阿里云
API 网关

用户指南（开放 API）

文档版本：20200422

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. 阿里云文档中所有内容，包括但不限于图片、架构设计、页面布局、文字描述，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
	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。	 **禁止：** 重置操作将丢失用户配置数据。
	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。
	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。
	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。	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 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函数计算接入文档。

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=request
/[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，详见章节3.4. 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"
${User}.${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API网关错误码处理，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5. Header转发规则
- **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。

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->RequestParameter
- 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5. Header转发规则
- **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

HeaderName	请求处理方式	应答处理方式
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	
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。

2 流量控制策略

流量控制策略和 API 分别是独立管理的，操作两者绑定后，流控策略才会对已绑定的 API 起作用。

在已有的流量控制策略上，可以额外配置特殊用户和特殊应用（APP），这些特例也是针对当前策略已绑定的API生效。

流量控制策略可以配置对 API、用户、应用三个对象的流控值，流控的单位可以是秒、分钟、小时、天。使用流量控制策略您需要了解以下几点：

- 流量控制策略可以涵盖下表中的维度：

API 流量限制	该策略绑定的API在单位时间内被调用的次数不能超过设定值，单位时间可选秒、分钟、小时、天，如5000次/分钟。
APP 流量限制	每个APP对该策略绑定的任何一个API在单位时间内的调用次数不能超过设定值。如50000次/小时。
用户流量限制	每个阿里云账号对该策略绑定的任何一个 API 在单位时间内的调用次数不能超过设定值。一个阿里云账号可能有多个 APP，所以对阿里云账号的流量限制就是对该账号下所有 APP 的流量总和的限制。如 50 万次/天。

  在一个流控策略里面，这三个值可以同时设置。请注意，用户流量限制应不大于 API 流量限制，APP 流量限制应不大于用户流量限制。即 APP 流量限制 <= 用户流量限制 <= API 流量限制。

  此外，您可以在流控策略下添加特殊应用（APP）和特殊用户。对于特例，流控策略基础的 **API 流量限制** 依然有效，您需要额外设定一个阈值作为该 APP 或者该用户的流量限制值，该值不能超过策略的 **API流量限制** 值，同时流控策略基础的 **APP流量限制** 和 **用户流量限制** 对该 APP 或用户失效。
- 与签名密钥相似，当您创建流量控制策略时，需要选择 **Region**，**Region** 一旦选定不可更改，且仅能被应用于同一个 Region 下的 API。
- 由于 API 网关限制，当您设置 **API 流量限制** 值时，考虑每个 API 分组的默认流控上限是500QPS（该值可以通过提交工单申请提高）。
- 绑定 API。您可以将策略绑定于多个 API，流控策略的限制值和特例将对该策略绑定的每一个 API 单独生效。当您绑定 API 时，如果该 API 已经与某个策略绑定，您的此次操作将替换之前的策略，实时生效。

- 特殊对象。如果您想要添加特殊应用或者特殊用户，您需要获得应用 ID 即 AppID 或者用户的阿里云邮箱账号。
- 在 API 网关控制台，您可以完成对流量控制策略的创建、修改、删除、查看等基本操作。还有流控策略与 API 的绑定解绑，流量控制策略特殊对象的添加删除等操作。

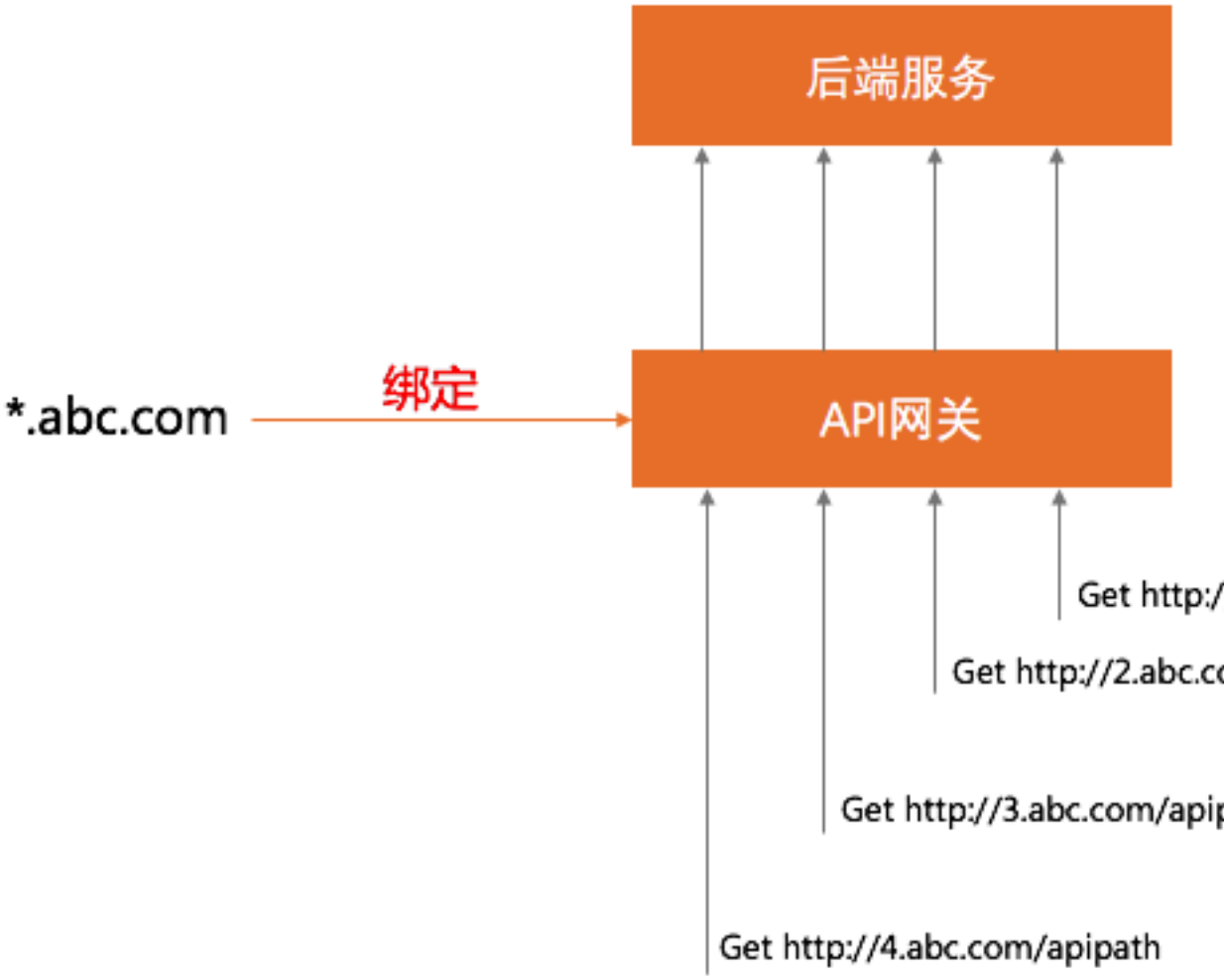
3 API网关支持泛域名绑定

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网关后直接在控制台上将对应的泛域名绑定自己的API分组上，然后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。

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：



下面我们将一个测试域名的泛域名*.test.yourdomain.com绑定到API网关，绑定后就可以使用1.test.yourdomain.com，2.test.yourdomain.com等域名来访问API网关，我们来具体走一遍绑定泛域名的流程。

1. 修改泛域名解析

步骤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

步骤2. 进入自己的域名解析管理页面，阿里云的域名解析管理页面入口在：https://dns.console.aliyun.com， 在域名列表页面找到要管理的域名，点击域名上的链接进入域名的管理页面。

步骤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*.test ，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. 在API网关上绑定泛域名

步骤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。 步骤2. 在页面右下方看到绑定域名的按钮，点击按钮：

