边缘安全加速平台 EO

工具指南

产品文档

腾讯云

文档目录

工具指南
诊断工具
自助调试

最近更新时间：2023-12-18 14:41:38

功能简介

如果您需要确认当前在 EdgeOne 内配置的节点缓存规则、自定义 Cache Key 等配置是否已针对您的资源生效，EdgeOne 提供了自助调试工具来帮助您获取节点缓存 TTL，资源是否可缓存，Cache key 等信息，方便您进行业务配置调试。开启自助调试后，您可以通过指定的客户端 IP 来发起 URL请求，在请求中携带 `EO-Debug-Headers: all` 头部，即可根据返回的响应头来查看该资源在节点内是否缓存、对应的 Cache Key 值、缓存时间。

```
curl -voa 'http://www.example.com/test.jpg?a=1' -H 'EO-Debug-Headers: all'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0*   Trying
.
* TCP_NODELAY set
* Connected to www.example.com ([redacted]) port 80 (#0)
> GET /test.jpg?a=1 HTTP/1.1
> Host: www.example.com
> User-Agent: curl/7.62.0
> Accept: */*
> EO-Debug-Headers: all
>
< HTTP/1.1 200 OK
< Last-Modified: Wed, 07 Dec 2022 12:32:32 GMT
< Etag: "639087e0-4"
< Server: nginx/1.20.2
< Date: Thu, 29 Jun 2023 12:19:58 GMT
< Content-Type: image/jpeg
< Content-Length: 4
< Accept-Ranges: bytes
< Connection: keep-alive
< EO-LOG-UUID: 6570353728066144953
< EO-Cache-Status: HIT
< EO-Debug-Status: on
< EO-Debug-CacheKey: www.example.com/test.jpg a=1
< EO-Debug-Cacheable: yes
< EO-Debug-CacheTTL: 00d00h10m00s
{ [4 bytes data]
100     4  100     4    0     0     47      0 --:--:-- --:--:-- --:--:--    47
* Connection #0 to host www.example.com left intact
```

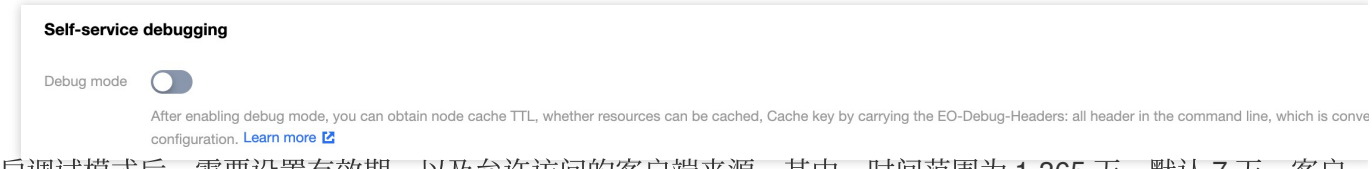
使用场景

若您在控制台规则引擎里配置了较复杂的缓存策略、自定义 cache key，需要验证配置是否生效，可以通过该功能进行验证。

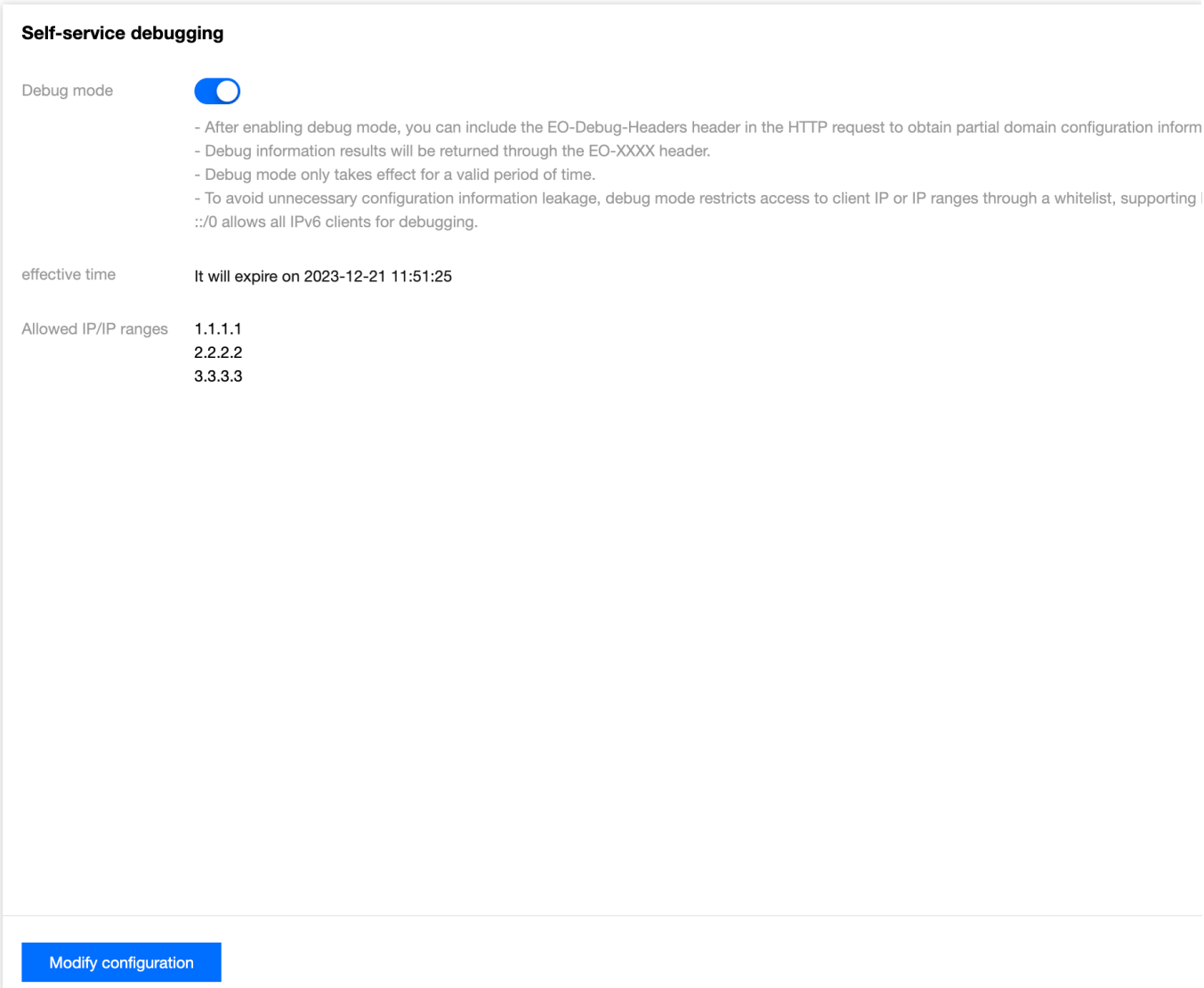
操作步骤

例如：当前站点 `example.com` 下的域名 `www.example.com`，当前已针对 `.jpg` 后缀文件配置需要在 EdgeOne 节点内缓存 600 秒；缓存 Cache Key 配置为保留指定参数 a 作为缓存键。配置完成后，需要验证当前配置是否已生效，可按照如下操作步骤验证：

1. 登录 边缘安全加速平台 EO 控制台，在左侧菜单栏中，单击**站点列表**，在站点列表内单击需配置的**站点**。
2. 在站点详情页面，单击**诊断工具 > 自助调试**。
3. 在自助调试页面，单击 “开关”，开启自助调试功能。



4. 开启调试模式后，需要设置有效期，以及允许访问的客户端来源。其中，时间范围为 1-365 天，默认 7 天。客户端 IP 可输入 100 个，支持填写 IPv4 以及 IPv6 的 IP/IP 段，0.0.0.0/0 表示允许所有 IPv4 客户端进行调试；::/0 表示允许所有 IPv6 客户端进行调试。



