ALIBABA CLOUD

Alibaba Cloud

API 网关

创建API

文档版本：20220601

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

本文讲述该如何创建API。

1 如何创建API

创建API的方式有三种：通过控制台创建API、通过管控API创建API、通过导入Swagger创建API

- **通过控制台创建API**

登录阿里云API网关控制台，左上角选择地域，然后在分组中创建API。具体步骤可参考创建 API

- **使用管控API创建API**

可以通过管控API创建API，具体操作可参考创建 API

- **通过导入Swagger创建API**

Swagger是一种用于描述API定义的规范，被广泛应用于定义和描述后端应用服务的API。现在，API网关支持导入Swagger 2.0的文件来创建API。目前有两种方式：

- 通过管控API导入Swagger创建，具体参考导入Swagger创建API;

- 在控制台上进行操作，具体可参考：通过导入Swagger创建API

2 API网关实例选择及分组

2.1 实例选择

API网关实例指用于接入并处理您的API的一组资源，包含公网IP、内网IP、公网出口、负载均衡，API网关支持共享实例和专享实例两种，适用于不同的使用场景：

- 专享实例适用场景：提供高性能和高SLA保证，适合生产环境使用。
- 共享实例适用场景：开发测试、评估和小规模的生产环境使用。

关于两种实例的具体差异可以参见 实例类型与选择指南。

2.2 分组

API 分组是 API 的管理单元。创建 API 之前，需要先创建分组，分组创建时，系统会为分组分配一个二级域名。**该二级域名仅供测试使用，客户端直接调用时会有每天1000次访问限制**。 建议为分组绑定一个在阿里云系统备案成功的独立域名，进而可以通过域名调用API。

关于分组域名绑定的详细信息可参考分组的域名绑定

3 创建API的步骤

3.1 定义请求的基本信息

API基本信息包括API分组、API名称、安全认证方式、API选项、和描述。

3.2 定义API请求

定义用户如何请求API，包括请求类型、协议、自定义域名、请求Path、HTTP Method、入参请求模式、和入参定义。

API网关支持的请求协议包括HTTP、HTTPS、WEBSOCKET。

3.3 定义后端服务信息

定义一些参数的前后端映射，即API后端服务的配置，包括后端服务类型、后端服务地址、后端Path、HTTP Method、后端超时时间、参数映射、常量参数、系统参数。用户请求到达API网关后，API网关会根据您的后端配置，映射为对应的后端服务的请求形式，请求后端服务。

API网关支持的后端服务包括HTTP(S)、函数计算、VPC、和Mock。

HTTP(S)，创建后端服务为HTTP的API

函数计算，使用函数计算做为API后端服务

VPC，使用VPC内资源作为API的后端服务

Mock，使用Mock方式做为API后端服务

3.4 定义返回结果

录入返回ContentType、返回结果示例、失败返回结果示例、和错误码定义。

4 环境管理

当前每个API分组支持设置三个环境：测试（TEST）、预发（PRE）和线上（RELEASE）。目的是能够满足不同研发场景下的API 调用需求，如API测试环境，后端服务对应到测试环境资源，从而可以保证在同一套API配置的情况下，供测试人员进行测试使用。

更多详情可参考环境管理

2.创建 API

创建API即录入API的定义，需要录入API的基本信息、服务信息、请求信息、和返回信息，然后对创建的API进行调试，及进行安全配置。经测试证明API可用后，可发布上线供用户使用。

1 定义 API

在API网关控制台中API列表页面，单击创建API，即进入API的创建和定义流程。

1.1 定义请求的基本信息

参数	描述
API分组	分组是API的管理单元，创建API之前您需要先创建分组，选择分组即选择Region。
API名称	所创建的API的名称。API名称标识需在所属分组内具有唯一性。
安全认证方式	目前支持阿里云APP、OpenID Connect、OpenID Connect&阿里云APP和无认证四种认证方式。 • **阿里云APP**：要求请求者调用该API时，需通过对APP的身份认证。 • **OpenID Connect**：是一套基于OAuth 2.0协议的轻量级规范，提供通过RESTful APIs进行身份交互的框架。可以使用OpenID Connect和您的自有账号系统无缝对接。详细介绍请参见API 网关 OpenID Connect 使用指南。（仅经典网络实例可用） • **- OpenID Connect & 阿里云APP**：同时进行OpenID Connect和阿里云APP认证。（仅经典网络实例可用） • **无认证**：即任何人知晓该API的请求定义后，均可发起请求，网关不对其做身份验证，均会将请求转发至您的后端服务。（强烈不建议使用此模式。）
签名算法	参与签名的算法 • HmacSHA256 • HmacSHA1和HmacSHA256：表示同时支持这2种签名算法。
描述	填写API的相关描述。

1.2 定义API请求

定义用户如何请求API，包括协议、请求Path、HTTP Method、入参请求模式、和入参定义。

参数	描述

参数	描述
请求类型	请求类型： 有4种类型可选。 普通请求：普通的HTTP(S)协议请求。 注册请求（双向通信）：是双向通信的管理信令，是客户端发送给服务端，注册设备使用。 注销请求（双向通信）：是双向通信的管理信令，是客户端发送给服务端，注销设备，注销成功后，将不再接受服务端推送的下行通知。 下行通知请求（双向通信）：用户后端服务，在收到客户端发送的注册信令后，记住注册信令中的设备ID字段，然后就可以向API网关发送接收方为这个设备的下行通知信令了。只要这个设备在线，API网关就可以将此下行通知发送到端。
请求协议	支持HTTP、HTTPS、WEBSOCKET。
请求Path	Path指相对于服务host，API的请求路径。请求Path可以与后端服务实际Path不同，您可以随意撰写合法的有明确语义的Path给用户使用。您可以在请求Path中配置动态参数，即要求用户在Path中传入参数，同时您的后端又可以不在Path中接收，可以映射为在Query、Header等位置接收。
HTTP Method	支持标准的HTTP Method，可选择PUT、GET、POST、DELETE、PATCH、HEAD、OPTIONS或ANY。
入参定义模式	可选入参映射（过滤未知参数）、入参映射（透传未知参数）和入参透传。 • **入参映射（过滤未知参数）**：表示Query，Path 和 Body Form位置的参数需要配置前后端映射，网关只会透传给后端配置过的参数，其他参数会被过滤掉。 • **入参映射（透传未知参数）**：表示Query，Path 和 Body Form位置的参数需要配置前后端映射，网关除配置的请求参数外，不对其他位置的请求参数进行映射与校验，用户参数会透明传递给后端。 • **入参透传**：表示不需要在入参定义处配置Query，Body Form参数（PATH参数仍需要配置）。客户端传给网关的参数都会被网关透传给后端

配置入参定义：定义您API的请求入参，即配置用户需要在什么位置传入什么参数。可选位置有Head、Query、Body、Path（Parameter Path），尤其当您在Path中配置了动态参数，那么在入参配置时需要对动态参数做配置说明。支持的参数类型有String、Integer、Boolean。

- 需要注意所有参数的名称会校验是否唯一。
- 您可以使用左侧的快捷键快速调整参数顺序。
- 您可以使用操作图标下的**移除**选项，移除不需要的参数。

> **说明**
>
> 如果您在Path中配置了动态参数，存在参数位置为Parameter Path的同名参数。

设置参数校验规则： 您可以单击操作图标中的**编辑更多**配置校验规则。如String的长度，Number的枚举等等。网关会参照校验规则对请求做初步校验，如果入参不合法，则不会到达您的后端服务，大大的降低了后端服务的压力。

1.3 定义后端服务信息

这部分主要是定义一些参数的前后端映射，具体描述的是您后端真实服务的API配置。用户请求到达网关后，网关会根据您的后端配置映射为对应实际后端服务的请求形式，去请求您的后端。包括后端服务地址、后端Path、后端超时时间、参数映射、常量参数、系统参数。

后端基础定义

参数	描述
后端服务类型	目前支持HTTP/HTTPS、函数计算、VPC、Mock四种类型。 • HTTP(s): 默认类型，后端是HTTP或者HTTPS协议的服务接入，当网关和后端服务网络能直接连通时使用。若是HTTPS服务，后端服务必须有SSL证书。 • 函数计算：若选择函数计算为后端服务，需先在函数计算控制台中创建函数，并需填入函数服务名称和函数名称，和获取函数计算角色Arn。 • VPC: 当后端服务在某个VPC内时使用。 • Mock: 用于模拟最初预定的返回结果HTTP/HTTPS：若您的服务为HTTP/HTTPS服务，则选择此项。
VPC授权名称	当您的后端服务在VPC中时，需要填写创建VPC授权时设置的VPC授权名称。
后端服务地址	后端服务的Host，可以是一个域名，也可以是http(s)://host:port的形式。填写时，必须包含http://或https://。
后端请求Path	Path是您的API服务在您后端服务器上的实际请求路径。若您后端Path需要接收动态参数，则需要声明调用者需传入参数的具体位置和参数名，即声明映射关系。
HTTP Method	支持标准的HTTP Method，可选择PUT、GET、POST、DELETE、PATCH、HEAD、OPTIONS或ANY。
后端超时	指API请求到达网关后，网关去调API后端服务的响应时间。由网关请求后端开始到网关收到后端返回结果。该值不能超过30秒。超过该值网关会放弃请求后端服务，并给用户返回相应的错误信息。

配置后端服务参数： 网关支持参数在前端、后端的全映射，包括名称映射和位置映射。位置映射包括Path、Header、Query、Body的混排映射。也就是说，您可以将您的后端服务通过映射完成包装成更规范、更专业的API形态。这部分就是在声明前后端API映射关系的。

配置常量参数：您可以配置常量参数。您配置的常量参数对您的用户不可见，但是API网关会在中转请求时，将这些参数加入到请求中的指定位置，再传递至后端服务，实现您的后端的一些业务需求。比如您需要网关每次请求您后端时都带有参数abc，您可以直接将abc配置为常量参数。请求达到网关后，网关会自动在指定位置加上该参数再去请求您的后端。

配置系统参数：指API网关的系统参数。默认系统参数不会传递给您，但是如果您需要获取系统参数，您可以在API里配置接收位置和名称。具体内容如下表：

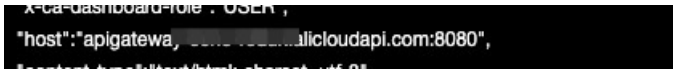
参数名称	参数含义
CaClientIp	发送请求的客户端IP（若您配置了WAF、CDN等，则记录的是回源IP，真实IP需要在X-Forwarded-For中查看）
CaDomain	发送请求的域名
CaRequestHandleTime	请求时间（格林威治时间）
CaAppId	请求的APP的ID
CaRequestId	RequestId
CaApiName	API名称
CaHttpSchema	用户调用API使用的协议，http或者https
CaProxy	代理（AliCloudApiGateway）
CaClientUa	请求客户端的User-Agent
CaCloudMarketInstanceId	请求的云市场商品的实例ID
CaAppKey	请求的APP的KEY
CaStage	请求的环境（RELEASE、TEST、PRE）

目前共享实例可以在分组详情中设置**透传HOST头（域名头）**，专享实例若有需求可提交工单开启此功能，开启后API网关会将请求域名透传至后端，若没有开启，后端收到的是用户在API网关填写的后端HOST。示例如下：

示例中API分组绑定的请求HOST为xuemeng.XXXX.com，API后端HOST为apigatewayXXXXXXalicloudapi.com:8080，配置前后如下

透传HOST头（域名头）开启时：

透传HOST头（域名头）未开启时：



> **说明**
>
> 您需确保您录入的所有参数的参数名称全局唯一，包括Path中的动态参数、Headers参数、Query参数、Body参数（非二进制）、常量参数、系统参数。如果您同时在Headers和Query里各有一个名为**name**的参数，将会导致错误

1.4 定义返回结果

录入返回ContentType、返回结果示例、失败返回结果示例、和错误码定义。

2 API 调试

API定义录入完成后，您可以在API调试页面调试API，以确定API的可用性。

API创建、定义完成后，页面自动跳转到**API列表**页。您可以通过此页面按钮，测试创建的API是否可用，请求链路是否正确。

1. 单击API名称或**管理**按钮，进入**API定义**页面。
2. 单击左侧导航栏中**调试API**。
3. 输入请求参数，单击**发送请求**。

返回结果将显示在右侧页面。

如果调试返回成功结果，则说明该API可以使用。

如果返回代码为4XX或5XX，则表示存在错误。请参见如何获取错误信息和错误代码表。

3 后续步骤

完成以上定义后和初步调试后，您就完成了API的创建。您可以发布API到测试、预发、线上环境，继续调试或供用户使用。还可以为API绑定客户端签名说明文档等安全配置。

3.分组的域名绑定

用户使用API网关对外提供服务时，需要使用自己拥有的域名开放自己的能力，本文主要描述如何将用户自己的域名绑定到API网关上，让客户端使用自己的域名来调用其开放的API。

1. 概述

1.1 域名与分组、API之间的关系

- 用户需要将自己拥有的域名绑定到API网关的分组上，建立域名与分组之间的映射关系。
- API网关在接收到客户端发出的HTTP请求时，根据HTTP请求中的域名来定位到这个请求所属的API分组，再通过HTTPMethod和PATH确定唯一的API。

API网关为每个分组默认提供了公网二级域名，如果客户端直接调用API分组提供的公网二级域名，将会受到每天1000次调用的限制。并且调用API网关提供的公网二级域名时，所有应答都会默认返回一个”Content-Disposition: attachment; filename=ApiResponseForInnerDomain“的头。您在正式生产环境开放API时，需要为API分组绑定独立域名才可正常使用，不受此项限制。

1.2 域名备案

在国内REGION绑定独立域名到分组上的前提是独立域名需要在阿里云备案或者在阿里云备案接入。海外REGION不需要域名备案。

> **说明**
>
> 绑定内网类型域名不需要进行备案。

1.3 域名所有权确认

域名在API网关上没有被同实例、同BasePath的分组绑定并且不和其他已经绑定的泛域名冲突，才能成功绑定。域名的所有权验证有两种，客户只需要满足任一条件即可

1. 用户将域名通过CNAME的方式解析到API网关分组上的二级域名上；
2. 用户在绑定的域名上增加一条记录类型为TXT的解析，记录名称为”分组ID.域名”，记录值为”apigateway-domain-verfication=公网分组二级域名“，下面举个例子：

一个分组，ID为b7eb2f79e64f4431b08bbb948ed2567e，二级域名为b7eb2f79e64f4431b08bbb948ed2567e-cn-hangzhou.alicloudapi.com，绑定的域名为单域名youdomain.com或者泛域名*.yourdomain，用户需要增加一条域名记录，类型为TXT，主机记录（RR）为b7eb2f79e64f4431b08bbb948ed2567e.yourdomain.com，记录值为apigateway-domain-verfication=b7eb2f79e64f4431b08bbb948ed2567e-cn-hangzhou.alicloudapi.com

> **说明**
>
> - 绑定内网类型域名将不进行域名所有权确认。
> - 如果不将域名通过CNAME绑定到API网关，客户端通过域名请求时，无法请求到API网关。

2.单域名绑定

将用户自己的单域名绑定到API分组需要以下两个步骤：

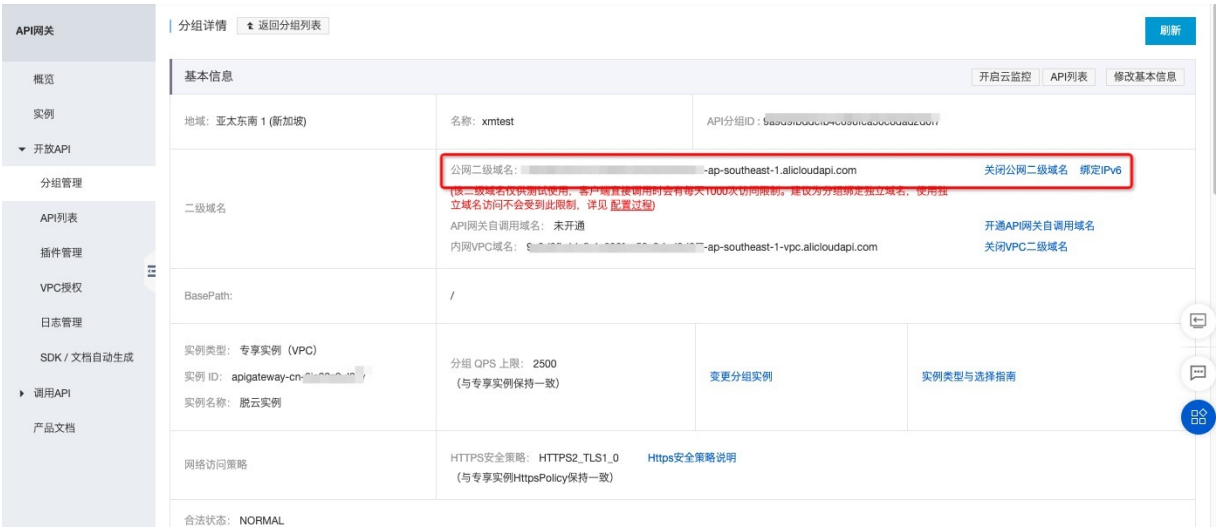
1. 域名解析：将域名通过CNAME方式或者TXT解析方式解析到API网关分组上提供的二级域名；