步骤3. 进入域名绑定页面，填写刚才做完解析的泛域名*.test.yourdomain.com，点击确定后域名绑定成功。

绑定成功后，我们就可以随意使用泛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://1.test.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

curl http://2.test.fredhuang.com/httpCommon -i
HTTP/1.1 200 OK
Date: Mon, 23 Mar 2020 08:40:56 GMT
Connection: keep-alive
Keep-Alive: timeout=25
Server: Jetty(7.2.2.v20101205)
X-Ca-Request-Id: C0688191-BFFC-4571-BE74-5F82B0C0A731
```

3.使用限制

1. 泛域名绑定之前一定要先将泛域名解析到分组二级域名，否则会绑定失败
2. 只有VPC实例会支持泛域名能力。

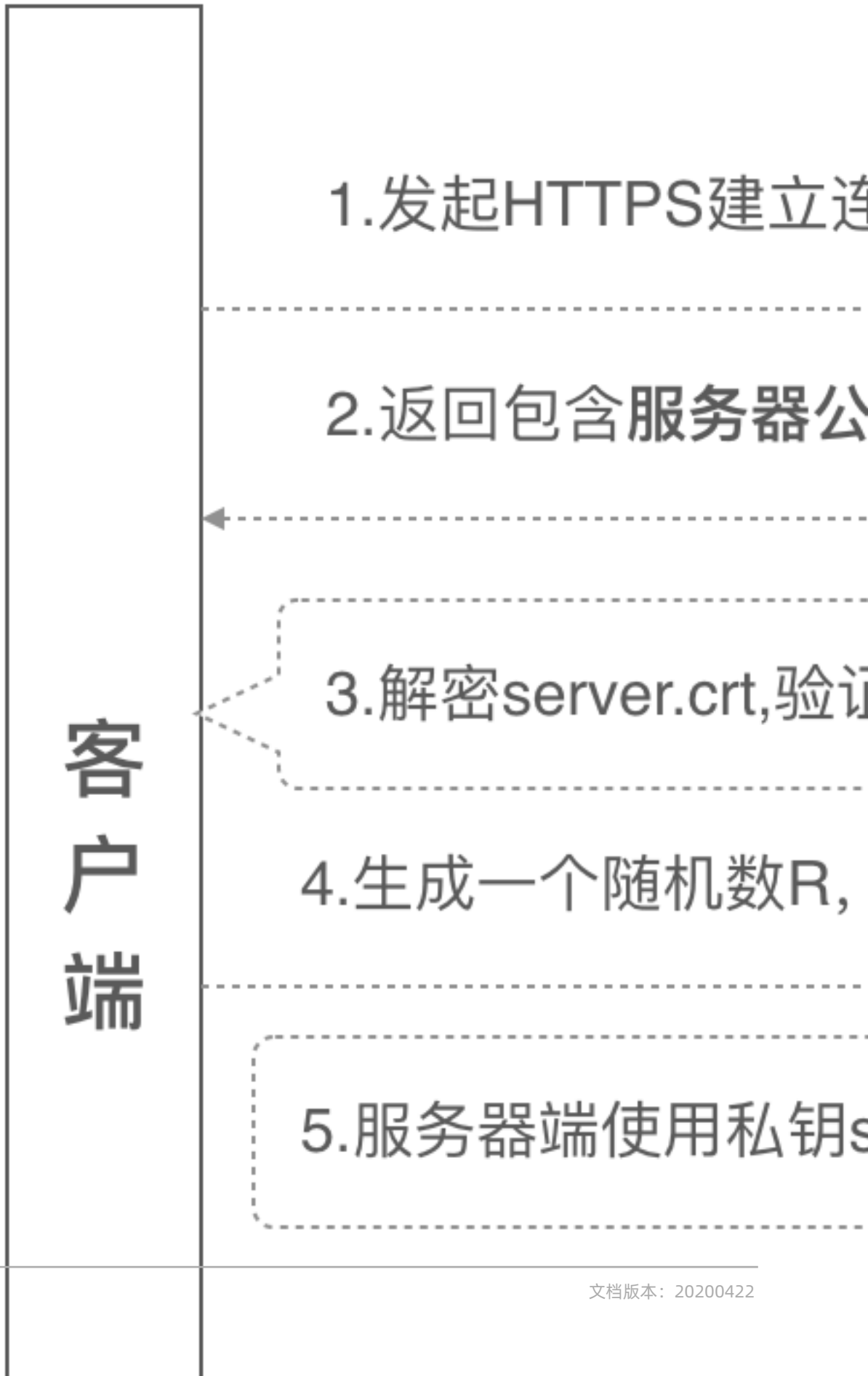
4 HTTPS双向认证（Mutual TLS authentication）

双向认证，顾名思义，客户端和服务器端都需要验证对方的身份，在建立Https连接的过程中，握手的流程比单向认证多了几步。单向认证的过程，客户端从服务器端下载服务器端公钥证书进行验证，然后建立安全通信通道。双向通信流程，客户端除了需要从服务器端下载服务器的公钥证书进行验证外，还需要把客户端的公钥证书上传到服务器端给服务器端进行验证，等双方都认证通过了，才开始建立安全通信通道进行数据传输。

1. 原理

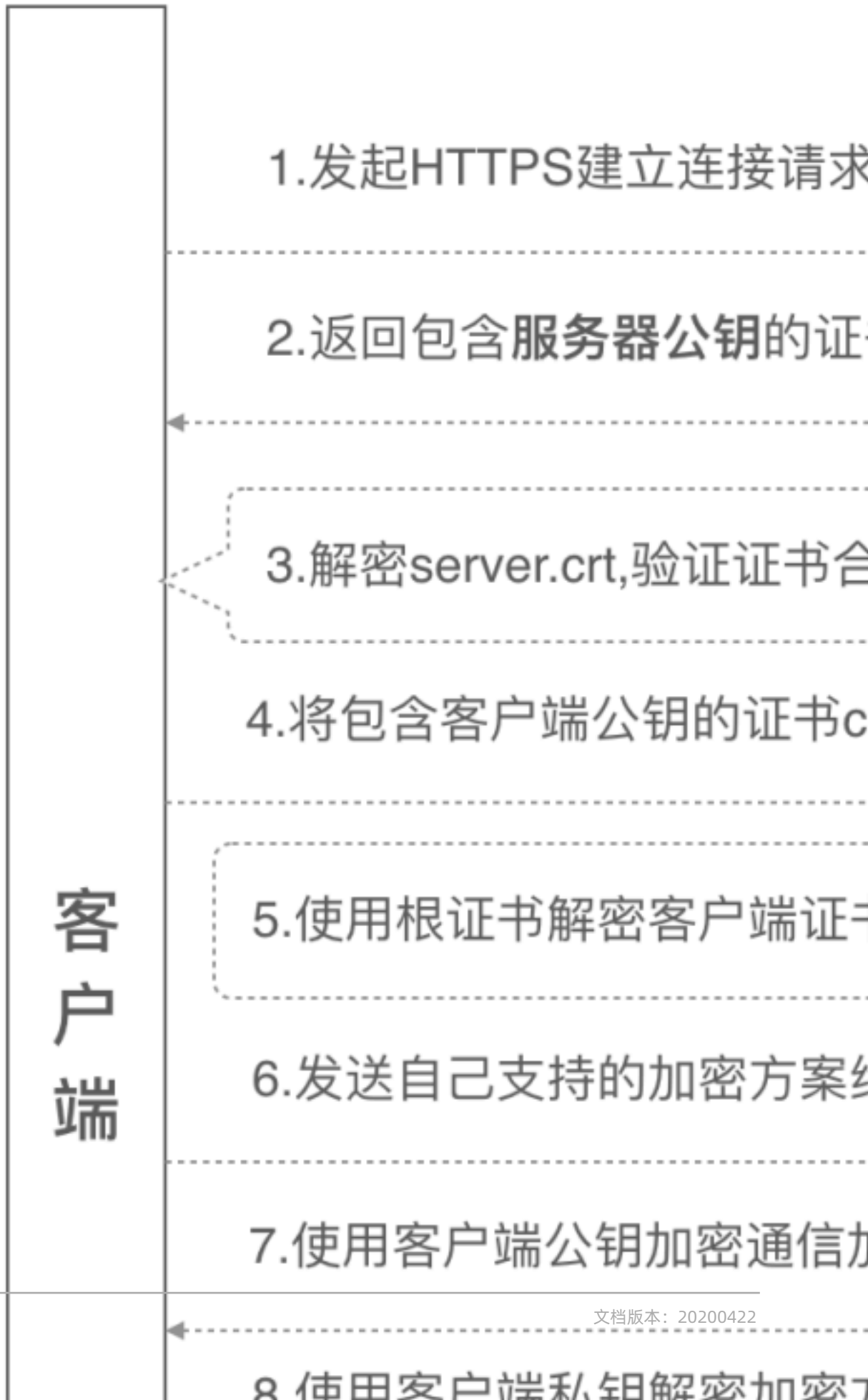
1.1 单向认证流程

单向认证流程中，服务器端保存着公钥证书和私钥两个文件，整个握手过程如下：


客户端
1.发起HTTPS建立连
2.返回包含服务器公
3.解密server.crt,验证
4.生成一个随机数R,
5.服务器端使用私钥s

1. 客户端发起建立HTTPS连接请求，将SSL协议版本的信息发送给服务器端；
2. 服务器端将本机的公钥证书（server.crt）发送给客户端；
3. 客户端读取公钥证书(server.crt)，取出了服务端公钥；
4. 客户端生成一个随机数（密钥R），用刚才得到的服务器公钥去加密这个随机数形成密文，发送给服务端；
5. 服务端用自己的私钥(server.key)去解密这个密文，得到了密钥R
6. 服务端和客户端在后续通讯过程中就使用这个密钥R进行通信了。

1.2 双向认证流程


客户端
1.发起HTTPS建立连接请求
2.返回包含服务器公钥的证
3.解密server.crt,验证证书合
4.将包含客户端公钥的证书c
5.使用根证书解密客户端证
6.发送自己支持的加密方案
7.使用客户端公钥加密通信

1. 客户端发起建立HTTPS连接请求，将SSL协议版本的信息发送给服务端；
2. 服务器端将本机的公钥证书(server.crt)发送给客户端；
3. 客户端读取公钥证书(server.crt)，取出了服务端公钥；
4. 客户端将客户端公钥证书(client.crt)发送给服务器端；
5. 服务器端使用根证书(root.crt)解密客户端公钥证书，拿到客户端公钥；
6. 客户端发送自己支持的加密方案给服务器端；
7. 服务器端根据自己和客户端的能力，选择一个双方都能接受的加密方案，使用客户端的公钥加密8. 后发送给客户端；
8. 客户端使用自己的私钥解密加密方案，生成一个随机数R，使用服务器公钥加密后传给服务器端；
9. 服务端用自己的私钥去解密这个密文，得到了密钥R
10. 服务端和客户端在后续通讯过程中就使用这个密钥R进行通信了。

2. 证书准备

从上一章内容中，我们可以总结出来，如果要把整个双向认证的流程跑通，最终需要六个证书文件：

- 服务器端公钥证书：server.crt
- 服务器端私钥文件：server.key
- 根证书：root.crt
- 客户端公钥证书：client.crt
- 客户端私钥文件：client.key
- 客户端集成证书（包括公钥和私钥，用于浏览器访问场景）：client.p12

所有的这些证书，我们都可以向证书机构去申请签发，一般需要收取一定的证书签发费用，此时我们需要选择大型的证书机构去购买。如果只是企业内部使用，不是给公众使用，也可以自行颁发自签名证书，具体的颁发办法请参见本文第四章。

3.在API网关配置HTTPS双向认证

在准备好上一章提到的六个证书文件后，就可以在API网关配置HTTPS的双向认证能力了，在配置之前，我们首先需要在API网关上拥有一个分组，并且在分组下绑定好了您的域名。本节介绍下将域名对应的服务器证书、根证书绑定到API网关来实现HTTPS双向认证的能力。

步骤1. 进入分组详情页面，找到要绑定的域名，点击域名对应的“选择证书”链接；

步骤2. 进入选择证书子页面，选择“手动添加证书”链接；

步骤3. 在手动添加证书页面，将第二章中说道的三个证书分别填写到本页面中：

- 服务器端公钥证书（server.crt）的内容填写到“证书内容”文本框中；
- 服务器端私钥文件（server.key）的内容填写到“私钥”文本框中；

- 根证书（root.crt）的内容填写到“根证书”的文本框中；

通过以上三个步骤就可以在API网关完成配置HTTPS双向认证的配置。

4. 自签名证书

生成这一些列证书之前，我们需要先生成一个CA根证书，然后由这个CA根证书颁发服务器公钥证书和客户端公钥证书。为了验证根证书颁发与验证客户端证书这个逻辑，我们使用根证书生成两套不同的客户端证书，然后同时用两个客户端证书来发送请求，看服务器端是否都能识别。下面是证书生成的内在逻辑示意图：


根证书及秘钥：root.crt，roo

4.1生成自签名根证书

