------+
  ----------------+------------+
  | cn-shanghai.i-uf6dj71ir6mh3gjmaz3a | 10.224.XX.XX | Ready | 16/6        | 0/0         | 62240M/4200M     | 2020-
  03-18 13:09:34 | ecs.g6.4xlarge | PostPaid   |
  +-----------------------------------+--------------+--------+---------------+---------------+------------------+--------------------+
  ----------------+------------+
  ```

配置资源组网络

- 功能

通过 `resource network` 命令可以设置某个资源组的直连情况，用于连通PAI-EAS VPC和用户VPC之间的网络。一方面可以在用户VPC内以直连软负载的方式调用PAI-EAS服务，另一方面可以在PAI-EAS Processor中反向访问用户VPC中的内网资源（例如RDS、Redis等）。

- 命令

```
eascmd resource network <resource_id> -s <network_cfg.json>
```

- 参数
  - <resource_id>：表示查待看的资源组ID，即 `resource list(ls)` 命令返回结果中的RESOURCENAME字段。

- <network_cfg.json>：网络配置文件，该文件格式如下所示。

```
{
 "Action":"create",
 "VSwitchId": "vsw-8vbsunr5bkcbyxh94****",
 "SecurityGroupId": "sg-8vbhwowdxzx5fjcx****",
 "VSwitchIdList": ["vsw-8xbsunr5abcbyqh93****", "vsw-8xbs1y7gu6cxbvqzw****"],
 "DestinationCIDR": "192.XX.XX.XX/16"
}
```

各参数的含义如下表所示。

参数	描述	是否必选	默认值
Action	网络设置的操作，取值范围如下： ■ create：开通直连。 ■ delete：关闭直连，此时无需配置其他参数。	是	无
VSwitchId	待连通的目标主vSwitch ID，PAI-EAS会自动在该vSwitch中创建ENI弹性网卡，请不要主动删除该ENI，否则会导致网络连通性问题。	是	无
SecurityGroupId	客户端ECS所在的安全组ID。 **说明** 客户端ECS必需归属于该安全组中，否则会导致网络连通性问题。	是	无
VSwitchIdList	待打通的附属vSwitch列表，必须与主vSwitch在同一个VPC中，这些VSwitch的IP网段会自动被加入到PAI-EAS的路由表规则中。	否	空数组（[]）
DestinationCIDR	待打通的客户端目标网段，必须与主vSwitch在同一个VPC中，该网段会被自动加入到PAI-EAS的路由表规则中。	否	空字符串（""）

说明 VSwitchIdList与DestinationCIDR原理相同，均是为了连通PAI-EAS集群与用户某个网段的网络。如果需要连通多个指定vSwitch，则使用VSwitchIdList。如果需要连通一个大网段（例如整个VPC），则使用DestinationCIDR字段。建议不要使用10.0.0.0/8、10.224.0.0/16或10.240.0.0/16网段，否则会导致网络冲突问题。如果有其他需求，可以提交工单。

预测

进行预测调用时，可以根据创建服务时生成的HTTP URL访问服务。预测服务的输入输出格式由Processor自定义，详细请参见模型服务调用章节中的通用Processor服务请求数据构造部分。

6.3. 预置Processor使用说明

Processor是包含在线预测逻辑的程序包。PAI-EAS已将一些常用的Processor预置为官方Processor，您可以直接使用该官方Processor部署常规模型，节省开发在线预测逻辑的成本。

PAI-EAS提供的Processor名称及Code如下表所示（使用EASCMD部署服务时，需要提供Processor Code）。

Processor名称	Processor Code（仅用于EASCMD部署）		文档
	CPU版	GPU版	
PMML	pmml	无	PMML Processor
TensorFlow1.12	tensorflow_cpu_1.12	tensorflow_gpu_1.12	TensorFlow1.12 Processor
TensorFlow1.14	tensorflow_cpu_1.14	tensorflow_gpu_1.14	TensorFlow1.14 Processor
TensorFlow1.15	tensorflow_cpu_1.15	tensorflow_gpu_1.15	TensorFlow1.15 Processor（内置PAI-Blade敏捷版优化引擎）
PyTorch1.6	pytorch_cpu_1.6	pytorch_gpu_1.6	PyTorch1.6 Processor（内置PAI-Blade敏捷版优化引擎）
Caffe	caffe_cpu	caffe_gpu	Caffe Processor
PS算法	parameter_sever	无	无
Alink	alink_pai_processor	无	无
xNN	xnn_cpu	无	无
EasyVision	easy_vision_cpu_tf1.12_torch151	easy_vision_gpu_tf1.12_torch151	EasyVision Processor
EasyNLP	easy_nlp_cpu_tf1.12	easy_nlp_gpu_tf1.12	EasyNLP Processor
Blade	blade_cpu	blade_cuda10.0_beta	无
MediaFlow	无	mediaflow	MediaFlow Processor

PMML Processor

PMML（Predictive Model Markup Language）是一种预测模型标记语言，在PAI-Studio训练的传统机器学习模型基本都能够以该格式导出。从PAI-Studio导出PMML模型的方法如下：

1. 模型训练前，在PAI-Studio首页，单击**设置 > 基本设置**，并选中**自动生成PMML**复选框。
2. 模型训练完成后，右键单击画布中的模型训练节点，在快捷菜单，单击**模型选项 > 导出PMML**。

> **说明** PAI-Studio中，可以生成PMML模型的算法包括GBDT二分类、线性支持向量机、逻辑回归二分类、逻辑回归多分类、随机森林、KMeans、线性回归、GBDT回归及评分卡训练等。

PAI-EAS预置的PMML Processor主要功能包括：

- 将PMML类型的模型文件加载为一个服务。
- 处理对模型服务进行调用的请求。
- 根据模型计算请求结果，并将其返回至客户端。

PMML Processor提供默认的缺失值填充策略。如果PMML模型文件中的特征字段无指定isMissing策略，则系统默认以如下值进行填充。

DataType	默认填充值
BOOLEAN	false
DOUBLE	0.0
FLOAT	0.0
INT	0
STRING	“”

您可以通过如下任何一种方式部署PMML模型：

- 控制台上传

  选择**Processor种类**为**PMML**，详情请参见控制台上传部署。

- PAI-Studio一键部署

  详情请参见PAI-Studio一键部署。

- 本地客户端部署

  在服务配置文件service.json中，将processor字段配置为相应的Processor Code，即pmml，示例如下。

```
{
 "processor": "pmml",
 "generate_token": "true",
 "model_path": "http://xxxxx/lr.pmml",
 "name": "eas_lr_example",
 "metadata": {
  "instance": 1,
  "cpu": 1 #自动为每个CPU配置4 GB内存，称为1 Quota。
 }
}
```

- PAI-DSW部署

  类似于本地客户端部署，编写服务配置文件service.json，详情请参见服务部署。

TensorFlow1.12 Processor

PAI-EAS提供的TensorFlow1.12 Processor可以加载SavedModel（推荐）或SessionBundle格式的TensorFlow模型。对于Keras和Checkpoint模型，需要先将其转换为Savedmodel模型，再进行部署，详情请参见TensorFlow模型如何导出为SavedModel。

> **说明** 官方通用Processor不支持TensorFlow自定义OP。

您可以通过如下任何一种方式部署TensorFlow模型：

- 控制台上传

  选择**Processor种类**为**TensorFlow1.12**，详情请参见控制台上传部署。

- PAI-Studio一键部署

  详情请参见PAI-Studio一键部署。

- 本地客户端部署

  在服务配置文件service.json中，将processor字段配置为相应的Processor Code，即tensorflow_cpu_1.12或tensorflow_gpu_1.12（需要根据部署所用的资源进行选择，如果processor与资源不匹配，则会导致部署报错），示例如下。

```
{
 "name": "tf_serving_test",
 "generate_token": "true",
 "model_path": "http://xxxxx/savedmodel_example.zip",
 "processor": "tensorflow_cpu_1.12",
 "metadata": {
  "instance": 1,
  "cpu": 1,
  "gpu": 0,
  "memory": 2000
 }
}
```

- PAI-DSW部署

  类似于本地客户端部署，编写服务配置文件service.json，详情请参见服务部署。

TensorFlow1.14 Processor

PAI-EAS提供的TensorFlow1.14 Processor可以加载SavedModel（推荐）或SessionBundle格式的TensorFlow模型。对于Keras和Checkpoint模型，需要先将其转换为Savedmodel模型，再进行部署，详情请参见TensorFlow模型如何导出为SavedModel。

> **说明** 官方通用Processor不支持TensorFlow自定义OP。

您可以通过如下任何一种方式部署TensorFlow模型：

- 控制台上传

  选择**Processor种类**为**TensorFlow1.14**，详情请参见控制台上传部署。

- PAI-Studio一键部署

  详情请参见PAI-Studio一键部署。

- 本地客户端部署

在服务配置文件service.json中，将processor字段配置为相应的Processor Code，即tensorflow_cpu_1.14或tensorflow_gpu_1.14（需要根据部署所用的资源进行选择，如果processor与资源不匹配，则会导致部署报错），示例如下。

```
{
 "name": "tf_serving_test",
 "generate_token": "true",
 "model_path": "http://xxxxx/savedmodel_example.zip",
 "processor": "tensorflow_cpu_1.14",
 "metadata": {
  "instance": 1,
  "cpu": 1,
  "gpu": 0,
  "memory": 2000
 }
}
```

- PAI-DSW部署

  类似于本地客户端部署，编写服务配置文件service.json，详情请参见服务部署。

TensorFlow1.15 Processor（内置PAI-Blade敏捷版优化引擎）

PAI-EAS提供的TensorFlow1.15 Processor可以加载SavedModel（推荐）或SessionBundle格式的TensorFlow模型。对于Keras和Checkpoint模型，需要先将其转换为Savedmodel模型，再进行部署，详情请参见TensorFlow模型如何导出为SavedModel。

> **说明**
>
> - 官方通用Processor不支持TensorFlow自定义OP。
> - 该Processor内置了PAI-Blade敏捷版优化引擎，您可以使用它部署PAI-Blade敏捷版优化后的TensorFlow模型。

您可以通过如下任何一种方式部署TensorFlow模型：

- 控制台上传

  选择**Processor种类**为**TensorFlow1.15**，详情请参见控制台上传部署。

- PAI-Studio一键部署

  详情请参见PAI-Studio一键部署。

- 本地客户端部署

  在服务配置文件service.json中，将processor字段配置为相应的Processor Code，即tensorflow_cpu_1.15或tensorflow_gpu_1.15（需要根据部署所用的资源进行选择，如果processor与资源不匹配，则会导致部署报错），示例如下。

```
{
 "name": "tf_serving_test",
 "generate_token": "true",
 "model_path": "http://xxxxx/savedmodel_example.zip",
 "processor": "tensorflow_cpu_1.15",
 "metadata": {
  "instance": 1,
  "cpu": 1,
  "gpu": 0,
  "memory": 2000
 }
}
```

- PAI-DSW部署

  类似于本地客户端部署，编写服务配置文件service.json，详情请参见服务部署。关于服务配置文件中的参数解释请参见创建服务。

PyTorch1.6 Processor（内置PAI-Blade敏捷版优化引擎）

PAI-EAS提供的PyTorch1.6 Processor可以加载TorchScript格式的模型，详情请参见TorchScript官方介绍。

> **说明**
> - 官方通用Processor不支持PyTorch扩展、不支持非Tensor类型的模型输入和输出。
> - 该Processor内置了PAI-Blade敏捷版优化引擎，您可以使用它部署PAI-Blade敏捷版优化后的PyTorch模型。

您可以通过如下任何一种方式部署TorchScript模型：

- 控制台上传

  选择**Processor种类**为**PyTorch1.6**，详情请参见控制台上传部署。

- PAI-Studio一键部署

  详情请参见PAI-Studio一键部署。

- 本地客户端部署

  在服务配置文件service.json中，将processor字段配置为相应的Processor Code，即pytorch_cpu_1.6或pytorch_gpu_1.6（需要根据部署所用的资源进行选择，如果processor与资源不匹配，则会导致部署报错），示例如下。

```
{
 "name": "pytorch_serving_test",
 "generate_token": "true",
 "model_path": "http://xxxxx/torchscript_model.pt",
 "processor": "pytorch_gpu_1.6",
 "metadata": {
  "instance": 1,
  "cpu": 1,
  "gpu": 1,
  "cuda": "10.0",
  "memory": 2000
 }
}
```

- PAI-DSW部署

  类似于本地客户端部署，编写服务配置文件service.json，详情请参见服务部署。关于服务配置文件中的参数解释请参见创建服务。

Caffe Processor

PAI-EAS提供的Caffe Processor可以加载Caffe框架训练得到的深度学习模型。因为Caffe框架比较灵活，所以部署Caffe模型时，需要指定模型包的Model文件名称和Weight文件名称。

> **说明** 官方通用Processor不支持自定义DataLayer。

您可以通过如下任何一种方式部署Caffe模型：

- 控制台上传

  选择**Processor种类**为**Caffe**，详情请参见控制台上传部署。

- PAI-Studio一键部署

  详情请参见PAI-Studio一键部署。

- 本地客户端部署

  在服务配置文件service.json中，将processor字段配置为相应的Processor Code，即caffe_cpu或caffe_gpu（需要根据部署所用的资源进行选择，如果processor与资源不匹配，则会导致部署报错），示例如下。

```
{
 "name": "caffe_serving_test",
 "generate_token": "true",
 "model_path": "http://xxxxx/caffe_model.zip",
 "processor": "caffe_cpu",
 "model_config": {
  "model": "deploy.prototxt",
  "weight": "bvlc_reference_caffenet.caffemodel"
 },
 "metadata": {
  "instance": 1,
  "cpu": 1,
  "gpu": 0,
  "memory": 2000
 }
}
```

- PAI-DSW部署

  类似于本地客户端部署，编写服务配置文件service.json，详情请参见服务部署。

EasyNLP Processor

PAI-EAS提供的EasyNLP Processor可以加载EasyTransfer框架训练得到的深度学习NLP模型。

您可以通过如下任何一种方式部署EasyTransfer模型：

- 控制台上传

  选择**Processor种类**为**EasyNLP**，详情请参见控制台上传部署。

- 本地客户端部署

  在服务配置文件service.json中，将processor字段配置为相应的Processor Code，即easy_nlp_cpu_tf1.12或easy_nlp_gpu_tf1.12（需要根据部署所用的资源进行选择，如果processor与资源不匹配，则会导致部署报错），在model_config的type字段指定训练时所使用的模型类型，示例如下。其他参数的详细解释请参见创建服务：

  - 使用GPU部署的配置

    ```
    {
     "name": "ev_app_demo",
     "generate_token": "true",
     "model_path": "http://xxxxx/your_model.zip",
     "processor": "easy_nlp_gpu_tf1.12",
     "model_config": "{\"type\":\"text_classify_bert\"}",
     "metadata": {
      "resource": "your_resource_name",
      "cuda": "9.0",
      "instance": 1,
      "memory": 4000,
      "gpu": 1,
      "cpu": 4,
      "rpc.worker_threads" : 5
     }
    }
    ```

- 使用CPU部署的配置

```
{
 "name": "easynlp_serving_test",
 "generate_token": "true",
 "model_path": "http://xxxxx/your_model.zip",
 "processor": "easy_nlp_cpu_tf1.12",
 "model_config": "{\"type\":\"text_classify_bert\"}",
 "metadata": {
  "resource": "your_resource_name",
  "instance": 1,
  "gpu": 0,
  "cpu": 4,
  "rpc.worker_threads" : 5
 }
}
```

EasyVision Processor

PAI-EAS提供的EasyVision Processor可以加载EasyVision框架训练得到的深度学习模型。

您可以通过如下任何一种方式部署EasyVision模型：

- 控制台上传

  选择**Processor种类**为**EasyVision**，详情请参见控制台上传部署。

- 本地客户端部署

  在服务配置文件service.json中，将processor字段配置为相应的Processor Code，即easy_vision_cpu_tf1.12_torch151或easy_vision_gpu_tf1.12_torch151（需要根据部署所用的资源进行选择，如果processor与资源不匹配，则会导致部署报错），在model_config的type字段指定训练时所使用的模型类型，示例如下。其他参数的详细解释请参见创建服务：

  - 使用GPU部署的配置

```
{
 "name": "ev_app_demo",
 "processor": "easy_vision_gpu_tf1.12_torch151",
 "model_path": "oss://path/to/your/model",
 "model_config": "{\"type\":\"classifier\"}",
 "metadata": {
  "resource": "your_resource_name",
  "cuda": "9.0",
  "instance": 1,
  "memory": 4000,
  "gpu": 1,
  "cpu": 4,
  "rpc.worker_threads" : 5
 }
}
```

- 使用CPU部署的配置

```
{
 "name": "ev_app_cpu_demo",
 "processor": "easy_vision_cpu_tf1.12_torch151",
 "model_path": "oss://path/to/your/model",
 "model_config": "{\"type\":\"classifier\"}",
 "metadata": {
  "resource": "your_resource_name",
  "instance": 1,
  "memory": 4000,
  "gpu": 0,
  "cpu": 4,
  "rpc.worker_threads" : 5
 }
}
```

MediaFlow Processor

PAI-EAS 提供的MediaFlow Processor是通用的编排引擎，可以进行视频、音频及图像分析处理。

您可以通过如下任何一种方式部署MediaFlow模型：

- 控制台上传

  选择**Processor种类**为**MediaFlow**，详情请参见控制台上传部署。

- 本地客户端部署

  在服务配置文件service.json中，将processor字段配置为相应的Processor Code，即mediaflow。此外，使用MediaFlow Processor部署模型，还需要增加如下特有字段，其他字段说明请参见创建服务：

  - graph_pool_size：图池的数量。
  - worker_threads：调度线程的数量。

  示例如下：

- 部署视频分类模型的配置。

```
{
 "model_entry": "video_classification/video_classification_ext.js",
 "name": "video_classification",
 "model_path": "oss://path/to/your/model",
 "generate_token": "true",
 "processor": "mediaflow",
 "model_config" : {
   "graph_pool_size":8,
   "worker_threads":16
 },
 "metadata": {
  "eas.handlers.disable_failure_handler" :true,
  "resource": "your_resource_name",
   "rpc.worker_threads": 30,
   "rpc.enable_jemalloc": true,
  "rpc.keepalive": 500000,
  "cpu": 4,
  "instance": 1,
  "cuda": "9.0",
  "rpc.max_batch_size": 64,
  "memory": 10000,
  "gpu": 1
 }
}
```

- 部署语音识别（ASR）模型的配置。