5. 单击**保存**，则配置允许的客户端 IP 在有效时间内可进行 debug 调试。

6. 通过以上指定的客户端 IP 来源，在 Mac/Linux 环境下，发起 curl 请求进行验证，例如：`curl -voa 'http://www.example.com/test.jpg?a=1' -H 'EO-Debug-Headers: all'` 。请求结果如下：

```
curl -voa 'http://www.example.com/test.jpg?a=1' -H 'EO-Debug-Headers: all'
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0*   Trying
.
* TCP_NODELAY set
* Connected to www.example.com (                ) port 80 (#0)
> GET /test.jpg?a=1 HTTP/1.1
> Host: www.example.com
> User-Agent: curl/7.62.0
> Accept: */*
> EO-Debug-Headers: all
>
< HTTP/1.1 200 OK
< Last-Modified: Wed, 07 Dec 2022 12:32:32 GMT
< Etag: "639087e0-4"
< Server: nginx/1.20.2
< Date: Thu, 29 Jun 2023 12:19:58 GMT
< Content-Type: image/jpeg
< Content-Length: 4
< Accept-Ranges: bytes
< Connection: keep-alive
< EO-LOG-UUID: 6570353728066144953
< EO-Cache-Status: HIT
< EO-Debug-Status: on
< EO-Debug-CacheKey: www.example.com/test.jpg a=1
< EO-Debug-Cacheable: yes
< EO-Debug-CacheTTL: 00d00h10m00s
{ [4 bytes data]
100     4  100     4    0     0     47      0 --:--:-- --:--:-- --:--:--    47
* Connection #0 to host www.example.com left intact
```

在响应头中，可以看到该请求对应的 Cache Key、缓存状态、缓存时间，与示例中的配置一致，即当前配置已生效。

相关参考

在开启自助调试模式下时，响应的 debug 头部说明如下：

头部名称	功能说明	返回值的含义说明
EO-Debug-Status	用于标识自助调试模式是否开启。	on：开启状态，且请求客户端 IP 在白名单内&请求时间在有效期内； off：关闭状态，或开启状态但请求时间超出有效期； forbidden：开启状态，但请求客户端 IP 不在白名单内。
EO-Debug-ClientIp	发起 Debug 请求的 ClientIp。	发起 Debug 请求的客户端 IP。当开启自助调试但 EO-Debug-Status 值为 forbidden 时，可以检查下是否请求的客户端 IP 不在允许的 IP/IP 段范围内。
EO-Debug-	本次请求的 URL，按照配置节点缓	yes：可缓存内容

Cacheable	存 TTL 的配置，最终该请求 URL 资源在 EdgeOne 节点内是否可缓存的状态。	no：不可缓存内容
EO-Debug-CacheKey	本次请求的 URL，按照自定义 Cache key，最终该请求 URL 资源在 EdgeOne 节点内生成的 Cache key。	例如：`www.example.com/test.jpg a=1` ，指该请求 URL 资源在 EdgeOne 内生成的 Cache Key
EO-Debug-CacheTTL	本次请求的 URL，按照配置节点缓存 TTL的配置，最终该请求 URL 资源在 EdgeOne 节点缓存 TTL 时长。	列表值，包括数字和时间单位。d 表示天，h 表示小时，m 表示分钟，s 表示秒，例如： 3d0h0m0s 表示缓存TTL是 3 天； 0d0h5m0s 表示缓存是 5 分钟； 0d0h0m5s 表示缓存是 5 秒。

测速工具
性能监控

最近更新时间：2024-01-25 11:03:17

功能概述

性能监控是一站式前端监控解决方案，该功能为 EdgeOne 与 前端性能监控 的联动功能，专注于 Web 等前端场景监控。用户只需要将 SDK 安装到自己的项目中，通过简单配置，即可全方位守护用户页面质量。实现低成本使用和无侵入的监控，确保页面性能和前端质量的实时可观测。

说明：

性能监控每个应用每天50万免费上报数据量额度，对超过50万上报次数的部分进行计费，该功能费用不计入 EdgeOne 套餐，由前端性能监控侧收取。计费详情请参见 前端性能监控-购买指南。

适用场景

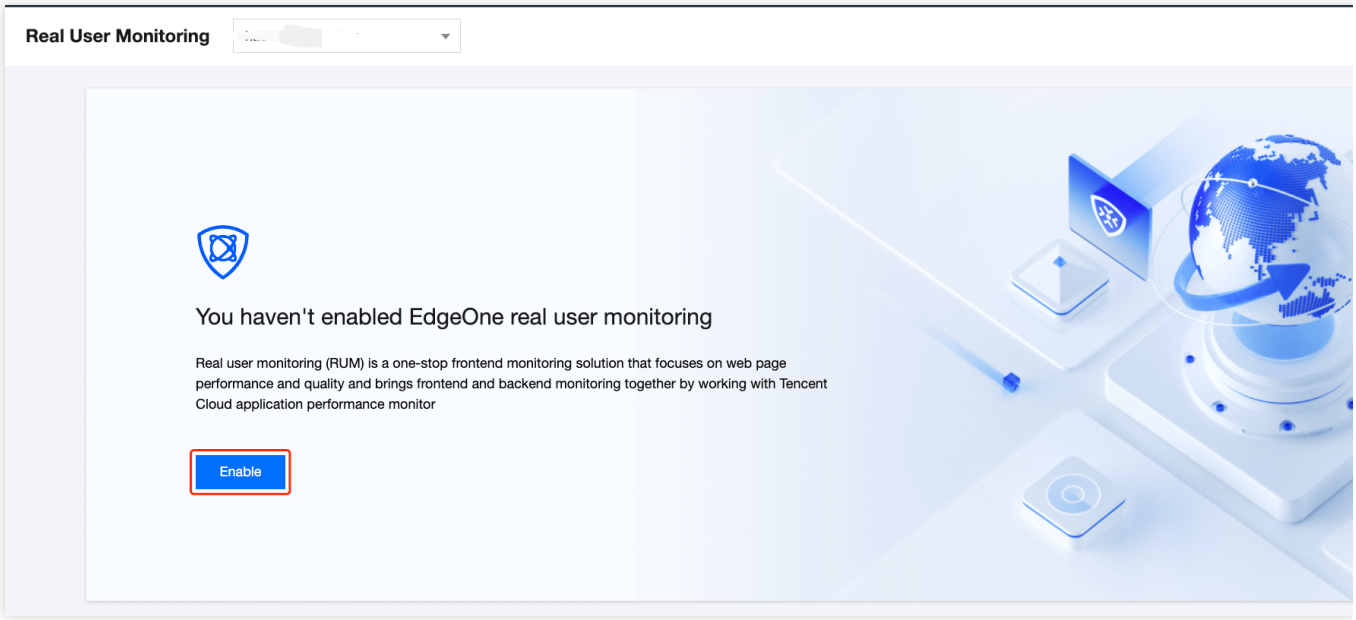
页面性能分析：包括首屏耗时、建立 TCP 连接耗时、TTFB 耗时、SSL 耗时等。同时还支持最新的 Web Vitals（谷歌针对网页加载速度和体验所提出的一套指标） 标准。全方位协助您优化用户体验。