2. 域名绑定：在API网关控制台的分组详情页面将域名绑定到对应的分组上。

2.1 域名解析

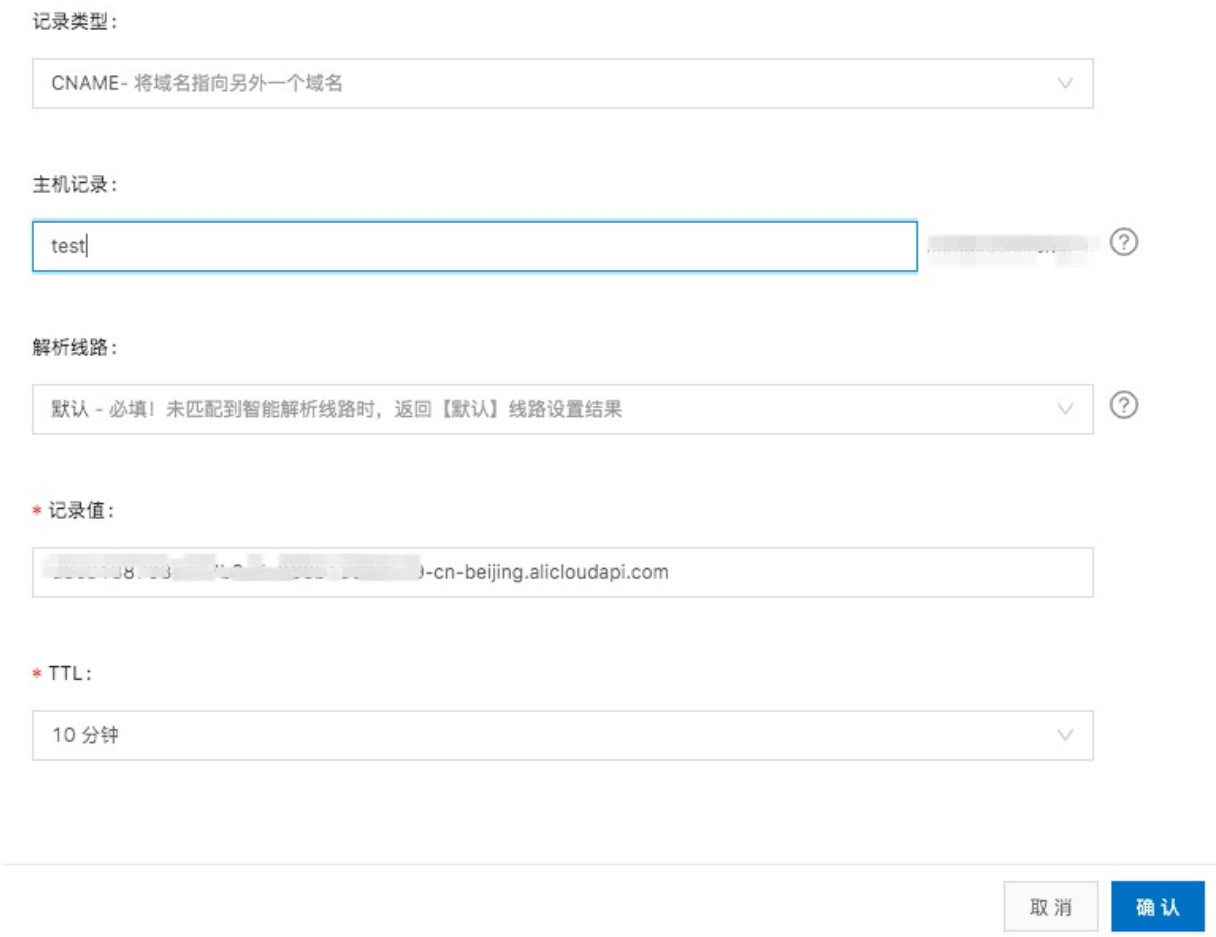
2.1.1 公网域名解析

1. 在API网关分组详情页面找到这个分组对应的API网关提供的公网二级域名



2. 登录云解析DNS 控制台， 在左侧栏点击域名解析，找到要解析的域名，点击域名上的链接进入域名的管理页面。

3. 增加一条需要绑定到API网关的域名的子记录，记录类型选择CNAME ，主机记录填写域名的前缀，记录值填写刚才第一步获取到的公网二级域名，点击确定就完成了。

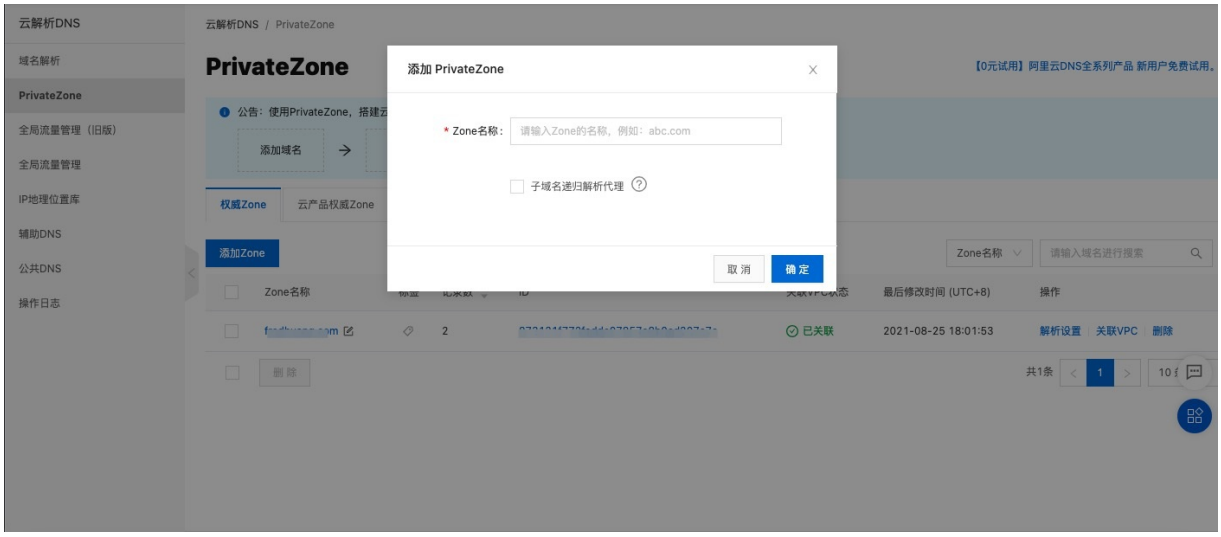


2.1.2 内网域名解析

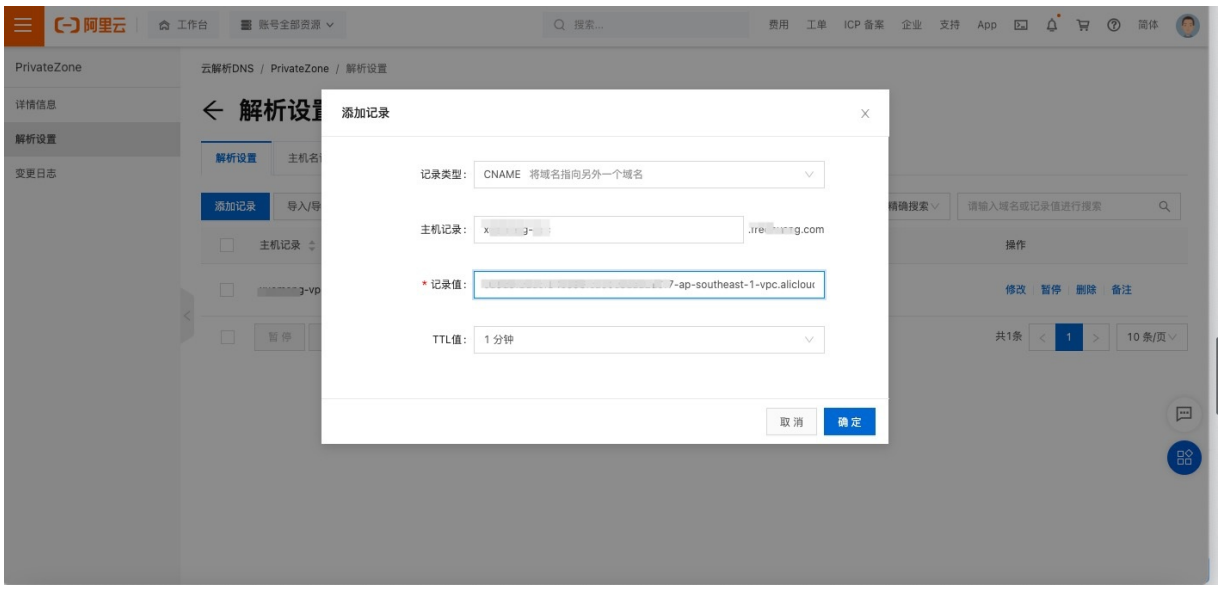
1. 在API网关分组详情页面找到这个分组对应的API网关提供的内网vpc域名。

2. 登录 PrivateZone 控制台，单击页面中的**添加Zone**，在 **添加PrivateZone** 对话框中，输入Zone名称。

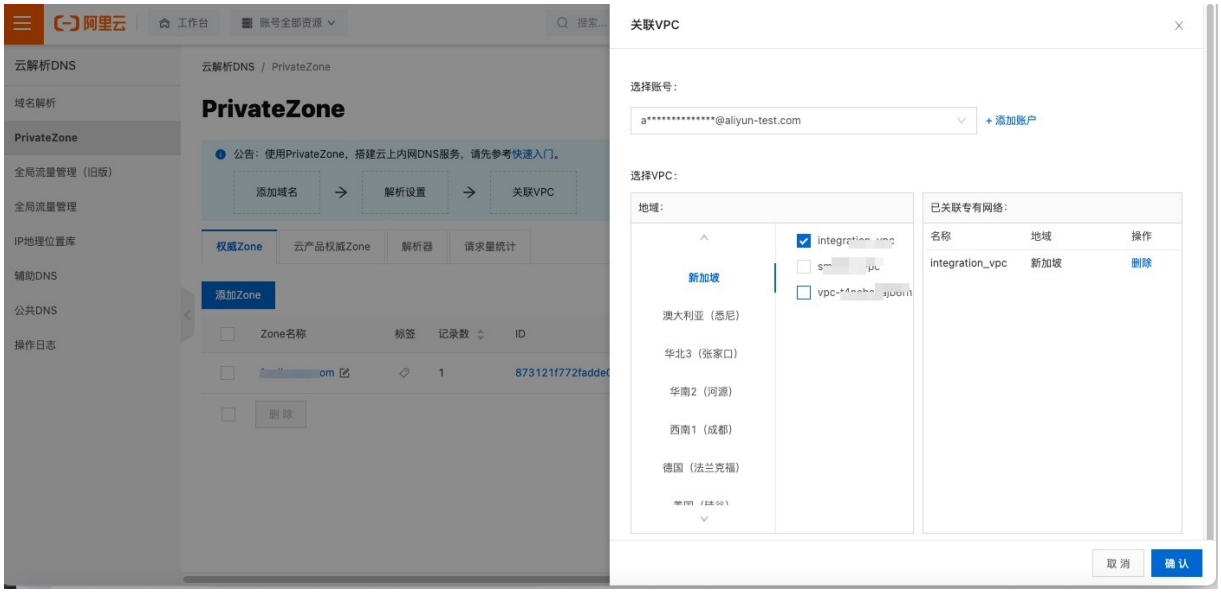
> **说明**
>
> 此处的 **Zone名称** 填写API网关分组所绑定的自定义的域名（私有域名），即您想要在VPC环境内专门为其设置PrivateZone解析记录的域名名称。



3. 单击Zone名称，可以进入解析记录控制台，为该私有域名添加CNAME解析记录。记录类型选择CNAME，主机记录填写域名前缀，记录值填写步骤1 获取到的内网vpc域名，点击确定。如图：



4. 前往 **PriviteZone** 页面。找到刚刚添加的Zone，单击其操作列下的 **关联VPC**。在 **关联VPC** 对话框中，从对应 **地域** 的 **专有网络列表** 中选择并添加需要关联的VPC。点击确认，完成关联。



> **说明**
>
> Zone关联VPC后，在被关联VPC内的ECS上，您的Zone（私有域名）将按照PrivateZone解析记录被解析，其公网解析记录则会被覆盖。**注意**：VPC环境外，该Zone的公网解析记录不受影响。通过为Zone添加PrivateZone解析记录，可以防止记录为空的Zone将需要使用的公网解析覆盖掉，造成异常。更多详情可参考开通PrivateZone 文档。

2.2 绑定域名的流程

1. 进入API网关控制台，点击左边菜单的分组管理，进入分组列表页面，然后选择要绑定域名的分组，进入分组详情页面，在页面右下方看到绑定域名的按钮，点击按钮。

2. 进入域名绑定页面，填写刚才做完解析的域名，点击确定。

- 域名：填写要绑定的域名。
- 环境：指的是域名绑定的环境，分别如下：
  - 指定为测试环境（TEST）：仅支持调用测试环境API。
  - 指定为预发环境（PRE）：仅支持调用预发环境API。
  - 指定为线上环境（RELEASE）：仅支持调用线上环境API。
  - 默认（使用X-Ca-Stage确定环境）：以上三种环境均可调用，调用时在请求的Header添加X-Ca-Stage参数指定调用的环境。
- 网络类型：公网类型仅支持通过公网调用API。内网类型仅支持通过内网调用API。

> 说明
>
> - 内网类型域名不进行域名所有权校验，若与实例内其他分组绑定域名冲突，将绑定失败。
> - 域名绑定成功后，不支持修改网络类型，若配置有误，可以删除域名重新绑定。

2.3 域名绑定常见问题

域名绑定失败，会有以下几种情况(附处理办法):

- 要绑定的域名已经在API网关被当前用户绑定到了当前实例的其他分组上，或者和当前用户已经绑定的其他域名有范围冲突(指泛域名和单域名之间存在的覆盖关系)，此时用户需要从原来的分组上解绑该自定义域名，才能重新在当前分组上绑定该域名；
- 要绑定的域名已经在API网关被其他用户绑定到了该用户名下的分组上，或者和当前用户已经绑定的其他域名有范围冲突(指泛域名和单域名之间存在的覆盖关系)，此时用户必须通过本文说的1.3中的办法证明此自定义域名的所有权才能成功绑定。

2.4 调用验证

绑定成功后，我们就可以随意使用绑定的域名来访问这个分组下的API了，假如您有一个API，可以通过简单的curl来访问：

```
curl http://yourdomain.com/apipath -i
HTTP/1.1 200 OK
Date: Mon, 23 Mar 2020 08:40:01 GMT
Connection: keep-alive
Keep-Alive: timeout=25
Server: Jetty(7.2.2.v20101205)
X-Ca-Request-Id: E2B8CBAB-D6EF-4576-838F-44DDC1A6B20D
```

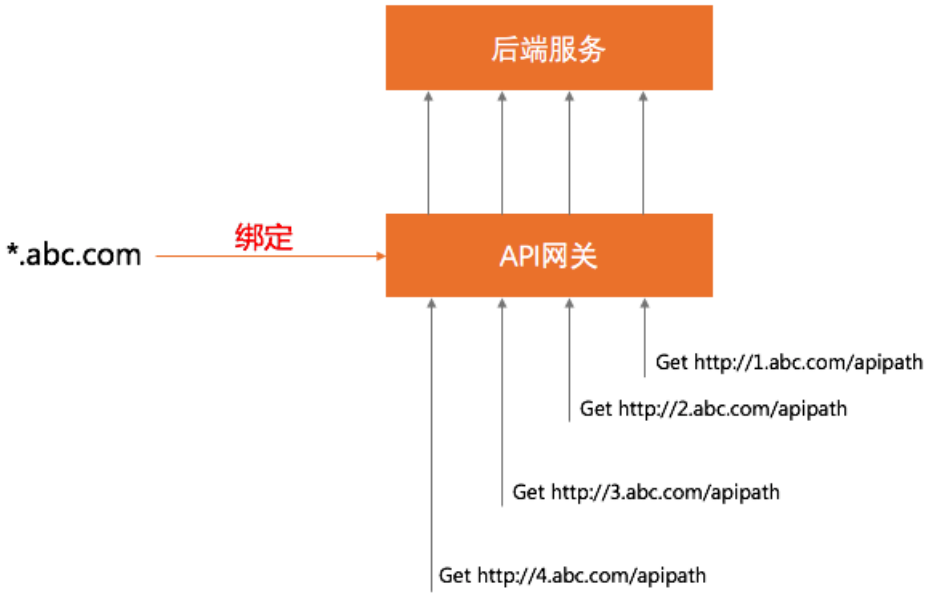
> **注意**
>
> 若绑定的是内网域名，需在您已关联的vpc内进行测试。

3.泛域名绑定

API网关目前已经支持了泛域名绑定，用户可以将泛域名解析到API网关的公网二级域名，之后在控制台上将对应的泛域名绑定至API分组，就可以通过泛域名来调用API网关上托管的对应分组下的所有API。

先介绍一下泛域名绑定的功能，假如您是abc.com这个域名的拥有者，您想将abc.com这个域名的所有子域名（比如1.abc.com，2.abc.com）都指向API网关对外提供服务，现在可以通过两个步骤实现这个能力：

1. 在您的域名解析管理平台将*.abc.com通过CNAME的方式解析到API网关分组的公网二级域名上；
2. 在API网关控制台的分组页面上，将*.abc.com绑定到对应的分组上。
3. 一旦绑定成功后，客户端就可以通过abc.com这个域名的所有子域名（比如1.abc.com，2.abc.com）来访问所绑定的分组下所有API了，比如对应分组下有个API可以通过Get方法匿名访问，那么在绑定了*.abc.com这个泛域名之后，就可以通过1.abc.com，2.abc.com等域名同时来访问了：



> 注意
>
> 仅VPC实例支持泛域名能力。

3.1 绑定泛域名的流程

泛域名的绑定和本文第二章描述的单域名版绑定的流程是大体一致的：

1. 绑定泛域名的时候必须要验证域名所有权，具体验证方法参见1.3节的描述；
2. 泛域名绑定成功之后，必须在分组详情页面设置对应的泛域名模板，泛域名的调用才会生效：

泛域名模板主要是为了配置域名参数所用，也就是泛域名中的可变字段实际可以作为一个参数传给后端服务的。

4. 设置分组默认域名

API网关允许用户上传域名对应的HTTPS证书，并对外提供安全级别更高的HTTPS调用能力。在分组下绑定了有多个域名时，并且多个域名同时支持HTTPS调用时，需要设置默认域名，才能在接收到不支持SNI客户端发送的SSL握手请求时返回默认域名证书，否则API网关会随机返回域名证书。设置分组默认域名仅对专享实例生效，共享实例默认不支持默认证书，不支持SNI的低版本客户端进行HTTPS访问时可能会产生证书混淆的问题。

在分组详情页面的配置该分组的默认域名：

> 注意
>
> 专享实例上如果绑定了多个分组，只能加载第一个分组的默认域名，其他设置无效。

4.通过导入Swagger创建API

Swagger是一种用于描述API定义的规范，被广泛应用于定义和描述后端应用服务的API。现在，API网关支持导入Swagger 2.0的文件来创建API，您可以参考Import Swagger，或在控制台上进行操作，入口见下图：

API网关的Swagger扩展基于Swagger 2.0，可以创建API实体的Swagger定义，并将Swagger导入API网关用于批量创建或者更新API实体。API网关的预配置是swagger2.0，它可支持并兼容大部分的Swagger规范，但存在一定的差异性（Swagger兼容性说明）。

本章节主要对基于Swagger的API网关扩展进行介绍，并提供相应的示例来阐述用法。

> 说明 Swagger中所有参数与取值均大小写敏感

一、Swagger扩展：

Swagger扩展主要是对于swagger原生Operation Object进行扩展，增加认证、参数映射以及后端服务等扩展。除此之外，增加处理any方法的扩展，用于捕获任意http请求方法。所有扩展均以 `x-aliyun-apigateway-` 开头，具体扩展内容如下：

1.1 x-aliyun-apigateway-auth-type：授权类型

授权类型，应用于Operation Object。用于指定该API的授权类型，其中：

取值范围：

- APP（默认值）：阿里云API网关APP授权
- ANONYMOUS：匿名

示例：

```
...
paths:
  'path/':
    get:
      x-aliyun-apigateway-auth-type: ANONYMOUS
...
```

1.2 x-aliyun-apigateway-api-market-enable：云市场支持

授权类型，应用于Operation Object。用于指定该API是否可以上架云市场，其中：

取值范围：

- true
- false（默认）

示例：

```
...
paths:
  'path/':
    get:
      x-aliyun-apigateway-api-market-enable: true
...
```

1.3 x-aliyun-apigateway-api-force-nonce-check: 强制NONCE校验

授权类型，应用于Operation Object。用于指定该API是否进行NONCE强制校验，其中：

取值范围：

- true
- false（默认）

示例：

```
...
paths:
  'path/':
    get:
      x-aliyun-apigateway-api-force-nonce-check: true
...
```

1.4 x-aliyun-apigateway-parameter-handling: API映射关系

API映射关系，应用于Operation Object，用于指定请求参数与后端服务参数的映射关系。当映射关系选择PASSTHROUGH时，Parameter Object不支持 `x-aliyun-apigateway-backend-location` 和 `x-aliyun-apigateway-backend-name` 属性。

取值范围：

- PASSTHROUGH（默认值）：入参透传
- MAPPING: 入参映射

示例：

```
...
paths:
  'path/':
    get:
      x-aliyun-apigateway-parameter-handling: MAPPING
...
```

1.5 x-aliyun-apigateway-backend: 后端类型

后端类型，应用于Operation Object。用于设置后端服务信息。根据后端服务类型，决定具体的属性，详情如下：

1.5.1 后端服务类型：HTTP

HTTP的后端类型用于直接配置后端服务地址，一般用于后端地址可以直接访问的场合。

属性说明：

属性名称	类型	描述
type	string	必填；值为HTTP
address	string	必填；用于标识后端服务地址
path	string	选填：用于标识后端服务路径。支持路径变量。默认与跟path值相同
method	string	必填：后端请求方法
timeout	int	选填，默认为10000。该属性取值范围为[500,30000]

示例：

```
...
x-aliyun-apigateway-backend:
  type: HTTP
  address: 'http://www.aliyun.com'
  path: '/builtin/echo'
  method: get
  timeout: 10000
...
```

1.5.2 后端服务类型：HTTP-VPC

HTTP-VPC的后端类型用于后端服务在VPC网络内的场合，需要先创建VPC授权，具体操作请参考创建后端服务为VPC内资源的API，使用VPC授权名导入。

属性说明：

属性名称	类型	描述
type	string	必填；值为：HTTP-VPC
vpcAccessName	string	必填；后端服务使用的vpc实例名称
path	string	选填：用于标识后端服务路径。支持路径变量。默认与根path相等
method	string	必填：后端请求方法
timeout	int	选填，默认为10000。该属性取值范围为[500,30000]

示例：

```
...
x-aliyun-apigateway-backend:
  type: HTTP_VPC
  vpcAccessName: vpcAccess1
  path: '/users/{userId}'
  method: GET
  timeout: 10000
...
```

1.5.3 后端服务类型：FC

API网关后端服务类型为FUNCTION COMPUTE。

属性说明：

属性名称	类型	描述
type	string	必填；值为：FC
fcRegion	string	必填；函数计算所在区域
serviceName	string	必填：函数计算服务名
functionName	string	必填：函数计算函数名
arn	string	选填，函数计算RAM授权

示例：

```
...
x-aliyun-apigateway-backend:
  type: FC
  fcRegion: cn-shanghai
  serviceName: fcService
  functionName: fcFunction
  arn: acs:ram::111111111:role/aliyunapigatewayaccessingfcrole
...
```

1.5.4 后端服务类型： MOCK

MOCK的后端类型，用来模拟最初预定的返回结果。

属性说明：

属性名称	类型	描述
type	string	必填；值为：MOCK
mockResult	string	必填；mock返回结果
mockStatusCode	Integer	选填

属性名称	类型	描述
mockHeaders	Header	选填

Header 的说明

属性名称	类型	描述
name	string	必填
value	string	必填

示例:

```
...
x-aliyun-apigateway-backend:
        type: MOCK
        mockResult: mock resul sample
        mockStatusCode: 200
        mockHeaders:
          - name: server
            value: mock
          - name: proxy
            value: GW
...
```

1.6 x-aliyun-apigateway-constant-parameters: 常量参数

常量参数，应用于Operation Object，用于定义后端服务的常量参数。

属性说明:

属性名称	类型	描述
backendName	string	必填；后端参数名称
value	string	必填；常量值
location	String	必填；常量参数存放位置，可选值：[query,header]
description	string	可选；用于对该常量进行说明

示例:

```
    ...
    x-aliyun-apigateway-constant-parameters:
      - backendName: swaggerConstant
        value: swaggerConstant
        location: header
        description: description of swagger
    ...
```

1.7 x-aliyun-apigateway-system-parameters： 后端服务参数

后端服务参数，应用于Operation Object，用于定义API后端服务的系统参数。

属性说明：

属性名称	类型	描述
systemName	string	必填；系统参数名称
backendName	string	必填；后端参数名称
location	String	必填；常量参数存放位置，可选值：[query,header]

示例：

```
    ...
    x-aliyun-apigateway-system-parameters:
        - systemName: CaAppId
          backendName: appId
          location: header
...
```

1.8 x-aliyun-apigateway-backend-location: 后端参数位置

后端参数位置，应用于 Parameter Object。该属性仅在 `x-aliyun-apigateway-parameter-handling: MAPPING` 设置时生效，用于设置参数映射后，在后端服务请求时的参数位置。