```
{
 "model_entry": "asr/video_asr_ext.js",
 "name": "video_asr",
 "model_path": "oss://path/to/your/model",
 "generate_token": "true",
 "processor": "mediaflow",
 "model_config" : {
   "graph_pool_size":8,
   "worker_threads":16
 },
 "metadata": {
  "eas.handlers.disable_failure_handler" :true,
  "resource": "your_resource_name",
   "rpc.worker_threads": 30,
   "rpc.enable_jemalloc": true,
  "rpc.keepalive": 500000,
  "cpu": 4,
  "instance": 1,
  "cuda": "9.0",
  "rpc.max_batch_size": 64,
  "memory": 10000,
  "gpu": 1
 }
}
```

语音识别与视频分类service.json配置的主要差异为model_entry、name及model_path字段，需要您根据部署的模型类型进行修改。

6.4. 自定义Processor

6.4.1. 使用说明

Processor是包含在线预测逻辑（模型加载和请求预测逻辑）的程序包，如果PAI-EAS提供的官方通用Processor无法满足模型部署需求，则可以根据Processor的开发标准自定义Processor。

前提条件

为保证模型服务安全，在公共资源组中无法使用自定义Processor。因此，您必须购买（创建）专属资源组，才能使用自定义Processor部署模型服务。购买（创建）专属资源组请参见购买专属资源组。

自定义Processor开发手册

支持使用以下编程语言开发自定义Processor：

- C或C++，详情请参见使用C或C++开发自定义Processor。
- Java，详情请参见使用Java开发自定义Processor。
- Python，详情请参见使用Python开发自定义Processor。

使用自定义Processor部署模型服务

自定义Processor开发完成后，建议在本地调试通过后再进行线上服务部署。您可以通过PAI-EAS控制台上传或EASCMD工具进行服务部署：

- 控制台上传

  选择**资源组种类**为**专属资源组**，选择**Processor种类**为**自定义processor**，详情请参见控制台上传部署。

- EASCMD工具部署

  部署时，将resource字段配置为您已购买的专属资源组ID，详情请参见命令使用说明。

6.4.2. 使用C或C++开发自定义Processor

本文为您介绍如何使用C或C++开发自定义Processor。

快速上手Demo

下载PAI-EAS预测服务示例，该项目包含两个自定义Processor，其中：

- echo：请求时将用户输入原样返回，同时返回模型中的文件列表。
- image_classification：mnist文本分类，输入mnist jpg图片，返回图片分类类别。

编译方法请参见项目下的README文件，每个Processor的本地调试方法请参见各目录下的README文件。

接口定义

使用C或C++开发自定义Processor，需要定义initialize()和Process()函数，分别用于服务初始化时加载模型和处理客户端请求并返回结果。两个函数的声明如下。

```
void *initialize(const char *model_entry, const char *model_config, int *state)
```

参数	类型	描述
model_entry	输入参数	对应创建服务时配置文件中的model_entry字段，关于该字段的更多信息请参见创建服务。您可以传入一个文件名（例如randomforest.pmml）或目录（例如./model）。
model_config	输入参数	对应创建服务时配置文件中的model_config字段，表示自定义的模型配置信息。关于该字段的更多信息请参见创建服务。
state	输出参数	模型加载状态。如果为0，则表示模型加载成功，否则表示模型加载失败。
返回值		自定义的model变量内存地址，可以为任意类型。

```
int process(void *model_buf, const void *input_data, int input_size,void **output_data, int *output_size)
```

参数	类型	描述
model_buf	输入参数	initialize()函数返回的模型内存地址。
input_data	输入参数	用户数入数据，可以为任意字符串或BINARY类型。
input_size	输入参数	用户数入数据的长度。
output_data	输出参数	Processor返回的数据，需要在堆为其分配内存，模型负责释放该内存。
output_size	输出参数	Processor返回的数据长度。
返回值		返回0或200表示成功，可以直接返回HTTP错误码。如果返回不识别的HTTP错误码，则自动转换为http 400 error。

代码示例

以下是一个简单的示例，未加载任何模型数据，预测服务将用户请求直接返回给客户端。

```
#include <stdio.h>
#include <string.h>
extern "C" {
  void *initialize(const char *model_entry, const char *model_config, int *state)
  {
    *state = 0;
    return NULL;
  }
  int process(void *model_buf, const void *input_data, int input_size,
      void **output_data, int *output_size)
  {
    if (inputSize == 0) {
      const char *errmsg = "input data should not be empty";
      *outputData = strndup(errmsg, strlen(errmsg));
      *outputSize = strlen(errmsg);
      return 400;
    }
    *outputData = strndup((char *)inputData, inputSize);
    *outputSize = inputSize;
    return 200;
  }
}
```

该Processor未读取任何模型信息，将用户输入原样输出，可以通过如下的Makefile将其编译为SO文件。

```
CC=g++
CCFLAGS=-I./ -D_GNU_SOURCE -Wall -g -fPIC
LDFLAGS= -shared -Wl,-rpath=./
OBJS=processor.o
TARGET=libpredictor.so
all: $(TARGET)
$(TARGET): $(OBJS)
    $(CC) -o $(TARGET) $(OBJS) $(LDFLAGS) -L./
%.o: %.cc
    $(CC) $(CCFLAGS) -c $< -o $@
clean:
    rm -f $(TARGET) $(OBJS)
```

如果有其它依赖的SO文件，则必须在Makefile文件中添加 `-rpath` 选项，指定SO文件的搜索目录。`-rpath` 的原理详情请参见Rpath。

6.4.3. 使用Java开发自定义Processor

本文为您介绍如何使用Java开发自定义Processor。

接口定义

使用Java开发自定义Processor仅需要定义一个类，该类中除构造函数以外，只需要Load()和Process()函数即可。类的原型如下。

```
package com.alibaba.eas;
import java.util.*;
public class TestProcessor {
 public TestProcessor(String modelEntry, String modelConfig) {
  /*传递模型文件名，可以添加初始化工作。*/
 }
 public void Load() {
  /*根据模型名加载模型信息。*/
 }
 public byte[] Process(byte[] input) {
  /*对输入数据进行预测处理并返回结果。支持BYTE[]和STRING，推荐使用BYTE[]，可以避免编码问题。*/
 }
 public static void main(String[] args) {
  /*主函数为非必选，可以在本地单机验证类功能。*/
 }
}
```

如果抛出异常，框架会捕获该异常，并将Exception中的message作为错误信息返回给客户端，同时HTTP响应400。您也可以自己捕获异常，输出自己的异常错误信息，示例如下。

```
try{
} catch (com.alibaba.fastjson.JSONException e) {
 throw new RuntimeException("bad json format, " + e.getMessage());
}
```

单机开发调试

单机调试功能用于非集群模式，用户在本地环境中对模型或Processor进行开发及调试，其开发接口及调用接口与线上集群环境完全兼容。该功能不仅可以避免您在开发测试阶段频繁部署新服务，而且可以节省调试所用的资源费用。

> **说明** 该功能依赖于Docker，需要在运行EASCMD的服务器上预装Docker。如果需要GPU及CUDA，则必须预先在本地安装CUDA及Nvidia-Docker。

单机调试的方法如下：

1. 如果未安装Docker，则参见Docker安装进行安装。
2. 下载客户端工具EASCMD，各版本的下载地址如下：
   - Windows 64版本
   - Linux 32版本
   - Linux 64版本
   - Mac 64版本
3. 创建服务配置文件。

   在配置文件中，指定需要部署的模型和编写的Processor，示例如下。

```
{
 "name": "diy_test",
 "generate_token": "true",
 "model_path": "model.tar.gz", #支持HTTP路径或本地路径。
 "model_entry": "./model_path/",
 "model_config": "{\"model\": \"deploy.prototxt\", \"weight\": \"bvlc_reference_caffenet.caffemodel\"}
",
 "processor_path": "diy_predictor_gpu.tar.gz", #支持HTTP路径或本地路径。
 "processor_entry": "diy_predictor_gpu.so",
 "processor_type": "java",
 "cuda": "/usr/local/cuda"
}
```

相关字段说明请参见命令使用说明。

4. 部署调试。

```
sudo eascmd test service.json
```

6.4.4. 使用Python开发自定义Processor

本文为您介绍如何使用Python开发自定义Processor。

PAI-EAS提供的Python SDK支持TensorFlow、PyTorch及Scikit-Learn等基于Python的开源机器学习框架，也支持集成Pandas等数据分析处理框架，使用PAI-EAS Python SDK，您可以快速将本地预测逻辑直接转换为在线服务。Python SDK中预置了PAI-EAS为AI Inference场景订制的高性能PRC框架及与PAI-EAS集群交互所需要的内部接口，您通过实现简单的接口就可以将模型服务部署至PAI-EAS集群，并使用PAI-EAS提供的模型监控、蓝绿发布、弹性伸缩及VPC高速直连等功能。

如下以简单案例介绍使用Python开发自定义Processor的步骤。

步骤一：构建开发环境

可以使用Pyenv或Conda等Python包管理工具构建开发环境，PAI-EAS提供的客户端工具EASCMD对构建过程进行了封装，您仅执行一条命令即可构建Python SDK开发环境。如果需求进行定制化，则可以选择手工方式构建开发环境。

您可以使用以下任何一种方式构建开发环境：

- EASCMD工具（仅Linux系统）

  EASCMD是PAI-EAS提供的客户端工具，将Python SDK的初始化过程进行了封装。下载该工具后，仅通过一条命令即可完成Python SDK环境初始化，并生成相关的文件模板。

  ```
  # 安装并初始化EASCMD，该示例为安装Linux环境的EASCMD工具。
  $ wget http://eas-data.oss-cn-shanghai.aliyuncs.com/tools/eascmd64
  # 下载完成后，可以修改访问权限，配置阿里云上AccessKey信息。
  $ chmod +x eascmd64
  $ ./eascmd64 config -i <access_id> -k <access_key>
  # 初始化环境。
  $ ./eascmd64 pysdk init ./pysdk_demo
  ```

  输入Python版本（默认为3.6版本），系统会自动创建Python环境ENV目录、预测服务代码模板app.py及服务部署模板app.json。

- 手工方式

如果EASCMD不能满足您的需求或初始化过程中遇到问题，则可以尝试手工初始化开发环境，推荐使用Conda部署环境。

```
mkdir demo
cd demo
# 使用Conda创建Python环境，目录必须指定为ENV。
conda create -p ENV python=2.6
# 安装PAI-EAS Python SDK。
ENV/bin/pip install http://eas-data.oss-cn-shanghai.aliyuncs.com/sdk/allspark-0.9-py2.py3-none-any.whl
# 安装其它依赖包，以TensorFlow 1.14为例。
ENV/bin/pip install tensorflow==1.14
```

如果本地未安装Conda，则必须先执行如下命令，安装Conda环境。

```
$ wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
$ sh Miniconda3-latest-Linux-x86_64.sh
```

- 预构建的开发镜像（推荐）

  PAI-EAS提供预先构建好的开发镜像，该镜像预装了Conda并生成了对应Python版本的ENV环境。PAI-EAS提供的三个预构建镜像如下。

  ```
  # 仅安装了Conda的基础镜像。
  registry.cn-shanghai.aliyuncs.com/eas/eas-python-base-image:latest
  # 安装了Conda及Python2.7和PAI-EAS Allspark 0.8的Python SDK。
  registry.cn-shanghai.aliyuncs.com/eas/eas-python-base-image:py2.7-allspark-0.8
  # 安装了Conda及Python3.6和PAI-EAS Allspark 0.8的Python SDK。
  registry.cn-shanghai.aliyuncs.com/eas/eas-python-base-image:py3.6-allspark-0.8
  ```

  对该镜像直接使用 `run` 命令即可获得Python开发环境，示例如下。

  ```
  $sudo docker run -ti registry.cn-shanghai.aliyuncs.com/eas/eas-python-base-image:py3.6-allspark-0.8
  (/data/eas/ENV) [root@487a04df4b21 eas]#
  (/data/eas/ENV) [root@487a04df4b21 eas]# ENV/bin/python app.py
  [INFO] initialize new lua plugin
  [INFO] loading builtin config script
  [INFO] current meritc id:0
  [INFO] loading builtin lua scripts
  [INFO] Success load all lua scripts.
  [INFO] create service
  [INFO] rpc binds to predefined port 8080
  [INFO] updating rpc port to 8080
  [INFO] install builtin handler call to /api/builtin/call
  [INFO] install builtin handler eastool to /api/builtin/eastool
  [INFO] install builtin handler monitor to /api/builtin/monitor
  [INFO] install builtin handler ping to /api/builtin/ping
  [INFO] install builtin handler prop to /api/builtin/prop
  [INFO] install builtin handler realtime_metrics to /api/builtin/realtime_metrics
  [INFO] install builtin handler tell to /api/builtin/tell
  [INFO] install builtin handler term to /api/builtin/term
  [INFO] Service start successfully
  [INFO] shutting down context ... press Ctrl+C again to force quit
  ```

  基于该基础镜像的ENV环境，您可以安装自己的依赖库（例如TensorFlow 1.12），再将修改过的Container提交为一个数据镜像。

```
ENV/bin/pip install tensorflow==1.12
```

您也可以在Docker外构建ENV开发环境，构建完成后，将其拷贝至任意Docker镜像的*/data/eas/*目录。使用镜像的方式构建开发环境，可以避免每次部署时都将整个ENV环境打包上传，从而提高部署速度。

步骤二：编写预测逻辑

在ENV同级目录下，创建预测服务主文件app.py（PAI-EAS预置的开发镜像中已经预先创建了该模板文件），PAI-EAS提供了如下SDK封装，使用EASCMD初始化环境时，系统会自动生成。

```
# -*- coding: utf-8 -*-
import allspark
class MyProcessor(allspark.BaseProcessor):
  """ MyProcessor is a example
      you can send mesage like this to predict
      curl -v http://127.0.0.1:8080/api/predict/service_name -d '2 105'
  """
  def initialize(self):
      """ load module, executed once at the start of the service
          do service intialization and load models in this function.
      """
      self.module = {'w0': 100, 'w1': 2}
  def pre_process(self, data):
      """ data format pre process
      """
      x, y = data.split(b' ')
      return int(x), int(y)
  def post_process(self, data):
      """ process after process
      """
      return bytes(data, encoding='utf8')
  def process(self, data):
      """ process the request data
      """
      x, y = self.pre_process(data)
      w0 = self.module['w0']
      w1 = self.module['w1']
      y1 = w1 * x + w0
      if y1 >= y:
          return self.post_process("True"), 200
      else:
          return self.post_process("False"), 400
if __name__ == '__main__':
  # paramter worker_threads indicates concurrency of processing
  runner = MyProcessor(worker_threads=10)
  runner.run()
```

上述代码即为Python SDK的简单示例，您需要继承PAI-EAS提供的基类BaseProcessor，实现initialize()和process()函数。其process()函数的输入输出均为BYTES类型，输出参数分别为response_data和status_code，正常请求status_code可以返回0或200。

函数	功能描述	参数描述
init(worker_threads=5, worker_processes=1,endpoint=None)	Processor构建函数。	• worker_threads：Worker线程数，默认值为5。 • worker_processes：进程数，默认值为1。如果worker_processes为1，则表示单进程多线程模式。如果worker_processes大于1，则worker_threads只负责读取数据，请求由多进程并发处理，每个进程均会执行initialize()函数。 • endpoint：服务监听的Endpoint，通过该参数可以指定服务监听的地址和端口，例如endpoint='0.0.0.0:8079'。
initialize()	Processor初始化函数。服务启动时，进行模型加载等初始化工作。	无参数。
process(data)	请求处理函数。每个请求会将Request Body作为参数传递给process()进行处理，并将函数返回值返回至客户端。	data为Request Body，类型为BYTES。返回值也为BYTES类型。
run()	启动服务。	无参数。

步骤三：本地测试服务

```
./ENV/bin/python app.py
curl http://127.0.0.1:8080/test  -d '10 20'
```

步骤四：发布线上服务

1. 打包代码。

   您可以通过以下任何一种方式操作：

   - 打包完整环境及代码

     您可以通过EASCMD或手动打包完整环境：

     - EASCMD封装的打包命令。

       ```
       $ ./eascmd64 pysdk pack ./demo
       [PYSDK] Creating package: /home/xingke.lwp/code/test/demo.tar.gz
       ```

- 手动打包（支持.zip或.tar.gz压缩包）。

  打包的根目录必须为*/ENV*目录。已经将该示例的完整.tar.gz包上传至OSS（下载完整包），您可以使用如下服务配置文件部署服务。

  ```
  {
   "name": "pysdk_demo",
   "processor_entry": "./app.py",
   "processor_type": "python",
   "processor_path": "oss://eas-model-beijing/1955570263925790/pack.tar.gz",
   "metadata": {
    "instance": 1,
     "memory": 2000,
     "cpu": 1
    }
  }
  ```

- 通过镜像上传环境（推荐）

  通常使用Conda生成的Python ENV环境文件较大，在开发部署过程中，每次都打包并上传环境会浪费大量时间和存储资源。PAI-EAS提供了Data Image部署方式，您可以基于预构建的镜像构建ENV环境，并安装所需的Python依赖包，之后将Container提交为自己的数据镜像并上传至镜像仓库，示例如下。

  ```
  sudo docker commit 487a04df4b21 registry.cn-shanghai.aliyuncs.com/eas-service/develop:latest
  sudo docker push registry.cn-shanghai.aliyuncs.com/eas-service/develop:latest
  ```

  每次只需要将app.py相关的文件打包上传至OSS即可。您可以使用如下服务描述文件部署服务。

  ```
  {
   "name": "pysdk_demo",
   "processor_entry": "./service.py",
   "processor_type": "python",
   "processor_path": "http://eas-data.oss-cn-shanghai.aliyuncs.com/demo/service.py",
   "data_image": "registry.cn-shanghai.aliyuncs.com/eas-service/develop:latest",
   "metadata": {
    "instance": 1,
     "memory": 2000,
     "cpu": 1
    }
  }
  ```

2. 部署服务。

```
$ ./eascmd64 create app.json
[RequestId]: 1202D427-8187-4BCB-8D32-D7096E95B5CA
+------------------+-------------------------------------------------------------------+
| Intranet Endpoint | http://1828488879222746.vpc.cn-beijing.pai-eas.aliyuncs.com/api/predict/pysdk_demo |
|         Token | ZTBhZTY3ZjgwMmMyMTQ5OTgyMTQ5YmM0NjdiMmNiNmJkY2M5ODI0Zg==       |
+------------------+-------------------------------------------------------------------+
[OK] Waiting task server to be ready
[OK] Fetching processor from [oss://eas-model-beijing/1955570263925790/pack.tar.gz]
[OK] Building image [registry-vpc.cn-beijing.aliyuncs.com/eas/pysdk_demo_cn-beijing:v0.0.1-20190806082810]
[OK] Pushing image [registry-vpc.cn-beijing.aliyuncs.com/eas/pysdk_demo_cn-beijing:v0.0.1-20190806082810]
[OK] Waiting [Total: 1, Pending: 1, Running: 0]
[OK] Service is running
# 测试服务。
$ curl http://1828488879222746.vpc.cn-beijing.pai-eas.aliyuncs.com/api/predict/pysdk_demo -H 'Authorization: ZTBhZTY3ZjgwMmMyMTQ5OTgyMTQ5YmM0NjdiMmNiNmJkY2M5ODI0Zg==' -d 'hello eas'
```

6.4.5. 常见问题

本文为您介绍自定义Processor的相关问题。

- 自定义Processor压缩包太大，如何提高上传效率？
- 开发Processor过程中，已经安装的Python包，为什么上线之后无法找到？
- PAI-EAS线上运行过程中，为什么无法找到ENV环境的某些系统依赖库？
- 如何解决导入cv2报错libSM.so.6: cannot open shared object file: No such file or directory的问题？
- 在PAI-EAS Python Processor中，如何配置环境变量？
- PAI-EAS Processor如何避免代码异常造成进程退出？
- 如何配置AccessKey和Endpoint？
- 使用EASCMD创建任务，为什么一直处于[OK] Waiting [Total: 1, Pending: 1, Running: 0]状态？

自定义Processor压缩包太大，如何提高上传效率？

PAI-EAS推荐使用Docker镜像上传运行环境和大文件。如果业务代码产生变动，则只需要增量更新改动部分即可，可以大幅度提升开发效率，详情请参见使用Python开发自定义Processor。

开发Processor过程中，已经安装的Python包，为什么上线之后无法找到？

因为Python Processor ENV环境的构建依赖于Conda，所以在开发环境中能够正常运行的程序，很可能依赖于用户 .local环境下的Python包。为保证Python程序所有依赖均已安装，在ENV环境使用 `pip install --force-reinstall` 命令，确保依赖包的安装位置正确。

PAI-EAS线上运行过程中，为什么无法找到ENV环境的某些系统依赖库？

由于用户的开发环境无法预测，很可能与PAI-EAS线上环境存在区别。为保证开发环境与线上环境一致，PAI-EAS提供了预先构建的开发镜像，用于配置Python Processor的ENV环境。开发镜像中已预装Conda并生成了对应Python版本的ENV环境，PAI-EAS提供的开发镜像如下。