```
（1）创建根证书私钥：
openssl genrsa -out root.key 1024

（2）创建根证书请求文件：
openssl req -new -out root.csr -key root.key
后续参数请自行填写，下面是一个例子：
Country Name (2 letter code) [XX]:cn
State or Province Name (full name) []:bj
Locality Name (eg, city) [Default City]:bj
Organization Name (eg, company) [Default Company Ltd]:alibaba
Organizational Unit Name (eg, section) []:test
Common Name (eg, your name or your servers hostname) []:root
Email Address []:a.alibaba.com
A challenge password []:
An optional company name []:

（3）创建根证书：
openssl x509 -req -in root.csr -out root.crt -signkey root.key -CAcreateserial -days 3650
```

在创建证书请求文件的时候需要注意三点，下面生成服务器请求文件和客户端请求文件均要注意这三点： 根证书的Common Name填写root就可以，所有客户端和服务器端的证书这个字段需要填写域名，一定要注意的是，根证书的这个字段和客户端证书、服务器端证书不能一样； 其他所有字段的填写，根证书、服务器端证书、客户端证书需保持一致 最后的密码可以直接回车跳过。

经过上面三个命令行，我们最终可以得到一个签名有效期为10年的根证书root.crt，后面我们可以用这个根证书去颁发服务器证书和客户端证书。

4.2 生成自签名服务器端证书

```
（1）生成服务器端证书私钥：
openssl genrsa -out server.key 1024

（2） 生成服务器证书请求文件，过程和注意事项参考根证书，本节不详述：
openssl req -new -out server.csr -key server.key

（3） 生成服务器端公钥证书
openssl x509 -req -in server.csr -out server.crt -signkey server.key -CA root.crt -CAkey
root.key -CAcreateserial -days 3650
```

经过上面的三个命令，我们得到：

server.key：服务器端的秘钥文件 server.crt：有效期十年的服务器端公钥证书，使用根证书和服务器端私钥文件一起生成

4.3 生成自签名客户端证书

```
（1）生成客户端证书秘钥：
openssl genrsa -out client.key 1024
```

```
openssl genrsa -out client2.key 1024

（2） 生成客户端证书请求文件，过程和注意事项参考根证书，本节不详述：
openssl req -new -out client.csr -key client.key
openssl req -new -out client2.csr -key client2.key

（3） 生客户端证书
openssl x509 -req -in client.csr -out client.crt -signkey client.key -CA root.crt -CAkey root.
key -CAcreateserial -days 3650
openssl x509 -req -in client2.csr -out client2.crt -signkey client2.key -CA root.crt -CAkey
root.key -CAcreateserial -days 3650


（4） 生客户端p12格式证书，需要输入一个密码，选一个好记的，比如123456
openssl pkcs12 -export -clcerts -in client.crt -inkey client.key -out client.p12
openssl pkcs12 -export -clcerts -in client2.crt -inkey client2.key -out client2.p12
```

重复使用上面的三个命令，我们得到两套客户端证书： client.key / client2.key： 客户端的私钥文件 client.crt / client2.key： 有效期十年的客户端证书，使用根证书和客户端私钥一起生成 client.p12/client2.p12：客户端p12格式，这个证书文件包含客户端的公钥和私钥，主要用来给浏览器访问使用

5. 验证

使用curl加上证书路径，可以直接测试Nginx的HTTPS双向认证是否配置成功。下面我们测试三个用例：

- 使用client.crt / client.key这一套客户端证书来调用服务器端
- 使用client.crt2 / client2.key这一套客户端证书来调用服务器端
- 不使用证书来调用服务器端

下面是三个用例的测试结果：

5.1 带证书的成功调用：

```
#--cert指定客户端公钥证书的路径
#--key指定客户端私钥文件的路径
#-k不校验证书的合法性，因为我们用的是自签名证书，所以需要加这个参数
#可以使用-v来观察具体的SSL握手过程

curl --cert ./client.crt --key ./client.key https://integration-fred2.fredhuang.com -k -v
* Rebuilt URL to: https://47.93.245.203/
*   Trying 47.93.245.203...
* TCP_NODELAY set
* Connected to 47.93.245.203 (47.93.245.203) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* Cipher selection: ALL:!EXPORT:!EXPORT40:!EXPORT56:!aNULL:!LOW:!RC4:@STRENGTH
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/cert.pem
  CApath: none
```

```
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Request CERT (13):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Certificate (11):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS handshake, CERT verify (15):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS change cipher, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* ALPN, server accepted to use http/1.1
* Server certificate:
*  subject: C=CN; ST=BJ; L=BJ; O=Alibaba; OU=Test; CN=integration-fred2.fredhuang.com;
emailAddress=a@alibaba.com
*  start date: Nov  2 01:01:34 2019 GMT
*  expire date: Oct 30 01:01:34 2029 GMT
*  issuer: C=CN; ST=BJ; L=BJ; O=Alibaba; OU=Test; CN=root; emailAddress=a@alibaba.com
*  SSL certificate verify result: unable to get local issuer certificate (20), continuing
anyway.
> GET / HTTP/1.1
> host:integration-fred2.fredhuang.com
> User-Agent: curl/7.54.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: nginx/1.17.5
< Date: Sat, 02 Nov 2019 02:39:43 GMT
< Content-Type: text/html
< Content-Length: 612
< Last-Modified: Wed, 30 Oct 2019 11:29:45 GMT
< Connection: keep-alive
< ETag: "5db97429-264"
< Accept-Ranges: bytes
<
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

```
* Connection #0 to host 47.93.245.203 left intact
```

使用client2.crt / client2.key这一套客户端证书来调用服务器端

```
curl --cert ./client2.crt --key ./client2.key https://integration-fred2.fredhuang.com -k
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

5.2 不带证书的调用