用户访问分析：支持查看业务 PV/UV 数据，每个页面访问的 TOP 数据等，支持通过网络、浏览器、地区等多维度分析用户访问数据，实时了解并分析用户访问情况。

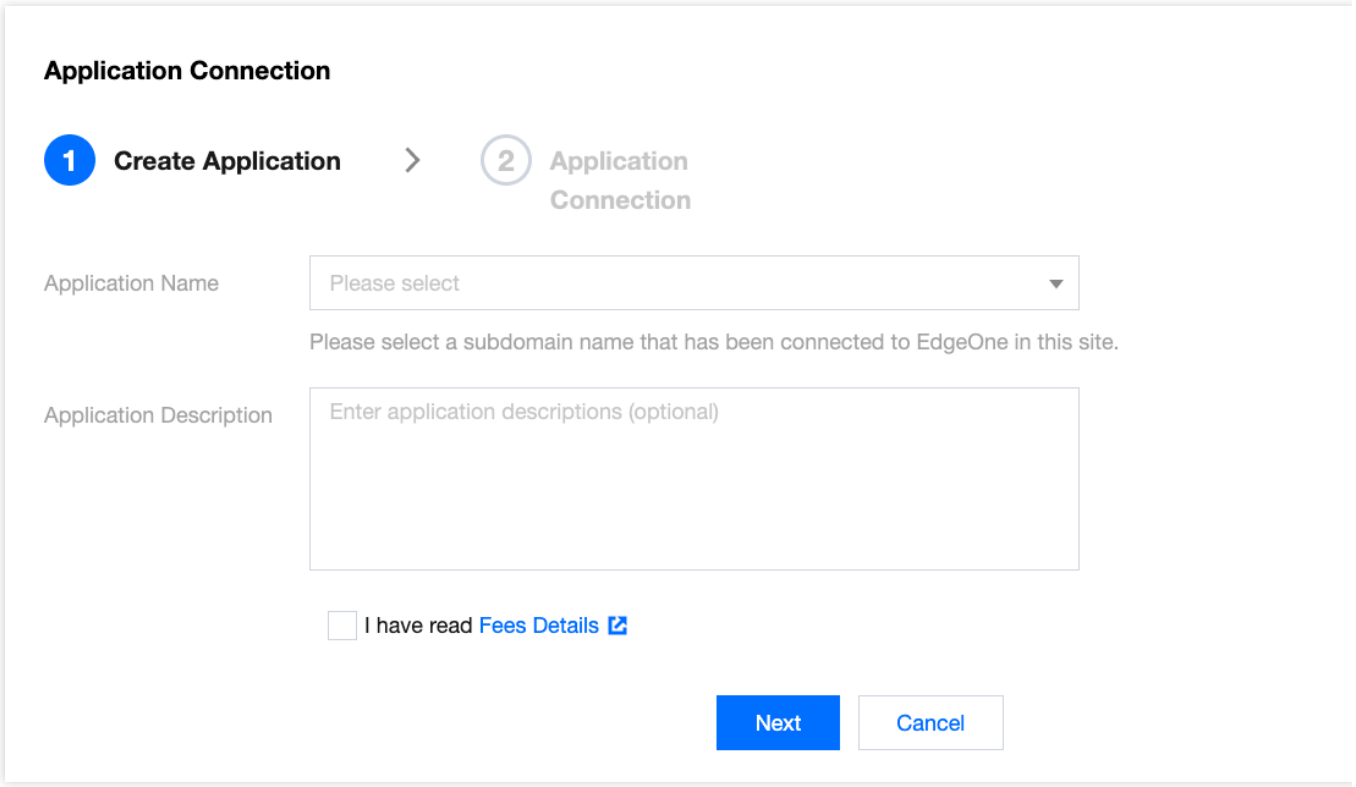
静态资源测速：支持资源测速，包括图片加载耗时和 CDN 资源耗时等。开发者可以查看某个页面下具体使用了哪些资源，每个资源的耗时情况等信息。

操作步骤

1. 登录 边缘安全加速平台控制台，在左侧导览中，单击**测速工具 > 性能监控**。
2. 首次进入性能监控页面时，由于性能监控是与腾讯云前端性能监控联动功能，需要您的单击**开启**，开启对应权限。