```
# 仅安装了Conda的基础镜像。
registry.cn-shanghai.aliyuncs.com/eas/eas-python-base-image:latest
# 安装了Conda、Python 2.7及PAI-EAS Allspark 0.8的Python SDK。
registry.cn-shanghai.aliyuncs.com/eas/eas-python-base-image:py2.7-allspark-0.8
# 安装了Conda、Python 3.6及PAI-EAS Allspark 0.8的Python SDK。
registry.cn-shanghai.aliyuncs.com/eas/eas-python-base-image:py3.6-allspark-0.8
```

如何解决导入cv2报错 libSM.so.6: cannot open shared object file: No such file or directory 的问题？

常见的 pip install opencv 方法依赖于libXext、libSM及libXrender库。因为PAI-EAS线上环境没有预装libXext、libSM及libXrender库，所以使用该命令安装的cv2在离线测试环境中可能正常运行，但是在线上环境可能运行失败。您可以通过以下任何一种方式解决该报错：

- 使用 pip install openv-python-headless 安装cv2，该方法不依赖于额外安装的libXext、libSM及libXrender库。
- 查找系统中已安装的libXext、libSM及libXrender的二进制库ISO文件，将其拷贝至*ENV/lib*并随Proccessor上传。（这种方法可能出现其他依赖库，需要根据实际情况解决。）

在PAI-EAS Python Processor中，如何配置环境变量？

因为PAI-EAS Python Processor会自动将Processor目录下的所有目录添加至LD_LIBRARY_PATH环境变量，所以将额外的依赖库存放至Processor目录的任意位置即可。如果是其他环境变量，则推荐在Python程序中使用 os.environ[‘key’]=‘val’ 进行配置。

PAI-EAS Processor如何避免代码异常造成进程退出？

在开发业务服务逻辑过程中，在重要的地方增加异常检测机制（try-catch），以保证业务代码不会因为异常而退出。即使由于某些特殊原因导致进程退出，PAI-EAS可以自动重启退出的进程，从而保证服务的稳定性。

如何配置AccessKey和Endpoint？

预测服务使用阿里云AccessKey进行身份认证，提交任务时需要使用AccessKey ID和AccessKey Secret。Endpoint默认地域为华东2（上海），如果需要将模型部署至其它地域，可以使用-e参数指定地域对应的Endpoint，示例如下。

```
./eascmd64 config -i <yourAccessKey ID> -k <yourAccessKey Secret> -e pai-eas.cn-beijing.aliyuncs.com
```

使用EASCMD创建任务，为什么一直处于 [OK] Waiting [Total: 1, Pending: 1, Running: 0] 状态？

创建任务所需的资源可能不足或无法找到。首先需要确认配置的资源地域及资源名称是否正确，然后确认是否有足够的可用资源。如下的PAI-EAS信息描述文件可以查看相关配置。

```
{
 "name": "service",
 "token": "[Authorization-token]",
 "data_image":"[your-public-docker-image-repo]",
 "processor_entry": "app.py",
 "processor_type": "python",
 "processor_path": "[oss://eas-model-shenzhen/xxxxxxxxx/codes.tar.gz]",
 "metadata": {
  "region": "cn-shenzhen", # 确保资源组所在的地域正确。
  "resource": "resource-name", # 确保资源组名称（类似:EAS-LsFlrwBP56）正确且与地域对应。
  "gpu": 1,
  "cpu": 6,
  "memory": 2000,
  "instance": 2, # 每个实例占用（gpu=1, cpu=6, memory=2000B）的资源。
        # 如果可用资源不足，则该任务一直处于waiting状态。
  "cuda": "10.0"
 }
}
```

6.5. 定时自动部署模型服务

本文为您介绍如何搭配使用PAI-EAS和DataWorks，进行模型服务定时部署。

前提条件

- 购买DataWorks**独享调度**资源组，详情请参见DataWorks独享资源。
- 购买PAI，详情请参见开通。
- 完成实验的定时训练，并将模型保存至固定地址，详情请参见离线调度。
- 如果RAM用户进行定时自动模型部署，则需要对其赋予DataWorks相关权限及PAI-EAS模型部署权限，详情请参见RAM用户使用DataWorks和授权。

步骤一：创建独享调度资源组

1. 登录DataWorks控制台。
2. 在左侧导航栏，单击**资源组列表**。
3. 在**独享资源组**页签，单击**创建调度资源组**。
4. 在**创建独享资源组**面板，配置参数。

参数	描述
资源组类型	选择**独享调度资源组**。
资源组名称	资源组名称，必须在租户内唯一。 **说明** 租户即阿里云账号，一个租户（阿里云账号）下可以有多个RAM用户。
资源组备注	资源组的简单描述，便于区分各资源组。

参数	描述
订单号	选择已购买的独享资源组订单号。如果未购买，请单击**购买**。

5. 单击**确定**。

> **说明** 独享资源组在20分钟内完成环境初始化，请耐心等待其状态更新为运行中。

步骤二：绑定归属的工作空间

独享调度资源组需要绑定归属的工作空间，才可以在对应的工作空间下选择该资源组。

1. 在**独享资源组**页签，单击相应资源组后的**修改归属工作空间**。
2. 在**修改归属工作空间**对话框的**分配到工作空间**区域，单击目标工作空间**操作**列下的**绑定**。

步骤三：安装部署工具包EASCMD

安装EASCMD工具包，需要联系管理员授权。

1. 创建命令。
   i. 在**资源组列表**的**独享资源组**页签，单击目标资源组**操作**列下的**运维助手**。
   ii. 在运维助手页面，单击**创建命令**。
   iii. 在**创建命令**页面，配置参数。

参数	描述
命令名称	命令的名称。
命令描述	命令的简要描述，便于区分各命令。
命令类型	系统默认Shell类型，不支持修改。

命令内容	输入如下命令。
	`wget -P /home/admin/usertools/tools/ http://eas-data.oss-cn-shanghai.aliyuncs.com/tools/eascmd`
	`chmod +x /home/admin/usertools/tools/eascmd`

安装目录	安装至*/home/admin/usertools/tools/*目录。
超时时间	命令执行的超时时间，单位为秒。如果命令执行超时，则系统强制结束命令。建议配置为60秒。

   iv. 单击**创建**。
2. 执行命令。
   i. 在运维助手页面，单击上一步已创建命令**操作**列下的**运行命令**。

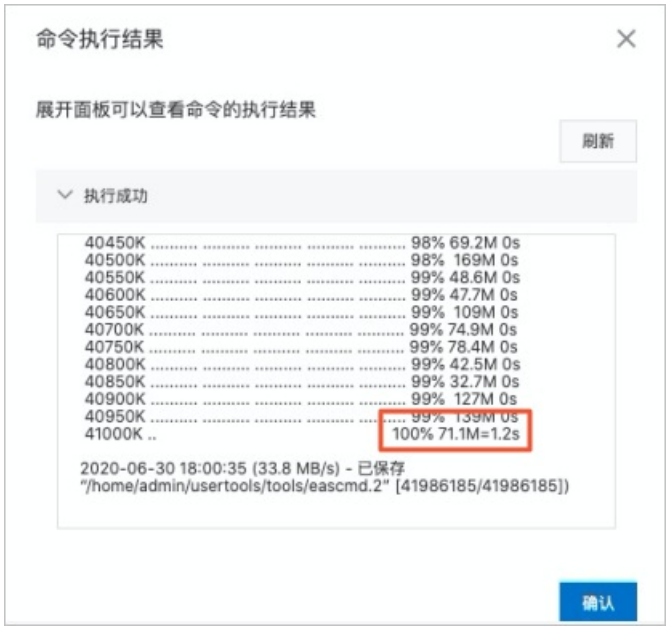
ii. 在**运行命令**面板，选择**基线命令**为**是**，**运行成功后纳入环境基线**，其他参数使用默认配置。

iii. 单击**运行**。

3. 查看命令执行情况。

i. 在运维助手页面下方的列表，单击相应命令后的**查看结果**。

ii. 在**命令执行结果**对话框，查看命令执行情况。如果执行进度为100%，则EASCMD工具包安装成功。



步骤四：创建工作流

1. 进入**数据开发**页面。

i. 登录DataWorks控制台。

ii. 在左侧导航栏，单击**工作空间列表**。

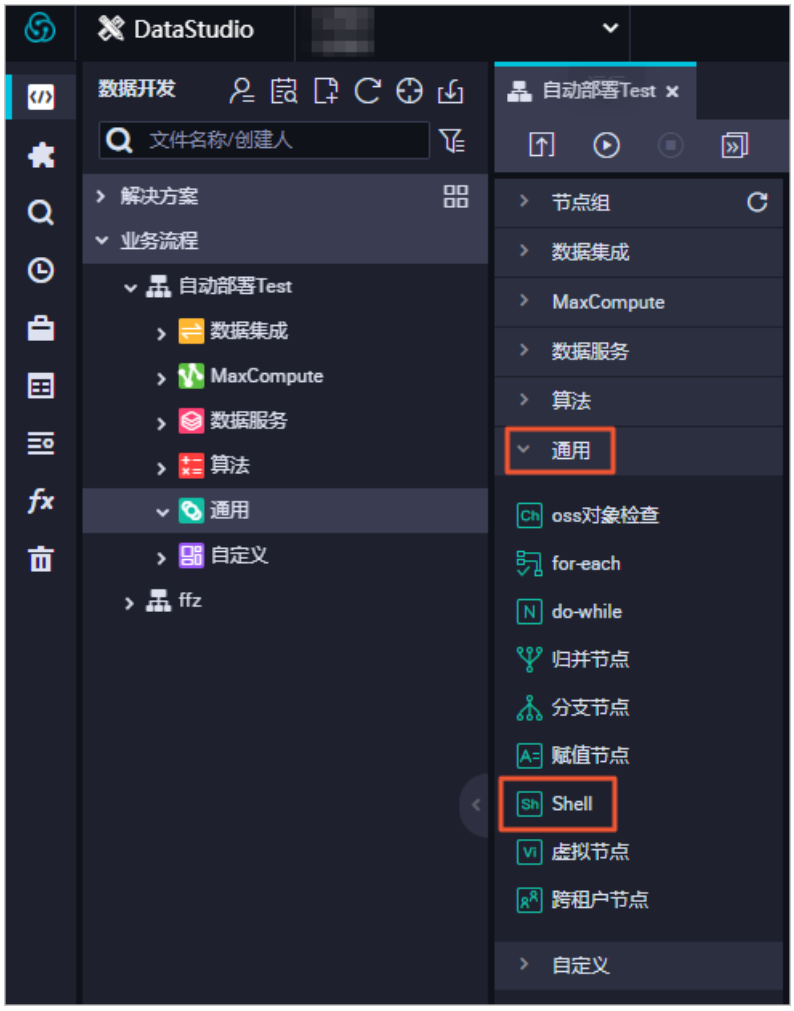
iii. 选择工作空间所在地域后，单击相应工作空间后的**进入数据开发**。

2. 鼠标悬停至图标，单击**业务流程**。

3. 在**新建业务流程**对话框，输入**业务名称**和**描述**。

> 注意　业务名称必须是大小写字母、中文、数字、下划线（_）及小数点（.），且不能超过128个字符。

4. 单击**新建**。
5. 在业务流程页面，拖拽**通用 > Shell**至右侧画布。



6. 在**新建节点**页面的**节点名称**文本框，输入**部署节点**。
7. 单击**提交**。

步骤五：部署初始模型

因为定时自动部署是在初始模型服务基础上，增加一个模型服务版本，作为线上运行服务。所以定时自动部署之前，需要先部署初始模型。如果已经存在初始模型服务，则直接执行步骤六。

1. 编辑部署脚本。
   i. 在业务流程页面，双击已创建的Shell节点（部署节点）。
   ii. 在Shell节点页面，输入如下命令。