```
curl https://integration-fred2.fredhuang.com -k
<html>
<head><title>400 No required SSL certificate was sent</title></head>
<body>
<center><h1>400 Bad Request</h1></center>
<center>No required SSL certificate was sent</center>
<hr><center>nginx/1.17.5</center>
</body>
</html>
```

三个用例都符合预期，从第一个测试日志中，我们可以看到，整个通信过程较长，客户端验证服务器端的证书，客户端也将自己的证书上传到服务器端进行验证。使用根证书颁发的两个客户端证书都可以正常发起双向HTTPS认证的调用。没有带客户端证书的调用会被服务器端拒绝服务。

6. 使用Java调用

由于使用的是自签名证书，使用ApacheHttpClient去调用的话，需要将服务器证书加入可信任证书库中，才能成功调用，也可以在代码中简单忽略证书。

```
cd $JAVA_HOME
```

```
sudo ./bin/keytool -import -alias ttt -keystore cacerts -file /Users/fred/temp/cert5/
server.crt
```

将服务器端公钥证书设置为可信证书后，使用以下代码可以直接发起带客户端证书的HTTPS请求：

```
import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.conn.ssl.SSLConnectionSocketFactory;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.ssl.SSLContexts;
import org.apache.http.util.EntityUtils;

import javax.net.ssl.SSLContext;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.security.KeyStore;

public class HttpClientWithClientCert {

   private final static String PFX_PATH = "/Users/fred/temp/cert5/client.p12";    //客户端证
书路径
   private final static String PFX_PWD = "123456";    //客户端证书密码

   public static String sslRequestGet(String url) throws Exception {
      KeyStore keyStore = KeyStore.getInstance("PKCS12");
      InputStream instream = new FileInputStream(new File(PFX_PATH));
      try {
         keyStore.load(instream, PFX_PWD.toCharArray());
      } finally {
         instream.close();
      }
      SSLContext sslcontext = SSLContexts.custom().loadKeyMaterial(keyStore, PFX_PWD.
toCharArray()).build();
      SSLConnectionSocketFactory sslsf = new SSLConnectionSocketFactory(sslcontext
            , new String[] { "TLSv1" }    // supportedProtocols ,这里可以按需要设置
            , null    // supportedCipherSuites
            , SSLConnectionSocketFactory.getDefaultHostnameVerifier());

      CloseableHttpClient httpclient = HttpClients.custom().setSSLSocketFactory(sslsf).build
();
      try {
         HttpGet httpget = new HttpGet(url);
         //httpget.addHeader("host", "integration-fred2.fredhuang.com");// 设置一些
heander等
         CloseableHttpResponse response = httpclient.execute(httpget);
         try {
            HttpEntity entity = response.getEntity();
            String jsonStr = EntityUtils.toString(response.getEntity(), "UTF-8");//返回结果
            EntityUtils.consume(entity);
            return jsonStr;
         } finally {
            response.close();
         }
      } finally {
         httpclient.close();
      }
   }

   public static void main(String[] args) throws Exception {
```

```
        System.out.println(System.getProperty("java.home"));
        System.out.println(sslRequestGet("https://integration-fred2.fredhuang.com"));
    }