3. 在性能监控页面，单击**应用接入**。
4. 在应用接入对话框中，输入应用名称和描述，并勾选“我已知悉费用详情”框，单击**下一步**。

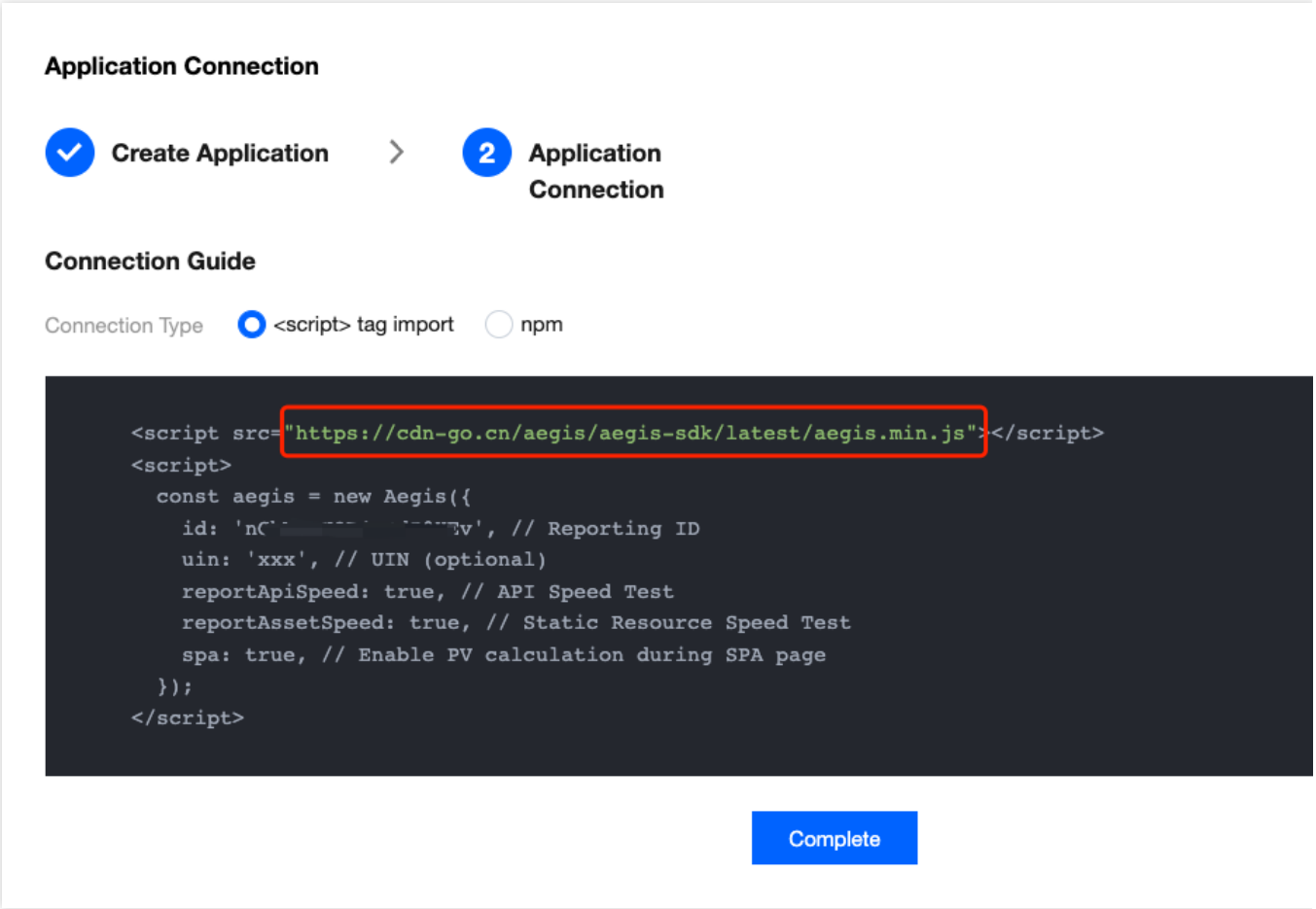


5. 根据接入类型安装 SDK。

`<script>` 标签引入的方式安装 SDK

5.1.1 在接入指引页面，复制提供的 `<script>` 标签代码。

5.1.2 把 `<script>` 标签引入类型下的代码引入到需要监控的网站的 `<head></head>` 标签中即可。



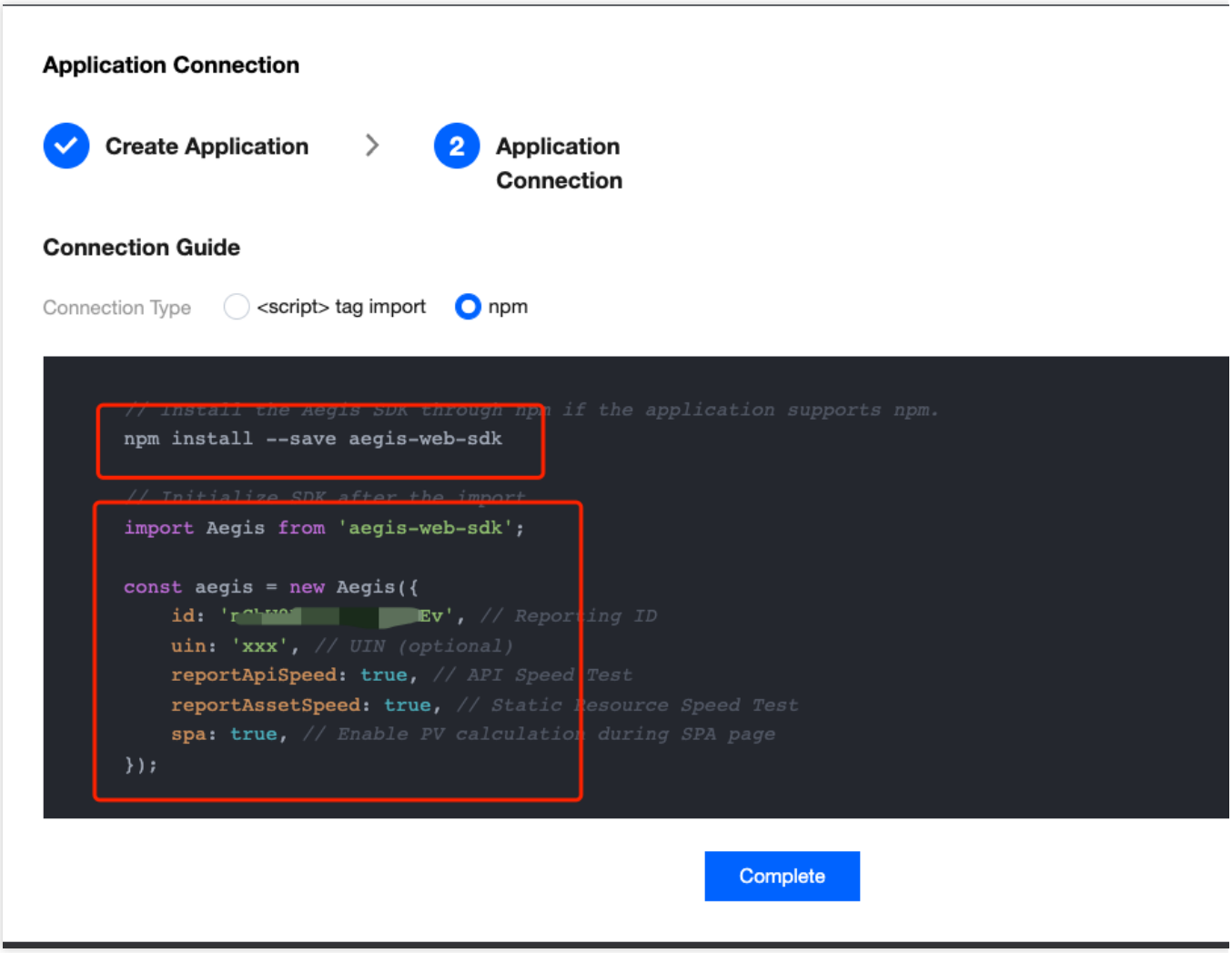
说明：

该接入方式使用 “h3-Q050” 协议，默认 cache-control 为 max-age=666，如果需要修改 cache-control，可以添加参数 max_age，例如：`<script src="https://cdn-go.cn/aegis/aegis-sdk/latest/aegis.min.js?max_age=3600"></script>` 。

npm 方式安装 SDK

5.1 在接入指引页面，复制提供的首行命令，在您的开发环境中引入 aegis sdk。

5.2 引入之后，复制提供的代码，在您的 js 代码中初始化 SDK 即可。



数据监控

完成上述步骤接入后即可以前往页面性能、页面访问和静态资源界面查看相关数据。

页面性能

页面性能模块支持多维度分析页面性能情况，您可以通过性能变化趋势图、页面加载瀑布图、地区视图等维度分析首屏时间、请求响应等页面性能关键指标。详情请参见 页面性能。

页面访问

页面访问用于展示页面访问量情况（UV、PV、WAU、MAU），并支持多维度分析页面访问情况。详情请参见 页面访问。

静态资源

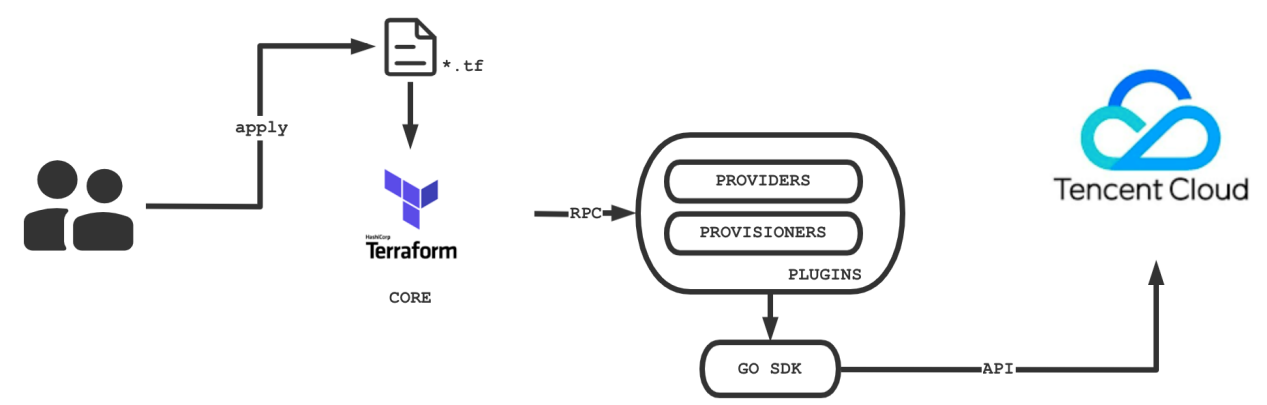
前端 HTML 页面中主要包含的静态资源有：JS 文件、CSS 文件和图片文件，若这些文件加载耗时较长、失败等，将直接对页面造成影响甚至瘫痪，静态资源监控将协助您分析前端静态资源情况。详情请参见 静态资源。

Terraform
Terraform 简介

最近更新时间：2024-01-25 11:20:13

Terraform 是什么？

Terraform 是一款使用 Go 编写、运行在客户端、开源的资源编排工具。基于 HashiCorp Plugin 的架构设计，赋予 Terraform 高度可扩展的特性。目前腾讯云也基于 Terraform Plugin 实现了 TencentCloud Provider，支持通过 Terraform 管理腾讯云上资源。示意图如下：



TencentCloud Provider 基于 tencentcloud-sdk-go 实现，目前已经提供了超过183个 Resource 和158个 Data Source，覆盖计算、存储、网络、容器服务、负载均衡、中间件、数据库、云监控等超过30款产品，已满足众多用户的基本上云需求。
您可通过 EdgeOne 提供的 Terraform 文档 及 Terraform 编写样例，来快速了解并开始使用 Terraform。