```
# 编写服务部署描述文件。
cat << EOF > echo.json
{
 "name": "yourModelName",
 "generate_token": "true",
 "model_path": "yourModelAdress",
 "processor": "yourProcessorType",
 "metadata": {
  "instance": 1, #可以根据实际情况修改Instance数量。
  "cpu": 2 #可以根据实际情况修改CPU数量。
 }
}
EOF
# 执行部署命令。
/home/admin/usertools/tools/eascmd -i <yourAccessKeyID> -k <yourAccessKeySecret>  -e pai-eas.cn-shanghai.aliyuncs.com create echo.json
```

   echo.json是描述服务相关信息（模型存储位置及使用资源等）的JSON文件，以下参数需要根据实际情况配置：

   - name：模型服务名称必须在同地域内唯一，是模型服务的唯一标识，可以结合业务含义命名。
   - model_path：训练模型的存储位置，支持HTTP和OSS地址。

     如果使用HTTP地址，则所需的文件必须为TAR、GZ、BZ2及ZIP等压缩包格式。如果使用OSS地址，则可以指定压缩包文件路径或目录路径。使用OSS地址时，需要指定Endpoint，即在上述服务部署描述文件中增加 `"oss_endpoint":"oss-cn-beijing.aliyuncs.com",` 代码行（按照实际情况修改地域）。

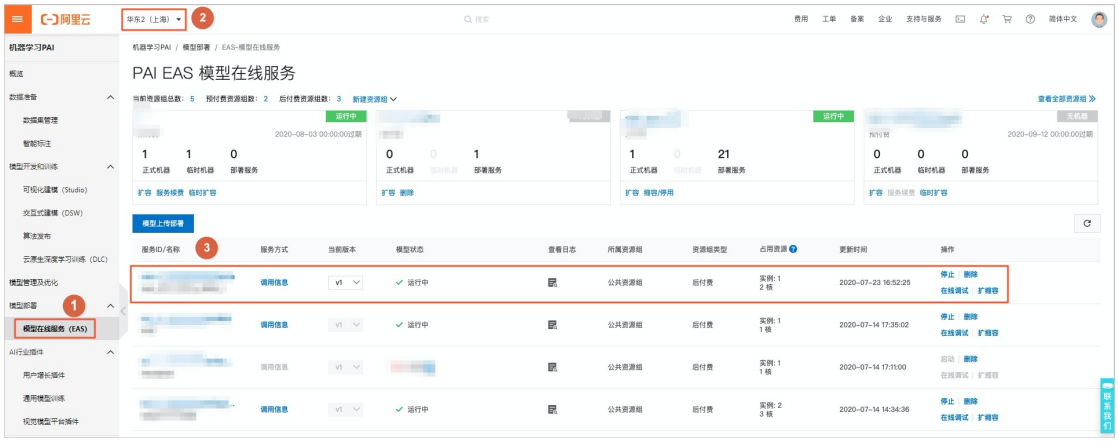
     > **说明** 使用OSS存储模型时，需要为PAI赋予OSS访问权限，详情请参见OSS授权。

   - processor：Processor类型。
   - metadata：服务的Meta信息，需要根据实际情况修改。详细的字段信息请参见命令使用说明。
   - yourAccessKeyID：Access Key ID。
   - yourAccessKeySecret：Access Key Secret。

- Endpoint：部署命令 `-e` 后的Endpoint参数与地域的对应关系如下。

地域	Endpoint
华东2（上海）	pai-eas.cn-shanghai.aliyuncs.com
华北2（北京）	pai-eas.cn-beijing.aliyuncs.com
华东1（杭州）	pai-eas.cn-hangzhou.aliyuncs.com
华南1（深圳）	pai-eas.cn-shenzhen.aliyuncs.com
中国（香港）	pai-eas.cn-hongkong.aliyuncs.com
新加坡（新加坡）	pai-eas.ap-southeast-1.aliyuncs.com
印度（孟买）	pai-eas.ap-south-1.aliyuncs.com
印度尼西亚（雅加达）	pai-eas.ap-southeast-5.aliyuncs.com
德国（法兰克福）	pai-eas.eu-central-1.aliyuncs.com

2. 运行脚本。
   i. 在Shell节点页面，单击页面上方的图标。
   ii. 在**警告**对话框，单击**继续运行**。
   iii. 在**运行参数**页面，选择**调度资源组**为已创建的独享资源组。
   iv. 单击**确定**。运行完成后，即可生成一个线上的模型服务。您可以参见以下步骤，在PAI控制台查看该模型服务。
3. （可选）查看部署的模型服务。
   i. 登录PAI控制台。
   ii. 在左侧导航栏，选择**模型部署 > 模型在线服务（EAS）**。
   iii. 在PAI控制台页面的左上方，选择相应的地域。
   iv. 在**PAI EAS模型在线服务**页面，查看已部署的模型服务。



后续步骤中，将不断对该模型服务增加服务版本，从而实现模型定时部署。

步骤六：编辑定时自动部署脚本

1. 编辑步骤五中的Shell节点脚本，示例如下（如果已执行步骤五，则前13行代码无需修改。如果未执行步骤五，则需要根据实际情况修改前13行代码中的参数值）。

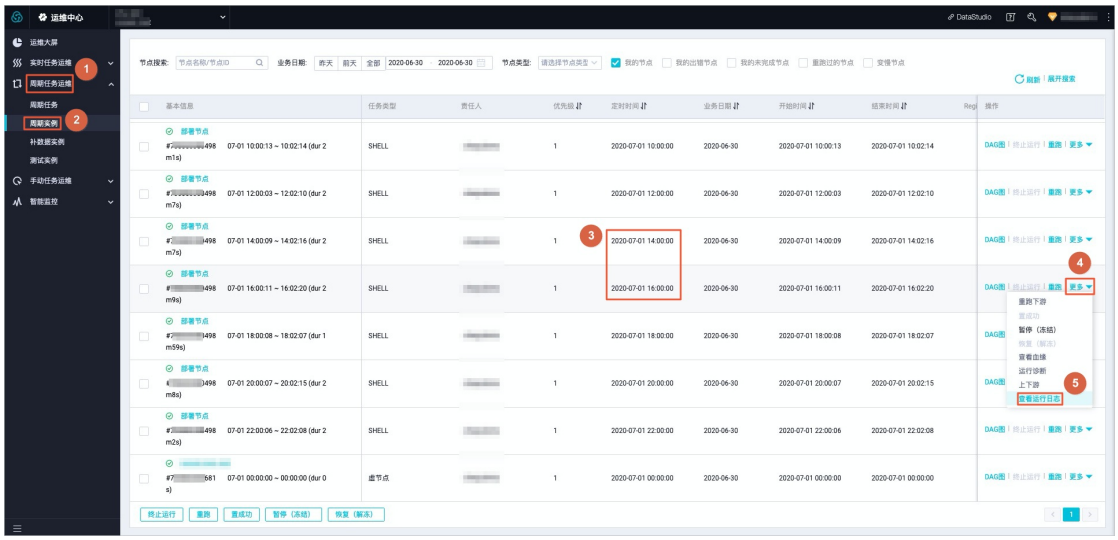
```
# 编写服务部署描述文件。
cat << EOF > echo.json
{
 "name": "yourModelName",
 "generate_token": "true",
 "model_path": "yourModelAdress",
 "processor": "yourProcessorType",
 "metadata": {
  "instance": 1,
  "cpu": 2
 }
}
EOF #第13行代码。
# 执行模型更新部署。每执行一次定时部署调度，系统就会在原模型服务基础上，增加一个模型服务版本，作为最新的线上运行服务。
/home/admin/usertools/tools/eascmd -i <yourAccessKeyID> -k <yourAccessKeySecret> -e pai-eas.cn-shanghai.aliyuncs.com modify <yourModelName> -s echo.json
# 此处可以编写服务的测试逻辑。
# 如果测试服务发生异常，则使用下述命令回滚模型服务。
#/home/admin/usertools/tools/eascmd -i <yourAccessKeyID> -k <yourAccessKeySecret> -e pai-eas.cn-shanghai.aliyuncs.com version -f <The name of the model to be rolled back> 1
```

其中的参数解释请参见步骤五：部署初始模型。

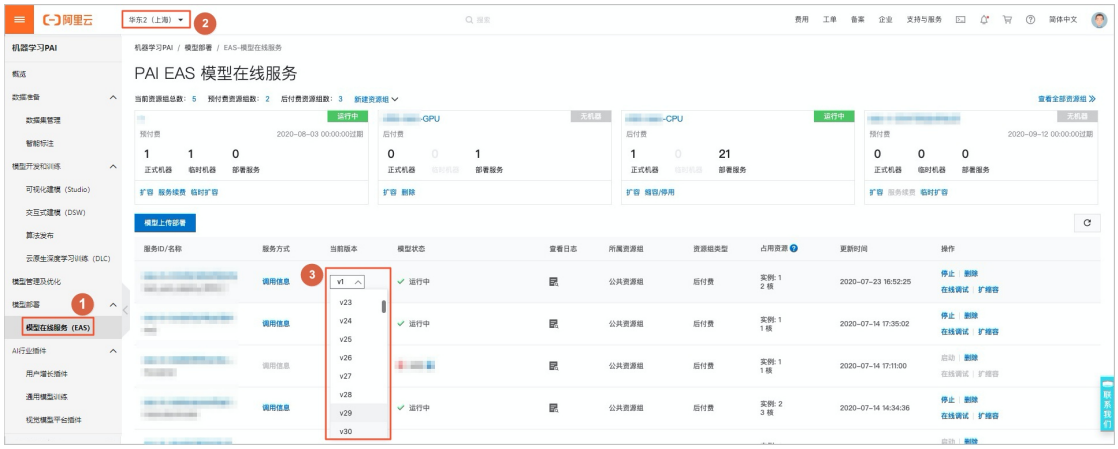
步骤七：执行定时调度

1. 执行调度任务。
   i. 在Shell节点页面，单击页面右侧的**调度配置**。
   ii. 在**调度配置**页面的**时间属性**区域，选择**调度周期**。
   iii. 在**调度依赖**区域，单击**依赖的上游节点**后的**使用工作空间根节点**。
   iv. 配置依赖关系，详情请参见配置调度依赖。
   v. 单击Shell节点页面上方的图标，保存配置。
   vi. 单击Shell节点页面上方的图标，提交调度任务。
2. 查看定时调度的运行实例。
   i. 在Shell节点页面，单击右上方的**运维中心**。
   ii. 在**运维中心**页面，选择**周期任务运维 > 周期实例**。
   iii. 在实例详情页面，查看模型自动部署的**定时时间**。

iv. 选择操作列下的**更多 > 查看运行日志**，查看每次定时部署的运行日志。



3. 查看定时部署的模型服务。
   i. 登录PAI控制台。
   ii. 在左侧导航栏，选择**模型部署 > EAS-模型在线服务**。
   iii. 在PAI控制台页面的左上方，选择相应的地域。
   iv. 在**PAI EAS模型在线服务**页面，从**当前版本**列下的列表，查看模型服务自动更新的所有历史版本。



6.6. 开通服务监控报警

通过服务监控报警功能，您可以监控服务运行情况。如果服务运行情况超过了配置的报警规则（条件），则发送报警通知。

背景信息

PAI-EAS支持对服务的以下项目进行监控报警。

监控项目	描述
CPU消耗	服务当前消耗的CPU核数。

监控项目	描述
GPU利用率	服务当前GPU使用量占部署GPU总量的比重。
内存消耗	服务当前内存消耗，单位MB。
每秒总调用次数	服务每秒总调用次数。
状态码2xx每秒响应	状态码为2xx码的每秒响应。
状态码2xx响应占比	状态码为2xx码的响应占比。
状态码4xx每秒响应	状态码为4xx码的每秒响应。
状态码4xx响应占比	状态码为4xx码的响应占比。
状态码5xx每秒响应	状态码为5xx码的每秒响应。
状态码5xx响应占比	状态码为5xx码的响应占比。
入流量	每秒进入服务的数据量，单位KB。
出流量	每秒流出服务的数据量，单位KB。

步骤一：配置报警联系人

1. 创建报警联系人。
   i. 登录云监控控制台。
   ii. 在左侧导航栏，单击**报警服务 > 报警联系人**。
   iii. 在**报警联系人**页面，单击**新建联系人**。
   iv. 在**设置报警联系人**面板，输入报警联系人姓名、手机号码、旺旺、邮箱和钉钉机器人。当您输入手机号码和邮箱时，需要进行验证，防止由于信息填写错误，而导致无法及时收到报警通知。
   v. 单击**确认**。
2. 创建报警联系组。
   i. 在**报警联系人**页面，单击**报警联系组**页签。
   ii. 在**报警联系组**页面，单击**新建联系组**。
   iii. 在**新建联系组**面板，输入**组名**，并选择**已有联系人**，其他参数使用默认配置。
   iv. 单击**确认**。

步骤二：配置报警规则

1. 在云监控控制台的左侧导航栏，单击**报警服务 > 报警规则**。
2. 在**报警规则列表**页面，单击**创建报警规则**。
3. 在**创建报警规则**页面，配置关联资源、报警规则及通知方式。

参数	描述
产品	云监控管理的产品名称，选择为**PAI-EAS在线预测服务**。

参数	描述
资源范围	报警规则的作用范围，分为服务和全部资源： ○ **全部资源**：PAI-EAS的任何服务满足报警规则，都会发送报警通知。 ○ **服务**：仅选中的单个或多个服务满足警报规则时，才发送警报通知。
规则名称	报警规则的名称。
规则描述	报警规则的主体，定义在监控数据满足指定条件时，触发报警规则。 **说明** 规则描述中的监控项（CPU消耗、GPU利用率及内存消耗等）单位需要与PAI-EAS服务监控页面的图表纵坐标单位一致。
通道沉默周期	指报警发生后如果未恢复正常，间隔多久重复发送一次报警通知。
生效时间	报警规则的生效时间，报警规则只在生效时间内才会检查监控数据是否需要报警。
通知对象	发送报警的联系人组，选择为已绑定报警联系人的报警组。
报警级别	支持以下三种级别： ○ **电话+短信+邮件+钉钉机器人（Critical）** ○ **短信+邮件+钉钉机器人（Warning）** ○ **邮件+钉钉机器人（Info）**
弹性伸缩	如果您选中**弹性伸缩**复选框，当报警发生时，会触发相应的伸缩规则。您需要配置弹性伸缩的**地域**、**弹性伸缩组**及**弹性伸缩规则**。 ○ 创建弹性伸缩组的操作方法，请参见创建伸缩组。 ○ 创建弹性伸缩规则的操作方法，请参见创建伸缩规则。
日志服务	如果您选中**日志服务**复选框，当报警发生时，会将报警信息写入日志服务。您需要配置日志服务的**地域**、**Project**及**Logstore**。 创建Project和Logstore的操作方法，请参见快速入门。
邮件备注	自定义报警邮件补充信息。配置**邮件备注**后，发送报警的邮件通知中会附带您的备注信息。
报警回调	可以访问的公网URL，云监控会将报警信息通过POST请求推送至该地址，仅支持HTTP协议。

4. 单击**确认**。

7.模型服务调用

7.1. 服务调用方式

7.1.1. 公网地址调用

PAI-EAS支持使用通用公网或API网关的公网对服务进行公网地址调用，通用公网调用又包括使用官方的Python SDK、Java SDK及自行实现调用逻辑三种方式。本文介绍详细介绍每种调用方式的实现方法。

PAI-EAS服务支持以下公网（普通公共网络，区别于VPC网络）调用方式：

- 通用公网调用
  - 官方SDK调用（Python）
  - 官方SDK调用（Java）
  - 自行实现调用逻辑
- 通过API网关的公网调用

如果没有特殊要求，则使用通用公网调用即可。如果您通过阿里云API网关创建API并生成公网调用地址，则可以使用API网关公网调用。

> **说明** 通过API网关的公网调用可能产生API网关费用，详情请参见API网关定价。

通用公网调用

在PAI-EAS部署模型服务后，系统会自动生成一个公网调用的服务地址。您可以在**PAI EAS模型在线服务**页面，单击待调用服务**服务方式**列下的**调用信息**，查看公网**访问地址**和**Token**。通过该调用信息可以进行调用测试，示例如下。

```
$ curl http://166408185518****.cn-hangzhou.pai-eas.aliyuncs.com/api/predict/heart_predict_online -H 'Authorization: ZGI1YWYxNmQwYjMzMDM1YTNlMmFmNmEzYjIzZTVlZGQ0NDJhYTRm****' -d '[{"sex":0,"cp":0,"fbs":0,"restecg":0,"exang":0,"slop":0,"thal":0,"age":0,"trestbps":0,"chol":0,"thalach":0,"oldpeak":0,"ca":0}]'
```

调用测试成功后，即可调用服务。为方便您调用服务，PAI-EAS提供了以下三种通用公网调用方式：

- 使用官方SDK调用（Python）

  PAI-EAS封装了调用逻辑并提供了Python官方SDK。
- 使用官方SDK调用（Java）

  PAI-EAS封装了调用逻辑并提供了Java官方SDK。
- 自行实现调用逻辑

  推荐使用以上两种官方SDK调用服务，从而有效减少编写调用逻辑的时间并提高调用稳定性。如果您需要使用其他语言或自己编写实现调用逻辑，下文也提供了调用Demo供您参考。此外，自行实现调用逻辑需要根据不同框架构建服务请求，详情请参见构建通用Processor服务请求。

使用官方SDK调用（Python）

1. 安装。

   ```
   pip install -U eas-prediction --user
   ```

该Python DSK的调用详情请参见GitHub官网。

2. 编写调用程序。

   以字符串作为输入输出的程序为例，其他格式（TensorFlow或PyTorch等）的输入输出程序示例请参见GitHub官网。

```
#!/usr/bin/env python
from eas_prediction import PredictClient
from eas_prediction import StringRequest
if __name__ == '__main__':
  #下面的client = PredictClient()入参源于公网调用的访问地址。
  client = PredictClient('http://166408185518****.cn-hangzhou.pai-eas.aliyuncs.com','heart_predict_online')
  #Token可以在公网地址调用信息中获取，详情请参见通用公网调用部分。
  client.set_token('ZGI1YWYxNmQwYjMzMDM1YTNlMmFmNmEzYjIzZTVlZGQ0NDJhYTRm****')
  client.init()
  #输入请求需要根据模型进行构造，此处仅以字符串作为输入输出的程序为例。
  request = StringRequest('[{"sex":0,"cp":0,"fbs":0,"restecg":0,"exang":0,"slop":0,"thal":0,"age":0,"trestbps":0,"chol":0,"thalach":0,"oldpeak":0,"ca":0}]')
  for x in range(0, 1):
    resp = client.predict(request)
    print(resp)
```

   其中 `client = PredictClient()` 的函数入参源于公网调用的访问地址。例如公网调用的访问地址为http://166408185518****.cn-hangzhou.pai-eas.aliyuncs.com/api/predict/heart_predict_online，则 `PredictClient()` 函数的调用格式为 `client = PredictClient('http://166408185518****.cn-hangzhou.pai-eas.aliyuncs.com','heart_predict_online')` 。

3. 执行调用程序。

```
$ python heart_predict.py
```

   其中 `heart_predict.py` 为Python程序的文件名，需要根据实际情况修改。获得的预测结果如下。

```
IT-C02YJ0V8JHD2:Desktop wowei$ python heart_predict.py
[{"p_0":0.9941226679708888,"p_1":0.005877332029111252}]
```

使用官方SDK调用（Java）

1. 添加依赖包。

   因为编写Java客户端代码需要使用Maven管理项目，所以需要在pom.xml文件中添加客户端依赖包eas-sdk。最新Release版本为2.0.1，代码示例如下。

```
<dependency>
 <groupId>com.aliyun.openservices.eas</groupId>
 <artifactId>eas-sdk</artifactId>
 <version>2.0.1</version>
</dependency>
```

   该Java SDK调用详情请参见GitHub官网。

2. 编写调用程序。

   以字符串作为输入输出的程序为例，其他格式（TensorFlow或PyTorch等）的输入输出程序示例请参见GitHub官网。

```
import com.aliyun.openservices.eas.predict.http.PredictClient;
import com.aliyun.openservices.eas.predict.http.HttpConfig;
public class Test_String {
  public static void main(String[] args) throws Exception{
    //启动并初始化客户端。
    PredictClient client = new PredictClient(new HttpConfig());
    //Token可以在公网地址调用信息中获取，详情请参见通用公网调用部分。
    client.setToken("ZGI1YWYxNmQwYjMzMDM1YTNlMmFmNmEzYjIzZTVlZGQ0NDJhYTRm****");
    //下面的Endpoint信息源于公网调用的访问地址。
    client.setEndpoint("http://166408185518****.cn-hangzhou.pai-eas.aliyuncs.com");
    //配置服务名称。
    client.setModelName("heart_predict_online");
    //定义输入字符串。输入请求需要根据模型进行构造，此处仅以字符串作为输入输出的程序为例。
    String request = "[{\"sex\":0,\"cp\":0,\"fbs\":0,\"restecg\":0,\"exang\":0,\"slop\":0,\"thal\":0,\"age\":0
,\"trestbps\":0,\"chol\":0,\"thalach\":0,\"oldpeak\":0,\"ca\":0}]";
    System.out.println(request);
    //通过PAI-EAS返回字符串。
    try {
      String response = client.predict(request);
      System.out.println(response);
    } catch(Exception e) {
      e.printStackTrace();
    }
    //关闭客户端。
    client.shutdown();
    return;
  }
}
```

其中 client.setEndpoint() 和 client.setModelName() 的函数入参源于公网调用的访问地址。例如公网调用的访问地址为http://166408185518****.cn-hangzhou.pai-eas.aliyuncs.com/api/predict/heart_predict_online，则 client.setEndpoint() 函数的调用格式为 client.setEndpoint("http://166408185518****.cn-hangzhou.pai-eas.aliyuncs.com")，client.setModelName() 函数的调用格式为 client.setModelName("heart_predict_online")。

3. 执行调用程序。

   执行调用逻辑程序可以获得如下结果。

```
[{"p_0":0.9941226679708888,"p_1":0.005877332029111252}]
```

自行实现调用逻辑

PAI-EAS支持使用Python、Java或其他语言自行实现调用逻辑，您需要根据不同框架构建服务请求，详情请参见构建通用Processor服务请求。编写调用逻辑的示例如下。

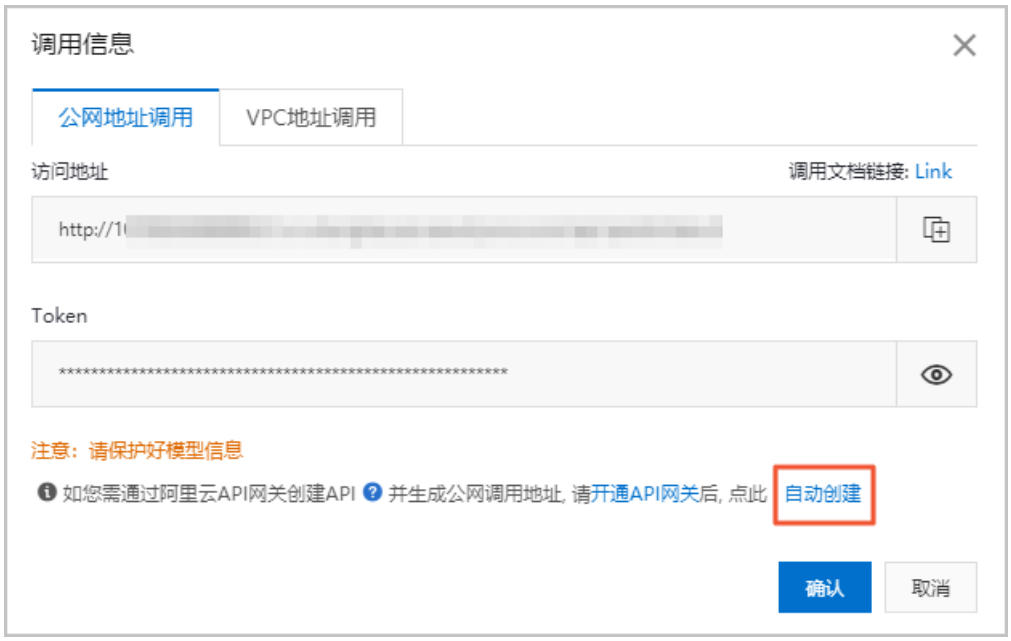
```
import requests
#URL信息可以在公网地址调用信息中获取，详情请参见通用公网调用部分。
url = 'http://166408185518****.cn-hangzhou.pai-eas.aliyuncs.com/api/predict/heart_predict_online'
#header信息（Token）可以在公网地址调用信息中获取，详情请参见通用公网调用部分。
headers = {"Authorization": 'ZGI1YWYxNmQwYjMzMDM1YTNlMmFmNmEzYjIzZTVlZGQ0NDJhYTRm****'}
#根据具体模型要求的数据格式构造服务请求，此处仅以字符串作为输入输出的程序为例。
data = '[{"sex":0,"cp":0,"fbs":0,"restecg":0,"exang":0,"slop":0,"thal":0,"age":0,"trestbps":0,"chol":0,"thalach":0,"oldpeak":0,"ca":0}]'
resp = requests.post(url, data=data, headers=headers)
print resp
print resp.content
```

通过API网关的公网调用

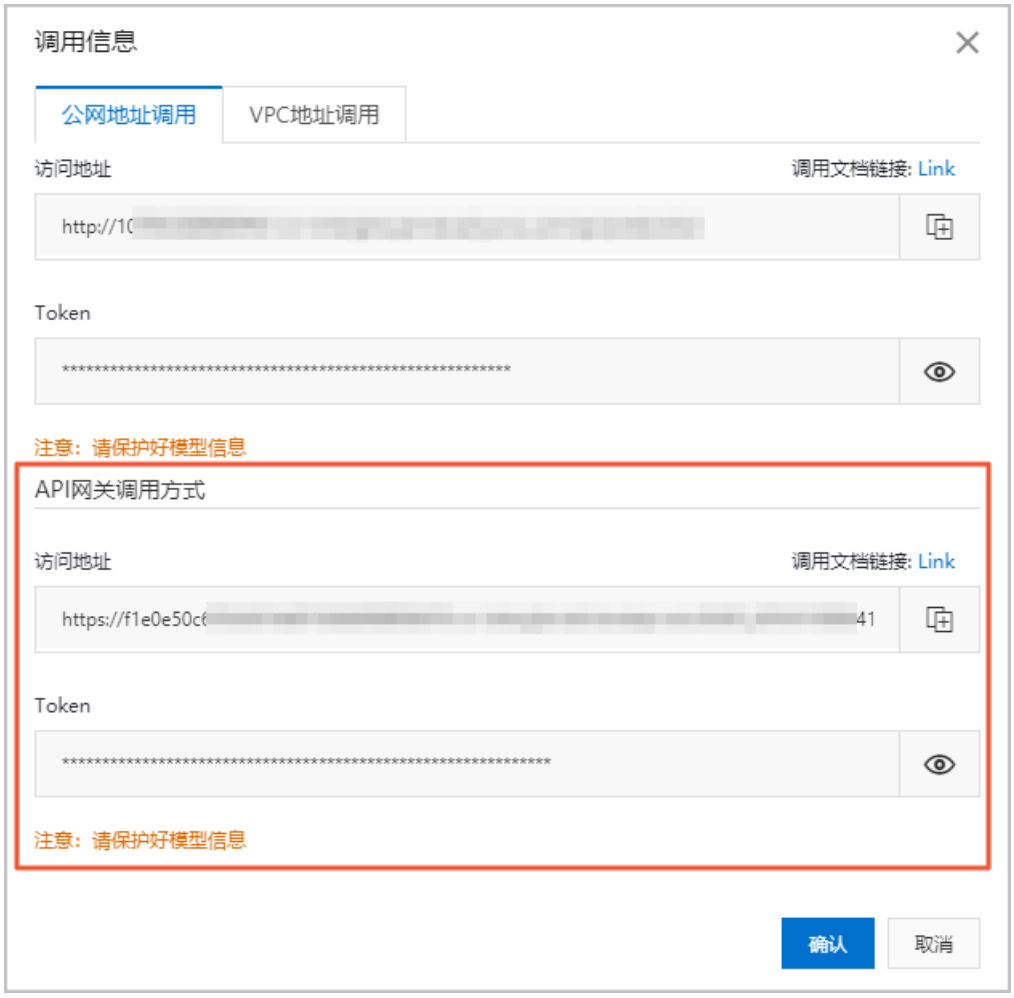
- 调用准备。

  通过API网关的公网调用需要先开通阿里云API网关，费用详情请参见API网关定价。

- 调用方式：
  i. 登录PAI控制台。
  ii. 在左侧导航栏，选择**模型部署 > 模型在线服务（EAS）**。
  iii. 在**PAI EAS模型在线服务**页面，单击待调用服务**服务方式**列下的**调用信息**。
  iv. 在**调用信息**对话框的**公网地址调用**页签，单击**自动创建**。



PAI-EAS会自动为该服务创建API，并生成一个API网关层的公网地址，您可以直接使用该地址（如下图所示）调用服务。



至此，已经完成了托管在API网关的API创建工作，下面即可进行调用。您可以自行实现调用代码，或参考下述指引：

- 登录API网关控制台-API列表（需要切换至您服务所在的地域），查找对应的API，并查看其所在分组。
- 登录API网关-SDK/文档自动生成（需要切换至您服务所在的地域），查找对应的分组，并下载对应语言的SDK，从而帮助您进行对应语言调用程序的开发。

更多关于调用的问题，请参见客户端调用API示例。

- API的高级功能（基于API网关）：
  - 查看API信息。

    您可以按照如下步骤查看API相关信息：

    a. 登录API网关控制台。

    b. 在左侧导航栏，单击**开放API > API列表**。

    c. 在**API列表**页面，单击待查看API**操作**列下的**管理**，即可查看该API的详细信息。
  - API授权。

    只有授权成功的用户才可以访问您的API，这样的用户身份称为App。PAI-EAS在自动创建API时，已经自动为您创建了App身份并完成了授权，如果需要创建其他App，需要手动为其授权，详情请参见4 创建应用和API授权。

- API分组与域名绑定。

  创建的每个API都隶属于某个API分组，可以通过该分组对这组API进行统一管理。API分组默认分配的是二级域名，此二级域名仅供调试使用，如果直接访问该域名，则每天访问次数不能超过1000次。如果有更大的访问需求，可以对该API所在分组进行域名绑定，详情请参见1. 绑定域名步骤。

7.1.2. VPC地址调用

PAI-EAS支持通过Python官方SDK、Java官方SDK或自行实现调用逻辑的方式对服务进行VPC地址调用。本文详细介绍每种调用方式的实现方法。

优势

- 避免了公网调用中的网络性能开销，可以提升调用速度。
- 内网传输可以节省流量费用。

使用限制

- 调用服务的服务器必须与部署的模型服务位于同一地域。例如，如果服务部署在华东2（上海），则只能从华东2（上海）的ECS服务器发送调用请求。
- PAI-DSW关于MaxCompute的服务，无法使用VPC地址调用。

调用方式

在PAI-EAS部署模型服务后，系统会自动生成一个VPC调用的服务地址。您可以在**PAI EAS模型在线服务**页面，单击待调用服务**服务方式**列下的**调用信息**，查看VPC地址调用的**访问地址**和**Token**。通过该调用信息可以进行调用测试，示例如下。