取值范围：

- path
- header
- query
- formData

示例：

```
...
parameters:
  - name: swaggerHeader
    in: header
    required: false
    type: number
    format: double
    minimum: 0.1
    maximum: 0.5
    x-aliyun-apigateway-backend-location: query
    x-aliyun-apigateway-backend-name: backendQuery
...
```

1.9 x-aliyun-apigateway-backend-name: 后端参数名称

后端参数名称，应用于Parameter Object。该属性仅在 `x-aliyun-apigateway-parameter-handling: MAPPING` 设置时生效，用于设置参数映射后，在后端服务请求时的参数名称。

示例:

```
...
parameters:
  - name: swaggerHeader
    in: header
    required: false
    type: number
    format: double
    minimum: 0.1
    maximum: 0.5
    x-aliyun-apigateway-backend-location: query
    x-aliyun-apigateway-backend-name: backendQuery
...
```

1.10 x-aliyun-apigateway-query-schema: query对Model支持

后端参数名称，应用于Parameter Object。用于对Query进行模型定义。当parameter为String类型，同时定义为query参数时，可以使用。

示例:

```
...
parameters:
  - name: event_info
    in: query
    required: true
    type: string
    x-aliyun-apigateway-query-schema:
      $ref: "#/definitions/EvnetInfo"
...
```

1.11 x-aliyun-apigateway-any-method：ANY方法

ANY方法，应用于Path Item Object，用于设置API可以接受任意类型的http请求。

示例：

```
...
paths:
  'path/':
     x-aliyun-apigateway-any-method:
     ...
...
```

1.12 x-aliyun-apigateway-app-code-type：Appcode简单认证

Appcode简单认证，应用于Operation Object。用于指定该API是否支持APPcode简单认证，其中：

取值范围：

- DEFAULT（默认值）
- DISABLE: 禁用
- HEADER: 通过HEADER传入appcode
- HEADER_QUERY: 可以通过HEADER或Qeury传入appcode

示例：

```
...
paths:
  'path/':
    get:
      x-aliyun-apigateway-app-code-type: HEADER
...
```

二、兼容性说明

API网关与Swagger规范在定义API时存在一定差异性，包括：

2.1 Swagger参数类型与原有API网关类型的对照表

Swagger类型	API网关类型	支持的校验参数及规则
• type:integer • format:int32	Int	• mininum • maxnum
• type:integer • format:int64	Long	• mininum • maxnum
• type:number • format:float	Float	• mininum • maxnum

Swagger类型	API网关类型	支持的校验参数及规则
• type:number • format:double	Double	• mininum • maxnum
• type:string	String	• maxLength • enumValues • pattern
• type:boolean • format:Boolean	Boolean	-

2.2 consumes字段支持

当Swagger配置文件存在formdata类型参数时，需要配置consumes节点。API网关现在只支持 `application/x-www-form-urlencoded` 类型。

```
consumes:
  - application/x-www-form-urlencoded
```

三、Swagger示例

本章节提供三个基于API网关的swagger扩展示例。示例几乎涵盖了所有的扩展内容，为基于swagger扩展自定义API实体提供便利。

> **说明** 示例仅供参考。

3.1 API网关后端服务为HTTP的Swagger示例

```
swagger: '2.0'
basePath: /
info:
  version: '0.9'
  title: Aliyun Api Gateway Swagger Sample
schemes:
  - http
  - https
x-aliyun-apigateway-paramater-handling: MAPPING
x-aliyun-apigateway-api-market-enable: true
x-aliyun-apigateway-api-force-nonce-check: true
x-aliyun-apigateway-backend:
  type: HTTP
  address: 'http://www.aliyun.com'
  method: get
  timeout: 10000
paths:
  '/http/get/mapping/{userId}':
    get:
      operationId: case1
      schemes:
```

```
        - http
        - https
      x-aliyun-apigateway-parameter-handling: MAPPING
      x-aliyun-apigateway-api-market-enable: true
      x-aliyun-apigateway-auth-type: ANONYMOUS
      parameters:
        - name: userId
          in: path
          required: true
          type: string
        - name: swaggerQuery
          in: query
          required: false
          default: '123465'
          type: integer
          format: int32
          minimum: 0
          maximum: 100
        - name: swaggerHeader
          in: header
          required: false
          type: number
          format: double
          minimum: 0.1
          maximum: 0.5
          x-aliyun-apigateway-backend-location: query
          x-aliyun-apigateway-backend-name: backendQuery
      x-aliyun-apigateway-constant-parameters:
        - backendName: swaggerConstant
          value: swaggerConstant
          location: header
          description: description of swagger
      x-aliyun-apigateway-system-parameters:
        - systemName: CaAppId
          backendName: appId
          location: header
      responses:
        '200':
          description: 200 description
        '400':
          description: 400 description
  '/echo/test/post/{userId}':
    post:
      operationId: testpost
      schemes:
        - http
        - https
      x-aliyun-apigateway-parameter-handling: MAPPING
      x-aliyun-apigateway-backend:
        type: HTTP
        address: 'http://www.aliyun.com'
        method: post
        timeout: 10000
      consumes:
```

```
      - application/x-www-form-urlencoded
    parameters:
      - name: userId
        required: true
        in: path
        type: string
      - name: swaggerQuery1
        in: query
        required: false
        default: '123465'
        type: integer
        format: int32
        minimum: 0
        maximum: 100
        x-aliyun-apigateway-enum: 1,2,3
      - name: swaggerQuery2
        in: query
        required: false
        type: string
        x-aliyun-apigateway-backend-location: header
        x-aliyun-apigateway-backend-name: backendHeader
        x-aliyun-apigateway-query-schema:
          $ref: '#/definitions/AiGeneratePicQueryVO'
      - name: swaggerHeader
        in: header
        required: false
        type: number
        format: double
        minimum: 0.1
        maximum: 0.5
        x-aliyun-apigateway-backend-location: query
        x-aliyun-apigateway-backend-name: backendQuery
      - name: swaggerFormdata
        in: formData
        required: true
        type: string
    responses:
      '200':
        description: 200 description
        schema:
          $ref: '#/definitions/ResultOfGeneratePicturesVO'
      '400':
        description: 400 description
  x-aliyun-apigateway-any-method:
    operationId: case2
    schemes:
      - http
      - https
    x-aliyun-apigateway-parameter-handling: MAPPING
    x-aliyun-apigateway-backend:
      type: HTTP
      address: 'http://www.aliyun.com'
      path: '/builtin/echo/{abc}'
      method: post
```

```
        timeout: 10000
      parameters:
        - name: userId
          in: path
          required: false
          default: '123465'
          type: integer
          format: int32
          minimum: 0
          maximum: 100
          x-aliyun-apigateway-backend-name: abc
          x-aliyun-apigateway-backend-location: path
      responses:
        '200':
          description: 200 description
        '400':
          description: 400 description
definitions:
  AiGeneratePicQueryVO:
    type: object
    properties:
      transactionId:
        type: string
        description: 异步任务ID
  GeneratePictureVO:
    type: object
    properties:
      id:
        type: integer
        format: int64
        description: 图片ID
      name:
        type: string
        description: 图片名称
  GeneratePicturesVO:
    type: object
    properties:
      failSize:
        type: integer
        format: int64
        description: 失败数量
      list:
        type: array
        description: 图片列表
        items:
          $ref: '#/definitions/GeneratePictureVO'
          title: GeneratePictureVO
      successSize:
        type: integer
        format: int32
        description: 成功数量
      totalSize:
        type: number
        format: float
        description: 请求总数
```

```
        description: 请求总数
  ResultOfGeneratePicturesVO:
    type: object
    properties:
      model:
        description: 返回内容
        $ref: '#/definitions/GeneratePicturesVO'
        title: GeneratePicturesVO
      requestId:
        type: string
        description: 请求ID
```

3.2 API网关后端服务为HTTP-VPC的Swagger示例

```
swagger: '2.0'
basePath: /
info:
  version: '0.9'
  title: Aliyun Api Gateway Swagger Sample
schemes:
  - http
  - https
paths:
  '/http/get/mapping/{userId}':
    get:
      operationId: case1
      schemes:
        - http
        - https
      x-aliyun-apigateway-parameter-handling: MAPPING
      x-aliyun-apigateway-backend:
        type: HTTP-VPC
        vpcAccessName: vpcName1
        path: '/builtin/echo/{userId}'
        method: get
        timeout: 10000
      parameters:
        - name: userId
          in: path
          required: true
          type: string
        - name: swaggerQuery
          in: query
          required: false
          default: '123465'
          type: integer
          format: int32
          minimum: 0
          maximum: 100
        - name: swaggerHeader
          in: header
          required: false
          type: number
          format: double
```

```
          minimum: 0.1
          maximum: 0.5
          x-aliyun-apigateway-backend-location: query
          x-aliyun-apigateway-backend-name: backendQuery
      responses:
        '200':
          description: 200 description
        '400':
          description: 400 description
  '/echo/test/post':
    post:
      operationId: testpost
      schemes:
        - http
        - https
      x-aliyun-apigateway-parameter-handling: MAPPING
      x-aliyun-apigateway-backend:
        type: HTTP-VPC
        vpcAccessName: vpcName2
        path: '/builtin/echo'
        method: post
        timeout: 10000
      consumes:
        - application/x-www-form-urlencoded
      parameters:
        - name: swaggerQuery1
          in: query
          required: false
          default: '123465'
          type: integer
          format: int32
          minimum: 0
          maximum: 100
        - name: swaggerQuery2
          in: query
          required: false
          type: string
          x-aliyun-apigateway-backend-location: header
          x-aliyun-apigateway-backend-name: backendHeader
        - name: swaggerHeader
          in: header
          required: false
          type: number
          format: double
          minimum: 0.1
          maximum: 0.5
          x-aliyun-apigateway-backend-location: query
          x-aliyun-apigateway-backend-name: backendQuery
        - name: swaggerFormdata
          in: formData
          required: true
          type: string
      responses:
        '200':
```

```
        description: 200 description
      '400':
        description: 400 description
  x-aliyun-apigateway-any-method:
    operationId: case2
    schemes:
      - http
      - https
    x-aliyun-apigateway-parameter-handling: PASSTHROUGH
    x-aliyun-apigateway-backend:
      type: HTTP-VPC
      vpcAccessName: vpcName3
      path: '/builtin/echo'
      method: post
      timeout: 10000
    responses:
      '200':
        description: 200 description
      '400':
        description: 400 description
```

3.3 API网关后端服务为FunctionCompute的Swagger示例

```
swagger: '2.0'
basePath: /
info:
  version: '0.9'
  title: Aliyun Api Gateway Swagger Sample
schemes:
  - http
  - https
paths:
  '/http/get/mapping/{userId}':
    get:
      operationId: case1
      schemes:
        - http
        - https
      x-aliyun-apigateway-parameter-handling: MAPPING
      x-aliyun-apigateway-backend:
        type: FC
        fcRegion: cn-shanghai
        serviceName: fcService
        functionName: fcFunction
        arn: acs:ram::111111111:role/aliyunapigatewayaccessingfcrole
      parameters:
        - name: userId
          in: path
          required: true
          type: string
      responses:
        '200':
          description: 200 description
        '400':
          description: 400 description
```

3.4 API网关后端服务为MOCK的Swagger示例

```
swagger: '2.0'
basePath: /
info:
  version: '0.9'
  title: Aliyun Api Gateway Swagger Sample
schemes:
  - http
paths:
  '/mock/get/mapping/{userId}':
    get:
      operationId: case1
      schemes:
        - http
        - https
      x-aliyun-apigateway-parameter-handling: MAPPING
      x-aliyun-apigateway-backend:
        type: MOCK
        mockResult: mock resul sample
        mockStatusCode: 200
        mockHeaders:
          - name: server
            value: mock
          - name: proxy
            value: GW
      parameters:
        - name: userId
          in: path
          required: true
          type: string
      responses:
        '200':
          description: 200 description
        '400':
          description: 400 description
```

四、特别说明

以下内容请密切关注，直接影响swagger的导入功能使用

4.1 全局域的支持

以下标签支持全局作用域的定义。当该标签原本作用域不存在该定义时，会自动填充全局域的定义值；若存在，则优先使用原本作用域的定义值。

- x-aliyun-apigateway-backend
- x-aliyun-apigateway-api-market-enable
- x-aliyun-apigateway-api-force-nonce-check
- x-aliyun-apigateway-parameter-handling
- x-aliyun-apigateway-auth-type

4.2 Swagger Definition说明

swagger导入现在支持模型定义，但是与原有swagger规范存在差异 模型的定义主要用于SDK的生成，所以会在原有swagger规范的基础上增加了一些限制条件：

- swagger中schema标签仅支持：$ref类型
- swagger中Definition的model仅支持object type类型的模型定义
- swagger中Definition的model中存在array定义，其引用$ref应与title标签配合使用。array类型默认在SDK生成时对应生成ArrayList

5.使用VPC内资源作为API的后端服务

可作为API网关后端服务的VPC内资源主要包括ECS以及SLB，本文主要讲述如何创建一个高可用的后端服务。

概述

使用阿里云专有网络VPC，可以构建出一个隔离的网络环境，并可以自定义IP地址范围、网段、路由表和网关等。API网关也支持您部署在专属网络VPC中的服务开放API。若您的后端服务在VPC环境，需要进行授权API网关访问才可开放相应API。

1.授权与绑定VPC

开放VPC环境的API，需要您先授权API网关可访问您VPC内的服务。授权时需指定API网关可以访问的资源+端口，如：SLB 的443端口、ECS 的80端口。

- 授权成功后，API网关将通过内网访问VPC内部资源。
- 此授权只会被用作API网关访问相应后端资源。
- API网关不可访问未被授权的资源或者端口。例如：只将VPC 中SLB 的80端口授权给API网关，那么API网关只能访问VPC 中SLB 的80端口。

可用作API网关后端服务的VPC内资源包括SLB实例以及ECS实例：

- ECS实例：创建授权时支持绑定VPC网络的ECS实例，实例id及地址处可填写ECS实例的实例ID，也可以填写ECS实例的私网IP。
- SLB实例：目前API网关仅支持绑定内网SLB实例，实例ID及地址处可填写SLB实例的实例ID或是实例私网IP。

2.构建高可用架构

为了构建高可用架构，建议您可以选择内网SLB作为API网关的后端服务，负载均衡可以将访问流量根据转发策略分发到后端多个ECS实例，既能提高整个系统的性能，又能提高应用的高可用。

2.1 准备VPC环境

购买VPC环境的SLB及ECS，并搭建服务，本示例SLB监听的是ECS的80端口，ECS中部署的是简单的NGINX环境。

SLB的实例使用的是内网实例，实例详情请看下图。

2.2 创建VPC授权

进入【API网关控制台】-【开放API】-【VPC授权】，点击“创建授权”。进入授权页面，填写相应信息

VPC名称：为此条授权的名称标识，供创建API时选择后端地址使用，所以为了便于后续管理，请保证此名称的唯一。

2.3 创建API

创建API的流程与其他类型API方式一致，创建API分组以及定义API可参考创建 API

创建应用以及授权可参考创建后端服务为VPC内资源的API

2.4 测试API

可以到通过以下方式测试您的API

- 调试API
- 下载SDK
- 使用简单认证（AppCode）方式调用API

2.5 安全

API网关通过内网调用VPC内资源的后端服务，如您有更高的安全要求，或是您的内网SLB中已经配置了黑白名单，您需要在白名单中放行API网关的出口地址。SLB黑白名单设置请参考负载均衡访问控制

ECS实例若设置了安全组，则需要在安全组中放行API网关的出口地址，ECS添加安全组规则请参考添加安全组规则

API网关的出口地址获取详见创建后端服务为VPC内资源的API

常见问题

1、是否支持公网SLB?

不支持，如您的API网关想要使用内网调用SLB，那么只支持绑定内网SLB；如您需要通过公网调用SLB，您可以创建后端服务为HTTP（S）的API来调用。

2、是否可以绑定多个VPC?

可以，若您的后端服务在多个VPC，可以添加多个授权。

3、为什么我无法授权我的VPC?

请确认VPCID、实例ID和端口号的正确，并保证授权策略和VPC在同一个区域。

4、授权API网关后，我的VPC安全么?

- 只有授权后，API网关才可调用
- 授权后，只有您授权过的API网关可以调用
- 您还可以在后端ECS和SLB实例中设置访问控制

5、API网关是否支持跨地域的VPC?

支持，使用cen打通网络之后就行，关于CEN的配置方式，详情见 什么是云企业网

6.使用函数计算做为API后端服务

函数计算是一个事件驱动的服务。函数的执行可以由事件驱动，即当某个事件发生时，该事件触发函数的执行。现在，函数计算支持以 API 网关作为事件源。当请求设置函数计算为后端服务的 API 时，API 网关会触发相应的函数，函数计算会将执行结果返回给 API 网关。

1 功能简介

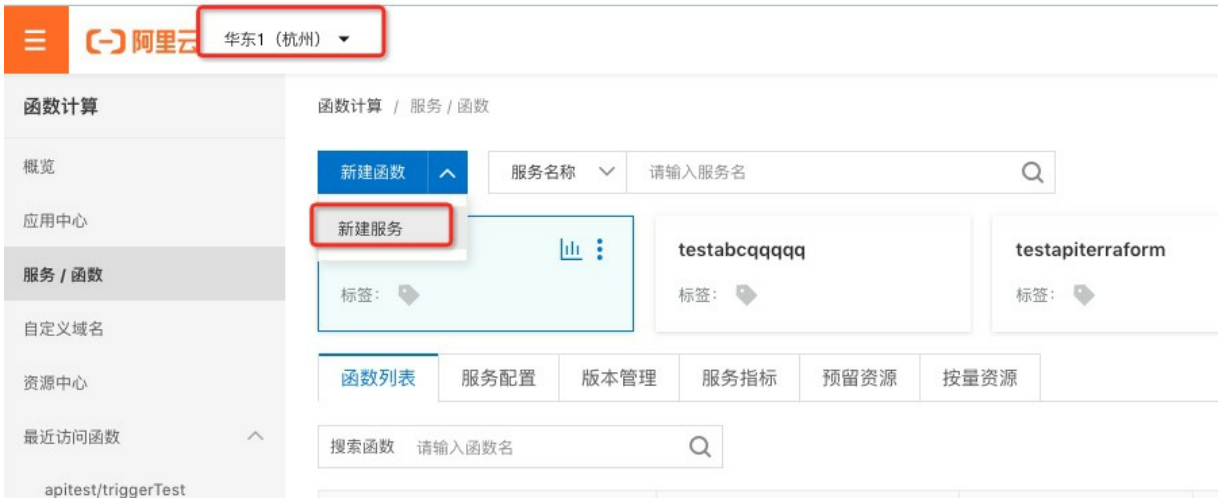
API 网关与函数计算对接，可以让您以 API 形式安全地对外开放您的函数，并且解决认证、流量控制、数据转换等问题（功能概览）。

API网关目前支持2种形式的函数，**HTTP函数**和**事件函数**，下面将分别介绍这两种函数的对接方法。

2 HTTP函数对接API网关

HTTP函数是函数计算创建的一个带HTTP触发器的函数。配置方法如下。

2.1. 创建服务。登录 函数计算控制台，选择您要创建服务和函数的**区域**，单击**新建服务**，并在弹出的对话框中完成服务创建。注意：服务创建成功后，无法更换Region，请 谨慎选择所属区域，建议和API网关选择同一个区域。



2.2. 创建函数。在服务页面上，单击 **新建函数** 进入函数创建流程：

a. 选择**HTTP函数**

注意配置触发器时，认证方式尽量选择“**function**”，表示调用时需要进行认证和签名调用，如果选择“anonymous”，则该函数支持匿名访问，不安全。

请求方式根据实际情况填写支持的Method，可以多选。

b. 接下来可以在 **代码执行** 页面进行代码编写和调试。

c. 在 **触发器** 页面查看触发器相关配置

2.3. 在API网关控制台创建分组和API

a. 单击控制台左侧导航栏中分组管理，选择分组列表的区域，再单击创建分组。（如果已创建分组，请忽略此步骤。）

> **说明**
>
> 如果函数计算与 API 不在同一地域，将通过公网访问您的函数计算服务。若您对数据安全和网络延迟有较高要求，请选择API 与函数计算为同一地域。

b. 创建和定义 API。在**API列表**页面，点击**创建API**，进行API定义配置，您可根据实际情况填写相关配置。

在后端基础定义页面中，后端类型选择：函数计算：

调用方式：选择“HTTP触发器”，

触发器路径：请拷贝函数计算控制台触发器页面中的路径（见上图2.2中c步骤的触发器页面的路径）；

如果是同region调用，可以将触发器路径上的域名更换为内网地址。可以在**函数计算控制台**的**概览->常用信息** 里面找到内网Endpoint。

后端请求path：可以自定义path，如果不需要自定义，请填写"/"

HTTP Method：请选择后端函数计算支持的method，如果多个请选择"any".