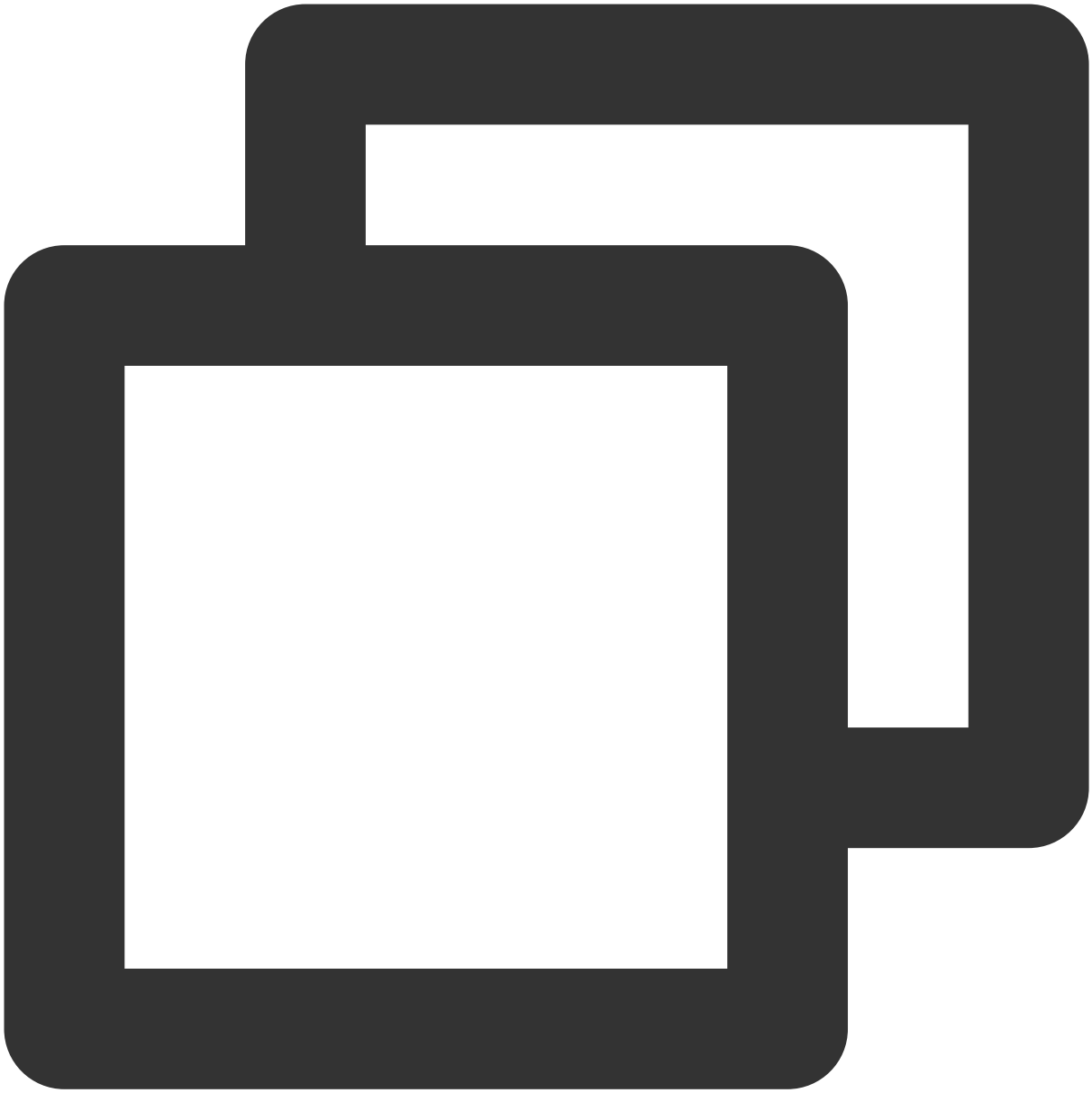
Terraform 优势

多云编排

Terraform 适用于多云方案，您可将相类似的基础结构部署到腾讯云、其他云提供商或本地数据中心。开发人员能够使用相同的工具和相似的配置文件同时管理不同云提供商的资源。

基础设施及代码

基础设施可以使用高级配置语法 HCL 进行描述，使得基础设施能够被代码化和版本化，从而可以进行共享和重复使用。示例如下：



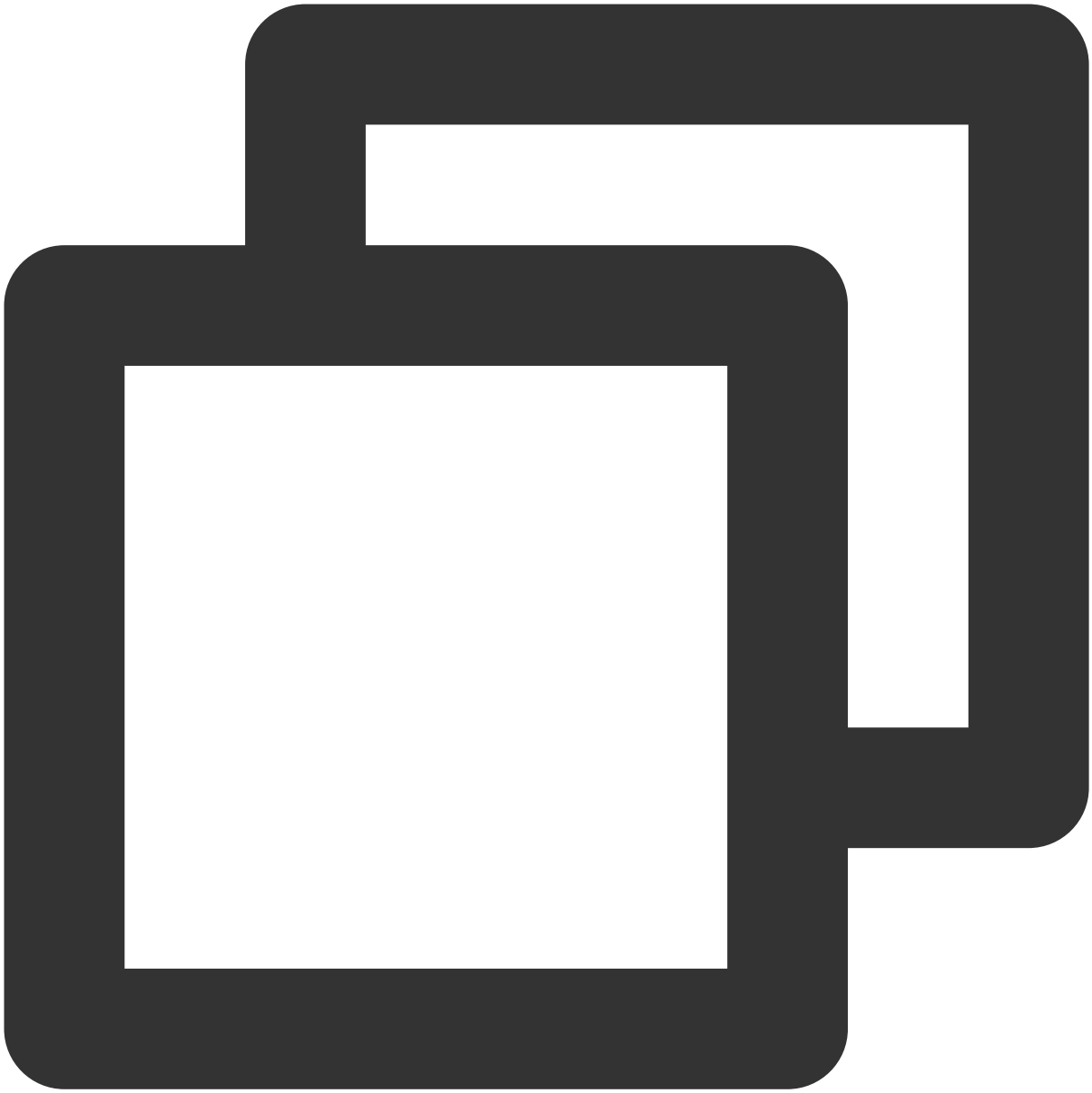
```
# 以 NS 接入方式创建站点 example.com
resource "tencentcloud_teo_zone" "zone" {
  zone_name      = "example.com"
  # 通过 zone_available_plans 查询您可用的套餐信息
  plan_type      = "<your-plan-type>"
  type           = "full"
  paused         = false
  cname_speed_up = "enabled"
}

# 创建 example.com 的 DNS 记录
```

```
resource "tencentcloud_teo_dns_record" "dns_record" {
  zone_id     = tencentcloud_teo_zone.zone.id
  type        = "A"
  name        = "example.com"
  # 开启 CDN 加速服务
  mode        = "proxied"
  content     = "<your-backend-ip>"
  ttl         = 60
}
```