```
$ curl http://http://166408185518****.vpc.cn-hangzhou.pai-eas.aliyuncs.com/api/predict/heart_predict_online -H 'Authorization: ZGI1YWYxNmQwYjMzMDM1YTNlMmFmNmEzYjIzZTVlZGQ0NDJhYTRm****' -d '[{"sex":0,"cp":0,"fbs":0,"restecg":0,"exang":0,"slop":0,"thal":0,"age":0,"trestbps":0,"chol":0,"thalach":0,"oldpeak":0,"ca":0}]'
```

调用测试成功后，即可调用服务。为方便用户调用服务，PAI-EAS提供了以下三种调用方式：

- 使用官方SDK调用（Python）

  PAI-EAS封装了调用逻辑并提供了Python官方SDK。

- 使用官方SDK调用（Java）

  PAI-EAS封装了调用逻辑并提供了Java官方SDK。

- 自行实现调用逻辑

  推荐使用以上两种官方SDK调用服务，从而有效减少编写调用逻辑的时间。如果您需要使用其他语言或自己编写实现调用逻辑，下文也提供了调用Demo供您参考。此外，自行实现调用逻辑需要根据不同框架构建服务请求，详情请参见构建通用Processor服务请求。

使用官方SDK调用（Python）

使用Python官方SDK调用服务的方法如下：

1. 安装。

   ```
   pip install -U eas-prediction --user
   ```

该Python SDK调用详情请参见GitHub官网。

2. 编写调用程序。

以字符串作为输入输出的程序为例，其他格式（TensorFlow或PyTorch等）的输入输出程序示例请参见GitHub官网。

```
#!/usr/bin/env python
from eas_prediction import PredictClient
from eas_prediction import StringRequest
if __name__ == '__main__':
  #下面的client = PredictClient()入参源于VPC地址调用的访问地址。
  client = PredictClient('http://166408185518****.vpc.cn-hangzhou.pai-eas.aliyuncs.com','heart_predict_online')
  #Token可以在VPC地址调用信息中获取，详情请参见调用方式部分。
  client.set_token('ZGI1YWYxNmQwYjMzMDM1YTNlMmFmNmEzYjIzZTVlZGQ0NDJhYTRm****')
  client.init()
  #输入请求需要根据模型进行构造，此处仅以字符串作为输入输出的程序为例。
  request = StringRequest('[{"sex":0,"cp":0,"fbs":0,"restecg":0,"exang":0,"slop":0,"thal":0,"age":0,"trestbps":0,"chol":0,"thalach":0,"oldpeak":0,"ca":0}]')
  for x in range(0, 1):
    resp = client.predict(request)
    print(resp)
```

其中 `client = PredictClient()` 的函数入参源于VPC调用的访问地址。例如VPC调用的访问地址为http://166408185518****.vpc.cn-hangzhou.pai-eas.aliyuncs.com/api/predict/heart_predict_online，则 `PredictClient()` 函数的调用格式为 `client = PredictClient('http://166408185518****.vpc.cn-hangzhou.pai-eas.aliyuncs.com','heart_predict_online')`。

3. 执行调用程序。

```
$ python heart_predict.py
```

其中 `heart_predict.py` 为Python程序的文件名，需要根据实际情况修改。获得的预测结果如下。

```
IT-C02YJ0V8JHD2:Desktop wowei$ python heart_predict.py
[{"p_0":0.9941226679708888,"p_1":0.005877332029111252}]
```

使用官方SDK调用（Java）

使用Java官方SDK调用服务的方法如下：

1. 添加依赖包。

因为编写Java客户端代码需要使用Maven管理项目，所以需要在pom.xml文件中添加客户端依赖包eas-sdk。最新Release版本为2.0.1，代码示例如下。

```
<dependency>
 <groupId>com.aliyun.openservices.eas</groupId>
 <artifactId>eas-sdk</artifactId>
 <version>2.0.1</version>
</dependency>
```

该Java SDK调用详情请参见GitHub官网。

2. 编写调用程序。

以字符串作为输入输出的程序为例，其他格式（TensorFlow或PyTorch等）的输入输出程序示例请参见GitHub官网。

```
import com.aliyun.openservices.eas.predict.http.PredictClient;
import com.aliyun.openservices.eas.predict.http.HttpConfig;
public class Test_String {
  public static void main(String[] args) throws Exception{
    //启动并初始化客户端。
    PredictClient client = new PredictClient(new HttpConfig());
    //Token可以在VPC地址调用信息中获取，详情请参见调用方式部分。
    client.setToken("ZGI1YWYxNmQwYjMzMDM1YTNlMmFmNmEzYjIzZTVlZGQ0NDJhYTRm****");
    //下面的Endpoint信息源于VPC调用的访问地址。
    client.setEndpoint("http://166408185518****.vpc.cn-hangzhou.pai-eas.aliyuncs.com");
    //配置服务名称。
    client.setModelName("heart_predict_online");
    //定义输入字符串。输入请求需要根据模型进行构造，此处仅以字符串作为输入输出的程序为例。
    String request = "[{\"sex\":0,\"cp\":0,\"fbs\":0,\"restecg\":0,\"exang\":0,\"slop\":0,\"thal\":0,\"age\":0
,\"trestbps\":0,\"chol\":0,\"thalach\":0,\"oldpeak\":0,\"ca\":0}]";
    System.out.println(request);
    //通过PAI-EAS返回字符串。
    try {
      String response = client.predict(request);
      System.out.println(response);
    } catch(Exception e) {
      e.printStackTrace();
    }
    //关闭客户端。
    client.shutdown();
    return;
  }
}
```

其中 client.setEndpoint() 和 client.setModelName() 的函数入参源于VPC调用的访问地址。例如VPC调用的访问地址为http://166408185518****.vpc.cn-hangzhou.pai-eas.aliyuncs.com/api/predict/heart_predict_online，则 client.setEndpoint() 函数的调用格式为 client.setEndpoint("http://166408185518****.vpc.cn-hangzhou.pai-eas.aliyuncs.com")，client.setModelName() 函数的调用格式为 client.setModelName("heart_predict_online")。

3. 执行调用程序。

   执行调用逻辑程序可以获得如下结果。

   ```
   [{"p_0":0.9941226679708888,"p_1":0.005877332029111252}]
   ```

自行实现调用逻辑

PAI-EAS支持使用Python、Java或其他语言自行实现调用逻辑，您需要根据不同框架构建服务请求，详情请参见构建通用Processor服务请求。自行实现调用逻辑的示例如下，采用HTTP接口调用。

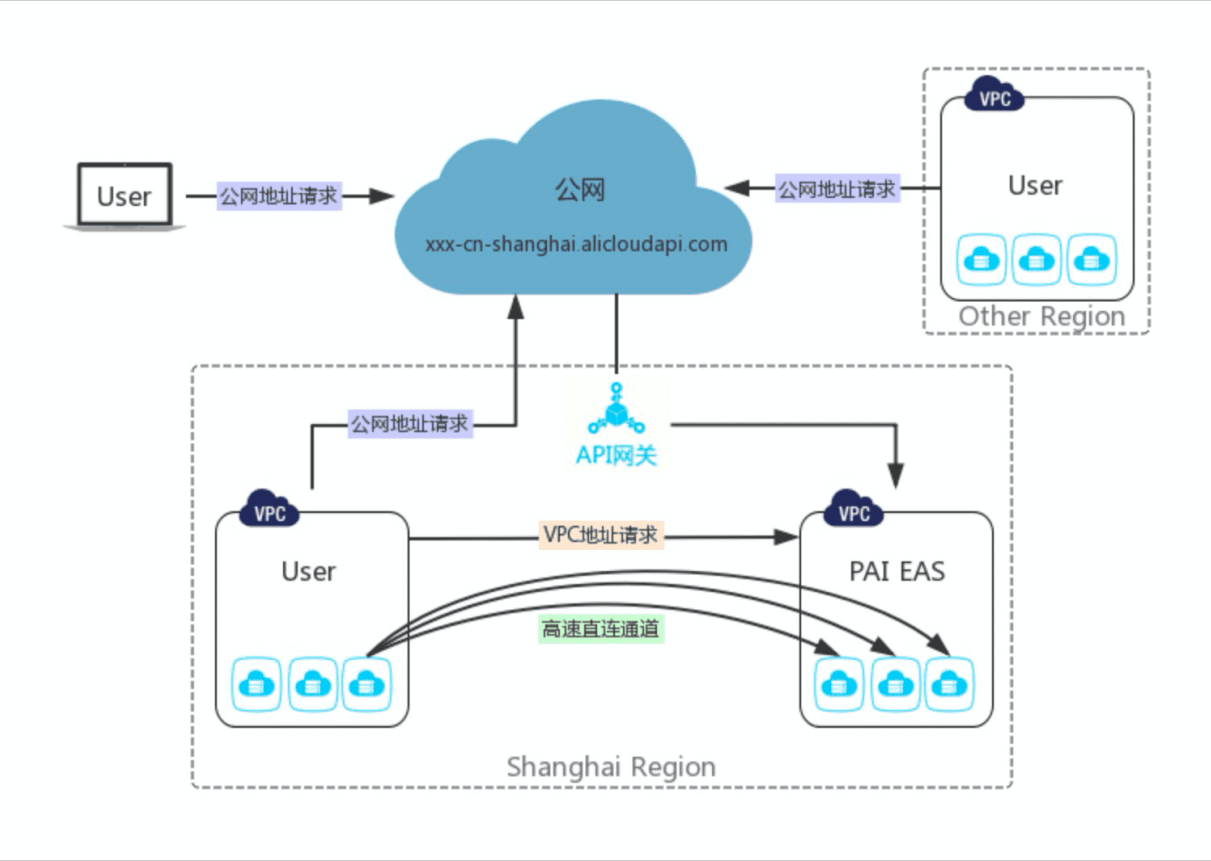
```
import requests
#URL信息可以在VPC地址调用信息中获取，详情请参见调用方式部分。
url = 'http://166408185518****.vpc.cn-hangzhou.pai-eas.aliyuncs.com/api/predict/heart_predict_online'
#header信息（Token）可以在VPC地址调用信息中获取，详情请参见调用方式部分。
headers = {"Authorization": 'ZGI1YWYxNmQwYjMzMDM1YTNlMmFmNmEzYjIzZTVlZGQ0NDJhYTRm****'}
#根据具体模型要求的数据格式构造服务请求，此处仅以字符串作为输入输出的程序为例。
data = '[{"sex":0,"cp":0,"fbs":0,"restecg":0,"exang":0,"slop":0,"thal":0,"age":0,"trestbps":0,"chol":0,"thalach":0,"oldpeak":0,"ca":0}]'
resp = requests.post(url, data=data, headers=headers)
print resp
print resp.content
```

7.1.3. VPC高速直连调用

PAI-EAS支持通过Python官方SDK或自行实现调用逻辑的方式对服务进行VPC高速直连调用。本文详细介绍这两种调用方式的实现方法。

调用原理

VPC高速直连调用、公网调用及VPC地址调用的链路如下图所示。



开启服务运行所在资源组的VPC高速直连功能后，无需通过网关访问服务，避免了四层SLB和七层网络转发，您可以在自己的VPC中直接访问PAI-EAS实例。同时，PAI-EAS预置的RPC实现了HTTP相关协议栈，对于高QPS（Queries Per Second）的大流量服务（例如图像服务），可以大幅度提高访问性能、降低访问延时。

前提条件

- 仅支持对部署在专属资源组中的服务进行VPC高速直连调用，因此需要购买（创建）专属资源组，详情请参见开通及购买。
- 部署服务前，需要为专属资源组开通VPC高速直连功能，详情请参见VPC高速直连。

调用方式

为方便用户调用服务，PAI-EAS提供了以下两种方式实现VPC高速直连调用：

- 使用官方SDK调用（Python）

  PAI-EAS封装了调用逻辑并提供了Python官方SDK，您可以直接使用该SDK实现VPC高速直连调用。

- 自行实现调用逻辑

  推荐使用官方SDK调用服务，从而有效减少编写调用逻辑的时间并提高服务调用稳定性。如果您需要使用其他语言或希望自己编写实现调用逻辑，下文也提供了方法指导供您参考。此外，自行实现调用逻辑需要根据不同框架构建服务请求，详情请参见构建通用Processor服务请求。

使用官方SDK调用（Python）

使用Python官方SDK调用服务的方法如下：

1. 安装。

   ```
   pip install -U eas-prediction --user
   ```

   该Python DSK的调用详情请参见GitHub官网。

2. 编写调用程序。

   以字符串作为输入输出的程序为例，其他格式（TensorFlow或PyTorch等）的输入输出程序示例请参见GitHub官网。

   ```
   #!/usr/bin/env python
   from eas_prediction import PredictClient
   from eas_prediction import StringRequest
   from eas_prediction import TFRequest
   from eas_prediction import ENDPOINT_TYPE_DIRECT
   if __name__ == '__main__':
     client = PredictClient('http://pai-eas-vpc.cn-shanghai.aliyuncs.com', 'mnist_saved_model_example')
     client.set_token('M2FhNjJlZDBmMzBmMzE4NjFiNzZhMmUxY2IxZjkyMDczNzAzYjFi****')
     client.set_endpoint_type(ENDPOINT_TYPE_DIRECT) # 表示通过直连通道访问服务。
     client.init()
     #request = StringRequest('[{}]')
     req = TFRequest('predict_images')
     req.add_feed('images', [1, 784], TFRequest.DT_FLOAT, [1] * 784)
     for x in range(0, 1000000):
       resp = client.predict(req)
       print(resp)
   ```

   其中 `client = PredictClient()` 的函数入参源于VPC调用的访问地址。例如VPC调用的访问地址为http://166408185518****.vpc.cn-hangzhou.pai-eas.aliyuncs.com/api/predict/heart_predict_online，则 `PredictClient()` 函数的调用格式为 `client = PredictClient('http://166408185518****.vpc.cn-hangzhou.pai-eas.aliyuncs.com','heart_predict_online')` 。

自行实现调用逻辑

如果您需要使用其他语言或希望自己编写调用逻辑，则可以参见如下方法，自行实现直连访问功能，采用HTTP接口调用。PAI-EAS提供服务发现机制，在VPC环境中，通过如下地址即可获得服务的后端地址列表。

地域	地址
华东2（上海）	http://pai-eas-vpc.cn-shanghai.aliyuncs.com/exported/apis/eas.alibaba-inc.k8s.io/v1/upstreams/
华北2（北京）	http://pai-eas-vpc.cn-beijing.aliyuncs.com/exported/apis/eas.alibaba-inc.k8s.io/v1/upstreams/
华东1（杭州）	http://pai-eas-vpc.cn-hangzhou.aliyuncs.com/exported/apis/eas.alibaba-inc.k8s.io/v1/upstreams/

例如访问华东1（杭州）的mnist_saved_model_example服务（该服务有两个Instance），示例如下。

```
$curl http://pai-eas-vpc.cn-shanghai.aliyuncs.com/exported/apis/eas.alibaba-inc.k8s.io/v1/upstreams/mnist_saved_model_example
```

获得的服务后端地址列表如下。