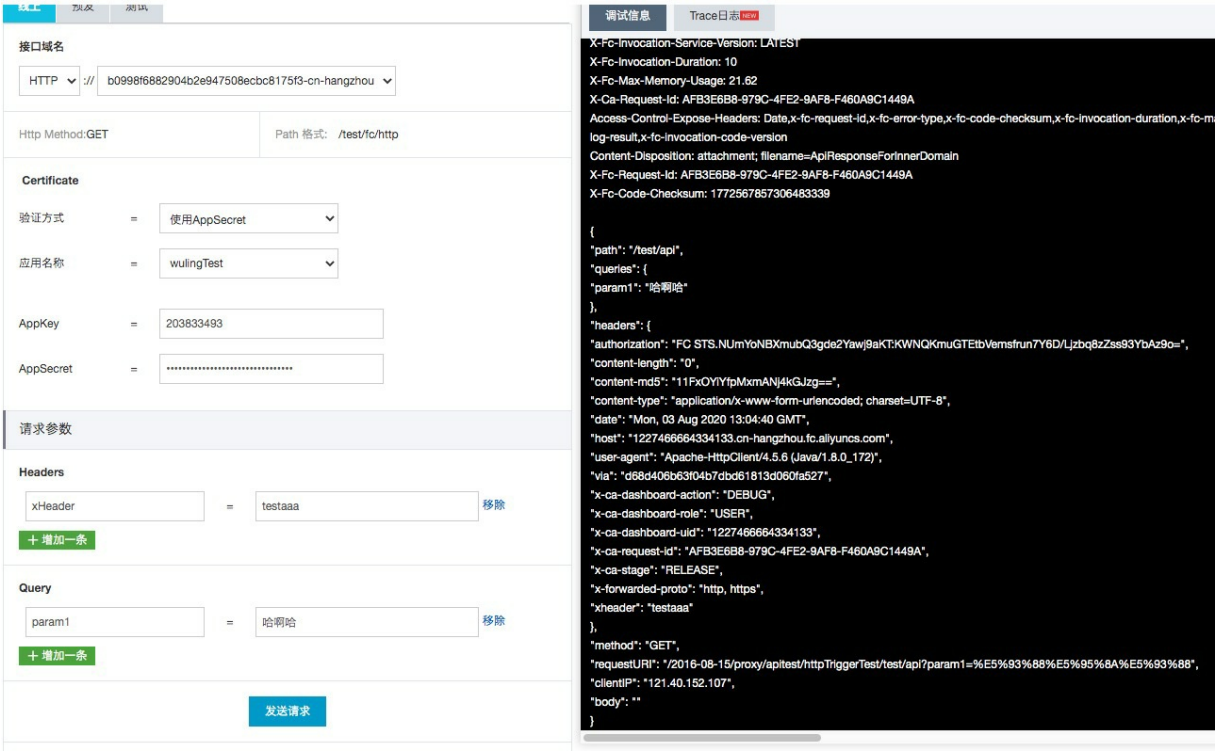
角色Arn: 您需要授权API网关访问您的函数计算服务。单击**获取授权**，自动获取角色 Arn。如果这是您第一次获取函数计算为 API 网关后端服务的角色授权，当您单击获取授权后，会弹出 RAM 控制台的授权页面。您需单击 RAM 控制台的授权权限，然后返回 API 创建页面再次单击获取授权，该角色Arn将自动显示在选项框中。如果是子账号操作，需要主账号给子账号授权RAM的查询和操作权限。

然后点击“下一步”，在页面上配置返回内容，这部分仅用于生成文档使用，不对API网关的返回产生影响。

2.4. 发布API，将API发布到测试或者线上环境，然后授权给某个APP（如果API的认证方式是“无认证”，则无需授权）。

2.5. 调试API。通过**API列表**，点击API名称，进入API详情，在左侧菜单中点击“调试API”，进入调试页面。

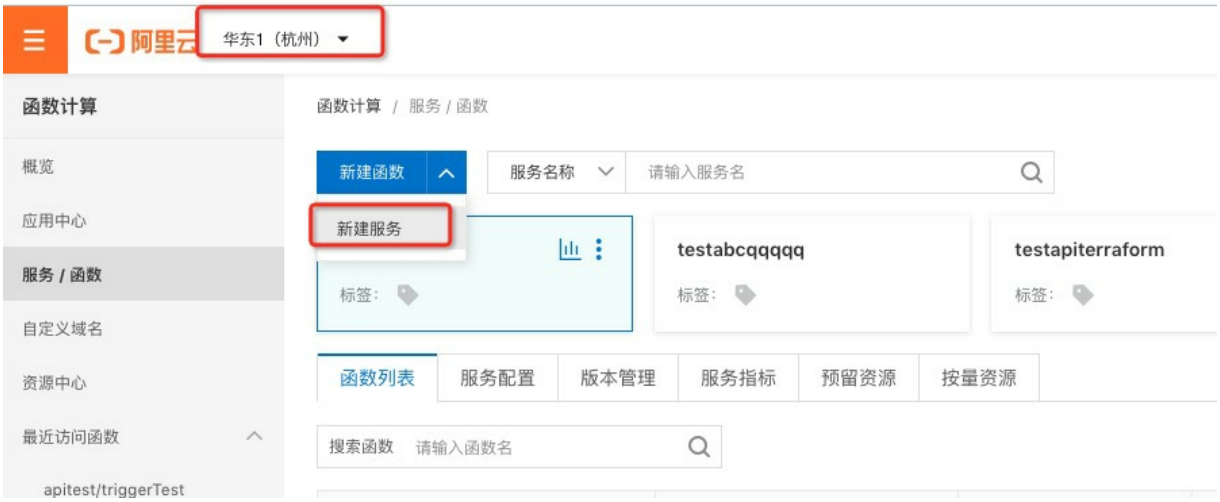
透传模式可以自行配置调用参数，进行API调试。



3 事件函数对接API网关

3.1 事件函数对接API网关创建服务

3.1.1选择您要创建服务和函数的 所属区域，单击 新建服务, 并在弹出对话框中完成服务创建。注意：服务创建成功后，无法更换区域，请谨慎选择所属区域。

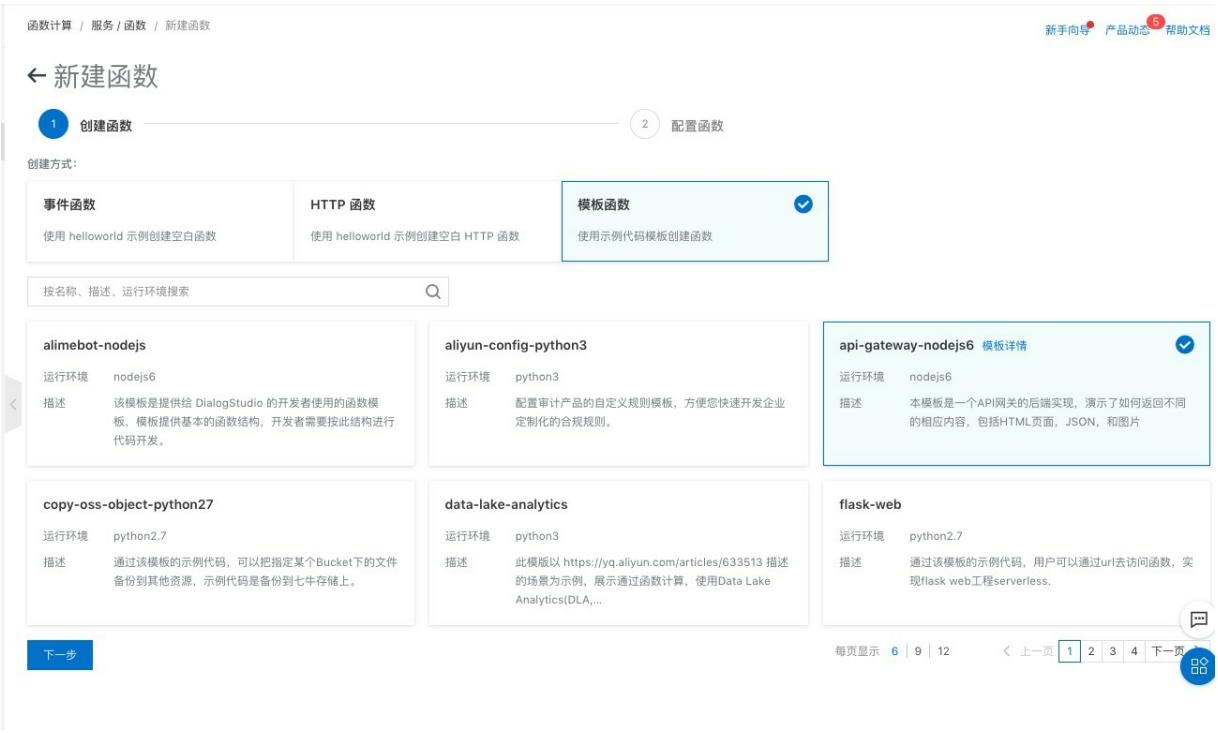


3.1.2 在已创建的服务中，创建函数。在该服务页面上，单击**新建函数**，进入函数创建流程：

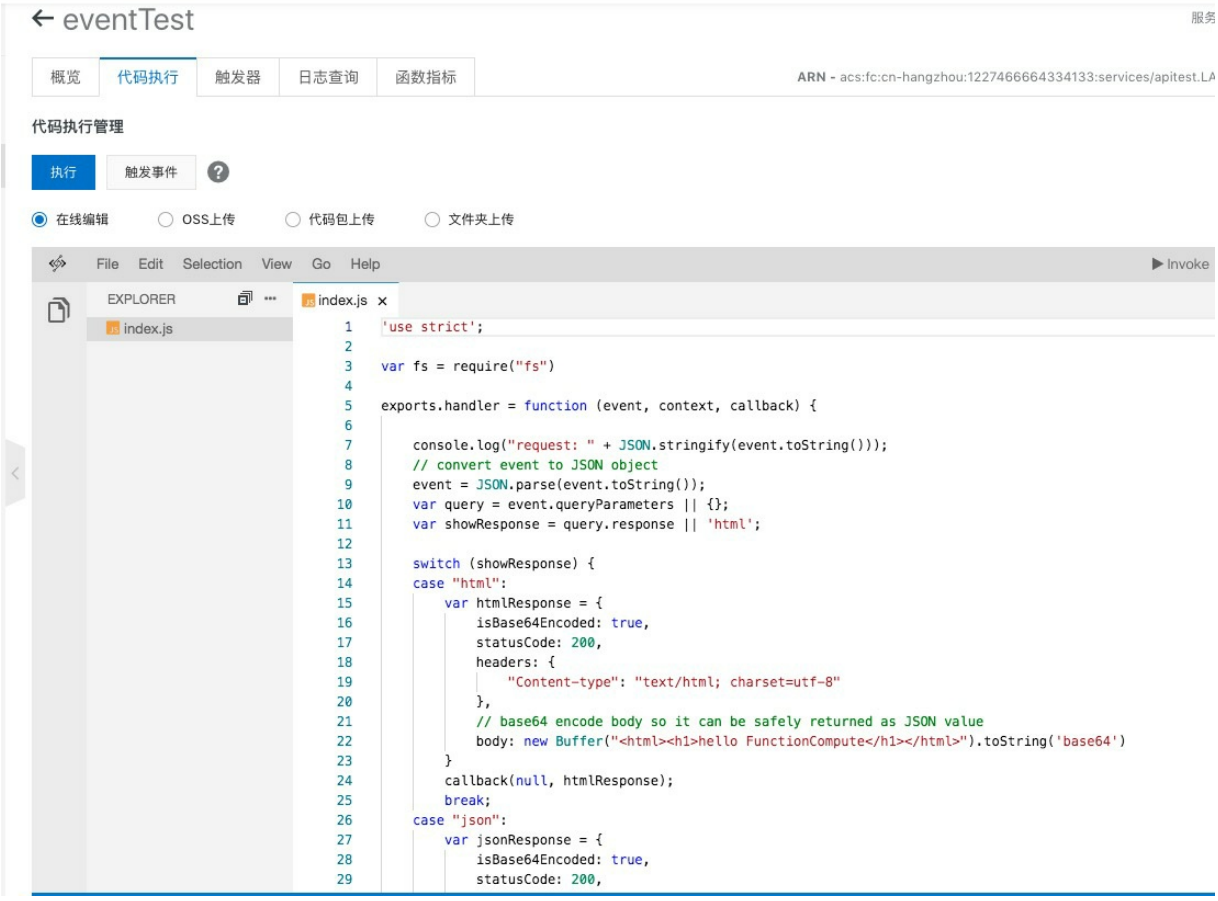
a. 选择函数模板。

函数计算控制台中，提供了 Node.js 6 环境的 API 网关后端实现模板 api-gateway-nodejs6 供您使用。

如果 api-gateway-nodejs6 模板不适用您的业务场景，请选择 **事件函数**。选择使用 **事件函数** 后，后续在**运行环境**中提交您自己编写的代码。请提前准备好代码包，以便上传。



b. 关于代码编写示例，可参见 配置API网关触发器 文档中，**编写函数代码** 部分。



3.1.3 在API网关控制台创建分组和API

其过程和“HTTP函数对接API网关的第2.3部分”一样。这里只详细说明函数计算后端配置部分。

调用方式：选择 **事件**

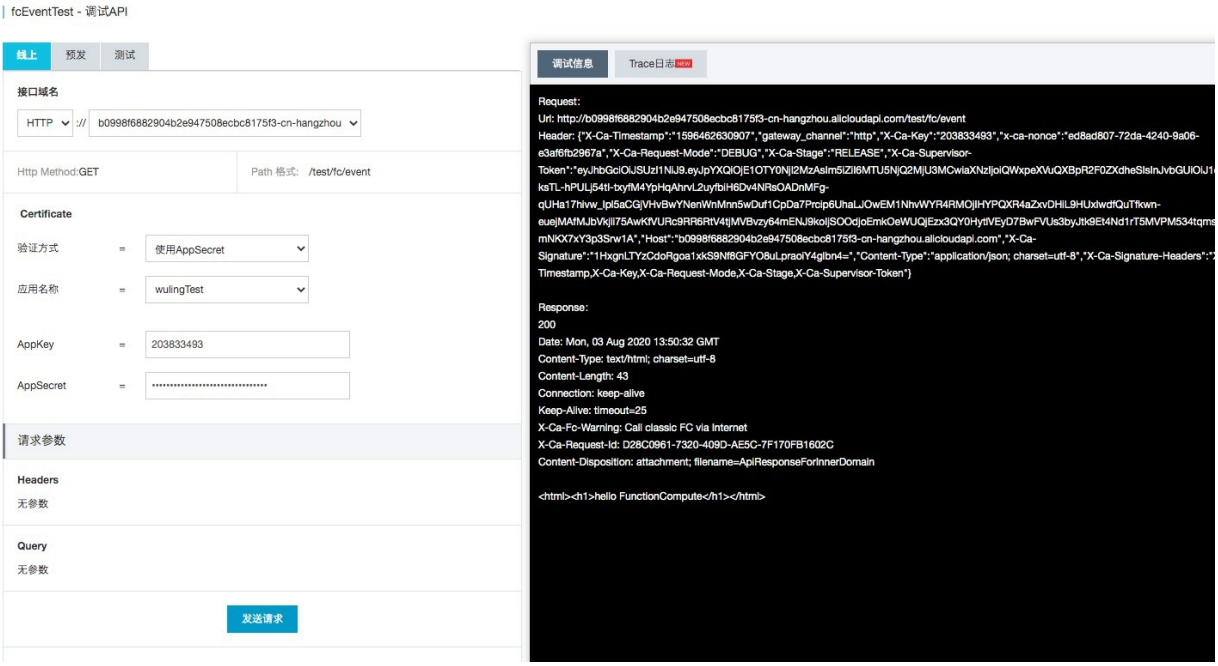
区域选择：选择函数计算服务所在region。如果函数计算与 API 不在同一地域，将通过公网访问您的函数计算服务。若您对数据安全和网络延迟有较高要求，请选择API 与函数计算为同一地域。

服务名称 和 **函数名称**：请根据实际的函数计算名称填写。当不同环境分别对应不同的函数计算时。可以通过环境管理方式配置。

角色Arn: 您需要授权API网关访问您的函数计算服务。单击获取授权，自动获取角色 Arn。如果这是您第一次获取函数计算为 API 网关 后端服务的角色授权，当您单击获取授权后，会弹出 RAM 控制台的授权页面。您需单击 RAM 控制台的授权权限，然后返回 API 创建页面再次单击获取授权，该角色Arn将自动显示在选项框中。如果是子账号操作，需要主账号给子账号授权RAM的查询和操作权限。

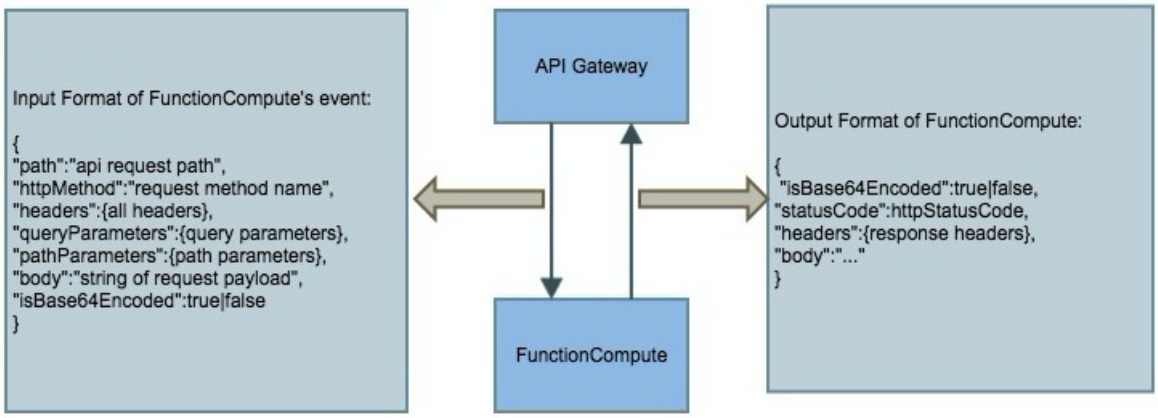
3.1.4. 发布API，将API发布到测试或者线上环境，然后授权给某个APP（如果API的认证方式是“无认证”，则无需授权）。

3.1.5. 调试API。通过API列表，点击API名称，进入API详情，在左侧菜单中点击“调试API”，进入调试页面。透传模式可以自行配置调用参数，进行API调试。



3.2.事件函数对接API网关的格式要求

API 网关调用函数计算的事件函数时，会将 API 的相关数据转换为 Map 形式传给函数计算服务。函数计算服务处理后，按照下图中 Output Format 的格式返回 statusCode、headers、body 等相关数据。API 网关再将函数计算返回的内容映射到 statusCode、header、body等位置返回给客户端。



API 网关向函数计算传入参数格式：

当以函数计算作为 API 网关的后端服务时，API 网关会把请求参数通过一个固定的 Map 结构传给函数计算的入参 event。函数计算通过如下结构去获取需要的参数，然后进行处理。

```
{
        "path":"api request path",
        "httpMethod":"request method name",
        "headers":{all headers,including system headers},
        "queryParameters":{query parameters},
        "pathParameters":{path parameters},
        "body":"string of request payload",
        "isBase64Encoded":"true|false, indicate if the body is Base64-encode"
}
```

- 如果 `"isBase64Encoded"` 的值为 `"true"`，表示 API 网关传给函数计算的 body 内容已进行 Base64 编码。函数计算需要先对 body 内容进行 Base64 解码后再处理。
- 如果 `"isBase64Encoded"` 的值为 `"false"`，表示 API 网关没有对 body 内容进行 Base64 编码。

函数计算的返回参数格式：

函数计算需要将输出内容通过如下 JSON 格式返回给 API 网关，以便 API 网关解析。

```
{
        "isBase64Encoded":true|false,
        "statusCode":httpStatusCode,
        "headers":{response headers},
        "body":"..."
}
```

- 当 body 内容为二进制编码时，需在函数计算中对 body 内容进行 Base64 编码，设置 `"isBase64Encoded"` 的值为 `"true"`。如果 body 内容无需 Base64 编码，`"isBase64Encoded"` 的值为 `"false"`。API 网关会对 `"isBase64Encoded"` 的值为 `"true"` 的 body 内容进行 Base64 解码后,再返回给客户端。
- 在 Node.js 环境中，函数计算根据不同的情况设置 callback。
  - 返回成功请求： callback(null,{"statusCode":200,"body":"..."})。

- 返回异常：callback(new Error('internal server error'),null)。
- 返回客户端错误： callback(null,{"statusCode":400,"body":"param error"})。

- 如果函数计算返回不符合格式要求的返回结果，API 网关将返回 503 Service Unavailable 给客户端。

3.3 事件函数调用示例

以下提供三个示例，分别为：事件函数代码示例、API 请求示例、和 API 网关返回示例。

3.3.1 事件函数代码示例

在函数计算的代码执行页面配置的代码示例。

```
module.exports.handler = function(event, context, callback) {
    var responseCode = 200;
    console.log("request: " + JSON.stringify(event.toString()));
    //将event转化为JSON对象
    event=JSON.parse(event.toString());
    var isBase64Encoded=false;
        //根据用户输入的statusCode返回，可用于测试不同statusCode的情况
    if (event.queryParameters !== null && event.queryParameters !== undefined) {
        if (event.queryParameters.httpStatus !== undefined && event.queryParameters.httpSta
tus !== null && event.queryParameters.httpStatus !== "") {
            console.log("Received http status: " + event.queryParameters.httpStatus);
            responseCode = event.queryParameters.httpStatus;
        }
    }
    //如果body是Base64编码的，FC中需要对body内容进行解码
    if(event.body!==null&&event.body!==undefined){
            if(event.isBase64Encoded!==null&&event.isBase64Encoded!==undefined&&event.isBas
e64Encoded){
                    event.body=new Buffer(event.body,'base64').toString();
            }
    }
    //input是API网关给FC的输入内容
    var responseBody = {
        message: "Hello World!",
        input: event
    };
        //对body内容进行Base64编码，可根据需要处理
    var base64EncodeStr=new Buffer(JSON.stringify(responseBody)).toString('base64');
        //FC给API网关返回的格式，须如下所示。isBase64Encoded根据body是否Base64编码情况设置
    var response = {
                isBase64Encoded:true,
                statusCode: responseCode,
                headers: {
                "x-custom-header" : "header value"
                },
                body: base64EncodeStr
    };
    console.log("response: " + JSON.stringify(response));
    callback(null, response);
};
```

3.3.2 事件函数请求示例

以 POST 形式请求 path 为如下的 API:

```
/fc/test/invoke/[type]
```

发起请求如下:

```
POST http://test.alicloudapi.com/fc/test/invoke/test?param1=aaa&param2=bbb
"X-Ca-Signature-Headers":"X-Ca-Timestamp,X-Ca-Version,X-Ca-Key,X-Ca-Stage",
"X-Ca-Signature":"TnoBldxxRHrFferGlzzkGcQsaezK+ZzySloKqCOsv2U=",
"X-Ca-Stage":"RELEASE",
"X-Ca-Timestamp":"1496652763510",
"Content-Type":"application/x-www-form-urlencoded; charset=utf-8",
"X-Ca-Version":"1",
"User-Agent":"Apache-HttpClient\/4.1.2 (java 1.6)",
"Host":"test.alicloudapi.com",
"X-Ca-Key":"testKey",
"Date":"Mon, 05 Jun 2017 08:52:43 GMT","Accept":"application/json",
"headerParam":"testHeader"
{"bodyParam":"testBody"}
```

3.3.3 API网关返回示例

```
200
Date: Mon, 05 Jun 2017 08:52:43 GMT
Content-Type: application/json; charset=UTF-8
Content-Length: 429
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,POST,PUT,DELETE,HEAD,OPTIONS , PATCH
Access-Control-Allow-Headers: X-Requested-With, X-Sequence,X-Ca-Key,X-Ca-Secret,X-Ca-Versio
n,X-Ca-Timestamp,X-Ca-Nonce,X-Ca-API-Key,X-Ca-Stage,X-Ca-Client-DeviceId,X-Ca-Client-AppId,
X-Ca-Signature,X-Ca-Signature-Headers,X-Forwarded-For,X-Ca-Date,X-Ca-Request-Mode,Authoriza
tion,Content-Type,Accept,Accept-Ranges,Cache-Control,Range,Content-MD5
Access-Control-Max-Age: 172800
X-Ca-Request-Id: 16E9D4B5-3A1C-445A-BEF1-4AD8E31434EC
x-custom-header: header value
{"message":"Hello World!","input":{"body":"{\"bodyParam\":\"testBody\"}","headers":{"X-Ca-A
pi-Gateway":"16E9D4B5-3A1C-445A-BEF1-4AD8E31434EC","headerParam":"testHeader","X-Forwarded-
For":"100.81.146.152","Content-Type":"application/x-www-form-urlencoded; charset=UTF-8"},"h
ttpMethod":"POST","isBase64Encoded":false,"path":"/fc/test/invoke/test","pathParameters":{"
type":"test"},"queryParameters":{"param1":"aaa","param2":"bbb"}}}
```

4 常见问题

4.1 为什么我无法录入我已有的函数?