执行计划

Terraform 具备 “Planning” 步骤，执行 terraform plan 命令后 Terraform 会生成执行计划。执行计划会显示当调用 apply 时 Terraform 的状态，您可通过该能力在 Terraform 操作基础设施时避免任何意外。示例如下：



```
Terraform used the selected providers to generate the following execution plan. Res
  + create

Terraform will perform the following actions:

  # tencentcloud_teo_dns_record.dns_record will be created
  + resource "tencentcloud_teo_dns_record" "dns_record" {
      + cname         = (known after apply)
      + content       = "<your-backend-ip>"
      + created_on    = (known after apply)
      + domain_status = (known after apply)
```

```
      + id             = (known after apply)
      + locked         = (known after apply)
      + mode           = "proxied"
      + modified_on    = (known after apply)
      + name           = "example.com"
      + priority       = (known after apply)
      + type           = "A"
      + status         = (known after apply)
      + ttl            = 60
      + zone_id        = (known after apply)
    }

  # tencentcloud_teo_zone.zone will be created
  + resource "tencentcloud_teo_zone" "zone" {
      + area                     = (known after apply)
      + cname_speed_up           = "enabled"
      + cname_status             = (known after apply)
      + created_on               = (known after apply)
      + id                       = (known after apply)
      + modified_on              = (known after apply)
      + name                     = "example.com"
      + name_servers             = (known after apply)
      + original_name_servers    = (known after apply)
      + paused                   = false
      + plan_type                = "sta"
      + status                   = (known after apply)
      + type                     = "full"
      + vanity_name_servers_ips = (known after apply)

      + vanity_name_servers {
          + servers = (known after apply)
          + switch  = (known after apply)
        }
    }

Plan: 2 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------

Note: You didn't use the -out option to save this plan, so Terraform can't guarante
```

自动变更

您可在基础设施上通过最少的人工干预工作，应用复杂的变更集。通过前文中的执行计划及资源拓扑，您可准确获取 Terraform 动态，避免可能的人为错误。

远程状态管理

Terraform 引入了远程状态存储机制 Backend，目前腾讯云可通过 对象存储 COS 来管理用户的 tfState 文件，避免将文件保存在本地，造成丢失。同时，远程存储使多人同时对 Terraform 资源进行管理成为可能。

安装和配置 Terraform

最近更新时间：2024-01-25 11:20:13

本文介绍如何安装和配置 Terraform。

步骤1：安装 Terraform

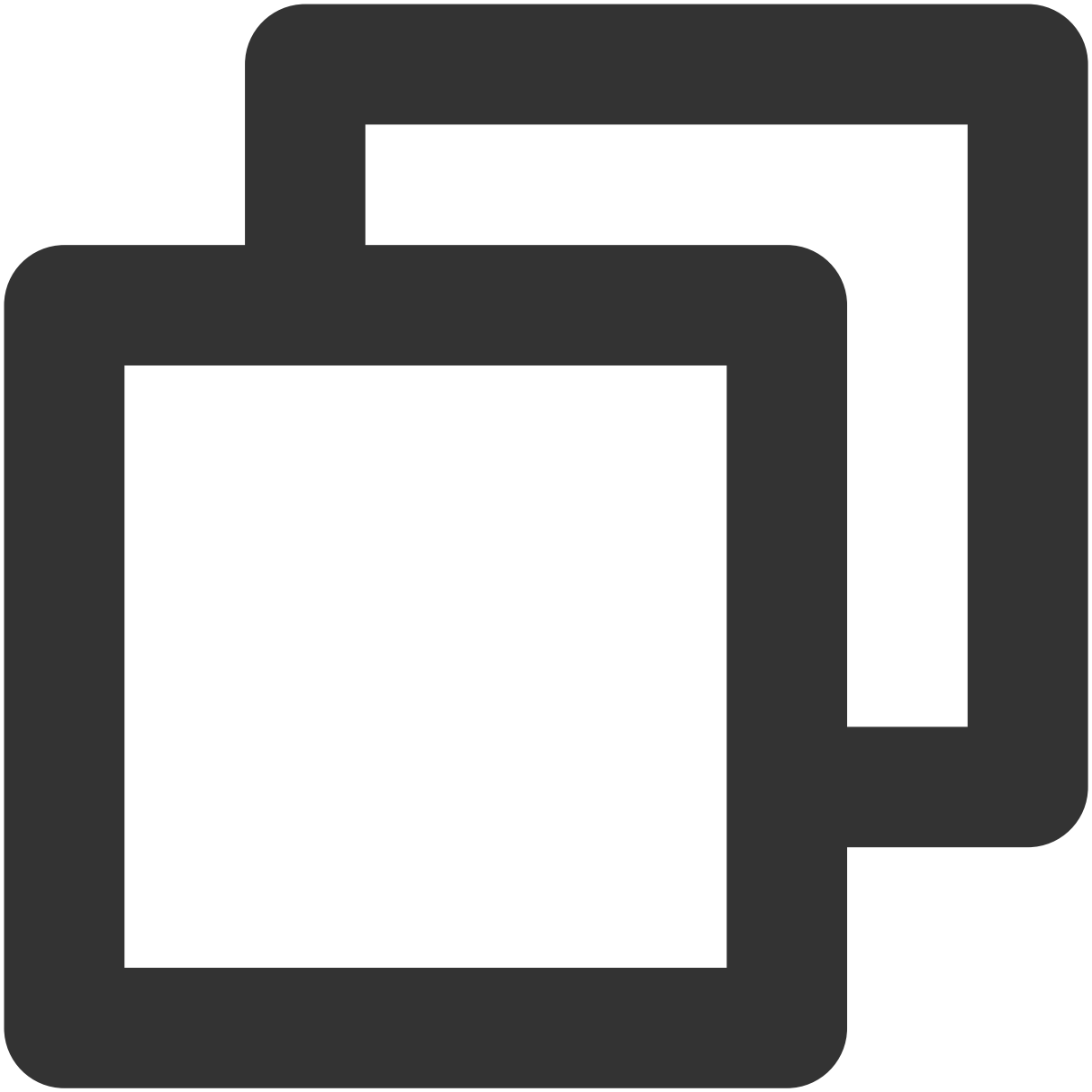
1. 前往 Terraform 官网，使用命令行直接安装 Terraform 或下载二进制安装文件。
2. 解压并配置全局路径。