}
```

5 VPC访问API网关

内网二级域名使用说明

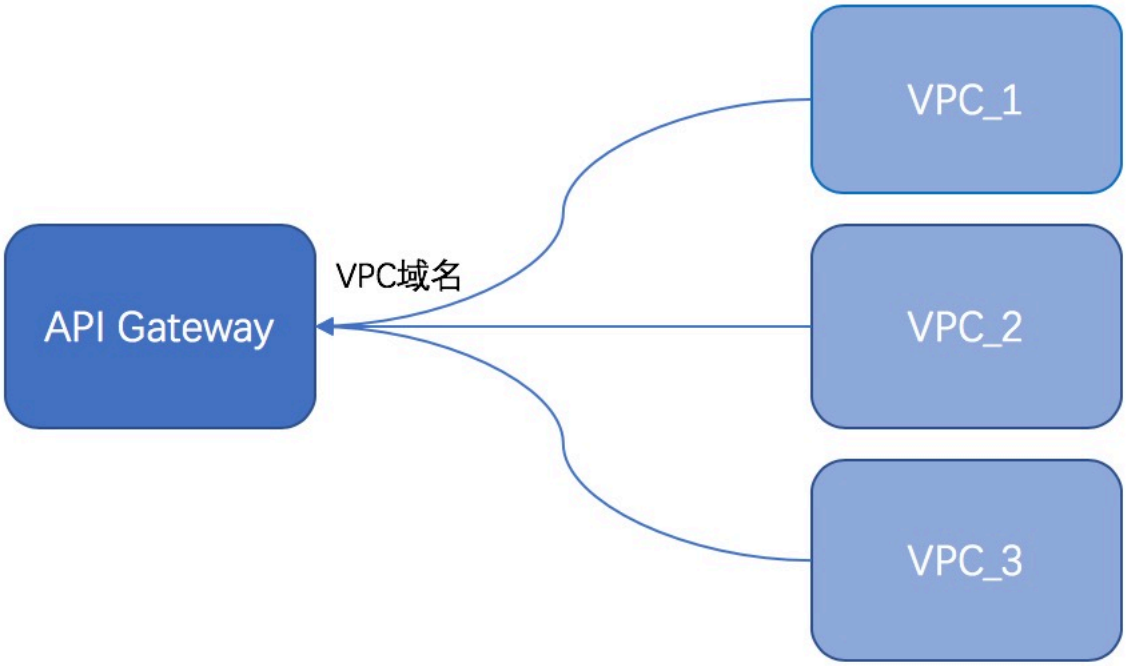
API网关不仅支持用户通过公网访问API，同时也支持内网访问，每个分组可以生成一个“内网VPC二级域名”。

- 内网VPC二级域名可用于VPC直接内网访问该分组的API，没有每天1000次调用限制。
- 内网VPC二级域名不支持HTTPS访问，如果要实现HTTPS访问，请绑定自己域名。
- 绑定自己的域名时，需要先将域名添加CNAME解析到VPC二级域名上，然后再在分组上绑定自己的域名。

不同的实例类型，内网二级域名的开通方法和生效范围不一样，具体参考如下说明。

共享实例内网接入点

API网关共享实例的内网访问支持同region所有用户VPC访问。



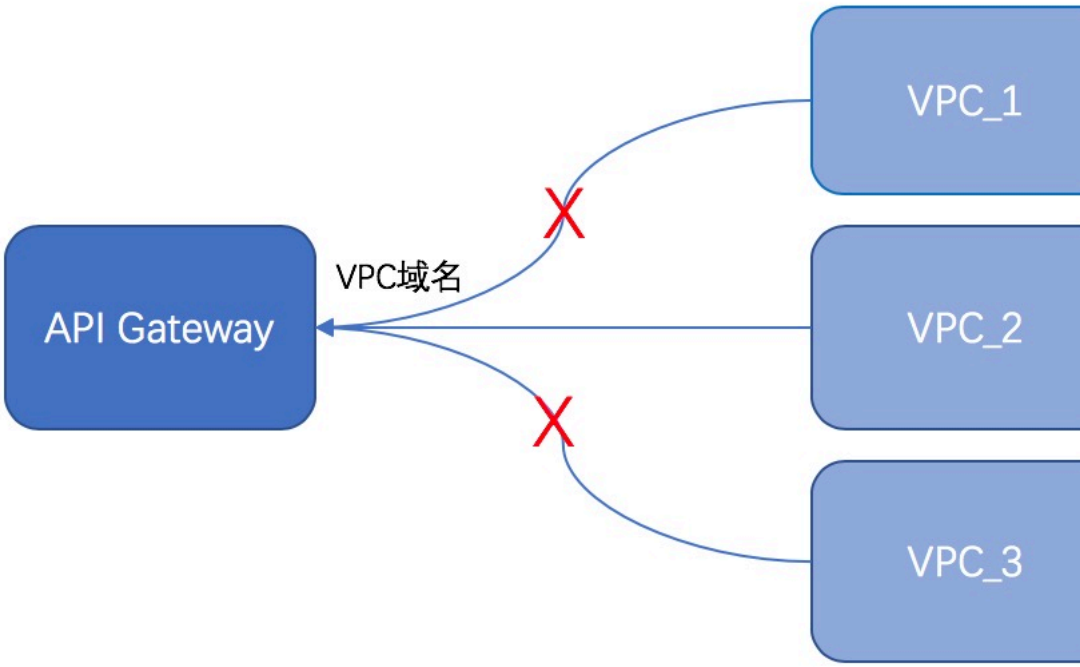
配置方法:

在API网关控制台->开放API->分组管理->分组详情，点击“开通VPC二级域名”，网关会给自动给该分组创建一个内网VPC二级域名，通过该域名可以直接访问API。


API网关
实例
开放API
分组管理
API列表
流量控制
签名秘钥
IP访问控制
插件管理
VPC授权
日志管理
SDK/文档自动生成
分组详情
返回分组列表
基本信息
地域： 华东 2 (上海)
二级域名
实例类型：
共享实例（VPC）
网络访问策略
合法状态： 正常
描述：

专享实例内网接入点

专享实例的VPC内网访问能力仅只对某一个VPC开放，其他VPC则无法通过内网方式访问该实例下的API，这种方式更加安全。



配置方法：

1、请到API网关控制台->实例页面 对应的专享实例上配置允许访问API的VPC。


API网关
实例
开放API
调用API
产品文档
实例列表
华东1（杭州）
华东2（上海）
华北1（青
新加坡
澳大利亚（悉尼）
马来西亚（吉隆
美国（弗吉尼亚）
阿联酋（迪拜）
集团云
专享实例（VPC）： apigateway-cn-459
实例名称
wuling_test
可用区
多可用区 1(g,h)
HTTPS安全策略
HTTPS2_TLS1
入访VPC
绑定到用户VPC
付费方式
按量计费
实例规格
api.s1.small
最大每秒请求数
SLA:
最大公网入访带
最大公网出访带
出口地址
公网:
内网VPC:

2、在该实例下分组详情页面中，开通内网访问，即可通过内网二级域名访问API（或者通过CNAME解析到该二级域名后，绑定自己的域名访问API）。

注意事项：

- 1、专享实例上未配置“入访VPC”，则分组不能开通内网VPC域名。
- 2、当专享实例上“入访VPC”发生变更，该实例下的所有分组的VPC二级域名将重新配置为对新的VPC开放，原来的VPC就不能再访问该API了。

- 3、当分组从共享实例（经典或者VPC）迁移到专享实例时，如果分组上开通了内网VPC域名，需要先开通专享实例的入访VPC后，才能完成实例的迁移。**请注意：此时通过内网访问API，只能从入访VPC发起访问**。