请确认您输入的函数计算的服务名称和函数名称是否与您在函数计算控制台创建的服务和函数的名称一致。

4.2 使用函数计算作为API后端服务时，API网关到后端服务是否可以走内网?

当选择事件函数时，API网关和函数计算在同一region时，默认走内网。

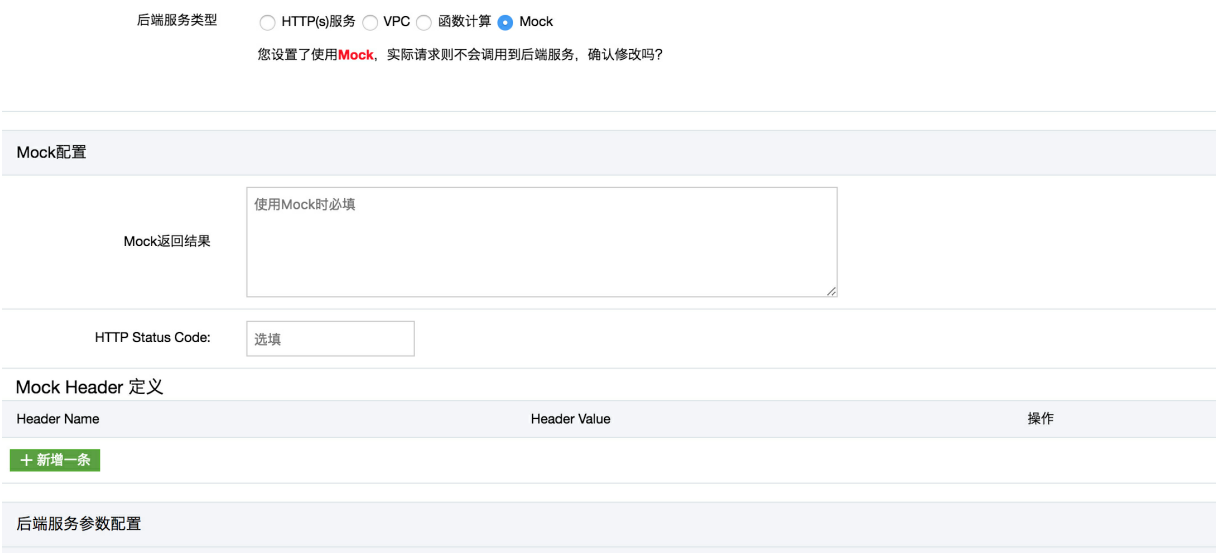
当选择HTTP函数时，需要您手工将触发器路径中的域名替换为内网Endpoint。内网Endpoint可以在**函数计算控制台**的**概览->常用信息**模块里面找到。

7.使用Mock方式做为API后端服务

在项目开发过程中，往往是多个合作方一同开发，多个合作方相互依赖，而这种依赖在项目过程中会造成相互制约，理解误差也会影响开发进度，甚至影响项目的工期。所以在开发过程中，一般都会使用 Mock 来模拟最初预定的返回结果，来降低理解偏差，从而提升开发效率。API 网关也支持 Mock 模式的简单配置。

配置 Mock

在 API 编辑页面--后端基础定义，来配置 Mock。



1. 填写 Mock 返回结果

Mock 返回结果，可以填写您真实的返回结果。目前支持是 Json、XMl、文本等格式作为 Mock 返回结果。如：

```
{
    "result": {
        "title": " API 网关 Mock 测试",
        ...
    }
}
```

保存后 Mock 设置成功，请根据实际需要 **发布** 到测试或线上环境进行测试，也可以在 API 调试页面进行调试。

2. 填写 Mock 请求响应statusCode

支持的statusCode取值如下表，兼容HTTP 1.1 Response Status Code的格式返回及其状态，如果您定义的statusCode不在下表中，将提示参数无效：

http code	http message
200	OK
201	Created

http code	http message
202	Accepted
203	Non-Authoritative Information
204	No Content
205	Reset Content
206	Partial Content
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
306	(Unused)
307	Temporary Redirect
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Timeout
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed

http code	http message
413	Request Entity Too Large
414	Request-URI Too Long
415	Unsupported Media Type
416	Requested Range Not Satisfiable
417	Expectation Failed
450	Parameter Requried
451	Method Connect Exception
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported

3. 定义Mock Header

API网关支持自定义Mock Header，允许Header同名，Header定义后取值不能为空，设置Mock时，Header Name不允许为空，并且只能由数字、字母、下划线、减号组成。Header Value不允许为空。

解除 Mock

若您需要解除 Mock，选择其他后端服务类型即可，而 Mock 返回结果中的值不会被清除，以便您进行下一次的 Mock。修改完成后请发布，只有发布后才会真正生效。

8.创建数据服务API

数据服务旨在为企业搭建统一的数据服务总线，帮助企业提升数据资产的价值，同时保证了数据的可靠性、安全性和有效性。使用场景包括：

- 最小粒度的数据输出：当企业需要对外提供数据时，相较于直接对外提供表数据，使用API的方式可以控制到行级（SQL过滤条件）和列级（查询字段），这样能帮助用户只对外暴露最小单元的数据，保障数据安全。
- 可视化制作：对于多数可视化制作工具，均支持API数据源。使用API数据源来制作可视化，而不是对外暴露数据库连接的用户名密码，可以避免账号泄漏，同时API的方式更简易。
- 加工后的数据供应用读取：通过数据工具对数据进行加工汇总后，业务应用希望读取这部分加工后的数据进行业务处理，则可通过API的方式快速输出。当需要变更读取的逻辑时，只需要调整API背后的数据逻辑，而不需要重新发布应用。

目前API网关可以与如下阿里云产品集成提供数据服务：

- 大数据产品：Dataworks，概述。
- 数据库产品：DMS，数据服务概览 。

9.创建在线预测服务API

阿里云机器学习平台PAI（Platform of Artificial Intelligence），为传统机器学习和深度学习提供了从数据处理、模型训练、服务部署到预测的一站式服务。其中在线预测服务部署是将算法模型应用至实际业务的重要环节。为了帮助用户更好的实现一站式端到端的算法应用，PAI平台针对在线推理场景提供了PAI EAS(Elastic Algorithm Service)在线预测服务。

PAI EAS可以将模型以REST API的形式发布到API网关，业务系统可以使用HTTP的方式加以调用，公网地址调用 。

10.参数映射与校验规则

> 说明
>
> 此文档仅适用于VPC实例（包含共享实例和专享实例），目前尚不适用于经典网络实例。

1. 概述

API网关支持参数映射与校验逻辑，本篇文档描述了API网关对转发客户端发起的HTTP请求到HTTP后端服务，以及转发后端服务的HTTP应答到客户端的处理规则，后端为阿里云函数计算的用户请参考使用函数计算做为API后端服务。

目前API网关支持以下几种传输处理方式：

- **透传模式**：`透传模式`下，API网关除`Path`位置的请求参数外，不对其他位置的请求参数进行映射与校验，用户参数会透明传递给后端，详细请阅读`章节2. 透传模式`。
- **映射模式**：`映射模式`下，API网关会根据用户配置的所有参数执行校验与映射，如果客户端传递了未在配置中的参数，参数将会被API网关过滤掉，不会转发给后端，详细请参考`章节3. 映射模式`。
- **透明映射模式**：类似`映射模式`，但在`透明映射模式`下，如果用户传递了未配置的参数，参数会被透明转发给后端，详细请参考`章节4. 透明映射模式`。

2. 参数位置与读取规则

API网关支持从`HTTP请求`的各种位置上读取参数，以及API网关提供的`系统参数`与用户可配置的`常量参数`。

2.1. `path`参数

API网关支持从`HTTP请求`的分段路径中提取参数的能力，使用`path`参数，需要首先将API的请求路径配置为`/path/[parameter]`的格式，API网关会使用参数路径方式去匹配`HTTP`请求中的路径，请参考如下的例子：

请求路径配置	输入	参数提取结果
/request/to/[path]	/request/to/user1	path=user1
/[path1]/[path2].	/group1/user1	path1=group1, path2=user1
/[root]/*	/root/user1	root=root

请求路径配置	输入	参数提取结果
/[root]	/root/user1	无法匹配

- API网关对于不符合RFC3986的非法输入，直接返回 `错误码:I400PH: Invalid Request Path` - API网关支持的最大RequestUri长度为128KBytes, 超过这个限制时会返回 `错误码:I413RL: Request Url too Large`

2.2. `query` 参数

`query` 参数为请求在 `QueryString` 中携带的参数，API网关会对请求 `QueryString` 通过 `=` 与 `&` 分割符进行键值对的拆分，并使用 `UTF-8` 编码做Url Decode，对于 `?a=` 和 `?a` 均被认为是值等于空字符串 `''` 的合法值，如：

QueryString输入	参数提取结果
?a=1&b=2	a: "1", b: "2"
?a=1&a=2	a: ["1", "2"]；参数为非 `array` 类型时仅使用第一个值
?a	a: ""
?a=	a: ""
?=a&b=1	b: "1"； `=a` 这个输入会被API网关忽略

- *API网关对于不符合RFC3986的非法输入，直接返回 `错误码:I400PH: Invalid Request Path`
- *API网关支持的最大RequestUri长度为128KBytes, 超过这个限制时会返回 `错误码:I413RL: Request Url too Large`

2.3. `formData` 参数

`formData` 参数为当请求的 `Content-Type` 为 `application/x-www-form-urlencoded` 时，消息体中携带的值。如果 `Content-Type` 没有指定 `charset=` 则网关会使用 `UTF-8` 编码，否则使用 `charset` 中指定的字符集来进行Url Decode，对于Form的拆分与处理逻辑与 `QueryString` 中描述的处理方式一致。

当 `Content-Type` 为 `multipart/formdata` 时，网关支持进行文件类型参数。

2.4. `header` 参数

`header` 参数会从HTTP请求的头中读取，比如 `X-User: aaa` 会被解析为 `X-User` = `aaa` ，特殊规则如下

- Header值中包含的前后空格会被截取掉
- 如果存在多个重名Header且参数类型被配置为 `ARRAY` 则参数会被解析为数组，否则只取第一个值
- API网关使用 `ISO-8859-1` 编码读取和转发Header，非法字符会导致乱码或其他非预期的结果

2.5. `host` 参数

`host` 参数仅在用户绑定了泛域名和有效的泛域名模板时才有效，例如：绑定泛域名 `*.api.foo.com` 且配置了泛域名模板 `${user}.api.foo.com` ，则当收到请求 `1234.api.foo.com` 时，会读取到 `host` 参数 `user` = `1234` ，用户可以配置多个泛域名模板，API网关会使用第一个匹配到的记录来解析 `host` 参数，如果没有记录，则不会读取到 `host` 参数，具体例子如下

泛域名模板配置	请求Host	参数提取结果
`${User}.api.io`	123.api.io	User: "123"
`${User}.${Group}.api.io`	123.g01.api.io	User: "123" Group: "g01"
`${Admin}.admin.api.io` `${User}.${Group}.api.io`	123.api.io	User: "123"
`${Admin}.admin.api.io` `${User}.${Group}.api.io`	123.admin.api.io	Admin: "123"
`${Admin}.admin.api.io` `${User}.${Group}.api.io`	123.u00.api.io	User: "123"GroupId: "u00"

泛域名模板配置	请求Host	参数提取结果
`${User}.${Group}.api.io` `${Admin}.admin.api.io`	123.admin.api.io	User: "123"Group: "admin"

在最后一个例子中，因为第一条记录就已经命中匹配了，所以忽略后面的所有记录。

3. 参数透传 模式

透传模式支持以下方法： `GET` 、 `PUT` 、 `POST` 、 `DELETE` 、 `PATCH` 、 `HEAD` 、 `OPTIONS`

3.1. 转发 客户端请求 到 后端服务

在透传模式下，API网关在处理签名和授权后，会将请求透明传输给后端，对于不同位置的透传规则如下

- **Path**: 当用户将API的请求路径配置为 `/path/to/[user]` 的格式时，可以同时配置 `/path/backend/[user]` 到后端服务路径上，API网关会识别前端路径参数并将其映射到后端服务路径上
- **QueryString**: 透明传输给后端，保持原始接收到的QueryString的顺序与格式
- **Header**: 除一些系统头和以 `X-Ca-` 开头的 `Header` 以外，网关会透传其余的Header给后端，API网关会使用 `ISO-8859-1` 编码读取和转发 `Header` ，所以如果你在Header中传递了非法的编码，可能会收到非预期的结果，对于系统以及API网关保留 `Header` 的处理逻辑，请参考“章节5. Http头处理规则”
- **Body**: 包体会透明转发给后端，如果用户在API配置中设置了自定义 `Content-Type` ，则使用设置的 `Content-Type` ，否则转发客户端提供的 `Content-Type` 头

3.2. 转发 后端应答 给 客户端

在透传模式下，如果后端成功返回应答，API网关会将来自 后端服务 的 `HTTP应答` 转发给 客户端 ，如果在处理过程中失败了，由API网关生成错误码，错误处理请参考错误代码表，透传规则如下

- **StatusCode**: 透传来自后端应答的错误码
- **Header**: API网关会过滤或添加一些 `系统Header` 和名字以 `X-Ca-` 起始的 保留Header ，透传来自后端应答的其他Header，详细参考 章节7. Http Header处理规则
- **Body**: API网关会将来自后端服务应答的包体转发给客户端，后端应答的 `Content-Type` 为空，则补充一个默认值： `application/oct-stream`

可以通过使用错误码映射插件插件，来改写客户端的应答码。

4. 参数映射 模式

参数映射 下，API网关会根据用户配置的所有参数执行校验与映射，如果客户端传递了未在配置中的参数，参数将会被API网关过滤掉，不会转发给后端，如果您希望网关转发未配置参数给后端，请参考 章节5．透传映射模式 。在参数映射模式下，API网关可以支持 query ， header ， host ， path ， formData 位置下的参数，并进行参数值的类型判断、校验，并进行向后端的映射。

4.1. 参数类型

API网关目前支持以下参数类型

类型	类型说明	格式支持	可选校验方式
String	字符串	不限	最小长度、最大长度、枚举值、正则
Integer	32位整数	1 ， -1 ， 100	最小值，最大值，枚举值
Long	64位整数	-1233 ， 1001	最小值，最大值，枚举值
Double	浮点数	100 ， 0.1 ， 9E-9 ， 1.01E16	最小值，最大值
Boolean	布尔值	true ， false ；（忽略大小写）	
File	文件类型	仅用于 multipart/form-data	最小长度、最大长度
Array	数组类型	参考数组字段类型	数组字段类型上的校验

1. Float类型与Double类型在处理标准与过程一致，不再单独列出。

4.2. 参数校验配置

- 参数校验可通过 控制台 、 OpenAPI 或 导入Swagger 的方式设置，设置方式与含义如下

名称	说明	OpenAPI的字段	Swagger中的字段
参数名	必选，API内唯一	ApiParameterName	name
参数位置	必选	Location	location
参数类型	可选，默认类型为 `String`	ParameterType	type
数组参数类型	可选，参数类型为 `Array` 时，指定数组字段类型	ArrayItemsType	items.type
是否必填	可选，默认为 `否`	Required	required
默认值	可选，空字符串 `''` 不是有效的默认值.	DefaultValue	default
最大值	可选，输入值必须 `小于等于` 最大值	MaxValue	maximum
最小值	可选，输入值必须 `大于等于` 最小值	MinValue	minimum
最大长度	可选，仅对 `String` 类型有效	MaxLength	maxLength
最小长度	可选，仅对 `String` 类型有效	MinLength	minLength
正则表达式	可选，仅对 `String` 类型有效	RegularExpression	pattern
枚举值	可选	EnumValue	enum

- *OpenAPI的参数配置参考：创建API->Request Parameter*
- *Swagger参数配置参考：通过导入Swagger创建API*

参数校验的一些匹配规则：

- OpenAPi与Swagger对参数类型的取值定义不同，本节的描述参考 `Swagger` 标准
- 如果不设置参数类型，默认的类型为 `String`
- 如果参数类型的输入格式与当前类型的支持格式不符，会报错误码： `I400IP Invalid Parameter :` ...``
- 如果参数设置为了必选，如果客户端的请求中不传递，网关会阻拦这个请求，并报错误码： `I400MP Invalid Parameter Requried:` ...``
- 可选参数可以配置缺省值，当客户端不传递这个参数时，使用配置的缺省值传递给后端，但API网关不认为 `空字符串：''` 是合法的缺省值，网关不会将配置为空字符串的缺省值传给后端。
- 当参数处于 `query` ， `formData` 位置时，对于形如 `a` 或 `a=` 格式的输入，如 `?b=1&a` ，API网关认为输入参数为空字符串 `''` ，``
  - 此时如果参数设置必选，不报错
  - 如果参数设置为可选且配置了缺省值，网关将不使用缺省值，将空串传递给后端
- 当参数类型为 `Integer` ， `Long` ， `Float` ， `Double` 值时，如果输入为空字符串 `''` ，则认为此参数没有传递，
  - 当此参数为必选时，网关会阻拦这个请求，并报错误码 `400: <I400MP> Invalid Parameter Requried: ...` 。
  - 当此参数为可选且存在缺省值配置时，使用缺省值发送给后端
- 最小长度与最大长度的判断依据为 `大于等于` 最小长度和 `小于等于` 最大长度，可以单独设置，或同时设置，只有大于 `0` 的配置才会生效
- 正则表达式的最大允许长度为40个字符，
- 字符串类型和数字类型均可以使用 `枚举` 设置，使用逗号分隔：比如 `江,河,湖,海` ，如果输入值不在列表中，会报错误码： `I400IP: Invalid Parameter :` ...``
- 如果参数类型设置为 `ARRAY` 数组类型，只有 `query` ， `formData` ， `header` 三个位置的参数支持数组格式，数组参数的校验规则对可以对每个数组元素生效，类型由数组参数类型字段指定，默认为字符串