```
{
 "correlative": [
  "mnist_saved_model_example"
 ],
 "endpoints": {
  "items": [
   {
    "app": "mnist-saved-model-example",
    "ip": "172.16.XX.XX",
    "port": 50000,
    "weight": 100
   },
   {
    "app": "mnist-saved-model-example",
    "ip": "172.16.XX.XX",
    "port": 50000,
    "weight": 100
   }
  ]
 }
}
```

如上所示，客户端可以获得该服务后端对应的两个Instance的ip、port及weight信息。您可以实现一个weighted round robin算法，在每次调用服务前获取一个Instance，并对其进行VPC直连访问。

> **说明** 您需要从服务端定期同步Endpoint List至本地，每次请求前，从本地Cache中根据WRR（Weighted Round Robin）算法随机访问一个Instance。如果每次请求前从服务端获取Endpoint List，则会大幅度降低访问性能。

服务更新或节点异常发生Failover时，部分Instance可能不可用。因此检测到请求失败时，客户端需要进行自动重试，以避免健康检查失败的Instance在从Instance List中移除的延时过程中被访问，进而造成服务质量下降。

7.2. 服务调用SDK

7.2.1. Java SDK使用说明

推荐使用PAI-EAS提供的官方SDK进行服务调用，从而有效减少编写调用逻辑的时间并提高调用稳定性。本文介绍官方Java SDK接口详情。同时，以字符串输入输出和TensorFlow输入输出为例，提供了使用Java SDK进行服务调用的完整程序示例。

添加依赖包

使用Java编写客户端代码时，需要使用Maven管理项目。因此，您必须在pom.xml文件中添加客户端所需的依赖包eas-sdk。依赖包eas-sdk的最新版本为2.0.3，*pom.xml*文件中的具体代码如下。

```
<dependency>
 <groupId>com.aliyun.openservices.eas</groupId>
 <artifactId>eas-sdk</artifactId>
 <version>2.0.3</version>
</dependency>
```

接口列表

类	接口	描述
PredictClient	PredictClient(HttpConfig httpConfig)	• 功能：PredictClient类构造器。 • 参数：httpConfig表示HttpConfig类的实例对象。
	void setToken(String token)	• 功能：设置HTTP请求的Token参数。 • 参数：token表示访问服务时需要使用的鉴权Token。
	void setModelName(String modelName)	• 功能：设置请求的在线预测服务的模型名称。 • 参数：modelName表示所设置的模型名称。
	void setEndpoint(String endpoint)	• 功能：设置请求服务的Host和Port，格式为"host:port"。 • 参数：endpoint表示接收消息的终端地址。
	void setDirectEndpoint(String endpoint)	• 功能：设置VPC高速直连通道访问服务的Endpoint，例如，pai-eas-vpc.cn-shanghai.aliyuncs.com。 • 参数：endpont表示设置的访问服务地址。
	void setRetryCount(boolean int retryCount)	• 功能：设置失败重试次数。 • 参数：retryCount表示失败的重试次数。
	void setContentType(String contentType)	• 功能：设置HTTP Client的Content类型，默认为"application/octet-stream"。 • 参数：contentType表示发送数据流的类型。

类	接口	描述

	void createChildClient(String token, String endpoint, String modelName)	• 功能：创建子Client对象，共用父Client对象的线程池。该接口用于多线程预测。
		• 参数：
		  ◦ token：服务的鉴权Token。
		  ◦ endpoint：服务的Endpoint。
		  ◦ modelName：服务的名称。

	void predict(TFRequest runRequest)	• 功能：向在线预测服务提交一个TensorFlow请求。 • 参数：runRequest表示TensorFlow请求的实例对象。
	void predict(String requestContent)	• 功能：向在线预测服务提交一个字符串请求。 • 参数：requestContent表示字符串格式的请求内容。
	void predict(byte[] requestContent)	• 功能：向在线预测服务提交一个Byte数组请求。 • 参数：requestContent表示Byte类型的请求内容。
HttpConfig	void setIoThreadNum(int ioThreadNum)	• 功能：设置HTTP请求的IO线程数，默认值为2。 • 参数：ioThreadNum表示发送HTTP请求的IO线程数。
	void setReadTimeout(int readTimeout)	• 功能：设置Socket的读取超时时间，默认值为5000，表示5s。 • 参数：readTimeout表示请求的读取超时时间。
	void setConnectTimeout(int connectTimeout)	• 功能：设置连接超时时间，默认值为5000，表示5s。 • 参数：connectTimeout表示请求的连接超时时间。
	void setMaxConnectionCount(int maxConnectionCount)	• 功能：设置最大连接数，默认值为1000。 • 参数：maxConnectionCount表示客户端连接池的最大连接数。
	void setMaxConnectionPerRoute(int maxConnectionPerRoute)	• 功能：设置每个路由的最大默认连接数，默认值为1000。 • 参数：maxConnectionPerRoute表示每个路由上的默认最大连接数。
	void setKeepAlive(boolean keepAlive)	• 功能：设置HTTP服务的keep-alive。 • 参数：keepAlive表示是否开启连接的keep-alive机制，默认为true。
	int getErrorCode()	返回最近一次调用的状态码。

类	接口	描述
	string getErrorMessage()	返回最近一次调用的状态信息。
TFRequest	void setSignatureName(String value)	• 功能：请求的在线服务的模型为TensorFlow的SavedModel格式时，设置请求模型的signatureDef的名称。 • 参数：请求模型的signatureDef的名称。
	void addFetch(String value)	• 功能：请求TensorFlow的在线服务模型时，设置模型输出Tensor的别名。 • 参数：value表示TensorFlow服务输出Tensor的别名。

类	接口	描述

	void addFeed(String inputName, TFDataType dataType, long[]shape, ?[]content)	• 功能：请求TensorFlow的在线预测服务模型时，设置需要输入的Tensor。
		• 参数：
		  ○ inputName：表示输入Tensor的别名。
		  ○ dataType：表示输入Tensor的DataType。
		  ○ shape：表示输入Tensor的TensorShape。
		  ○ content：表示输入Tensor的内容，采用一维数组表示。
		如果输入Tensor的DataType为DT_FLOAT、DT_COMPLEX64、DT_BFLOAT16或DT_HALF，则content中的元素类型为FLOAT。其中DataType为DT_COMPLEX64时，content中相邻两个FLOAT元素依次表示复数的实部和虚部。
		如果输入Tensor的DataType为DT_DOUBLE或DT_COMPLEX128，则content中的元素类型为DOUBLE。其中DataType为DT_COMPLEX128时，content中相邻两个DOUBLE元素依次表示复数的实部和虚部。
		如果输入Tensor的DataType为DT_INT32、DT_UINT8、DT_INT16、DT_INT8、DT_QINT8、DT_QUINT8、DT_QINT32、DT_QINT16、DT_QUINT16或DT_UINT16，content中的元素类型为INT。
		如果输入Tensor的DataType为DT_INT64，则content中的元素类型为LONG。
		如果输入Tensor的DataType为DT_STRING，则content中的元素类型为STRING。
		如果输入Tensor的DataType为DT_BOOL，则content中的元素类型为BOOLEAN。

	List<Long> getTensorShape(String outputName)	• 功能：获得指定别名的输出Tensor的TensorShape。 • 参数：outputName表示待获取TensorShape的模型输出的名称。 • 返回值：表示TensorShape的一维数组。

类	接口	描述
TFResponse	List<Float> getFloatVals(String outputName)	• 功能：如果输出Tensor的DataType为DT_FLOAT、DT_COMPLEX64、DT_BFLOAT16或DT_HALF，则可以调用该接口获取指定输出Tensor的data。 • 参数：outputName表示待获取FLOAT类型返回数据的模型输出的名称。 • 返回值：模型输出的TensorData展开成的一维数组。
	List<Double> getDoubleVals(String outputName)	• 功能：如果输出Tensor的DataType为DT_DOUBLE或DT_COMPLEX128，则调用该接口获取指定输出Tensor的data。 • 参数：outputname表示待获取DOUBLE类型返回数据的模型输出的名称。 • 返回值：模型输出的TensorData展开成的一维数组。
	List<Integer> getIntVals(String outputName)	• 功能：如果输出Tensor的DataType为DT_INT32、DT_UINT8、DT_INT16、DT_INT8、DT_QINT8、DT_QUINT8、DT_QINT32、DT_QINT16、DT_QUINT16或DT_UINT16，则调用该接口获取指定输出Tensor的data。 • 参数：outputname表示待获取INT类型返回数据的模型输出的名称。 • 返回值：模型输出的TensorData展开成的一维数组。
	List<String> getStringVals(String outputName)	• 功能：如果输出Tensor的DataType为DT_STRING，则调用该接口获取指定输出Tensor的data。 • 参数：outputName表示待获取STRING类型返回数据的模型输出的名称。 • 返回值：模型输出的TensorData展开成的一维数组。
	List<Long> getInt64Vals(String outputName)	• 功能：如果输出Tensor的DataType为DT_INT64，则调用该接口获取指定输出Tensor的data。 • 参数：outputName表示待获取的INT64类型返回数据的模型输出的名称。 • 返回值：模型输出的TensorData展开成的一维数组。
	List<Boolean> getBoolVals(String outputName)	• 功能：如果输出Tensor的DataType为DT_BOOL，则调用该接口获取指定输出Tensor的data。 • 参数：outputName表示待获取BOOL类型的返回数据的模型输出的名称。 • 返回值：模型输出的TensorData展开成的一维数组。

程序示例

字符串输入输出示例

对于使用自定义Processor部署服务的用户而言，通常采用字符串进行服务调用（例如，PMML模型服务的调用），具体的Demo程序如下。

```
import com.aliyun.openservices.eas.predict.http.PredictClient;
import com.aliyun.openservices.eas.predict.http.HttpConfig;
public class Test_String {
  public static void main(String[] args) throws Exception{
  // 启动并初始化客户端。client对象需要共享，不能每个请求都创建一个client对象。
    PredictClient client = new PredictClient(new HttpConfig());
    client.setToken("YWFlMDYyZDNmNTc3M2I3MzMwYmY0MmYwM2Y2MTYxMTY4NzBkNzdj****");

    // 如果需要使用网络直连功能，则使用setDirectEndpoint方法。
    // 例如，client.setDirectEndpoint("pai-eas-vpc.cn-shanghai.aliyuncs.com");
    // 网络直连功能需要在PAI-EAS控制台开通，提供用于访问PAI-EAS服务的源vswitch。开通后可以绕过网关以软负载的方式直接访问服务的实例，以实现更好的稳定性和性能。
    // 注意：通过普通网关访问时，需要使用以用户uid开头的Endpoint，在PAI-EAS控制台服务的调用信息中可以查到该信息。通过直连访问时，需要使用如上的pai-eas-vpc.{region_id}.aliyuncs.com的域名。
    client.setEndpoint("182848887922****.vpc.cn-shanghai.pai-eas.aliyuncs.com");
    client.setModelName("scorecard_pmml_example");
    //定义输入字符串。
    String request = "[{\"money_credit\": 3000000}, {\"money_credit\": 10000}]";
    System.out.println(request);
    //通过PAI-EAS返回字符串。
    try {
      String response = client.predict(request);
      System.out.println(response);
    } catch(Exception e) {
      e.printStackTrace();
    }
    return;
  }
}
```

如上述程序所示，使用Java SDK调用服务的流程如下：

1. 通过 `PredictClient` 接口创建客户端服务对象。如果在程序中需要使用多个服务，则创建多个Client对象。
2. 为PredictClient对象配置Token、Endpoint及ModelName。
3. 构造STRING类型的request作为输入，通过 `client.predict` 发送HTTP请求，系统返回response。

TensorFlow输入输出示例

使用TensorFlow的用户，需要将TFRequest和TFResponse分别作为输入和输出数据格式，具体Demo示例如下。

```
import java.util.List;
import com.aliyun.openservices.eas.predict.http.PredictClient;
import com.aliyun.openservices.eas.predict.http.HttpConfig;
import com.aliyun.openservices.eas.predict.request.TFDataType;
import com.aliyun.openservices.eas.predict.request.TFRequest;
import com.aliyun.openservices.eas.predict.response.TFResponse;
public class Test_TF {
  public static TFRequest buildPredictRequest() {
    TFRequest request = new TFRequest();
    request.setSignatureName("predict_images");
    float[] content = new float[784];
    for (int i = 0; i < content.length; i++)
      content[i] = (float)0.0;
    request.addFeed("images", TFDataType.DT_FLOAT, new long[]{1, 784}, content);
    request.addFetch("scores");
    return request;
  }
  public static void main(String[] args) throws Exception{
    PredictClient client = new PredictClient(new HttpConfig());
    // 如果使用网络直连功能，则调用setDirectEndpoint方法。
    // 例如，client.setDirectEndpoint("pai-eas-vpc.cn-shanghai.aliyuncs.com");
    // 网络直连功能需要在PAI-EAS控制台开通，提供用于访问PAI-EAS服务的源vswitch。开通后可以绕过网关以软负载的方式直接访问服务的实例，以实现更好的稳定性和性能。
    // 注意：通过普通网关访问时，需要使用以用户uid开头的Endpoint，在PAI-EAS控制台服务的调用信息中可以查到该信息。通过直连访问时，需要使用如上的pai-eas-vpc.{region_id}.aliyuncs.com的域名。
    client.setEndpoint("1828488879222746.vpc.cn-shanghai.pai-eas.aliyuncs.com");
    client.setModelName("mnist_saved_model_example");
    client.setToken("YTg2ZjE0ZjM4ZmE3OTc0NzYxZDMyNmYzMTJjZTQ1YmU0N2FjMTAy****");
    long startTime = System.currentTimeMillis();
    for (int i = 0; i < 1000; i++) {
      try {
        TFResponse response = client.predict(buildPredictRequest());
        List<Float> result = response.getFloatVals("scores");
        System.out.print("Predict Result: [");
        for (int j = 0; j < result.size(); j++) {
          System.out.print(result.get(j).floatValue());
          if (j != result.size() -1)
            System.out.print(", ");
        }
        System.out.print("]\n");
      } catch(Exception e) {
        e.printStackTrace();
      }
    }
    long endTime = System.currentTimeMillis();
    System.out.println("Spend Time: " + (endTime - startTime) + "ms");
  }
}
```

如上述程序所示，使用Java SDK调用TensorFlow服务的流程如下：

1. 通过 `PredictClient` 接口创建客户端服务对象。如果在程序中需要使用多个服务，则创建多个Client对象。

2. 为PredictClient对象配置Token、Endpoint及ModelName。
3. 使用TFRequest类封装输入数据，使用TFResponse类封装输出数据。

7.2.2. Python SDK使用说明

推荐使用PAI-EAS提供的官方SDK进行服务调用，从而有效减少编写调用逻辑的时间并提高调用稳定性。本文介绍官方Python SDK接口详情，并以常见类型的输入输出为例，提供了使用Python SDK进行服务调用的完整程序示例。

安装方法

```
pip install -U eas-prediction --user
```

接口列表

类	接口	描述

	PredictClient(endpoint, service_name)	• 功能：PredictClient类的构造方法。
		• 参数：
		  ◦ endpoint：服务端的Endpoint地址。
		  如果是普通服务，则设置为默认网关Endpoint。例如 182848887922***.cn-shanghai.pai-eas.aliyuncs.com 。
		  如果是VPC直连请求，则设置为当前地域的通用Endpoint。例如，华东2（上海）设置为 pai-eas-vpc.cn-shanghai.aliyuncs.com 。
		  ◦ service_name：服务名字。

	set_endpoint(endpoint)	• 功能：设置服务的Endpoint地址。 • 参数：endpoint表示服务端的Endpoint地址。 如果是普通服务，则设置为默认网关Endpoint。例如 182848887922***.cn-shanghai.pai-eas.aliyuncs.com 。 如果是VPC直连请求，则设置为当前地域的通用Endpoint。例如，华东2（上海）设置为 pai-eas-vpc.cn-shanghai.aliyuncs.com 。
	set_service_name(service_name)	• 功能：设置请求的服务名字。 • 参数：service_name请求的服务名字。

类	接口	描述

PredictClient	set_endpoint_type(endpoint_type)	• 功能：设置服务端的网关类型。
		• 参数：endpoint_type待设置的网关类型，支持以下网关类型：
		  ○ ENDPOINT_TYPE_GATEWAY：默认网关。
		  ○ ENDPOINT_TYPE_DIRECT：表示直连请求。如果没有手动设置该参数，则默认通过网关访问服务。

	set_token(token)	• 功能：设置服务访问的Token。 • 参数：token表示服务访问的Token。
	set_retry_count(max_retry_count)	• 功能：设置请求失败的重试次数。 • 参数：max_retry_count表示请求失败的重试次数，默认为5。 注意 对于服务端进程异常、服务器异常或网关长连接断开等情况导致的个别请求失败，均需要客户端重新发送请求。因此，请勿将该参数设置为0。
	set_max_connection_count(max_connection_count)	• 功能：设置客户端连接池中长链接数量的最大值。出于性能考虑，客户端会与服务端建立长连接，并将连接放入连接池中。每次请求时，从连接池中获取一个空闲连接访问服务。 • 参数：max_connection_count表示连接池中最大的长连接数量，默认值为100。
	set_timeout(timeout)	• 功能：设置请求的超时时间。 • 参数：timeout表示请求的超时时间。单位为ms，默认值为5000。
	init()	对PredictClient对象进行初始化。在上述设置参数的接口执行完成后，需要调用 init() 接口才能生效。
	predict(request)	• 功能：向在线预测服务提交一个预测请求。 • 参数：request是一个抽象类，可以输入不同类型的request，例如StringRequest或TFRequest。 • 返回值：返回请求对应的Response。
StringRequest	StringRequest(request_data)	• 功能：StringRequest类的构造方法。 • 参数：request_data表示待发送的请求字符串。
StringResponse	to_string()	• 功能：将StringResponse类转换为字符串。 • 返回值：返回请求的Response Body。

类	接口	描述
TFRequest	TFRequest(signature_name)	• 功能：TFRequest类构造方法。 • 参数：signature_name表示待请求模型中的Signature Name。

	add_feed(self, input_name, shape, data_type, content)	• 功能：请求TensorFlow在线预测服务模型时，设置需要输入的input数据。
		• 参数：
		  ○ input_name：输入Tensor的别名。
		  ○ shape：输入Tensor的TensorShape。
		  ○ data_type：输入Tensor的DataType，支持以下类型：
		  ■ TFRequest.DT_FLOAT
		  ■ TFRequest.DT_DOUBLE
		  ■ TFRequest.DT_INT8
		  ■ TFRequest.DT_INT16
		  ■ TFRequest.DT_INT32
		  ■ TFRequest.DT_INT64
		  ■ TFRequest.DT_STRING
		  ■ TFRequest.TF_BOOL
		  ○ content：输入Tensor的内容，通过一维数组展开表示。

	add_fetch(self, output_name)	• 功能： 请求TensorFlow在线预测服务模型时，设置需要输出的Tensor别名。 • 参数：output_name表示待输出Tensor的别名。 对于SavedModel模型，该参数可选。如果没有设置该参数，则输出所有的outputs。 对于Frozen Model，该参数必选。
	to_string()	• 功能：将TFRequest构建的用于请求传输的ProtoBuf对象序列化成字符串。 • 返回值：TFRequest序列化后的字符串。
TFResponse	get_tensor_shape(output_name)	• 功能：获得指定别名的输出Tensor的TensorShape。 • 参数：output_name表示待获取Shape的Tensor别名。 • 返回值：输出的TensorShape。

类	接口	描述
	get_values(output_name)	• 功能：获取输出Tensor的数据向量。 • 参数：output_name表示待获取结果数据的Tensor别名。 • 返回值：输出结果以一维数组的形式保存。您可以搭配 get_tensor_shape() 接口，获取对应Tensor的Shape，将其还原成所需的多维Tensor。接口会根据output的类型，返回不同类型的结果数组。
TorchRequest	TorchRequest()	TorchRequest类的构造方法。

	add_feed(self, index, shape, data_type, content)	• 功能：请求PyTorch在线预测服务模型时，设置需要输入的Tensor。
		• 参数：
		  ◦ index：待输入Tensor的下标。
		  ◦ shape：输入Tensor的TensorShape。
		  ◦ data_type表示输入Tensor的DataType，支持以下类型：
		  ▪ TFRequest.DT_FLOAT
		  ▪ TFRequest.DT_DOUBLE
		  ▪ TFRequest.DT_INT8
		  ▪ TFRequest.DT_INT16
		  ▪ TFRequest.DT_INT32
		  ▪ TFRequest.DT_INT64
		  ▪ TFRequest.DT_STRING
		  ▪ TFRequest.TF_BOOL
		  ◦ content：输入Tensor的内容，通过一维数组展开表示。

	add_fetch(self, output_index)	• 功能：请求PyTorch在线预测服务模型时，设置需要输出Tensor的Index。该接口为可选，如果您没有调用该接口设置输出Tensor的Index，则输出所有的outputs。 • 参数：output_index表示输出Tensor的Index。
	to_string()	• 功能：将TorchRequest构建的用于请求传输的ProtoBuf对象序列化成字符串。 • 返回值：TorchRequest序列化后的字符串。
	get_tensor_shape(output_index)	• 功能：获得指定下标的输出Tensor的TensorShape。 • 参数：待获取Shape的输出Tensor的Index。 • 返回值：下标Index对应的输出Tensor的Shape。

类	接口	描述
TorchResponse	get_values(output_index)	• 功能：获取输出Tensor的数据向量，输出结果以一维数组的形式保存。您可以搭配使用 get_tensor_shape() 接口，获取对应Tensor的Shape，将其还原成所需的多维Tensor。接口会根据output的类型，返回不同类型的结果数组。 • 参数：output_index表示待获取的输出 Tensor对应的下标。 • 返回值：返回的结果Tensor的数据数组。

程序示例

- **字符串输入输出示例**

  对于使用自定义Processor部署服务的用户而言，通常采用字符串进行服务调用（例如，PMML模型服务的调用），具体的Demo程序如下。