若您通过命令行安装，则请跳过此步骤。若您通过下载二进制安装文件，则请配置全局路径。

Linux 及 macOS

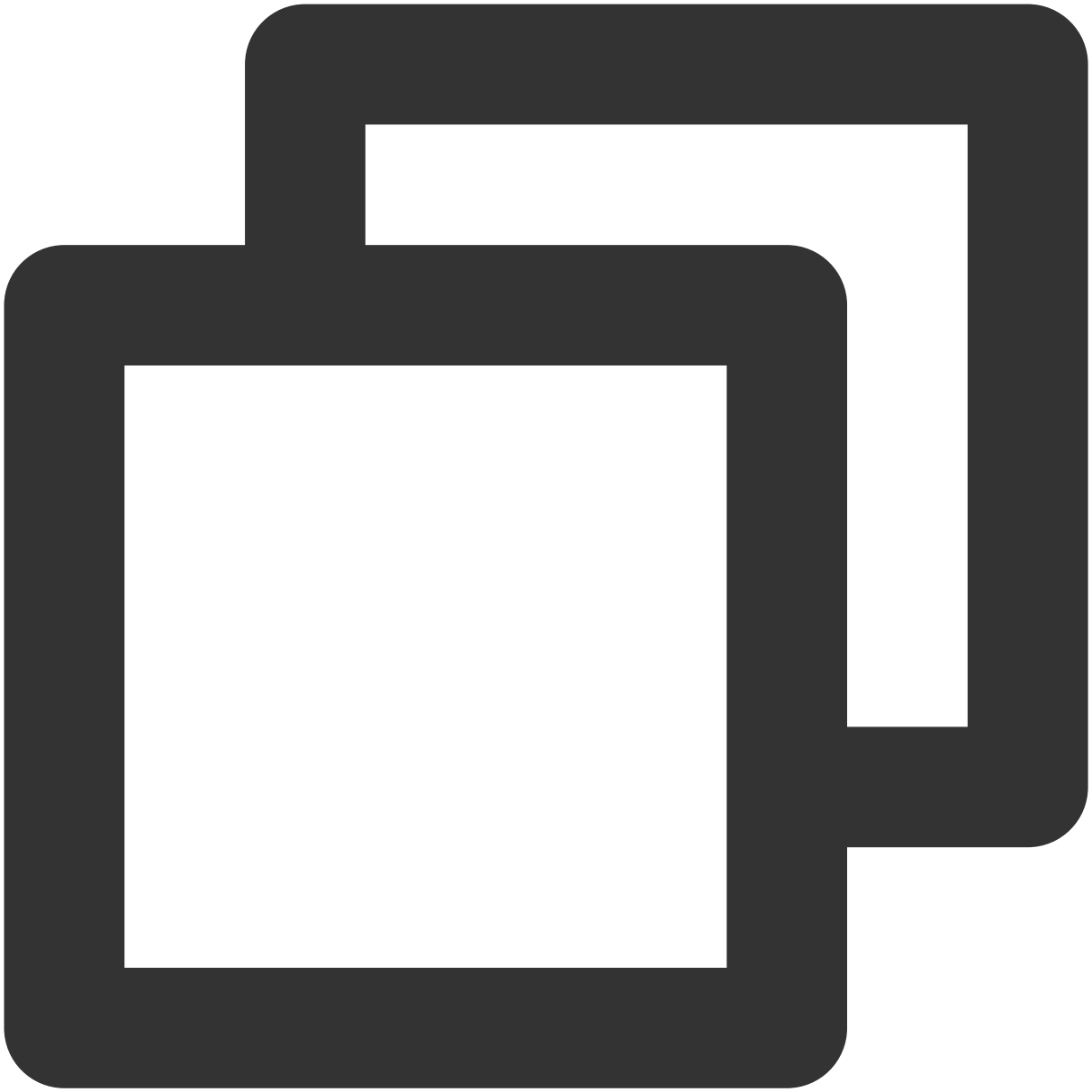
Windows

1. 执行以下命令进行解压，请将 1.x.x 替换为您实际安装 Terraform 版本号。



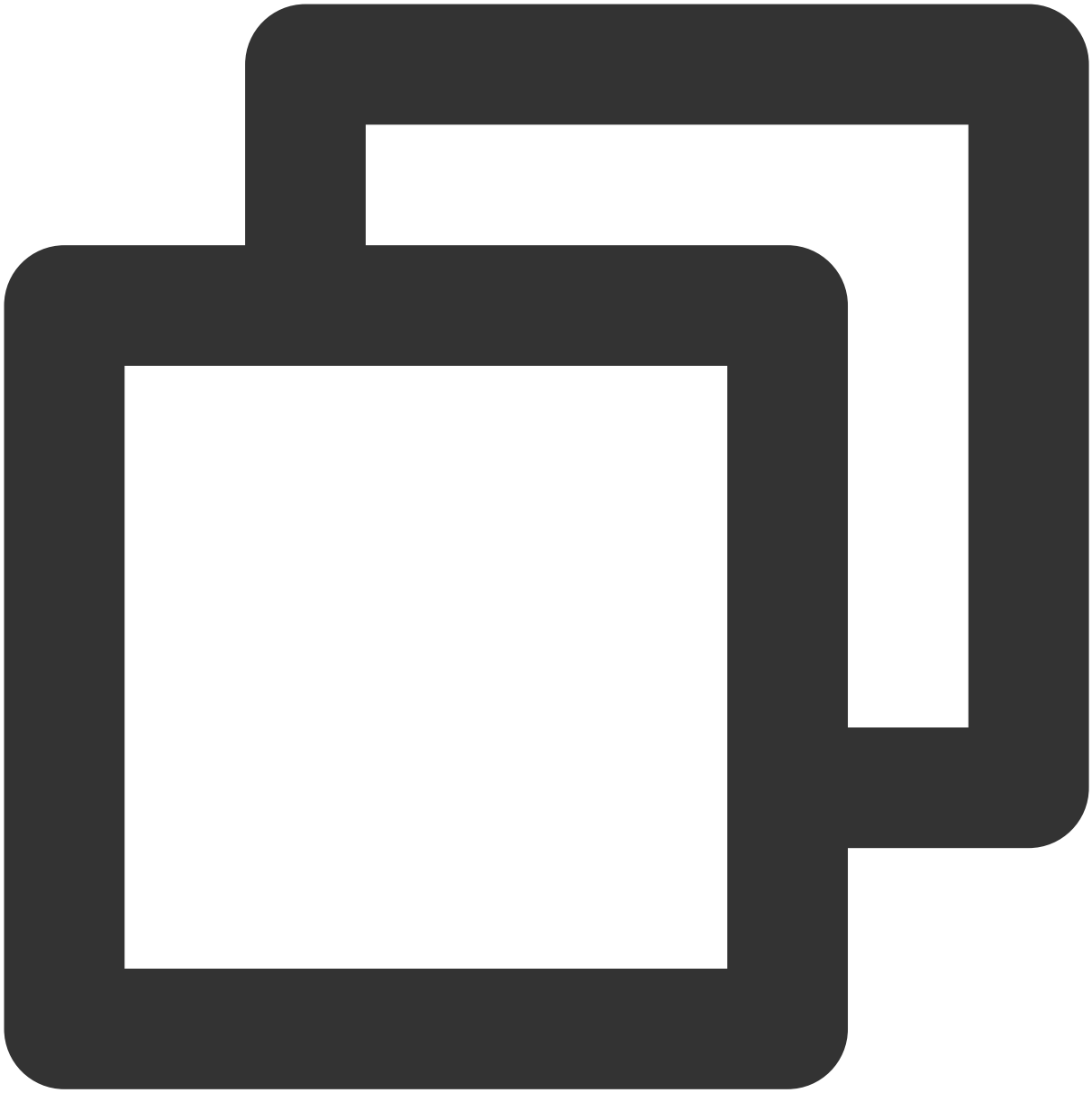
```
unzip terraform_1.x.x_linux_amd64.zip
```

2. 执行以下命令，将当前目录添加至 `~/.profile` 文件中。



```
echo $"export PATH=\\$PATH:$(pwd)" >> ~/.bash_profile
```