4.3. 后端参数映射规则

- 用户可以设置参数的 `后端位置` 与 `后端名称` ，API网关会在映发送请求给后端服务时，进行参数位置与名称的转换

- API网关的参数类型仅用于校验，不会变更传递到后端的形式，如Double类型的参数如果输入为 `a=1` 不会被更改为 `a=1.0` ，
- 参数类型为 `ARRAY` 时，后端位置只能为 `query` ， `formData` ， `header` ，发送给后端时使用会多个参数或多个Header的方式，如 `a=1&a=2` ，不使用 `a=1,2` 的方式
- 传递给后端的QueryString会使用 `UTF-8编码的URL Encode` 进行重新组装
- 如果参数中包含了Form参数时，会使用 `application/x-www-form-urlencoded; charset=utf-8` 或 `multipart/formdata; charset=utf-8` 作为包体格式发送给后端
  - 如果参数中包含 `File` 类型，会使用 `multipart/formdata; charset=utf-8` 进行组装，否则使用 `application/x-www-form-urlencoded; charset=utf-8` 进行组装
  - 如果用户在API定义的后端服务部分，设置了自定义 `Content-Type` ，则会以用户的Content-Type发给后端，如果用户自定义的 `Content-Type` 形势属于 `application/x-www-form-urlencoded; charset=???` 或 `multipart/formdata; charset=???` 形式，则使用自定义中描述的Encoding进行组装，对于其他的Content-Type，不进行编码的特殊处理
- 当转发的参数处于 `header` 位置时，网关会使用 `ISO8859-1` 编码进行转换和发送

4.4. 转发 `后端应答` 给 `客户端`

在映射模式下，如果后端成功返回应答，API网关会将来自 `后端服务` 的 `HTTP应答` 转发给 `客户端` ，如果在处理过程中失败了，由API网关生成错误码，错误处理请参考API网关异常处理，透传规则如下

- **StatusCode**: 透传来自后端应答的错误码
- **Header**: API网关会过滤或添加一些 `系统Header` 和名字以 `X-Ca-` 起始的 `保留Header` ，透传来自后端应答的其他Header，详细参考章节**7. Http Header处理规则**
- **Body**: API网关会将来自后端服务应答的包体转发给客户端，后端应答的 `Content-Type` 为空，则补充一个默认值： `application/oct-stream`

可以通过使用错误码映射插件，来改写客户端的应答码。

5. `透传映射` 模式

`透传映射模式` 与 `参数映射` 模式的校验与处理机制一致，区别在于 `透传映射` 模式下，请求中的未知参数会在原位置上透传给后端，而 `参数映射` 模式下，未知参数会被过滤掉。

6. `严格映射` 模式

`严格映射` 与 `参数映射` 模式的校验与处理机制一致，区别在于 `严格映射` 模式下，当请求中包含未知参数时会直接按报错处理。

7. Http Header处理规则

一般来讲，所有以 `X-Ca-` 开头的Header均为API网关保留Header, API网关会对 `X-Ca-` 的Header做特殊处理，请不要在您的业务中使用 `X-Ca-` 开头的头，否则会导致头被过滤，或产生未预期的行为。

HeaderName	请求处理方式	应答处理方式
Connection	重建	重建
Keep-Alive	重建	重建
Proxy-Authenticate	重建	重建
Proxy-Authorization	重建	重建
Trailer	重建	重建
TE	重建	重建
Transfer-Encoding	重建	重建
Upgrade	重建	重建
Host	重建	
Authorization	校验、映射或透传	
Date		透传或添加默认
Content-Type	映射或透传	透传或添加默认
Content-Length	映射或透传	
Content-MD5	校验并透传	

HeaderName	请求处理方式	应答处理方式
Via	添加网关记录	
X-Forwarded-For	在右侧添加客户端IP	
X-Forwarded-Proto	添加客户端请求协议：'http',https','ws','wss'	
User-Agent	透传或添加网关UserAgent	
Server		透传或添加默认

- 所有标记为重建的Header不会透传，网关会重新添加为网关设置的值。
- 未在表中出现的Header如果客户端请求为透传模式，则将请求头透传给后端，如果为映射模式，则除默认的HttpHeader外，其余的Header会被过滤。
- 未在表重出现的应答的Header默认均透传给客户端。

11.版本管理

1 编辑API

您可以查看API定义并根据需要进行编辑。

操作步骤：

1.登录API网关控制台

2.单击左侧导航栏的 开放API —— API列表 。

3.选择需要查看的API点击操作列下的 管理 。您可以看到当前所选API的定义信息。按照您的需要进行编辑，单击 编辑 按钮。

编辑API定义的流程和创建API定义一致，详细信息参见创建 API ，如果您想放弃编辑，可以点击页面顶部的返回图标。

API编辑完成后，需要发布API到测试，预发或者线上，API才能进行调用新的版本。

> **说明**
>
> 当您需要编辑某个API的定义时，如果该API已经发布，对定义的修改不会对线上产生影响，定义修改后需要再次发布才能把修改后的定义同步到线上环境。

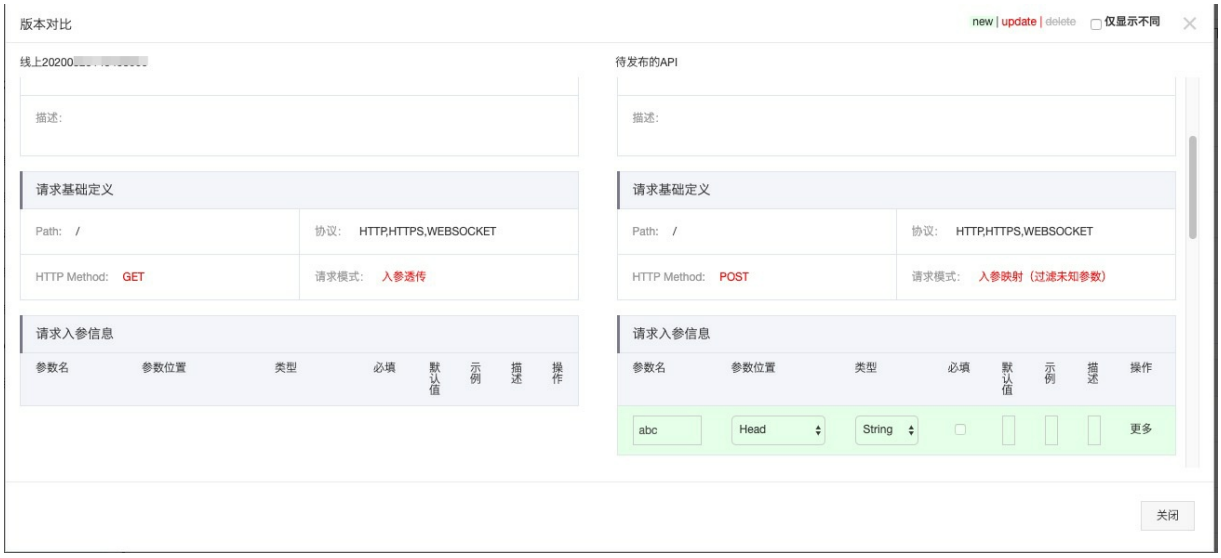
2 对比差异

API编辑后您可以通过**对比差异**功能来与之前发布的API进行配置对比，确认修改范围是否正确。

操作步骤：

1. API编辑完成后点击 保存 按钮，就会跳至API修改成功的页面，点击 发布 按钮。

2.在发布API页面，选择要发布的环境，之后可以点击 对比差异 ，与之前发布的API进行配置对比，确认修改范围后再发布。



3 查询历史发布及版本切换

API在环境中的每次发布，API 网关都会进行发布历史记录，发布历史包含了版本号，发布环境，发布时间和您填写的发布说明，您可以在发布历史中查看任何一次的发布记录。

3.1 查询历史发布

1.登录API网关控制台，在左侧导航栏点击 开放API —— API列表 ——找到目标API点击进入。

2.在左侧栏点击发布历史，就可以看到API的发布历史版本。单击右侧的查看，就可以看到该版本API定义的详情。

3.2 切换发布版本

查看发布历史时，您可以选定某个版本然后操作切换到此版本，该操作会使该版本直接在指定环境中替换之前的版本，实时生效。

1. 选择需要的版本并单击右侧的**切换至此版本**。
2. 在弹出的确认对话框中，填写备注并单击**切换**。

12.环境管理

1. 什么是环境管理

当前每个API分组支持设置三个环境：测试(TEST)、预发(PRE)和线上(RELEASE)。目的是能够满足不同您的不同研发场景下的API 调用需求，如API测试环境，后端服务对应到您的测试环境资源，从而可以保证在同一套API配置的情况下，供您的测试人员进行测试使用。

在API网关上进行环境管理的时候，您需要做两部分工作：

- API的后端配置：通过设置 `API分组` 的 **环境变量** ，为API分组的测试、预发、线上环境分别定义不同值，从而当调用API时，API网关可以调用到不同的后端地址。
- API的前端调用：需要client端显式的说明需要调用哪个环境。目前API网关支持两种方式，一种是为API分组的不同环境绑定不同的域名，另一种方式是在 Header 中增加入参 `X-Ca-Stage` 的信息。

本文将会分别介绍在API网关三种不同的后端服务类型(**VPC** 、 **http** 、 **函数计算**)情况下，如何结合两种不同的前端调用方式（ **X-Ca-Stage** 、 **域名** ），从而实现环境管理。受篇幅限制，本文并未将后端服务类型和前端调用方式的所有组合罗列出来进行讲解，您了解原理之后，可以根据情况自由组合。

2. VPC后端+X-Ca-Stage

2.1 前期准备

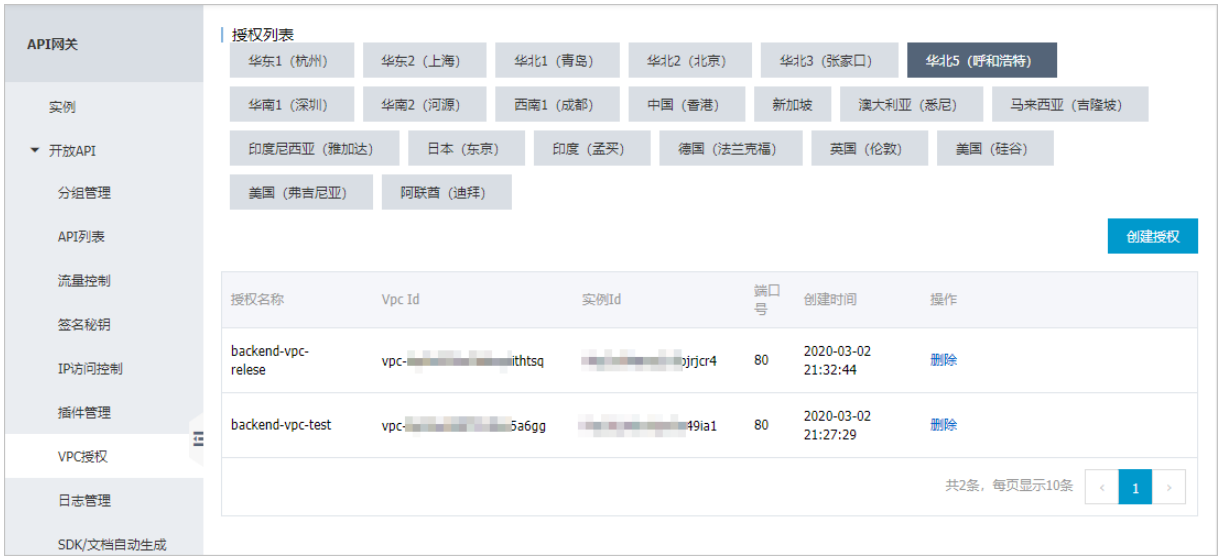
在本例中，首先在ECS中创建了两个不同的VPC，每个VPC中各创建了一个ECS实例，各代表线上环境和测试环境的后端服务器，如下图所示：

每个ECS实例都开放了TCP 80端口，并且安装了nginx做为http的web server，使用http访问时，分别会返回{"env":"test env"}和{"env":"relese env"}。

> **说明**
>
> 本示例仅用于介绍如何使用API网关进行环境管理，因此并未考虑高可用、可扩展性、安全性等重要因素，故本示例的后端服务架构请勿做为您正式使用时的架构参考。

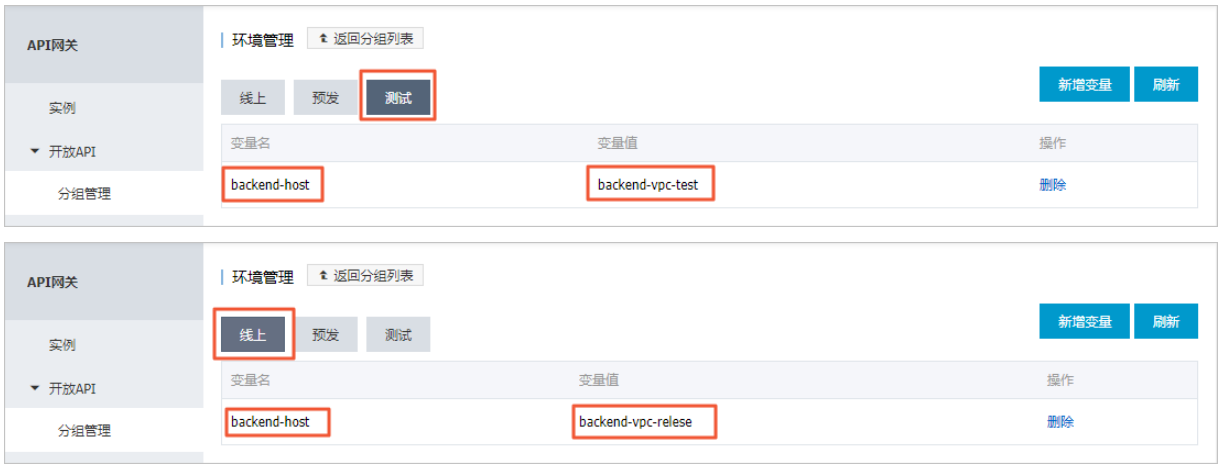
2.2 API后端配置

步骤1：由于本例中的API后端服务类型是VPC，因此需要分别创建两个VPC授权 backend-vpc-release 和 backend-vpc-test ，分别对应 **线上环境** 和 **测试环境** ，如何使用VPC做为API的后端服务的配置过程可详见创建后端服务为VPC内资源的API 。如下图所示：



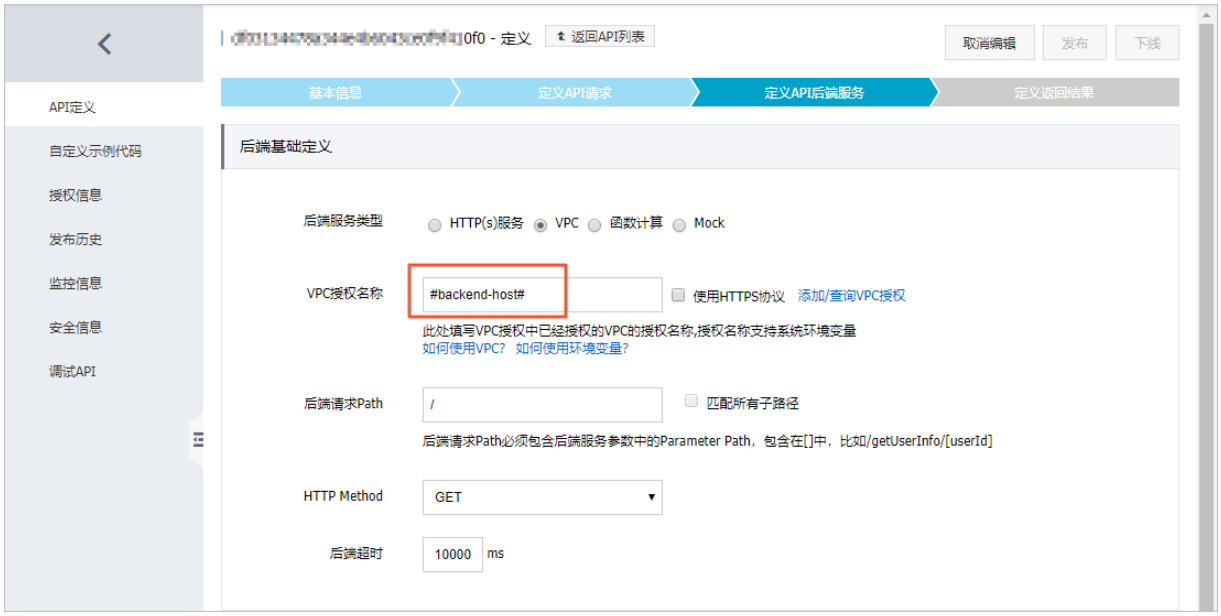
步骤2：在API网关控制台中创建API分组，进入 **环境管理** 。

步骤3：在 **环境管理** 中，在线上环境和测试环境中分别创建一个同名的变量（本例中为 backend-host ），但值的内容不同，分别对应 **步骤1** 中的两个VPC授权名称，如下图所示：



> **说明**
>
> 环境变量的变量名，需要保持三个环境中的对应的变量名称相同。如果您有多个 API，建议变量名标识有实际意义，以便后续查询。

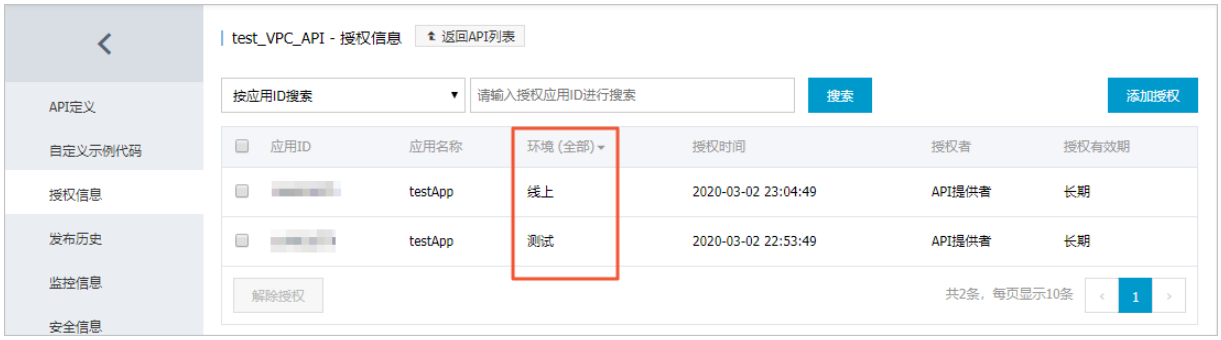
步骤4：在此分组下创建API，略过API的其他定义，重点在 定义API后端服务 的页面中，在 VPC授权名称 的位置填写变量名称，填写 #backend-host# 。



> **说明**
>
> API网关中，环境变量的表示方法为 #变量名#。如，#Service#、#Function#。

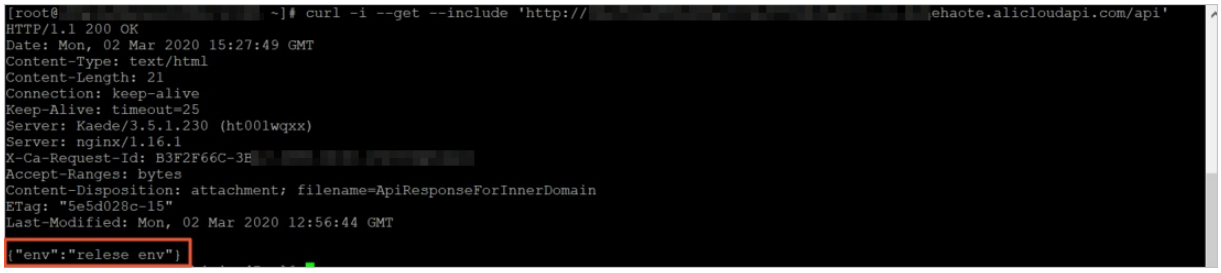
步骤5：完成其他API配置，注意需要发布到线上环境和测试环境中。如下图所示：

步骤6：添加授权信息。需要注意授权的添加过程是和环境相关的，如果使用统一授权访问不同环境，需要给每个环境都进行授权，如下图所示。本例后续的调用环节，为了演示方便，API可匿名访问。



2.3 使用X-Ca-Stage访问

- 线上环境调用。直接发起 API 调用，即调用线上环境。本例中如下图：



- 预发环境调用。调用预发环境的API，则在调用API时，在 Header 中增加入参 **X-Ca-Stage: PRE**， 即可访问预发环境的 API。
- 测试环境调用。调用测试环境的API，则在调用API时，在 Header 中增加入参 **X-Ca-Stage: TEST**， 即可访问测试环境的 API。本例中如下图：

3. Http后端+域名

3.1 前期准备

复用上一章节中的两个ECS，把这两个ECS实例做为API的http后端，如下图所示。每个ECS实例都开放了TCP 80端口，安装了nginx做为http的web server，使用http访问时，分别会返回{"env":"test env"}和{"env":"relese env"}。

> **说明**
>
> 本示例仅用于介绍如何使用API网关进行环境管理，因此并未考虑高可用、可扩展性、安全性等重要因素，故本示例的后端服务架构请勿做为您正式使用时的架构参考。

3.2 API后端配置

步骤1：在API网关控制台中创建API分组，进入 **环境管理** 。

步骤2：在 **环境管理** 中，在线上环境和测试环境中分别创建一个同名的变量（本例中为 backend-host ），但值的内容不同，分别对应本例中的两个ECS的公网地址，如下图所示：

步骤3：在此分组下创建API，略过API的其他定义，重点在**定义API后端服务**的页面中，在**后端服务地址**的位置填写变量名称，填写 http://#backend-host# 。