6 函数计算内网访问API网关

本文介绍在函数计算中如何通过VPC访问API网关，包括两种情况：在同一Region，以及在跨Region情况下如何访问。

1 概述

API网关即能够和函数计算集成，构建起serverless架构，同时在实际场景中，也会出现在函数中需要调用API网关上发布的API，同时很多时候出于安全考虑，往往会希望函数从内网能够访问API。因此本文着重介绍两个场景的实现方式：

- 在同一Region内，函数计算如何内网访问API网关
- 跨Region情况下，函数计算如何内网访问API网关

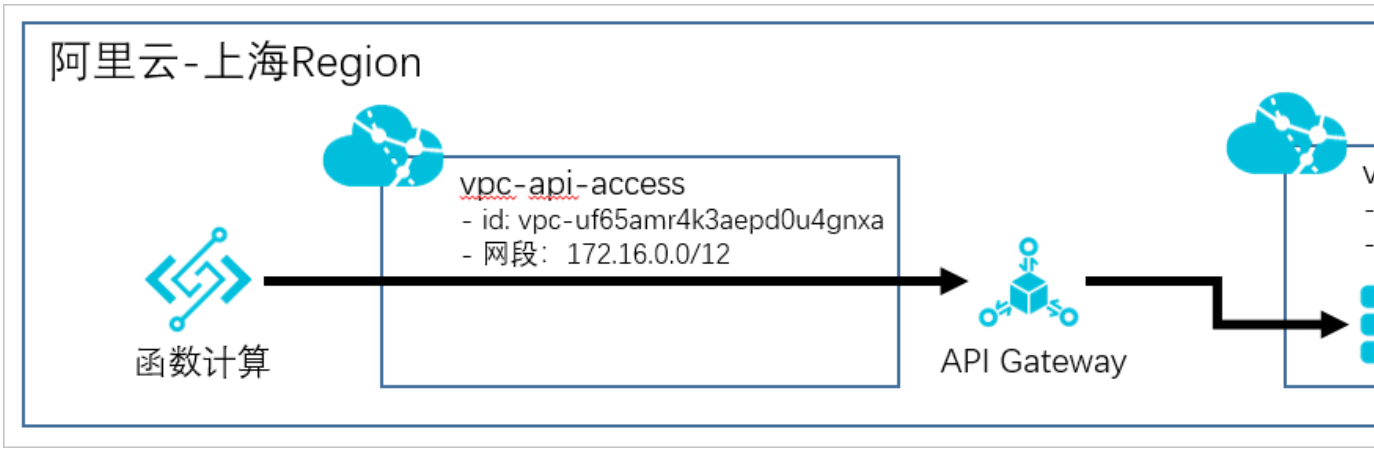
无论是哪种场景，主要的配置原则如下：

- 需要基于VPC实现内网访问；
- API网关的内网访问，需要设定一个VPC允许API网关接入，具体过程详见 VPC访问API网关；
- 函数计算的内网访问，也需要通过VPC实现，具体过程详见 配置VPC功能 。

2 场景1 同一Region内访问

步骤1: 准备工作

构建如下图所示的结构：



说明：

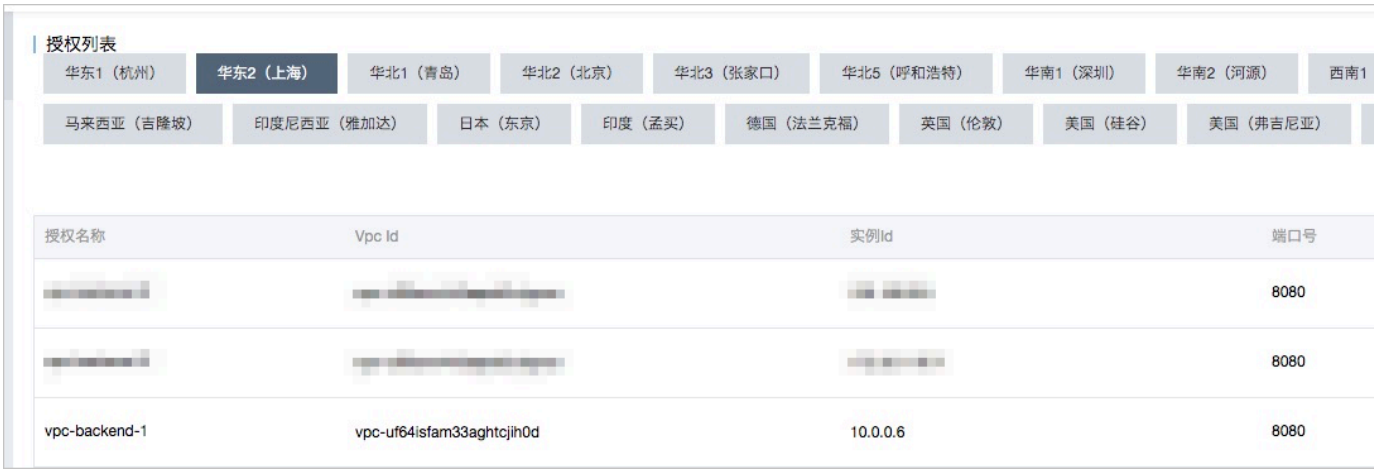
- 在上海Region内创建2个VPC，分别为vpc-api-access和vpc-backend-1；
- 在上海Region内创建了1个API网关专享实例；

- 在vpc-backend-1中，创建一个ecs实例做为API网关的后端服务，此ecs实例提供了http服务接口可对外访问，且对外可以访问的http服务地址为 http://localhost:8080/web/cloudapi 。同时为此ecs实例也设置了对应的安全组，允许API网关的访问。

步骤2：配置后端服务类型为VPC的API

此步骤的详细过程可参见 专有网络 VPC 环境开放 API，配置要点如下：

- 创建VPC授权，创建成功后如下图所示：



- 创建后端服务为VPC的API，为了方便后续的调用测试，API使用"无认证"方式，如下图所示：

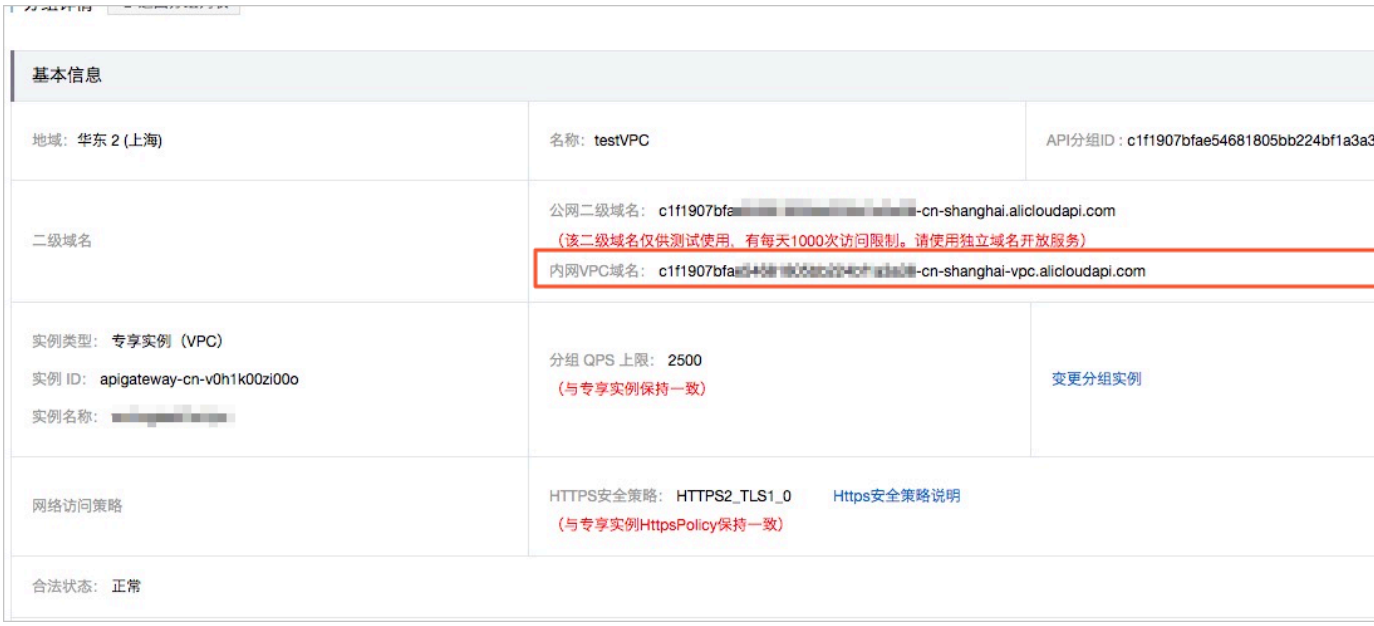
- API保存完后，需要进行发布，本例为了测试方便，发布到“线上”环境。

步骤3：配置VPC到API网关的内网访问权限

在专享实例页面。点击“入访VPC”后的选择VPC，选择vpc-api-access的vpc id，表示可以通过vpc-api-access内网访问到API网关。

步骤4：开通API分组的内网域名

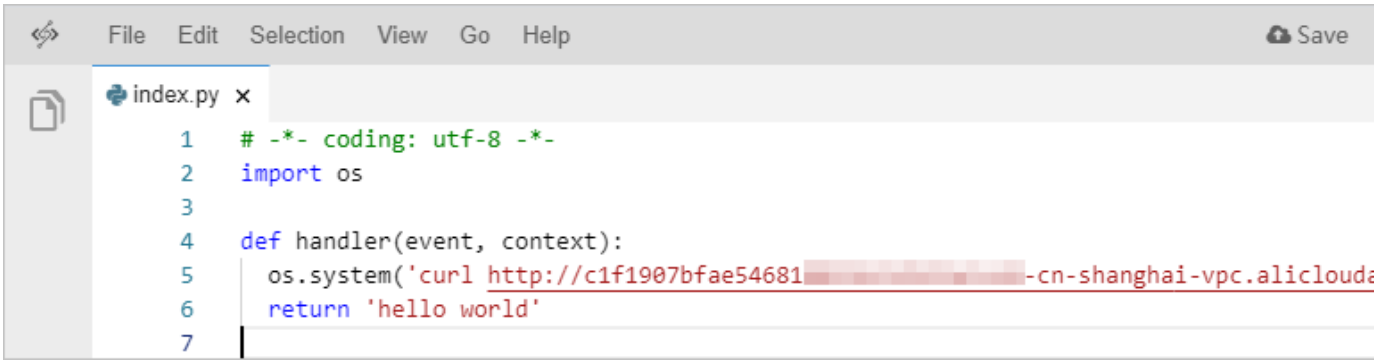
在分组详情页面，开通内网二级域名，点击开通后，API网关会给分组分配一个内网VPC二级域名。该域名可以直接调用该分组下的API。



注意：API分组默认开通互联网访问，您可以根据业务情况通过关闭公网二级域名来停止互联网访问，但注意禁止后，将不能通过API网关控制台进行在线调试。