```
#!/usr/bin/env python
from eas_prediction import PredictClient
from eas_prediction import StringRequest
if __name__ == '__main__':
  client = PredictClient('http://182848887922****.cn-shanghai.pai-eas.aliyuncs.com', 'scorecard_pmml_example')
  client.set_token('YWFlMDYyZDNmNTc3M2I3MzMwYmY0MmYwM2Y2MTYxMTY4NzBkNzdj****')
  client.init()
  request = StringRequest('[{"fea1": 1, "fea2": 2}]')
  for x in range(0, 1000000):
    resp = client.predict(request)
    print(resp)
```

- **TensorFlow输入输出示例**

  使用TensorFlow的用户，需要将TFRequest和TFResponse分别作为输入和输出数据格式，具体Demo示例如下。

```
#!/usr/bin/env python
from eas_prediction import PredictClient
from eas_prediction import StringRequest
from eas_prediction import TFRequest
if __name__ == '__main__':
  client = PredictClient('http://182848887922****.cn-shanghai.pai-eas.aliyuncs.com', 'mnist_saved_model_example')
  client.set_token('YTg2ZjE0ZjM4ZmE3OTc0NzYxZDMyNmYzMTJjZTQ1YmU0N2FjMTAy****')
  client.init()
  #request = StringRequest('[{}]')
  req = TFRequest('predict_images')
  req.add_feed('images', [1, 784], TFRequest.DT_FLOAT, [1] * 784)
  for x in range(0, 1000000):
    resp = client.predict(req)
    print(resp)
```

- **通过VPC网络直连方式调用服务的示例**

通过网络直连方式，您只能访问部署在PAI-EAS专属资源组的服务，且需要为该资源组与用户指定的vSwitch连通网络后才能使用。关于如何购买PAI-EAS专属资源组和连通网络，请参见专属资源组和VPC高速直连。该调用方式与普通调用方式相比，仅需增加一行代码 `client.set_endpoint_type(ENDPOINT_TYPE_DIRECT)` 即可，特别适合大流量高并发的服务，具体示例如下。

```
#!/usr/bin/env python
from eas_prediction import PredictClient
from eas_prediction import StringRequest
from eas_prediction import TFRequest
from eas_prediction import ENDPOINT_TYPE_DIRECT
if __name__ == '__main__':
  client = PredictClient('http://pai-eas-vpc.cn-hangzhou.aliyuncs.com', 'mnist_saved_model_example')
  client.set_token('M2FhNjJlZDBmMzBmMzE4NjFiNzZhMmUxY2IxZjkyMDczNzAzYjFi****')
  client.set_endpoint_type(ENDPOINT_TYPE_DIRECT)
  client.init()
  request = TFRequest('predict_images')
  request.add_feed('images', [1, 784], TFRequest.DT_FLOAT, [1] * 784)
  for x in range(0, 1000000):
    resp = client.predict(request)
    print(resp)
```

- **PyTorch输入输出示例**

  使用PyTorch的用户，需要将TorchRequest和TorchResponse分别作为输入和输出数据格式，具体Demo示例如下。

```
#!/usr/bin/env python
from eas_prediction import PredictClient
from eas_prediction import TorchRequest
if __name__ == '__main__':
  client = PredictClient('http://182848887922****.cn-shanghai.pai-eas.aliyuncs.com', 'pytorch_gpu_wl')
  client.init()
  req = TorchRequest()
  req.add_feed(0, [1, 3, 224, 224], TorchRequest.DT_FLOAT, [1] * 150528)
  # req.add_fetch(0)
  import time
  st = time.time()
  timer = 0
  for x in range(0, 10):
    resp = client.predict(req)
    timer += (time.time() - st)
    st = time.time()
    print(resp.get_tensor_shape(0))
    # print(resp)
  print("average response time: %s s" % (timer / 10) )
```

- **BladeProcessor输入输出示例**

  使用BladeProcessor的用户，需要将BladeRequest和BladeResponse分别作为输入和输出数据格式，具体Demo示例如下。

```
#!/usr/bin/env python
from eas_prediction import PredictClient
from eas_prediction import BladeRequest
if __name__ == '__main__':
  client = PredictClient('http://182848887922****.cn-shanghai.pai-eas.aliyuncs.com', 'nlp_model_example
')
  client.init()
  req = BladeRequest()
  req.add_feed('input_data', 1, [1, 360, 128], BladeRequest.DT_FLOAT, [0.8] * 85680)
  req.add_feed('input_length', 1, [1], BladeRequest.DT_INT32, [187])
  req.add_feed('start_token', 1, [1], BladeRequest.DT_INT32, [104])
  req.add_fetch('output', BladeRequest.DT_FLOAT)
  import time
  st = time.time()
  timer = 0
  for x in range(0, 10):
    resp = client.predict(req)
    timer += (time.time() - st)
    st = time.time()
    # print(resp)
    # print(resp.get_values('output'))
    print(resp.get_tensor_shape('output'))
  print("average response time: %s s" % (timer / 10) )
```

- **兼容PAI-EAS默认TensorFlow接口的BladeProcessor输入输出示例**

  BladeProcessor用户可以使用兼容PAI-EAS默认TensorFlow接口的TFRequest与TFResponse作为数据的输入输出格式，具体Demo示例如下。

```
#!/usr/bin/env python
from eas_prediction import PredictClient
from eas_prediction.blade_tf_request import TFRequest # Need Importing blade TFRequest
if __name__ == '__main__':
  client = PredictClient('http://182848887922****.cn-shanghai.pai-eas.aliyuncs.com', 'nlp_model_example
')
  client.init()
  req = TFRequest(signature_name='predict_words')
  req.add_feed('input_data', [1, 360, 128], TFRequest.DT_FLOAT, [0.8] * 85680)
  req.add_feed('input_length', [1], TFRequest.DT_INT32, [187])
  req.add_feed('start_token', [1], TFRequest.DT_INT32, [104])
  req.add_fetch('output')
  import time
  st = time.time()
  timer = 0
  for x in range(0, 10):
    resp = client.predict(req)
    timer += (time.time() - st)
    st = time.time()
    # print(resp)
    # print(resp.get_values('output'))
    print(resp.get_tensor_shape('output'))
  print("average response time: %s s" % (timer / 10) )
```

7.3. 通用Processor服务请求数据构造

7.3.1. TensorFlow服务请求构造

本文为您介绍如何为基于通用Processor的TensorFlow服务构造请求数据。

输入数据

PAI-EAS预置了TensorFlow Processor，为保证性能，其输入输出为ProtoBuf格式。

调用案例

PAI-EAS在华东2（上海）的VPC环境中部署了一个Public的测试案例，其服务名称为mnist_saved_model_example，访问Token为空。您可以通过URLhttp://pai-eas-vpc.cn-shanghai.aliyuncs.com/api/predict/mnist_saved_model_example访问该服务。具体方式如下：

1. 获取模型信息。

   通过GET请求可以获取模型的相关信息，包括signature_name、name、type及shape，示例如下。

```
$curl http://pai-eas-vpc.cn-shanghai.aliyuncs.com/api/predict/mnist_saved_model_example | python -mjson.tool
{
  "inputs": [
    {
      "name": "images",
      "shape": [
        -1,
        784
      ],
      "type": "DT_FLOAT"
    }
  ],
  "outputs": [
    {
      "name": "scores",
      "shape": [
        -1,
        10
      ],
      "type": "DT_FLOAT"
    }
  ],
  "signature_name": "predict_images"
}
```

   该模型是一个MNIST数据集（下载MNIST数据集）分类模型。输入数据为DT_FLOAT类型，例如shape为[-1,784]，其中第一维表示batch_size（如果单个请求只包含一张图片，则batch_size为1），第二维表示784维的向量。因为训练该测试模型时，将其输入展开成了一维，所以单张图片输入也需要变换为28*28=784的一维向量。构建输入时，无论shape取值如何，都必须将输入展开成一维向量。该示例中，如果输入单张图片，则输入为1*784的一维向量。如果训练模型时输入的shape为[-1, 28, 28]，则构建输入时就需要将输入构建为1*28*28的一维向量。如果服务请求中指定的shape与模型的shape不一致，则预测请求报错。

2. 安装ProtoBuf并调用服务（以Python为例，介绍如何对TensorFlow服务进行调用）。

   PAI-EAS为Python预先生成了ProtoBuf包，您可以使用如下命令直接安装。

```
$ pip install http://eas-data.oss-cn-shanghai.aliyuncs.com/sdk/pai_tf_predict_proto-1.0-py2.py3-none-any.whl
```

   调用服务进行预测的示例代码如下。

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
import json
from urlparse import urlparse
from com.aliyun.api.gateway.sdk import client
from com.aliyun.api.gateway.sdk.http import request
from com.aliyun.api.gateway.sdk.common import constant
from pai_tf_predict_proto import tf_predict_pb2
import cv2
import numpy as np
with open('2.jpg', 'rb') as infile:
  buf = infile.read()
  # 使用numpy将字节流转换成array。
  x = np.fromstring(buf, dtype='uint8')
  # 将读取到的array进行图片解码获得28 × 28的矩阵。
  img = cv2.imdecode(x, cv2.IMREAD_UNCHANGED)
  # 因为预测服务API需要长度为784的一维向量，所以将矩阵reshape成784。
  img = np.reshape(img, 784)
def predict(url, app_key, app_secret, request_data):
  cli = client.DefaultClient(app_key=app_key, app_secret=app_secret)
  body = request_data
  url_ele = urlparse(url)
  host = 'http://' + url_ele.hostname
  path = url_ele.path
  req_post = request.Request(host=host, protocol=constant.HTTP, url=path, method="POST", time_out=6000)
  req_post.set_body(body)
  req_post.set_content_type(constant.CONTENT_TYPE_STREAM)
  stat,header, content = cli.execute(req_post)
  return stat, dict(header) if header is not None else {}, content
def demo():
  # 输入模型信息，单击模型名称即可获取。
  app_key = 'YOUR_APP_KEY'
  app_secret = 'YOUR_APP_SECRET'
  url = 'YOUR_APP_URL'
  # 构造服务。
  request = tf_predict_pb2.PredictRequest()
  request.signature_name = 'predict_images'
  request.inputs['images'].dtype = tf_predict_pb2.DT_FLOAT  # images参数类型。
  request.inputs['images'].array_shape.dim.extend([1, 784])  # images参数的形状。
  request.inputs['images'].float_val.extend(img)  # 数据。
  request.inputs['keep_prob'].dtype = tf_predict_pb2.DT_FLOAT  # keep_prob参数的类型。
  request.inputs['keep_prob'].float_val.extend([0.75])  # 默认填写一个。
  # 将ProtoBuf序列化成string进行传输。
  request_data = request.SerializeToString()
  stat, header, content = predict(url, app_key, app_secret, request_data)
```

```
    if stat != 200:
        print 'Http status code: ', stat
        print 'Error msg in header: ', header['x-ca-error-message'] if 'x-ca-error-message' in header else ''
        print 'Error msg in body: ', content
    else:
        response = tf_predict_pb2.PredictResponse()
        response.ParseFromString(content)
        print(response)
if __name__ == '__main__':
    demo()
```

该示例的输出如下。

```
outputs {
  key: "scores"
  value {
    dtype: DT_FLOAT
    array_shape {
      dim: 1
      dim: 10
    }
    float_val: 0.0
    float_val: 0.0
    float_val: 1.0
    float_val: 0.0
    float_val: 0.0
    float_val: 0.0
    float_val: 0.0
    float_val: 0.0
    float_val: 0.0
    float_val: 0.0
    float_val: 0.0
  }
}
```

其中 `outputs` 为10个类别对应的得分，即输入图片为2.jpg时，除value[2]外，其他均为0。因此最终预测结果为2，预测结果正确。

其它语言的调用方法

除Python外，使用其它语言客户端调用服务都需要根据.proto文件手动生成预测的请求代码文件。调用示例如下：

1. 编写请求代码文件（例如创建tf.proto文件），内容如下。

```
syntax = "proto3";
option cc_enable_arenas = true;
option java_package = "com.aliyun.openservices.eas.predict.proto";
option java_outer_classname = "PredictProtos";
enum ArrayDataType {
  // Not a legal value for DataType. Used to indicate a DataType field
  // has not been set.
  DT_INVALID = 0;
  // Data types that all computation devices are expected to be
  // capable to support.
  DT_FLOAT = 1;
  DT_DOUBLE = 2;
```

```
  DT_INT32 = 3;
  DT_UINT8 = 4;
  DT_INT16 = 5;
  DT_INT8 = 6;
  DT_STRING = 7;
  DT_COMPLEX64 = 8;  // Single-precision complex.
  DT_INT64 = 9;
  DT_BOOL = 10;
  DT_QINT8 = 11;    // Quantized int8.
  DT_QUINT8 = 12;   // Quantized uint8.
  DT_QINT32 = 13;   // Quantized int32.
  DT_BFLOAT16 = 14;  // Float32 truncated to 16 bits.  Only for cast ops.
  DT_QINT16 = 15;   // Quantized int16.
  DT_QUINT16 = 16;  // Quantized uint16.
  DT_UINT16 = 17;
  DT_COMPLEX128 = 18;  // Double-precision complex.
  DT_HALF = 19;
  DT_RESOURCE = 20;
  DT_VARIANT = 21;  // Arbitrary C++ data types.
}
// Dimensions of an array.
message ArrayShape {
  repeated int64 dim = 1 [packed = true];
}
// Protocol buffer representing an array.
message ArrayProto {
  // Data Type.
  ArrayDataType dtype = 1;
  // Shape of the array.
  ArrayShape array_shape = 2;
  // DT_FLOAT.
  repeated float float_val = 3 [packed = true];
  // DT_DOUBLE.
  repeated double double_val = 4 [packed = true];
  // DT_INT32, DT_INT16, DT_INT8, DT_UINT8.
  repeated int32 int_val = 5 [packed = true];
  // DT_STRING.
  repeated bytes string_val = 6;
  // DT_INT64.
  repeated int64 int64_val = 7 [packed = true];
  // DT_BOOL.
  repeated bool bool_val = 8 [packed = true];
}
// PredictRequest specifies which TensorFlow model to run, as well as
// how inputs are mapped to tensors and how outputs are filtered before
// returning to user.
message PredictRequest {
  // A named signature to evaluate. If unspecified, the default signature
  // will be used.
  string signature_name = 1;
  // Input tensors.
  // Names of input tensor are alias names. The mapping from aliases to real
  // input tensor names is expected to be stored as named generic signature
  // under the key "inputs" in the model export.
```

```
  // Each alias listed in a generic signature named "inputs" should be provided
  // exactly once in order to run the prediction.
  map<string, ArrayProto> inputs = 2;
  // Output filter.
  // Names specified are alias names. The mapping from aliases to real output
  // tensor names is expected to be stored as named generic signature under
  // the key "outputs" in the model export.
  // Only tensors specified here will be run/fetched and returned, with the
  // exception that when none is specified, all tensors specified in the
  // named signature will be run/fetched and returned.
  repeated string output_filter = 3;
}
// Response for PredictRequest on successful run.
message PredictResponse {
  // Output tensors.
  map<string, ArrayProto> outputs = 1;
}
```

其中 `PredictRequest` 定义TensorFlow服务的输入格式， `PredictResponse` 定义服务的输出格式。关于ProtoBuf的详细用法请参见ProtoBuf介绍。

2. 安装protoc。

```
#/bin/bash
PROTOC_ZIP=protoc-3.3.0-linux-x86_64.zip
curl -OL https://github.com/google/protobuf/releases/download/v3.3.0/$PROTOC_ZIP
unzip -o $PROTOC_ZIP -d ./ bin/protoc
rm -f $PROTOC_ZIP
```

3. 生成请求代码文件：
   - Java版本

     ```
     $ bin/protoc --java_out=./ tf.proto
     ```

     命令执行完成后，系统会在当前目录生成com/aliyun/openservices/eas/predict/proto/PredictProtos.java，在项目中导入该文件即可。
   - Python版本

     ```
     $ bin/protoc --python_out=./ tf.proto
     ```

     命令执行完成后，系统会在当前目录生成tf_pb2.py，通过 `import` 命令导入该文件即可。
   - C++版本

     ```
     $ bin/protoc --cpp_out=./ tf.proto
     ```

     命令执行完成后，系统在当前目录生成tf.pb.cc和tf.pb.h。在代码中使用 `include tf.pb.h` 命令，并将tf.pb.cc添加至compile列表即可。

7.3.2. Caffe服务请求构造

本文为您介绍如何为基于通用Processor的Caffe服务构造请求数据。

输入数据说明

PAI-EAS预置了Caffe Processor，为保证性能，其输入输出为ProtoBuf格式。

调用案例

PAI-EAS在华东2（上海）的VPC环境中部署了一个Public的测试案例，其服务名称为caffenet_serving_example，访问Token为空。您可以通过URLhttp://pai-eas-vpc.cn-shanghai.aliyuncs.com/api/predict/caffenet_serving_example访问该服务。具体方式如下：

1. 获取模型信息。

   通过Caffe服务的model文件，可以查看模型每个layer的相关信息。例如该案例的model文件内容如下。

   ```
   name: "CaffeNet"
   layer {
    name: "data"
    type: "Input"
    top: "data"
    input_param {
     shape {
      dim: 10
      dim: 3
      dim: 227
      dim: 227
     }
    }
   }
   ....
   layer {
    name: "prob"
    type: "Softmax"
    bottom: "fc8"
    top: "prob"
   }
   ```