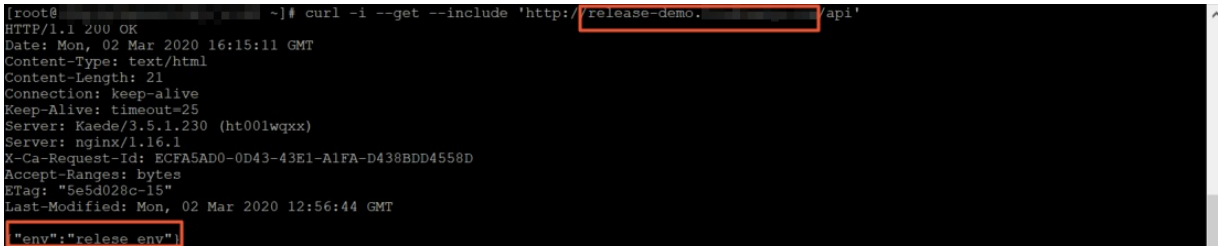
步骤4：完成其他API配置，注意需要发布到线上环境和测试环境中。

步骤5：为环境绑定域名。在 `API分组` 菜单中，进入 `绑定域名` 。

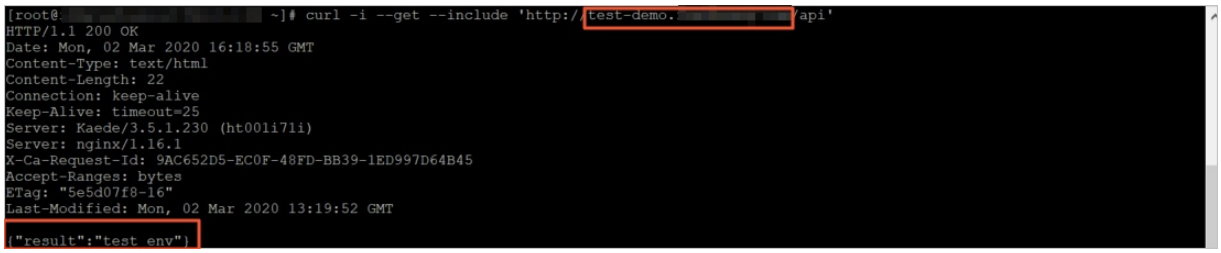
步骤6：为测试环境和线上环境分别绑定两个域名，如下图所示。

3.3 使用域名访问

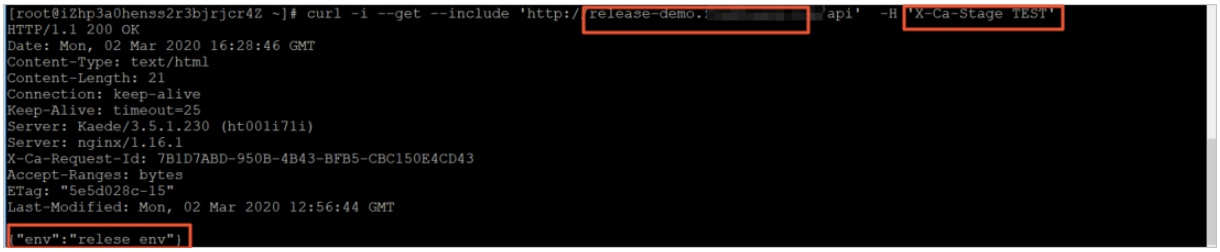
- 线上环境调用。使用绑定好的线上域名进行调用，如下图所示：



- 预发环境调用。使用绑定好的线上域名进行调用。
- 测试环境调用。使用绑定好的线上域名进行调用，如下图所示：



注意：绑定环境的域名优先级大于 X-Ca-Stage，即在调用绑定环境的域名中，仍然在header中添加 X-Ca-Stage 信息，API网关会以域名的环境配置为准，如下图所示：



4. 函数计算后端+域名和X-Ca-Stage访问

4.1 API后端配置

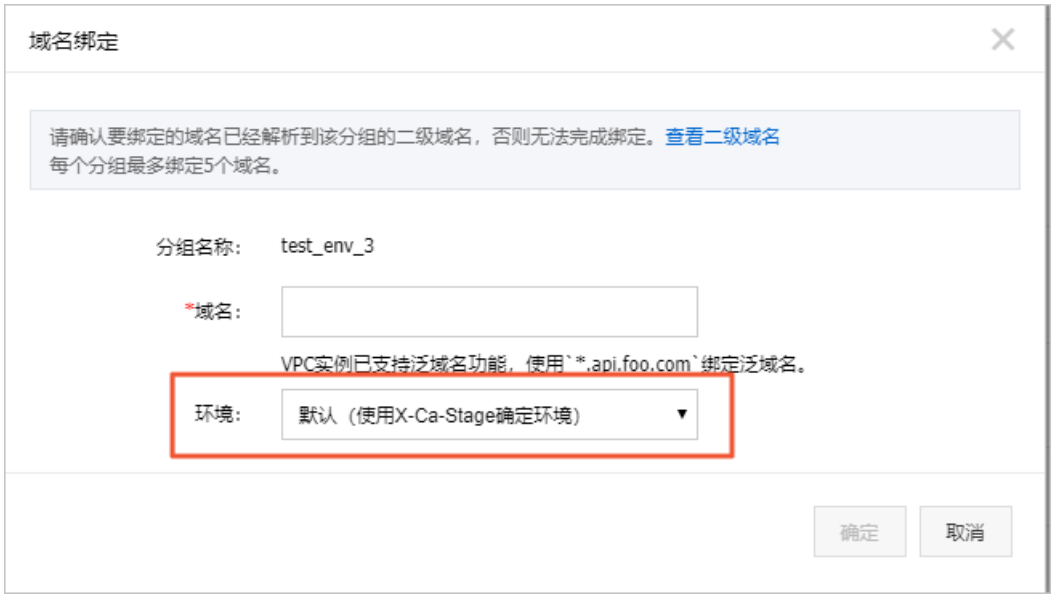
API后端服务类型为函数计算时，如下图所示，更多函数计算的配置参考 创建后端为函数计算的API。

可以使用两种环境变量的方式进行环境管理：

- 类似后端服务类型为VPC时的配置方式，环境变量可以配置在 服务名称 、 函数名称 的位置，从而实现不同的API环境对应不同的函数计算后端；
- 采用函数别名的方式。函数别名是函数计算版本管理的一个功能，可以为函数计算不同的服务版本创建不同的自定义名称，具体可以参考 别名操作。因此可以将**环境变量**的内容设置为不同的别名，从而实现由不同版本的函数来进行处理。

4.2 使用域名+X-Ca-Stage访问

如果想用同一个域名，通过X-Ca-Stage方式访问多套环境，需在API分组绑定域名时，可以按下图所示配置：



5. 使用限制

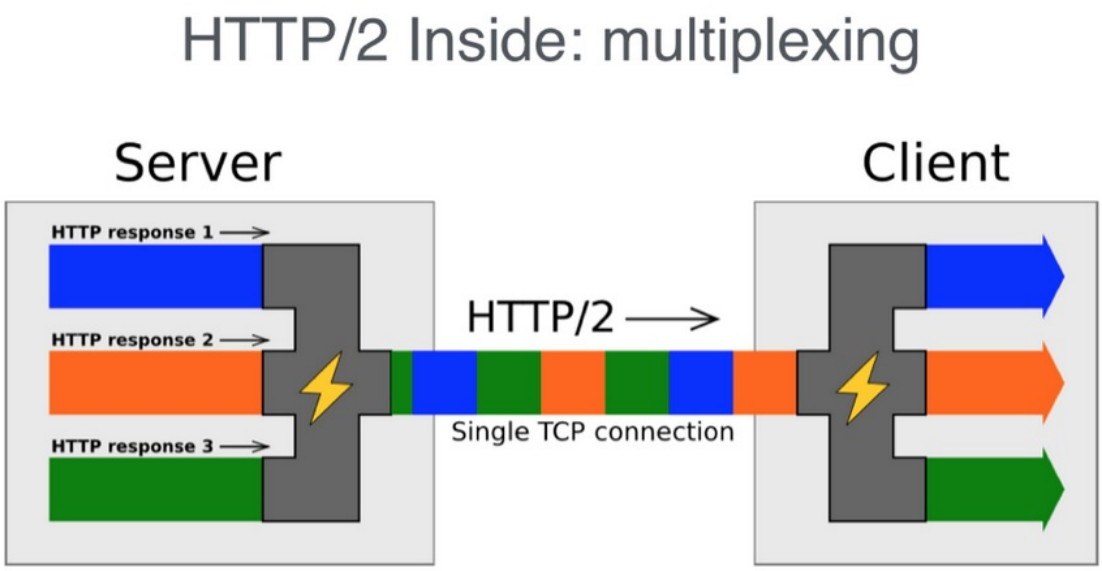
- 环境变量的内容发生变化后，注意需要将分组内对应的**API重新发布**，环境变量才能生效。
- 每个环境允许配置最多 **50** 个环境变量。

13.支持HTTP2.0

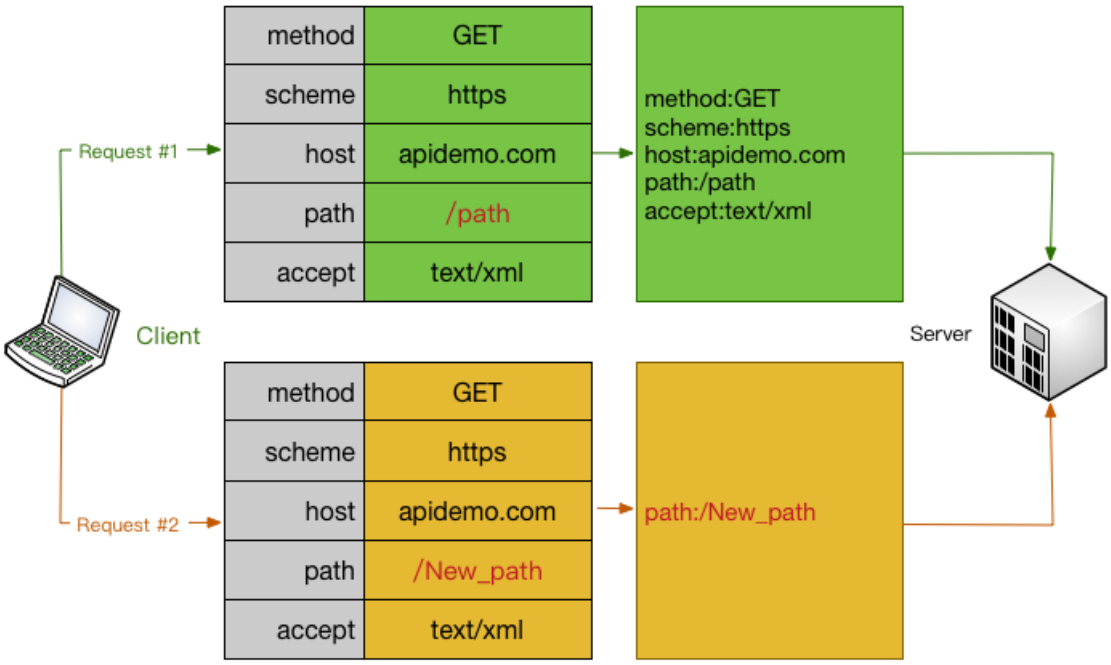
API网关支持HTTP2.0

API网关支持HTTP2.0新特性，支持API请求多路复用、支持请求头压缩。

- 多路复用（MultiPlexing）：消除了 HTTP 1.x 中并行处理和发送请求及响应时对多个连接的依赖。可客户端和服务器可以把HTTP消息分解为互不依赖的帧，然后乱序发送，最后再在另一端把它们重新组合起来。从而避免不必要的延迟，提升效率，在请求量比较大的场景，客户端也可以轻松使用少量连接完成大量请求数据的传输。



- header压缩：如上文中所言，HTTP1.x 的header带有大量信息，而且每次都要重复发送。HTTP 2.0 使在客户端和服务器端使用“首部表”来跟踪和存储之前发送的键值对，对于相同的数据，不再通过每次请求和响应发送；“首部表”在 HTTP 2.0 的连接存续期内始终存在，由客户端和服务器共同渐进地更新；每个新的首部键值对要么追加到当前表的末尾，要么替换表中之前的值。从而减少每次请求的数据量。



如何开启HTTP 2.0?

- 新建的API分组（2017年7月14日以后）

  HTTPS的API都可以使用HTTP2协议进行客户端和API网关的通信。（由于 HTTP 2.0 只许在HTTPS下运行，所以需要您使用HTTPS并用域名访问方可启用 HTTP 2.0）

- 存量API分组

  您需稍做等待，后续将提供功能手动开启。

14.模型管理

1. 模型的简介及限定

模型用于描述HTTP协议的请求数据和响应数据。API网关通过使用JSON Schema定义模型，用来描述用户API约定数据的组织方式，比如参数或者返回值有哪些字段，这些字段的取值范围等。同时，通过定义模型，并在用户创建的API中加以引用，用户在API的SDK导出时，关联的Model会自动生成对应的POJO类。这样可以增强用户传入参数的便利性，同时可以方便用户反序列化返回的数据。

API网关模型定义基于JSON架构草案4的规范，但存在一定的条件限制：1. 仅支持创建元素属性为Object类型的JSON Schema；2. $ref仅支持本用户的内部Model引用。Model的‘ref’引用地址可以通过创建模型和获取已创建的模型获取。‘ref’不支持循环引用。

Api网关支持的模型可以参考如下定义：

```
{
        "required": ["name", "photoUrls"],
        "type": "object",
        "properties": {
                "id": {
                        "format": "int64",
                        "type": "integer"
                },
                "category": {
                        "$ref": "https://apigateway.aliyun.com/models/bbc725be4b0b48b79bdd2
f6ebbdcc8c0/a5e7741d8a3a4bcb9746275a0db15fcf"
                },
                "name": {
                        "pattern": "^\\d{3}-\\d{2}-\\d{4}$",
                        "type": "string"
                },
                "status": {
                        "type": "string"
                },
                "dogProject": {
                        "type": "object",
                        "properties": {
                                "id": {
                                        "format": "int64",
                                        "maximum": 100,
                                        "exclusiveMaximum": true,
                                        "type": "integer"
                                },
                                "name": {
                                        "maxLength": 10,
                                        "type": "string"
                                }
                        }
                }
        }
}
```

2. 创建模型

您可以通过阿里云提供的创建模型进行模型创建。同样，您也可以通过API网关的控制台进行创建。

模型相关操作的控制台入口：

1. 点击分组管理
2. 点击模型管理，进入模型管理界面来创建模型

Swagger导入创建模型：

Api网关支持通过导入Swagger创建API。Swagger文件中的Model相关内容会在Swagger导入成功后，会在该分组下自动生成模型。注意：**通过Swagger导入模型时，同名模型将直接被覆盖，不会进行用户确认。**

3. 修改和查看模型

完成模型的创建后，可以在模型管理界面点击查看所需的模型。在模型的详情页，可以看到模型的名称，模型的定义，以及系统为其分配的URI。API网关模型间可以通过'$ref:{URI}'来实现模型间的项目引用。

如果用户希望对当前模型的信息进行修改，可以点击右上角修改按钮完成模型的修改。需要注意的是：**模型的URI不随模型的更改发生改变**

4. 删除模型

用户可以对分组下的模型进行删除操作。注意：**API网关不维护模型和API的关联关系，删除模型时可能会引起线上API的SDK导出失败等问题。因此，删除模型请谨慎操作。**

15.双向通信使用指南

一、概述

移动端APP大多数功能都能通过客户端向服务器端发送请求，服务器应答来完成。比如：用户注册，获取商品列表等能力。

但有一些场景需要服务器向客户端推送应用内通知，如：用户之间的即时通信等功能。这种时候就需要建立一个通信通道，让服务器能够给指定的客户端发送下行通知请求。也就是客户端和服务器端之间具备双向通信的能力。

具备双向通行能力的架构对于移动APP属于刚性需求。

使用须知

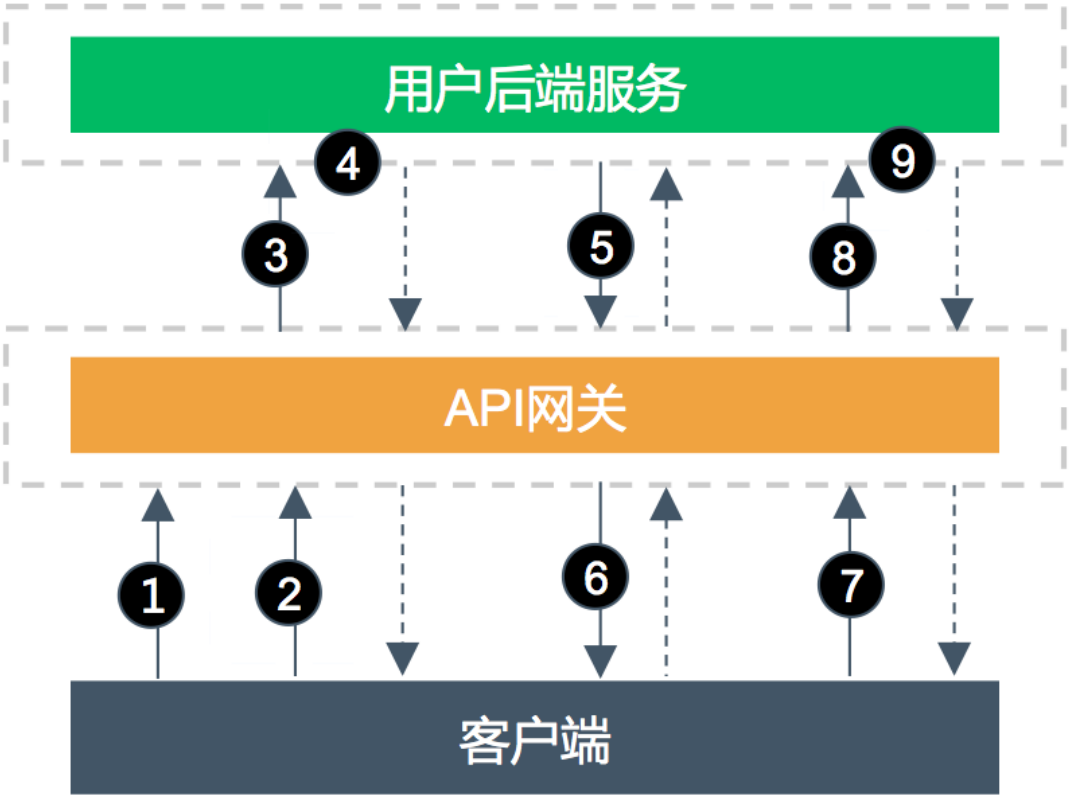
API网关已经在所有Region开放双向通信能力，双向通信能力构建于WebSocket协议之上，目前Android，Objective-C，JAVA三种SDK均支持双向通信。

（若您有此之外的需求可以工单或者钉钉群：11747055进行咨询）

实现方式

API网关目前已经在所有Region提供双向通信的能力，用户只需要在API网关上设置三个API，然后下载自动生成的SDK到客户端，简单嵌入到客户端就能完美实现客户端和服务器端之间的双向通信的功能。

下面是利用API网关实现双向通信的能力的业务流程简图：



流程描述

(1) 客户端在启动的时候和API网关建立了WebSocket连接，并且将自己的设备ID告知API网关；

(2) 客户端在WebSocket通道上发起注册信令；

(3) API网关将注册信令转换成HTTP协议发送给用户后端服务，并且在注册信令上加上设备ID参数（增加在名称为x-ca-deviceid的header中）；

(4) 用户后端服务验证注册信令，如果验证通过，记住用户设备ID，返回200应答；

(5) 用户后端服务通过HTTP/HTTPS/WebSocket三种协议中的任意一种向API网关发送下行通知信令，请求中携带接收请求的设备ID;

(6) API网关解析下行通知信令，找到指定设备ID的连接，将下行通知信令通过WebSocket连接发送给指定客户端；

(7) 客户端在不想收到用户后端服务通知的时候，通过WebSocket连接发送注销信令给API网关，请求中不携带设备ID;

(8) API网关将注销信令转换成HTTP协议发送给用户后端服务，并且在注册信令上加上设备ID参数；

(9) 用户后端服务删除设备ID，返回200应答。

二、双向通信三种管理信令

要使用API网关的双向通信能力，首先要了解API网关双向通信相关的三种信令，需要注意的是，这三个信令其实就是API网关上的三个API，需要用户去API网关创建后才能使用。

1.注册信令

注册信令是客户端发送给用户后端服务的信令，起到两个作用：

（1）将客户端的设备ID发送给用户后端服务，用户后端服务需要记住这个设备ID。用户不需要定义设备ID字段，设备ID字段由API网关的SDK自动生成；

（2）用户可以将此信令定义为携带用户名和密码的API，用户后端服务在收到注册信令的验证客户端的合法性。用户后端服务在返回注册信令应答的时候，返回非200时，API网关会视此情况为注册失败。

客户端要想收到用户后端服务发送过来的通知，需要先发送注册信令给API网关，收到用户后端服务的200应答后正式注册成功。

2.下行通知信令

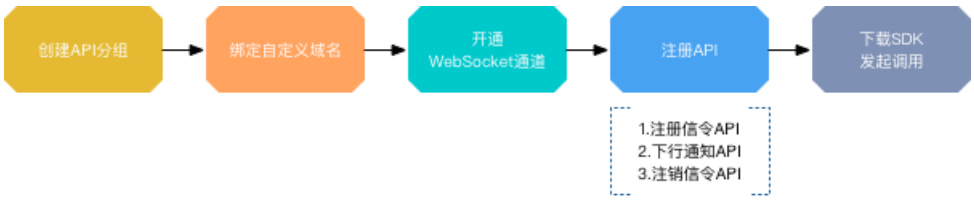
用户后端服务，在收到客户端发送的注册信令后，记住注册信令中的设备ID字段，然后就可以向API网关发送接收方为这个设备的下行通知信令了。只要这个设备在线，API网关就可以将此下行通知发送到客户端。