步骤5：新建函数

在函数计算中，创建应用，并创建运行环境为python的函数，函数内容如下所示：



函数中仅是使用curl去访问vpc二级域名下的API，如果此时执行，将无法访问。

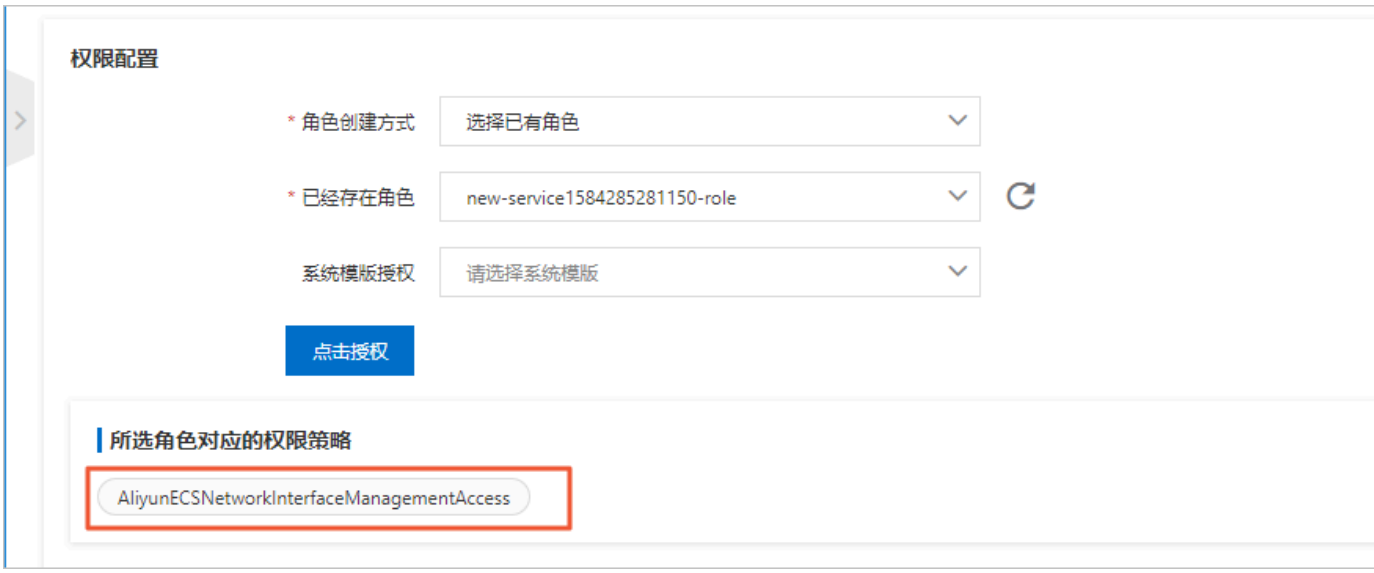
步骤6：配置函数计算的VPC访问

首先需要在vpc-api-access中再创建一个vswitch，用于函数计算的接入，如下图所示：

其次在函数计算控制台中，在上一步骤中新建应用的配置服务菜单中进行配置，具体过程以函数计算的文档为准 配置VPC功能。

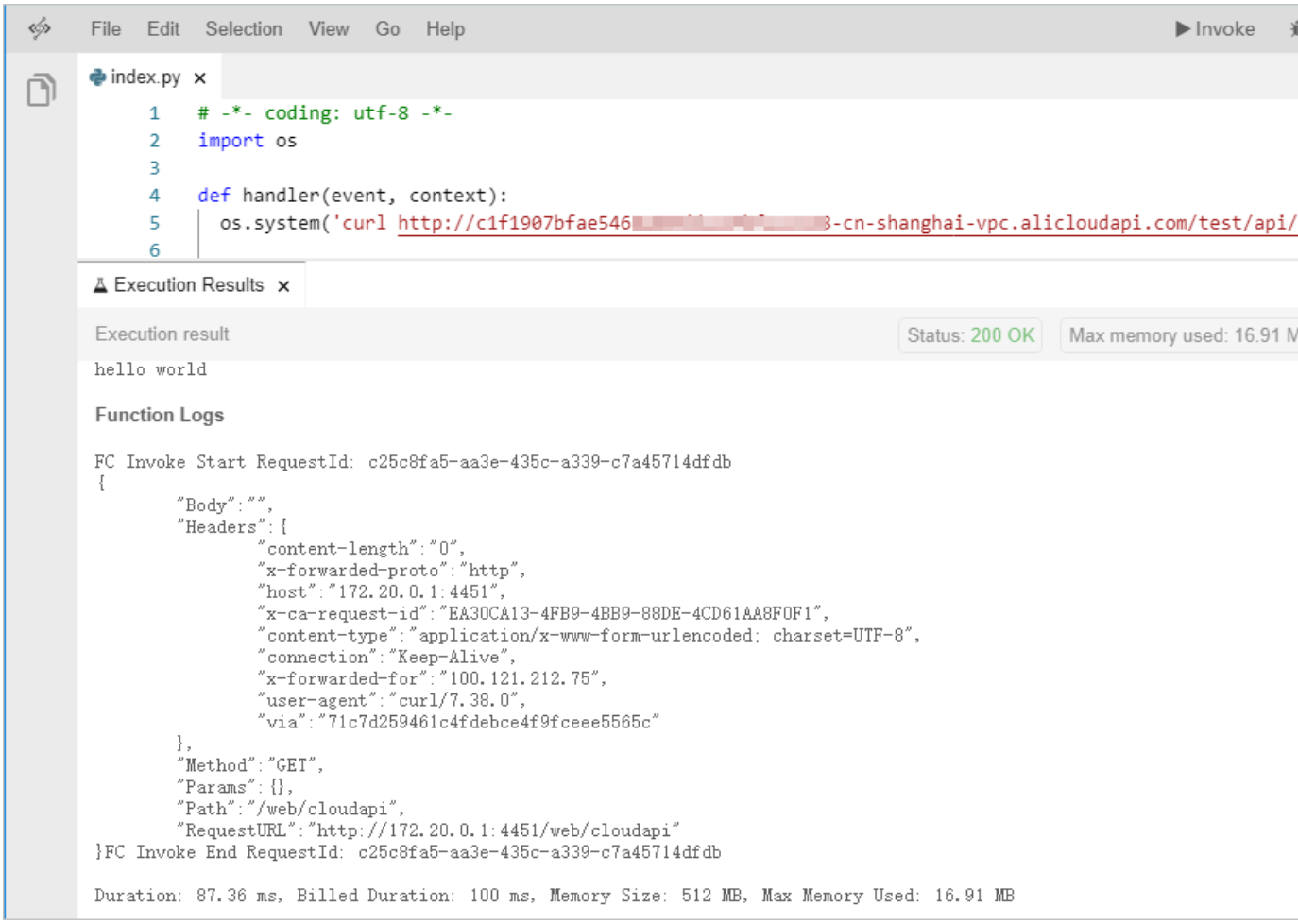
在专有网络配置中，专有网络选择vpc-api-access，交换机选择本步骤创建的vswitch。

权限配置中，需要新增一个角色，系统模板授权选择AliyunECSNetworkInterfaceManagementAccess，按控制台操作向导执行完后如下图所示：



步骤7：执行函数进行测试

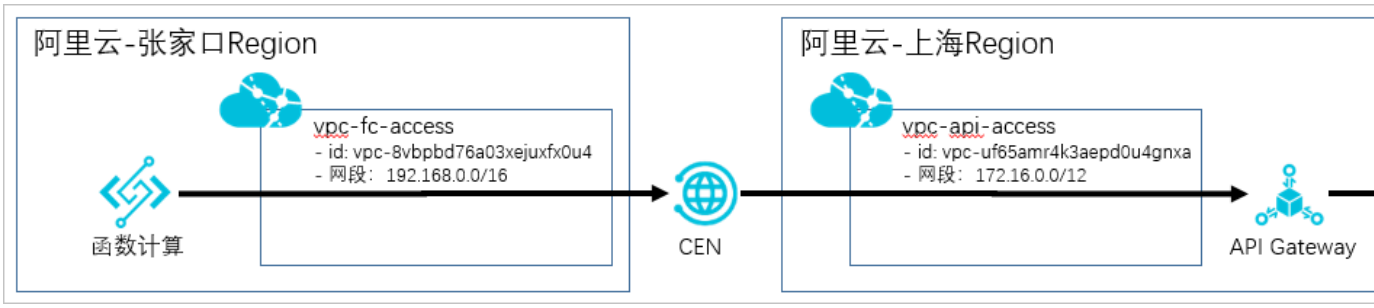
执行后可以看到函数计算已经可以通过vpc二级域名访问到API。



3 场景2 跨Region访问

步骤1: 准备工作

构建如下图所示的结构：



说明：

- 张家口Region内，创建1个函数计算应用，用于本例中发起API调用请求；创建1个VPC（vpc-fc-access），用于帮助函数计算接入云企业网（CEN）；
- 上海Region内，1个API网关专享实例；同时1个VPC（vpc-api-access），用于帮助API网关接入云企业网（CEN）；

- 上海Region内，创建VPC（vpc-backend-1），并创建一个ecs实例做为API网关的后端服务，此ecs实例提供了http服务接口可对外访问，且对外可以访问的http服务地址为 http://localhost:8080/web/cloudapi 。同时为此ecs实例也设置了对应的安全组，允许API网关的访问。

步骤2：创建云企业网

首先创建一个云企业网（CEN），打通上海VPC(vpc-api-access)和张家口的VPC(vpc-fc-access)，实现内网互通。关于CEN的更多配置方式，请详见 云企业网帮助文档 。先通过 云企业网控制台 创建一个云企业网实例，多次加载网络实例，将vpc-api-access，vpc-fc-access都添加到云企业网中，完成后如下图所示。

步骤3：配置带宽

购买一个带宽包，作为云企业网内通信需要。本例购买了一个最低的2M的带宽，您可以根据实际需要按需购买。

配置跨地域带宽设置，指定的互通地域配置带宽值，还可以将1个带宽包拆分到多个互通地域中。

步骤4：配置跨VPC路由

本步骤需要给CEN团队提工单。注意按照 ResolveAndRouteServiceInCen 接口的参数说明，提供配置信息。打通API网关和张家口VPC的互通。