   该模型是一个经典的CaffeNet模型，type取值为Input的layer（通常为第一层layer）声明模型的输入，最后一层layer声明模型的输出。该模型的输入shape为[10, 3, 227, 227]，其中第一维表示batch_size（如果单个请求只包含一张图片，则batch_size为1）。构建输入时，无论shape取值如何，都必须将输入展开成一维向量。该示例中，构建输入时，需要将输入构建为1*3*227*227的一维向量。如果服务请求中指定的shape与模型的shape不一致，则预测请求报错。

2. 安装ProtoBuf并调用服务（以Python为例，介绍如何对Caffe服务进行调用）。

   PAI-EAS为Python预先生成了ProtoBuf包，您可以使用如下命令直接安装。

   ```
   $ pip install http://eas-data.oss-cn-shanghai.aliyuncs.com/pai_caffe_predict_proto-1.0-py2.py3-none-any.whl
   ```

   调用服务进行预测的示例代码如下。

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
import requests
from pai_caffe_predict_proto import caffe_predict_pb2
# build request.
request = caffe_predict_pb2.PredictRequest()
request.input_name.extend(['data'])
array_proto = caffe_predict_pb2.ArrayProto()
array_proto.shape.dim.extend([1, 3, 227, 227])
array_proto.data.extend([1.0]*3*227*227)
request.input_data.extend([array_proto])
# 将ProtoBuf转换成string进行传输。
data = request.SerializeToString()
# 该API需要在华东2（上海）的VPC内访问。
url = 'http://pai-eas-vpc.cn-shanghai.aliyuncs.com/api/predict/caffenet_serving_example'
s = requests.Session()
resp = s.post(url, data=data)
if resp.status_code != 200:
  print resp.content
else:
  response = caffe_predict_pb2.PredictResponse()
  response.ParseFromString(resp.content)
  print(response)
```

其它语言的调用方法

除Python外，使用其它语言客户端调用服务都需要根据.proto文件手动生成预测的请求代码文件。调用示例如下：

1. 编写请求代码文件（例如创建caffe.proto文件），内容如下。

```
syntax = "proto2";
package caffe.eas;
option java_package = "com.aliyun.openservices.eas.predict.proto";
option java_outer_classname = "CaffePredictProtos";
message ArrayShape {
 repeated int64 dim = 1 [packed = true];
}
message ArrayProto {
 optional ArrayShape shape = 1;
 repeated float data = 2 [packed = true];
}
message PredictRequest {
 repeated string input_name = 1;
 repeated ArrayProto input_data = 2;
 repeated string output_filter = 3;
}
message PredictResponse {
 repeated string output_name = 1;
 repeated ArrayProto output_data = 2;
}
```

其中 `PredictRequest` 定义caffe服务的输入格式， `PredictResponse` 定义服务的输出格式。关于ProtoBuf的详细用法请参见ProtoBuf介绍。

2. 安装protoc。

```
#/bin/bash
PROTOC_ZIP=protoc-3.3.0-linux-x86_64.zip
curl -OL https://github.com/google/protobuf/releases/download/v3.3.0/$PROTOC_ZIP
unzip -o $PROTOC_ZIP -d ./ bin/protoc
rm -f $PROTOC_ZIP
```

3. 生成请求代码文件：
   - Java版本

     ```
     $ bin/protoc --java_out=./ caffe.proto
     ```

     命令执行完成后，系统会在当前目录生成com/aliyun/openservices/eas/predict/proto/CaffePredictProtos.java，在项目中导入该文件即可。
   - Python版本

     ```
     $ bin/protoc --python_out=./ caffe.proto
     ```

     命令执行完成后，系统会在当前目录生成caffe_pb2.py，通过 `import` 命令导入该文件即可。
   - C++版本

     ```
     $ bin/protoc --cpp_out=./ caffe.proto
     ```

     命令执行完成后，系统在当前目录生成caffe.pb.cc和caffe.pb.h。在代码中使用 `include caffe.pb.h` 命令，并将caffe.pb.cc添加至compile列表即可。

7.3.3. PMML服务请求构造

本文为您介绍如何为基于通用Processor的PMML服务构造请求数据。

输入数据

使用通用PMML Processor部署的模型服务，其请求数据格式为List序列化的字符串，且支持多线程并发处理请求。List中的每个对象格式需要满足PMML模型文件中DataDictionary标签内的数据要求。

以如下PMML模型为例，该模型的特征包括sex、cp及fbs，Label字段为ifhealth。您构造的服务请求数据格式可以为[{“sex”:0,”cp”:1,”fbs”:1},{“sex”:1,”cp”:0,”fbs”:1},{“sex”:0,”cp”:0,”fbs”:0}]。

```
<PMML version="4.3">
<Header/>
<DataDictionary numberOfFields="4">
 <DataField name="sex" optype="continuous" dataType="double"/>
 <DataField name="cp" optype="continuous" dataType="double"/>
 <DataField name="fbs" optype="continuous" dataType="double"/>
 <DataField name="ifhealth" optype="categorical" dataType="double">
  <Value value="1"/>
  <Value value="0"/>
 </DataField>
</DataDictionary>
... a model ...
</PMML>
```

7.3.4. Blade服务请求构造

本文为您介绍如何为基于通用Processor的Blade服务构造请求数据。

官方SDK

您可以直接下载Blade服务请求的官方SDK，详情请参见GitHub官方SDK。

示例代码

使用Python SDK调用Blade服务的示例代码请参见示例代码。

8.使用案例

8.1. PAI-Studio模型部署及预测

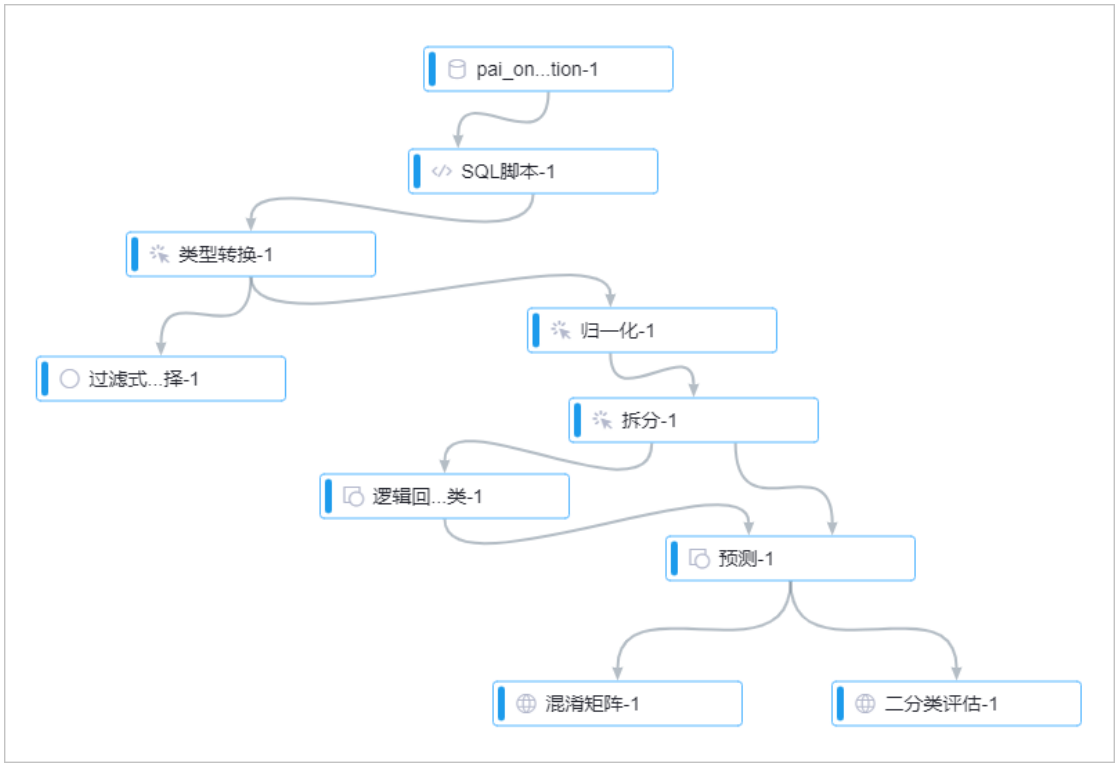
本文使用心脏病预测案例，为您介绍如何通过PAI进行一站式机器学习，包括模型训练、模型部署及在线预测服务调用。

部署模型

1. 进入PAI-Studio项目空间。
   i. 登录PAI控制台。
   ii. 在左侧导航栏，选择**模型开发和训练 > 可视化建模（Studio）**。
   iii. 在页面左上方，选择使用服务的地域。
   iv. （可选）在**PAI可视化建模**页面的搜索框，输入项目名称，搜索项目。
   v. 单击待打开的项目**操作**列下的**进入机器学习**。
2. 构建并运行实验。
   i. 在左侧导航栏，单击**首页**。
   ii. 在**模板列表**，单击**心脏病预测案例**下的**从模板创建**。
   iii. 在**新建实验**对话框，配置参数。

参数	描述
名称	实验的名称，不能超过32个字符。例如，心脏病预测案例。
项目	PAI-Studio项目空间名称，系统自动带入，不能修改。
描述	对实验进行简要描述，以区分多个实验。例如，该实验包括数据预处理、特征工程、模型训练及预测等全套机器学习流程。
名称	实验的存储位置，默认存储在我的实验文件夹。

iv. 单击**确定**，等待大约十秒钟，实验构建成功后如下图所示。



v. 单击画布上方的**运行**，运行过程中右键单击组件，可以查看组件的输出。

3. 部署模型。

i. 实验运行结束后，鼠标悬停在画布上方的**部署**，单击**模型在线部署**。

ii. 在**选择要部署的模型**对话框，选择待部署的模型并单击**下一步**。

iii. 在**资源和模型**面板，配置参数。

区域	参数	描述
资源组	**资源组种类**	支持使用公共资源组或已购买（创建）的专属资源组部署模型服务。
模型	**Processor种类**	心脏病模型的**Processor种类**为PMML，系统自动配置。
	资源种类	仅**资源组种类**为**公共资源组**时，该参数生效，且取值为CPU。

> **说明** 部署不同的模型时，需要配置的参数可能不同。关于资源和模型的更多参数请参见控制台上传部署。

iv. 单击**下一步**。

v. 在**部署详情及配置确认**面板，配置部署方式相关参数。PAI-EAS支持以下两种在线部署方式，您可以根据实际需要选择合适的部署方式：

- 新建服务

  新部署一个服务。

- 新增蓝绿部署

  为已有的服务新增一个关联服务，可以配置两个服务的流量分配。

新建服务

1. 在**部署详情及配置确认**面板，单击**新建服务**，并配置相关参数。

参数		描述
自定义模型名称		只能包含数字、小写字母及下划线（_），且必须以字母开头。
模型部署占用资源	**实例数**	建议配置多个服务实例，以避免单点部署带来的风险。
	Quota	仅**资源组种类**为公共资源组时，该参数生效。1 Quota等于1核加4 GB内存，**Quota**取值范围为1~100。
	核数	仅**资源组种类**为专属资源组时，该参数生效。
	内存数（M）	仅**资源组种类**为专属资源组时，该参数生效。

> 说明
> - 单实例内的CPU、GPU及内存需要位于同一台机器上。如果资源不够，则会导致部署失败。
> - 对于高稳定性要求的正式服务，建议使用包含多台机器的资源组，并部署多个服务实例。

2. 单击**部署**。

新增蓝绿部署

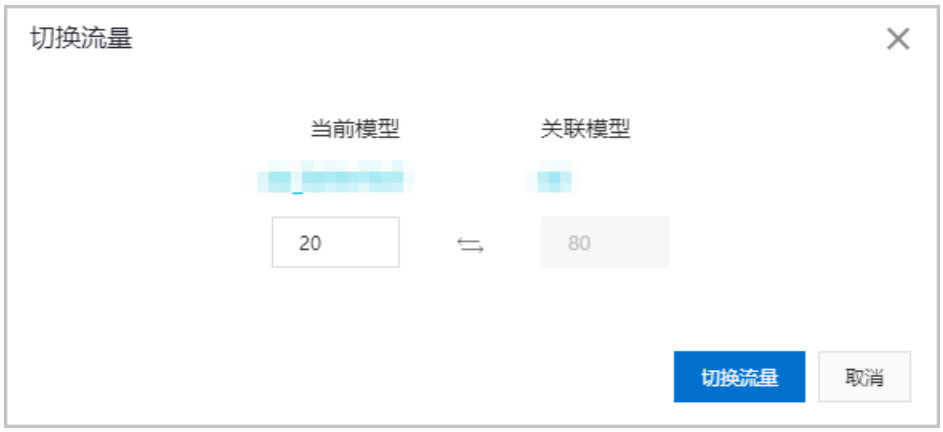
1. 在**部署详情及配置确认**面板，单击**新增蓝绿部署**，并配置相关参数。

参数		描述
选择已部署模型服务		选择已部署的模型服务，默认为正在服务的版本。
模型部署占用资源	**实例数**	建议配置多个服务实例，以避免单点部署带来的风险。
	Quota	仅**资源组种类**为公共资源组时，该参数生效。1 Quota等于1核加4 GB内存，**Quota**取值范围为1~100。
	核数	仅**资源组种类**为专属资源组时，该参数生效。
	内存数（M）	仅**资源组种类**为专属资源组时，该参数生效。

2. 单击**部署**。
3. 部署完成后，可以修改两个模型的流量分配。

i. 在**PAI EAS模型在线服务**页面，单击待切换流量模型的**切换流量**。

ii. 在**切换流量**对话框，修改**当前模型**流量，系统会自动更新**关联模型**流量。**当前模型**的初始流量为100%，您可以根据实际情况配置模型流量。下图中**当前模型**取值20，表示**当前模型**承载20%流量，**关联模型**承载80%流量。



iii. 单击**切换流量**。

iv. 在**消息提醒**对话框，单击**确认**。

在线调试

模型部署完成后，您可以在线调试服务。

1. 进入**PAI EAS模型在线服务**页面。
   i. 登录PAI控制台。
   ii. 在左侧导航栏，选择**模型部署 > 模型在线服务（EAS）**。
2. 在**PAI EAS模型在线服务**页面，单击待调试服务**操作**列下的**在线调试**。
3. 在调试页面的**在线调试请求参数**区域，配置参数。

参数	描述
接口地址	系统自动填入，无需手动配置。
Token	系统自动填入，无需手动配置。
Request Body	输入数据（特征）。以心脏病预测案例的逻辑回归模型为例，**Request Body**信息如下。 [{"sex":0,"cp":0,"fbs":0,"restecg":0,"exang":0,"slop":0,"thal":0,"age":0,"trestbps":0,"chol":0,"thalach":0,"oldpeak":0,"ca":0}]

4. 单击**发送请求**，即可在**调试信息**区域查看预测结果。

```
Request:

http://[redacted]1.cn-shanghai.pai-eas.aliyuncs.com/api/predict/heart_test
Authorization: EAS [redacted]
Date: Thu, 13 Aug 2020 11:01:13 GMT
Content-MD5: [redacted]
Content-Type: application/octet-stream
[{"sex":0,"cp":0,"fbs":0,"restecg":0,"exang":0,"slop":0,"thal":0,"age":0,"trestbps":0,"chol":0,"thalach":0,"oldpeak":0,"ca":0}]

Response:

200
Server: nginx/1.13.11
Connection: keep-alive
Content-Length: 56
Date: Thu, 13 Aug 2020 11:01:13 GMT
Content-Type: text/html; charset=utf-8
[{"p_0":0.9963996241581827,"p_1":0.0036003758418172898}]
```

8.2. 部署PS-LR模型

PS-LR（Parame Server-Logistical Regression）是基于参数服务器的逻辑回归算法，适用于超大规模特征和样本的推荐系统及广告引擎。本文介绍如何使用PAI-EAS部署PS-LR模型并进行在线调试。

前提条件

- 已购买PAI-EAS专属资源组，您可以根据需要选择预付费或后付费模式，详情请参见专属资源组。
- 完成PS-LR模型训练，从而获得可以公开访问的模型存储地址或本地模型。

背景信息

因为PS-LR模型需要使用自定义Processor，所以只能将其部署至专属资源组。

部署模型

1. 进入**PAI EAS模型在线服务**页面。
   i. 登录PAI控制台。
   ii. 在左侧导航栏，选择**模型部署 > 模型在线服务（EAS）**。
2. 在**PAI EAS模型在线服务**页面，单击**模型上传部署**。
3. 在**资源和模型**面板，配置参数。

参数	描述
资源组种类	选择已购买的PAI-EAS专属资源组（根据资源组名称选择）。
Processor种类	选择自定义processor。
Processor语言	选择cpp。

参数	描述
Processor包	您可以通过以下任何一种方式配置该参数： ○ 在**Processor包**文本框，输入http://easprocessor.oss-cn-shanghai.aliyuncs.com/public/pslr_processor_release.tar.gz。 ○ a. 下载Processor包至本地。 b. 单击**Processor包**下的**上传本地文件**，并根据提示上传已下载的Processor文件。 系统会将文件上传至当前地域的官方OSS路径，并自动配置**Processor包**。 **说明** 通过本地上传的方式，可以使系统在模型部署时，快速加载Processor。
Processor主文件	输入libpslr.so。
模型文件	您可以通过以下任何一种方式配置该参数： ○ **本地上传**： a. 单击**本地上传**。 b. 单击**上传本地文件**，并根据提示上传本地模型文件。 ○ **OSS文件导入**。 单击**OSS文件导入**，并下方列表中选择模型文件所在的OSS路径。 ○ **公网下载地址**。 单击**公网下载地址**，并在下方文本框中输入可以公开访问的URL地址。

4. 单击**下一步**。
5. 在**部署详情及配置确认**面板，配置参数。
   i. 选择**新建服务**。
   ii. 输入**自定义模型名称**。
   iii. 根据实际需要，配置**实例数**、**核数**及**内存数**。
   iv. 单击**部署**。

测试接口

1. 在**PAI EAS模型在线服务**页面，单击**操作**列下的**在线调试**。
2. 在调试页面，核对模型的**接口地址**和**Token**。通常系统会自动配置已部署模型的**接口地址**和**Token**，无需手动修改。
3. 在调试页面的**Request Body**区域，输入测试数据。在单个请求中，可以输入多条数据。例如[{"1":1,"2":1},{"1":1,"3":1}]。
4. 单击**发送请求**。
5. 在**调试信息**区域，查看接口返回结果。

8.3. 示例代码

本文为您介绍PAI-EAS的服务调用示例和Python SDK。

PAI-EAS相关的示例代码都可以从GitHub官网获取。

服务调用示例（Python）

支持通过公网地址或VPC内网地址调用服务（简单的HTTP调用），服务请求的输入输出数据格式均由部署服务的Processor指定。通过公网地址或VPC内网地址调用，接口使用的鉴权方式不同。PAI-EAS提供了基于官方Processor部署的服务调用代码示例，详情请参见服务调用示例。该示例包括：

- 公网地址调用的PMML示例。
- VPC地址调用的PMML示例。
- 公网地址调用的TensorFlow示例（调用Caffe可以参考该示例，二者的输入输出一致）。
- VPC地址调用的TensorFlow示例（调用Caffe可以参考该示例，二者的输入输出一致）。

服务调用Python SDK

服务调用的Python SDK封装了多线程连接池、请求数据构造及服务请求等功能，详情请参见官方Python SDK。