3.注销信令

客户端在不想收到用户后端服务的通知时发送注销信令发送给API网关，收到用户后端服务的200应答后注销成功，不再接受用户后端服务推送的下行消息。

三.API网关设置双向通信

1.开通绑定分组域名的WebSocket通道



1.1 创建分组

已经有分组的情况可忽略本节。

要使用API网关的基本功能，首先需要在API网关上创建一个分组。

1.2 在分组上绑定域名

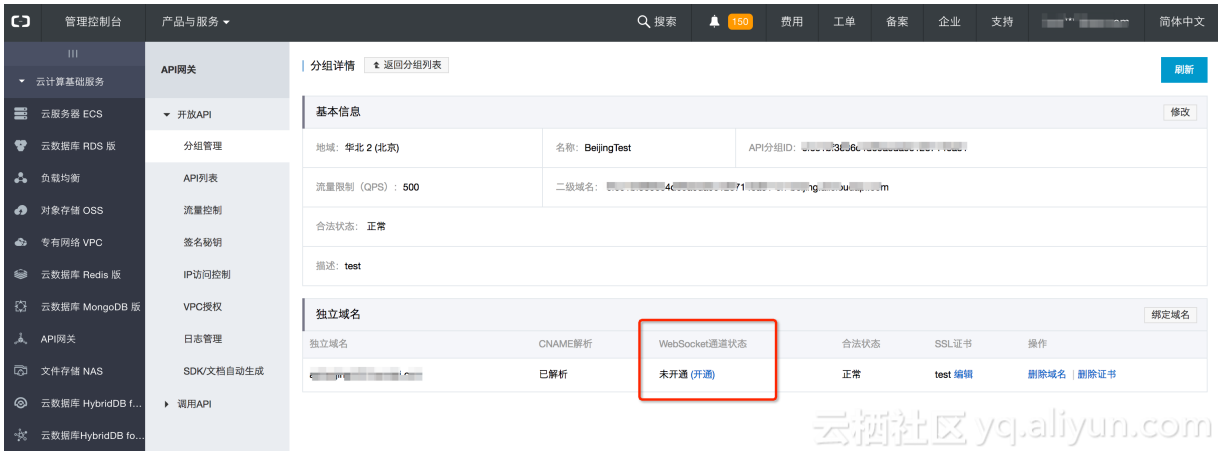
已经绑定了域名的情况可忽略本节。

创建完分组后，需要在分组上绑定一个域名。

1.3 开通域名的WebSocket通道

绑定好域名后，需要开通域名上的WebSocket通道。

具体开通方法如下： 开通页面路径：【API网关控制台】->【开放API】->分组详情。



2.创建注册、下行通知、注销信令的API

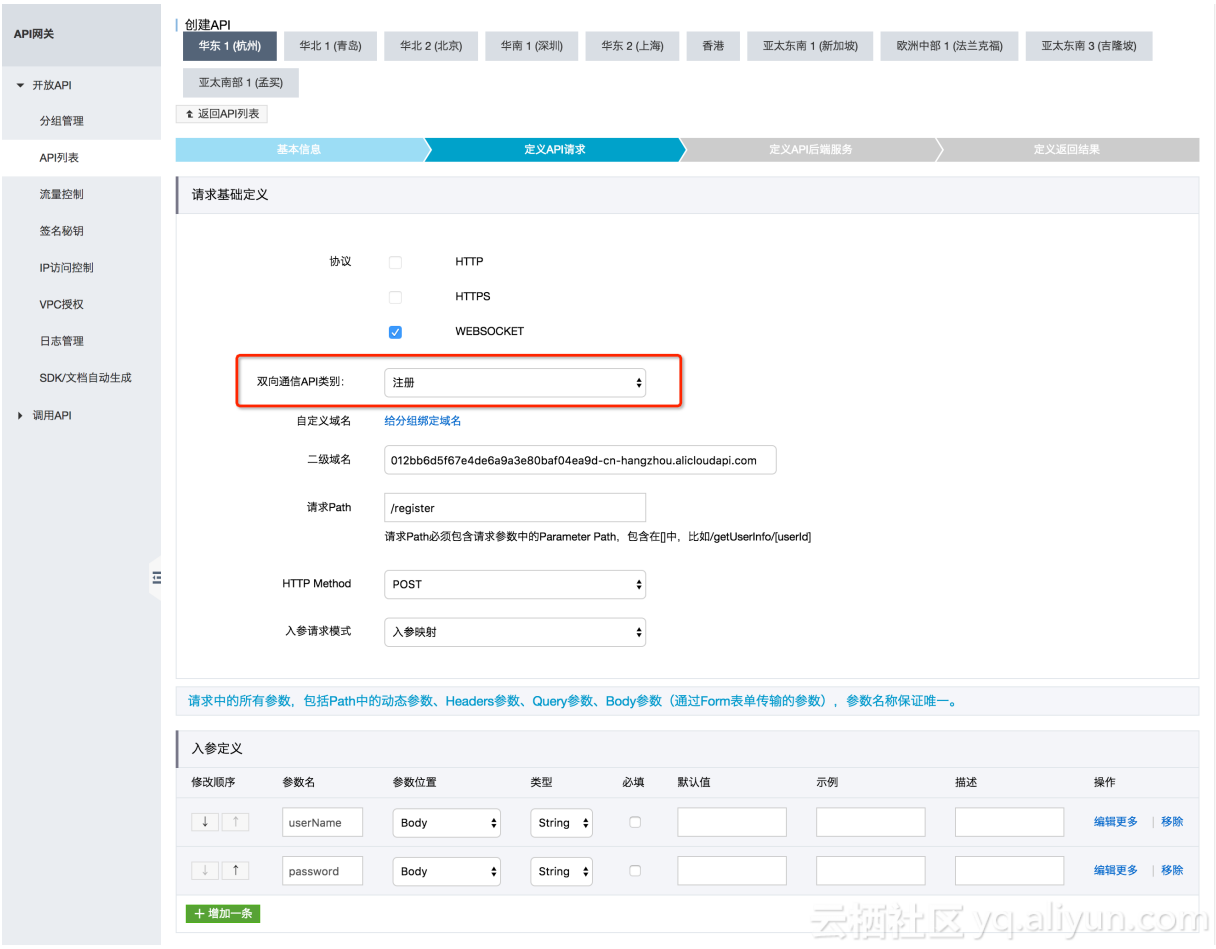
需要刚才创建的分组下创建三个API。

这三个API在创建的时候，需要特别注意的是，需要选择“双向通信API类别”选项。“双向通信API类别”选项在勾选“WebSocket协议”时会自动弹出，如下图：

2.1 注册信令API

注册信令是客户端发送给用户后端服务的信令，创建的时候需要注意的是：

(1) 一般会包含用户名和密码两个字段，如图所示：



(2) 只能通过WebSocket协议传输

当然这个信令的定义由用户自己去定义，包含什么参数都是可以的。重要的是，这个信令的应答必须是200，客户端才算注册成功。

2.2 下行通知API

下行通知信令是用户后端服务发送给客户端的信令，创建的时候需要注意的是：

(1) 强烈建议使用和客户端不同的APP进行授权，区分用户后端服务和客户端调用权限；

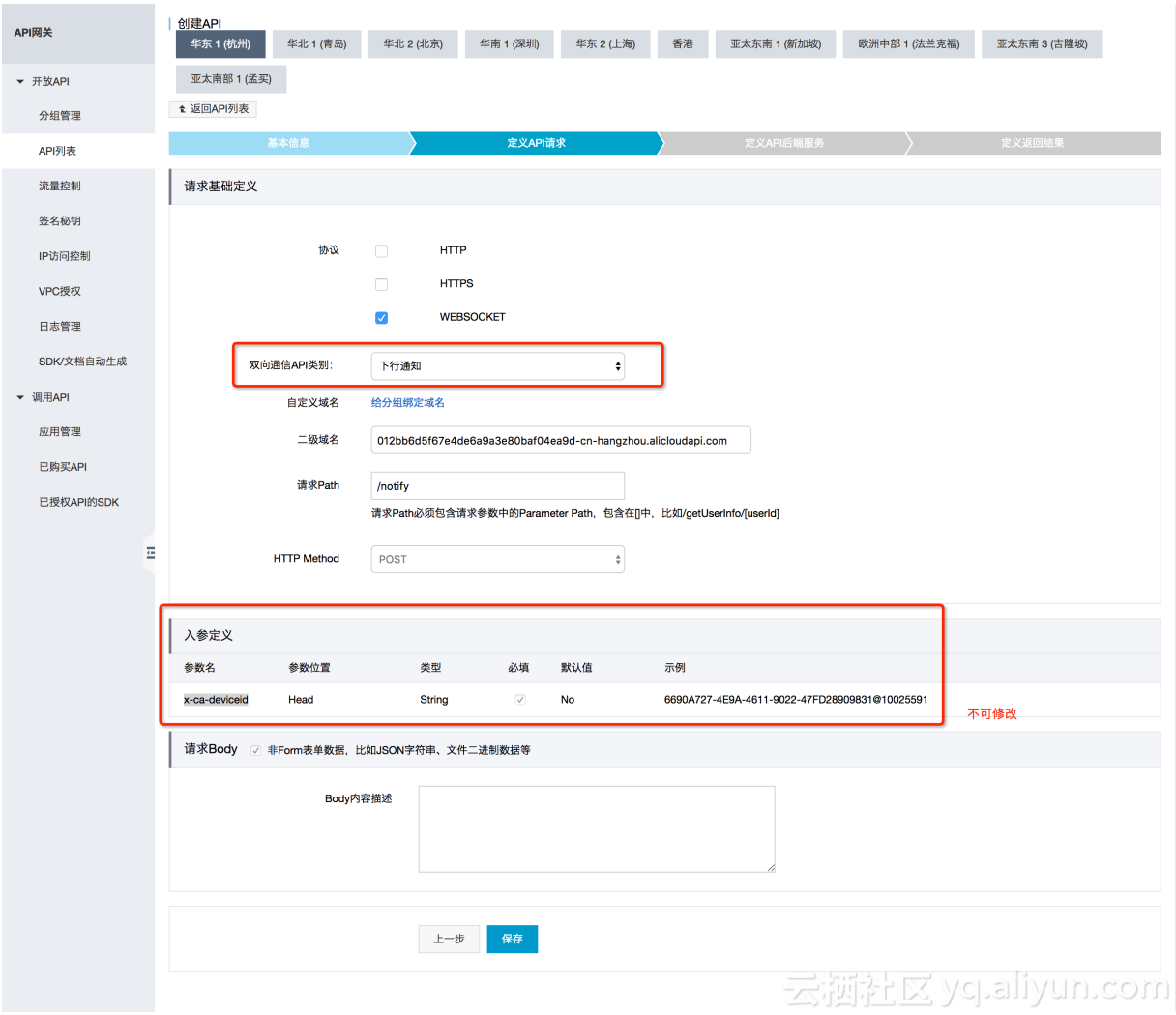
(2) 可以使用HTTP/HTTPS/WebSocket中任意协议调用此API;

(3) 因为是发送给客户端的，因此不需要和其他API一样定义后端服务参数；

(4) 请求中必须携带接收通知的客户端ID的x-ca-deviceid头，且不可修改；

(5) 下行通知请求最大限制为8KB；

创建页面如下图：



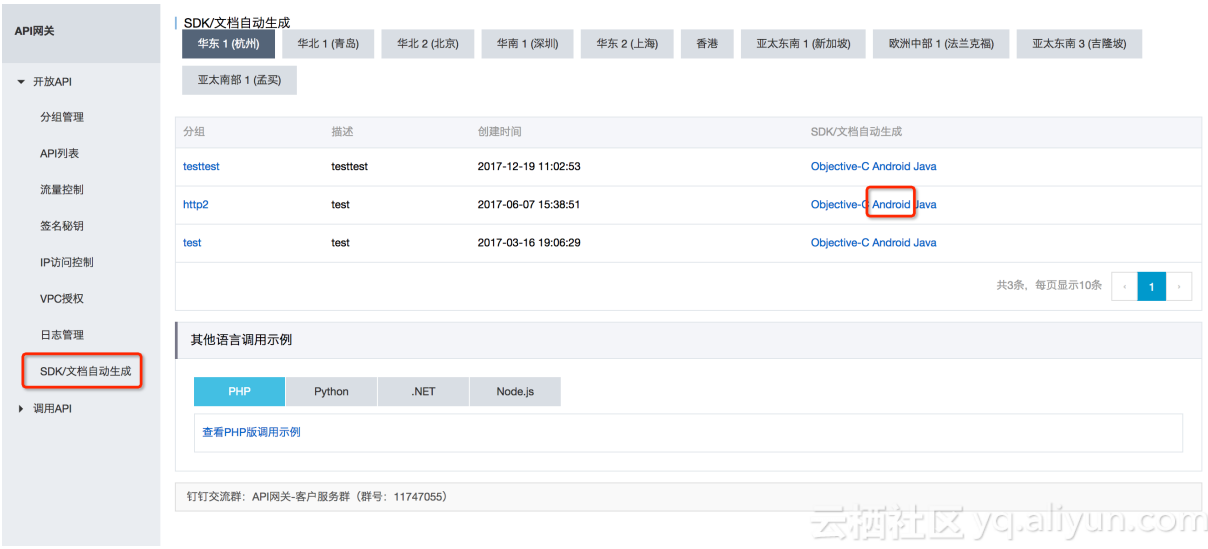
2.3 注销信令API

注销信令是客户端发送给用户后端服务的信令，创建的时候需要注意的是只能通过WebSocket协议传输。

3.生成、下载SDK

在三个信令API全部创建成功后，需要将分别授权到指定APP上。授权后，需要发布到线上环境。在授权、发布完成后，就可以到SDK文档自动生成页面去生成、下载SDK了。

因此您可以下载Android的SDK作为客户端SDK，使用这个SDK来使用WebSocket和API网关进行通信，并在此SDK上发送注册信令，接收用户后端服务发送的下行通知信令。您可以下载JAVA语言的SDK作为服务器端SDK，用来发送下行通知信令。两个SDK配合完成双向通信功能。下面是下载页面：



4.调用SDK的注册信令，并在SDK中读取下行通知信令的内容

Android的SDK下载下去后，需要仔细阅读SDK的安装和使用说明，也就是ReadMe.txt文件。

自动生成的SDK里面有所有API的调用入口，我们找到调用入口，就可以调用这个API来发送注册信令了。下面是一个Demo示例，在此Demo中，我们在启动的时候先初始化一个WebSocket通道出来，在通道中注册接收用户后端服务发送的下行通知的函数onNotify，客户端接收到用户后端服务发送的下行通知，SDK会调用这个函数来通知客户端。然后Demo提供注册函数registerWebsocketTest供外部调用。

```
public class Demo_HangZhou {
    static{
        WebSocketClientBuilderParams websocketParam = new WebSocketClientBuilderParams();
        websocketParam.setAppKey("12345678");
        websocketParam.setAppSecret("12345678");
        websocketParam.setApiWebSocketListner(new ApiWebSocketListner() {
            @Override
            //客户端接收到用户后端服务发送的下行通知，SDK会调用这个函数来通知客户端
            public void onNotify(String message) {
                System.out.println(message);
            }

            @Override
            public void onFailure(Throwable t, ApiResponse response) {
                if(null != t){
                    t.printStackTrace();
                }

                if(null != response){
                    System.out.println(response.getCode());
                    System.out.println(response.getMessage());
                }

            }
        });

        WebSocketApiClient_hangzhou.getInstance().init(websocketParam);


    }



    public static void registerWebsocketTest(){
        WebSocketApiClient_hangzhou.getInstance().register("fred" , "123456" , new ApiCallb
ack() {

            @Override
            public void onFailure(ApiRequest request, Exception e) {
                e.printStackTrace();
            }

            @Override
            public void onResponse(ApiRequest request, ApiResponse response) {
                try {
                    System.out.println(getResultString(response));
                }catch (Exception ex){
                    ex.printStackTrace();
                }
            }
        });
    }
}
```

5.用户后端服务发送下行通知

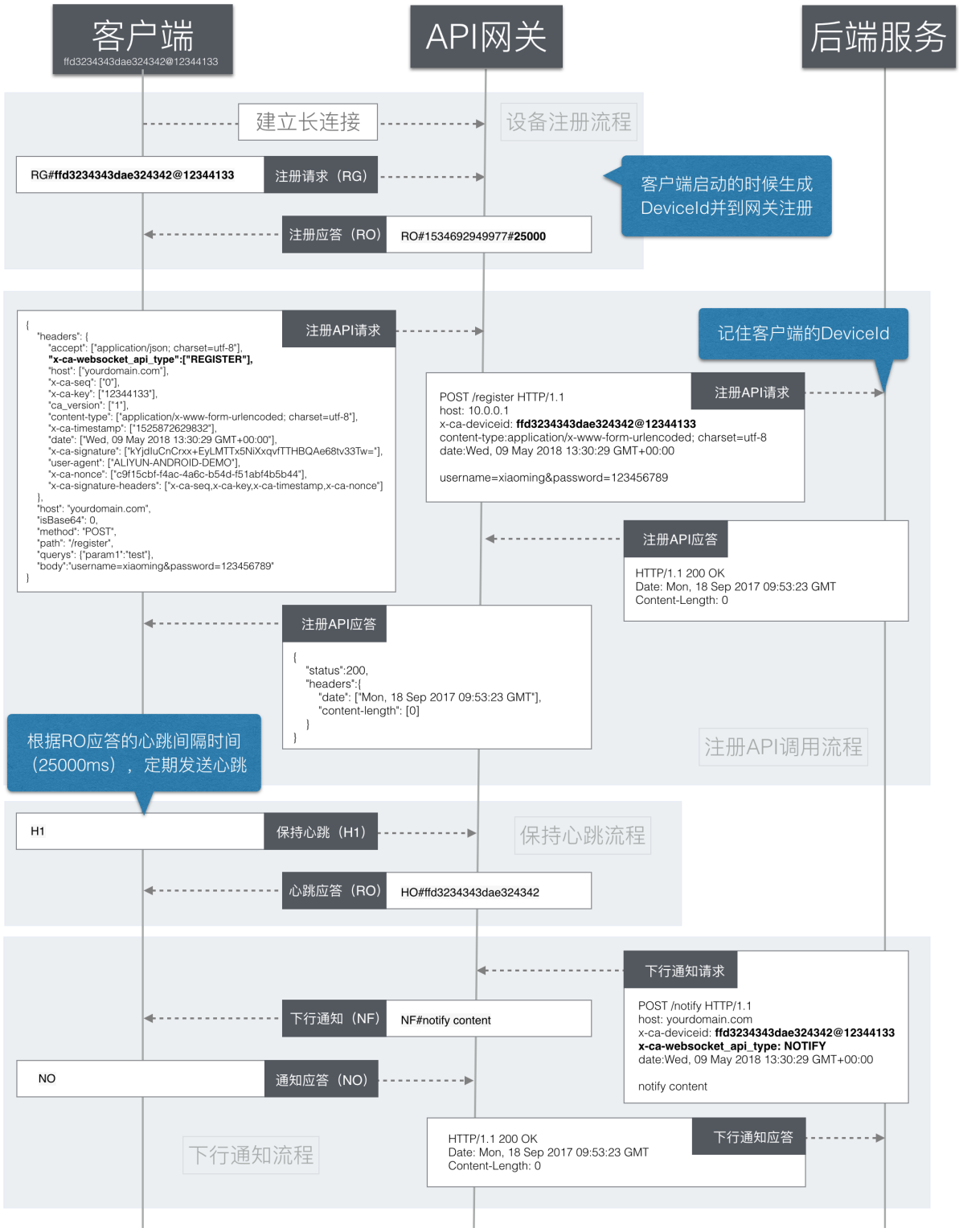
用户后端服务在接收到客户端发送的注册信令后，需要记住信令请求中设备ID。然后用户后端服务给客户端发送下行通知就变得非常容易，就和调用普通API一样，发送一个标准的API调用给API网关就可以了。下面是调用代码示例：

```
public class Demo_Hanzhou {

            static{
                HttpClientBuilderParams param = new HttpClientBuilderParams();
                param.setAppKey("123456");
                param.setAppSecret("123456");
                HttpApiClient_BeiJing.getInstance().init(param);

            }

            public static void HanZhouNotifyTest(){
                HttpApiClient_HanZhou.getInstance().notify("NotifyContent" , new ApiCallbac
k() {

                    @Override
                    public void onFailure(ApiRequest request, Exception e) {
                        e.printStackTrace();
                    }

                    @Override
                    public void onResponse(ApiRequest request, ApiResponse response) {
                        try {
                            System.out.println(response.getCode());
                        }catch (Exception ex){
                            ex.printStackTrace();
                        }
                    }
                });
            }
}
```

让我们再来整理一下API网关双向通信能力的使用流程：

1. 开通分组绑定的域名的WebSocket通道；
2. 创建注册、下行通知、注销三个API，给这三个API授权、并上线；
3. 用户后端服务实现注册，注销信令逻辑，下载JAVA的SDK发送下行通知；
4. 下载AndroidSDK，嵌入到客户端，建立WebSocket连接，发送注册请求，监听下行通知；

这篇文章中的一个结合报文的流程图可以帮助大家更好理解双向通信的原理，贴出来给大家参考：



如果在使用中遇到棘手的问题，请加入我们钉钉交流群：API网关-客户服务群（群号：11747055）。