```
AccessRegionIds.1=cn-zhangjiakou
AccessRegionIds.2=cn-shanghai
CenId=cen-uggzcthgz7cwsl7prr     #云企业网的实例ID
Host=100.104.255.128/26          #通过API网关专享实例内网VPC出口地址
HostRegionId=cn-shanghai
HostVpcId=vpc-uf65amr4k3aepd0u4gnxa    #API网关在上海region，这个是vpc-api-access的VPCID
```

其中API网关专享实例接入内网VPC的出口地址，可以在实例管理中查询到，如下图所示

工单回复配置完成后，在云企业网控制台查看配置的路由，能看到上海和张家口region都加了一些路由策略。而且都有一条自定义路由，是根据上面提供的信息添加的。

步骤5：配置后端服务类型为VPC的API

同 **场景1** 中的 **步骤2：**配置后端服务类型为VPC的API。

步骤6：配置VPC到API网关的内网访问权限

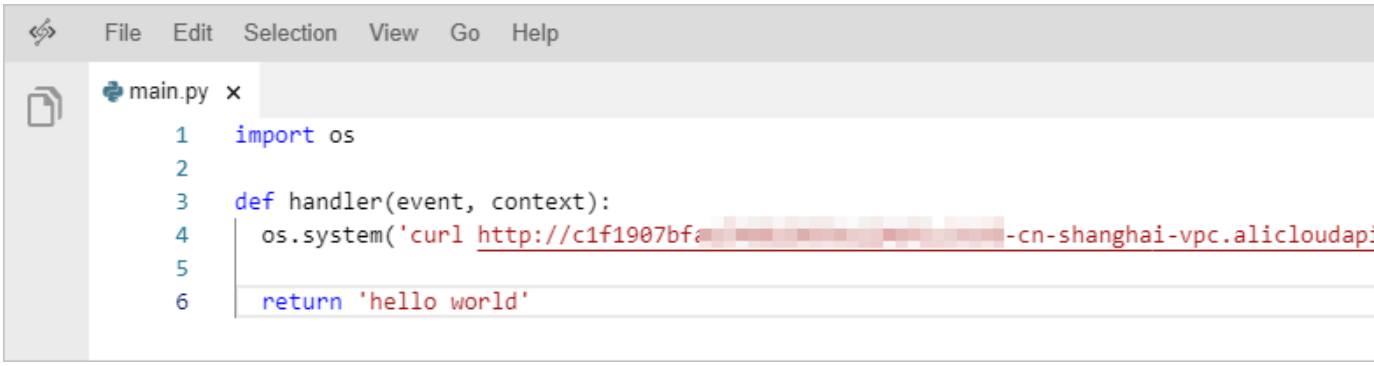
同 **场景1** 中的 **步骤3：**配置VPC到API网关的内网访问权限。

步骤7：开通API分组的内网域名

同 **场景1** 中的 **步骤4：**开通API分组的内网域名。

步骤8：新建函数

在张家口Region的函数计算中，创建应用，并创建运行环境为python的函数，函数内容如下所示：

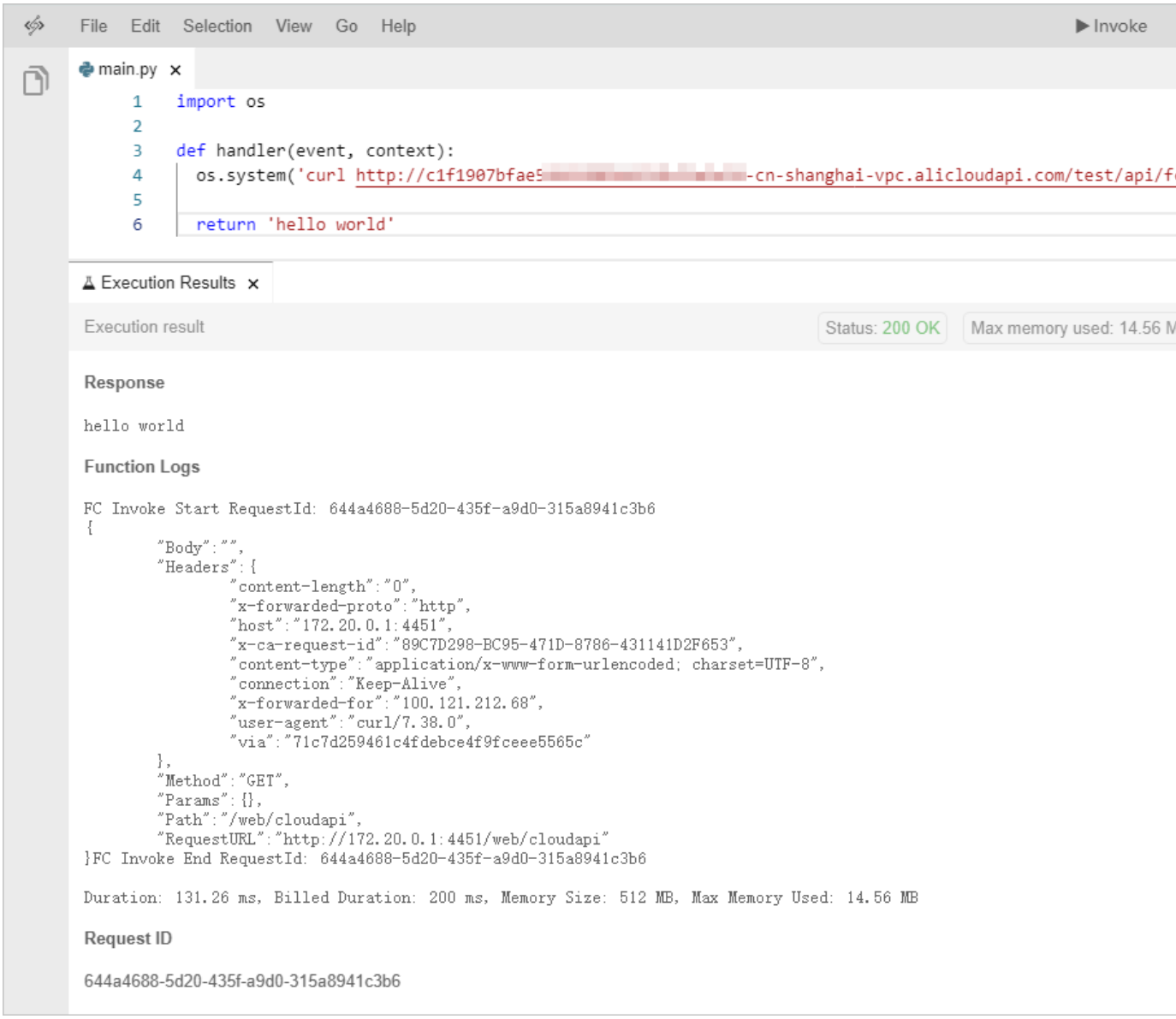


步骤9：配置函数计算的VPC访问

与 **场景1** 的 **步骤6：**配置函数计算的VPC访问 类似，在vpc-fc-access中新建一个vswitch，然后在函数计算控制台中的配置服务菜单中进行配置。

步骤10：执行函数进行测试

执行后可以看到函数计算已经可以通过vpc二级域名访问到API。



4 使用限制

- 仅限API网关专享实例。

7 WAF接入配置

本文主要介绍如何配置WAF，对API网关上发布的API进行增强安全防护。

1 概述

API网关的核心是为API提供认证、防篡改、防重放、参数验证、全链路签名、限流等诸多安全功能，因此针对恶意攻击者精心构造的攻击请求，进行应用层攻击（如OWASP TOP10常见Web攻击等）、暴力破解等情况，您可以考虑接入 云盾Web应用防火墙（简称WAF），从而避免遭到入侵导致数据泄露，更好的保障您的业务安全。

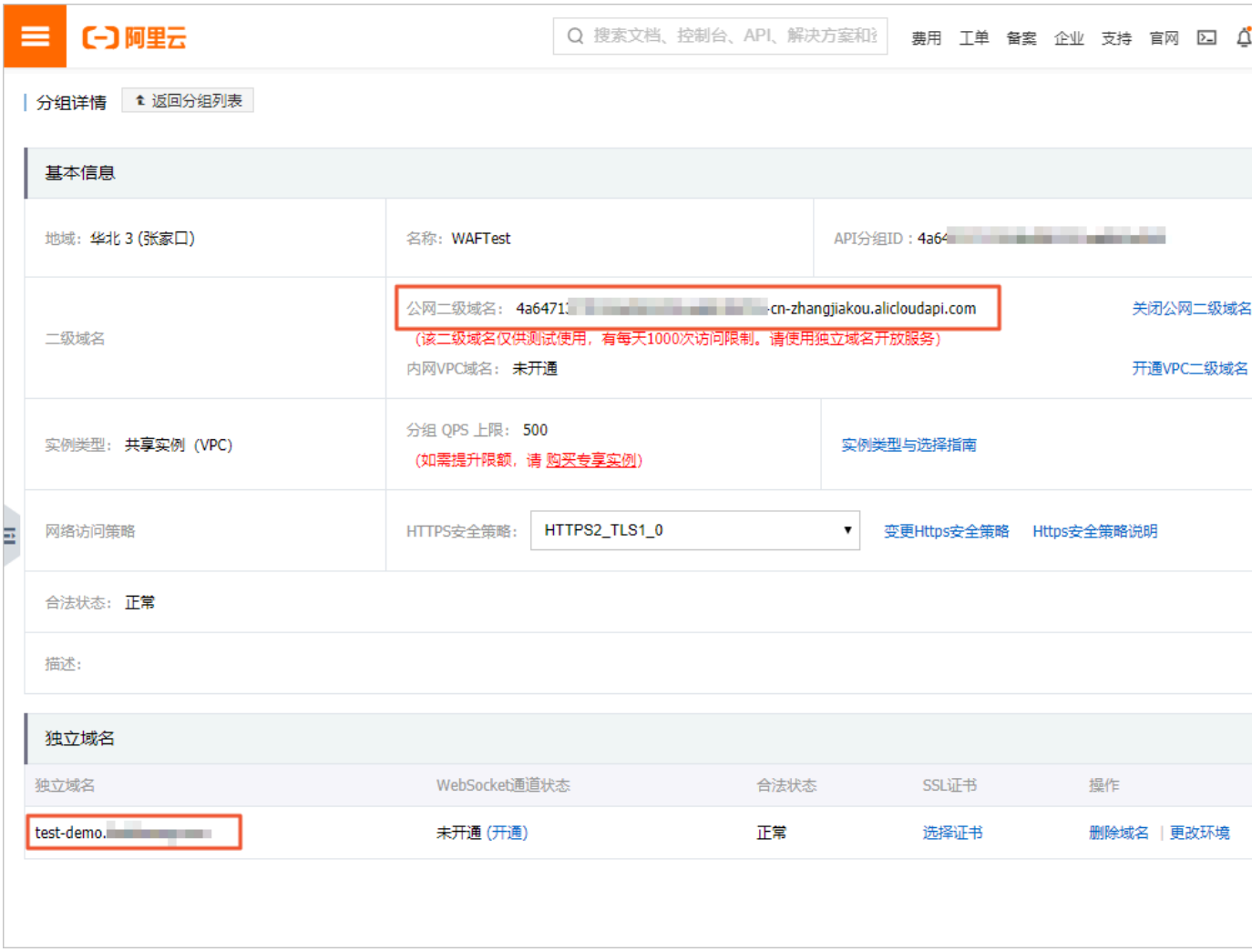
API网关和WAF完全兼容，您可以参考以下步骤为API接入WAF。

2 前提条件

- 开通Web应用防火墙
- 在API网关上已经发布了API

3 操作步骤

步骤1：在API分组上绑定您的域名，操作过程详见 使用HTTPS并用域名访问。绑定成功后如下图所示：



由于后续步骤中还需配置WAF，建议当前阶段您不进行的CNAME配置。

步骤2：在WAF上添加网站。进入 WAF控制台，在 管理 - 网站配置 菜单中 添加站点。

主要的填写信息包括：

- 域名：填写您的域名，需要和 **步骤一** 中API网关分组上绑定的域名一致；
- 协议类型：需要和您在API网关在发布API的协议类型一致；
- 服务器地址：选择“其他地址”，填写API分组为您分配的公网二级域名。

点击下一步，按照WAF的提示，站点添加成功。同时您为您的域名添加CNAME解析记录，逐个完成业务流量的切换。更多关于WAF的安全配置方式，请详见 WAF接入配置最佳实践 。