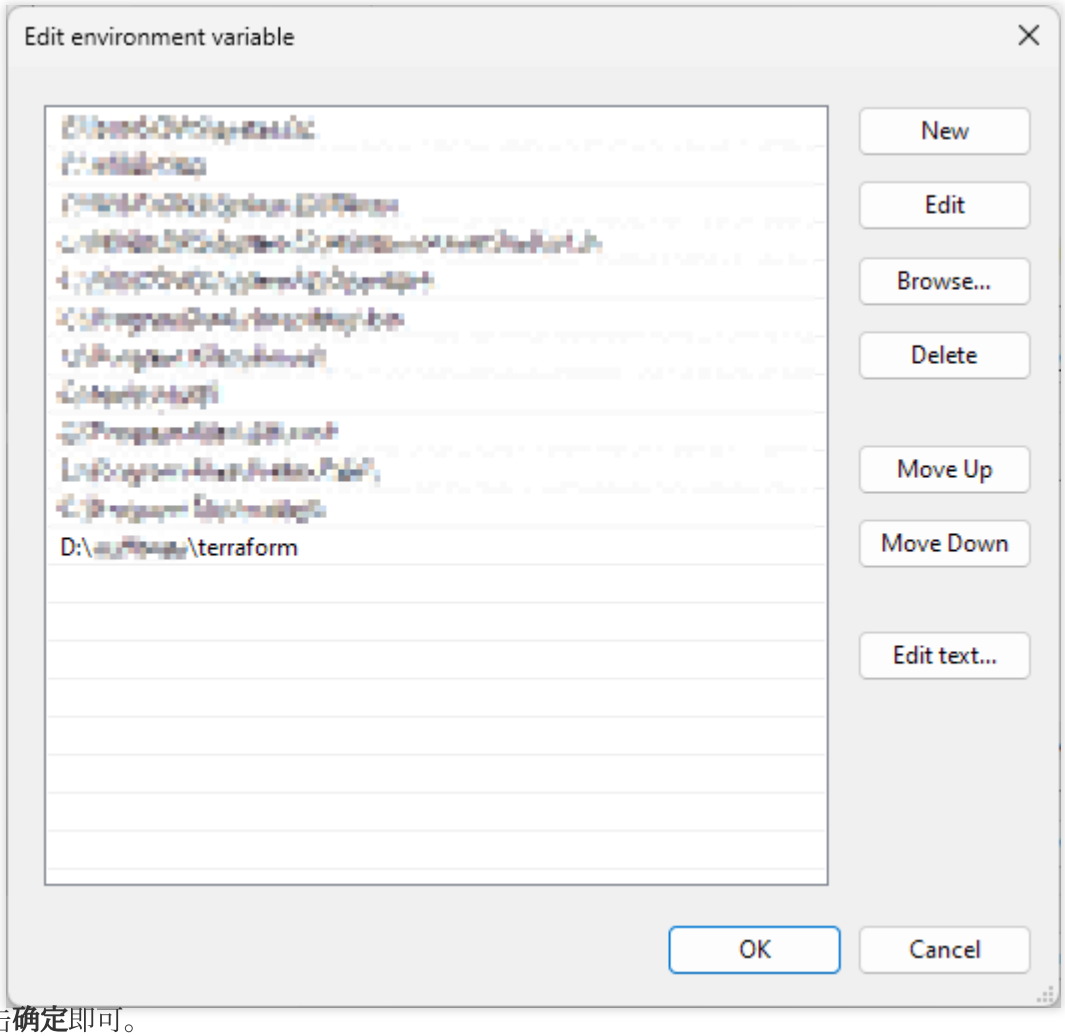
3. 执行以下命令，使全局路径配置生效。



```
source ~/.bash_profile
```

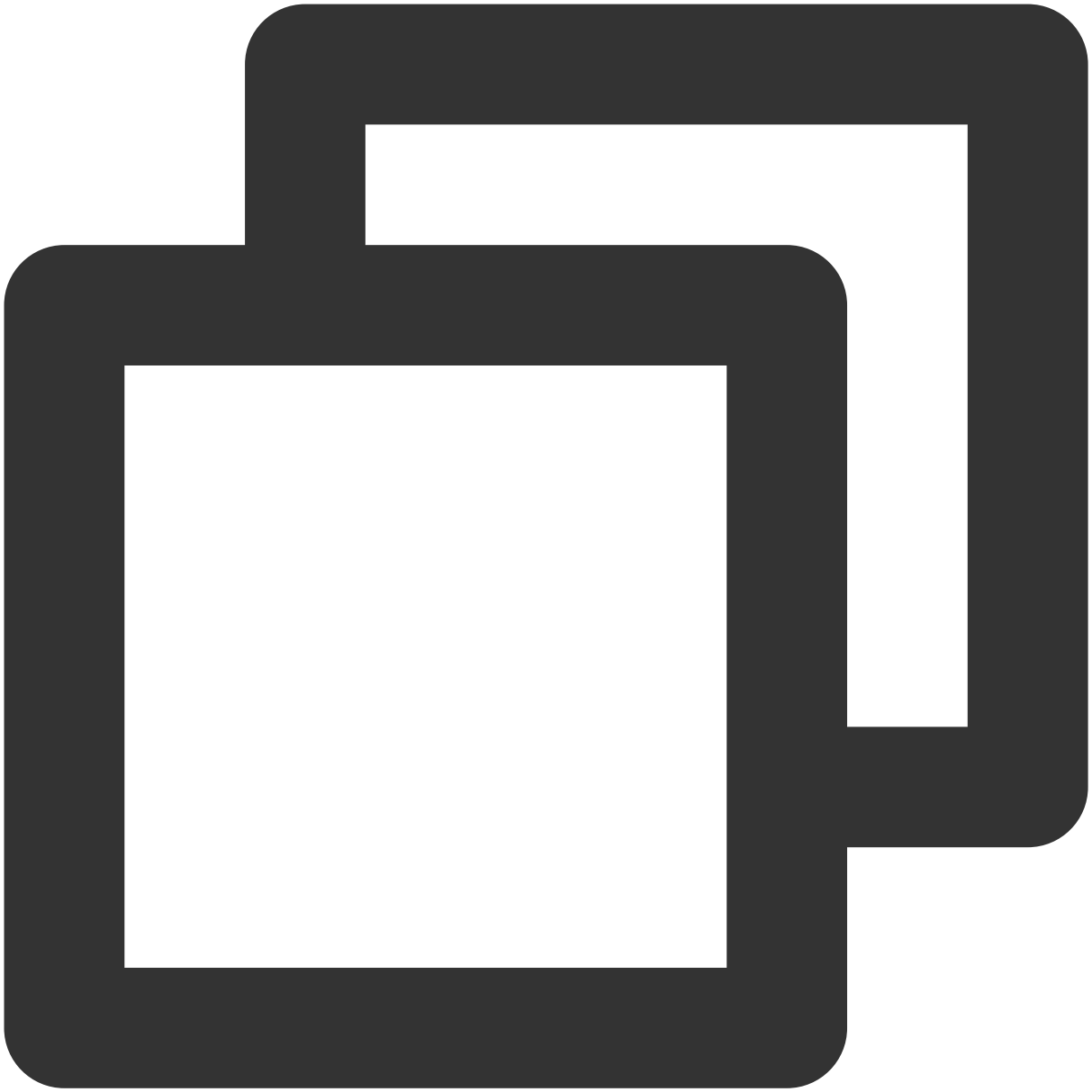
1. 在桌面中，右键选择“计算机”，并在弹出菜单中选择**属性**。
2. 在弹出窗口中，找到并单击**高级系统设置**。
3. 在弹出的“系统属性”窗口中，单击**环境变量**。
4. 在“系统变量”中，找到 `Path` ，将 `terraform.exe` 的绝对路径加入其中。如下图所示：

本文以 Windows 10 操作系统为例， `terraform.exe` 位于 `D:\\xxxx\\terraform` 文件下。



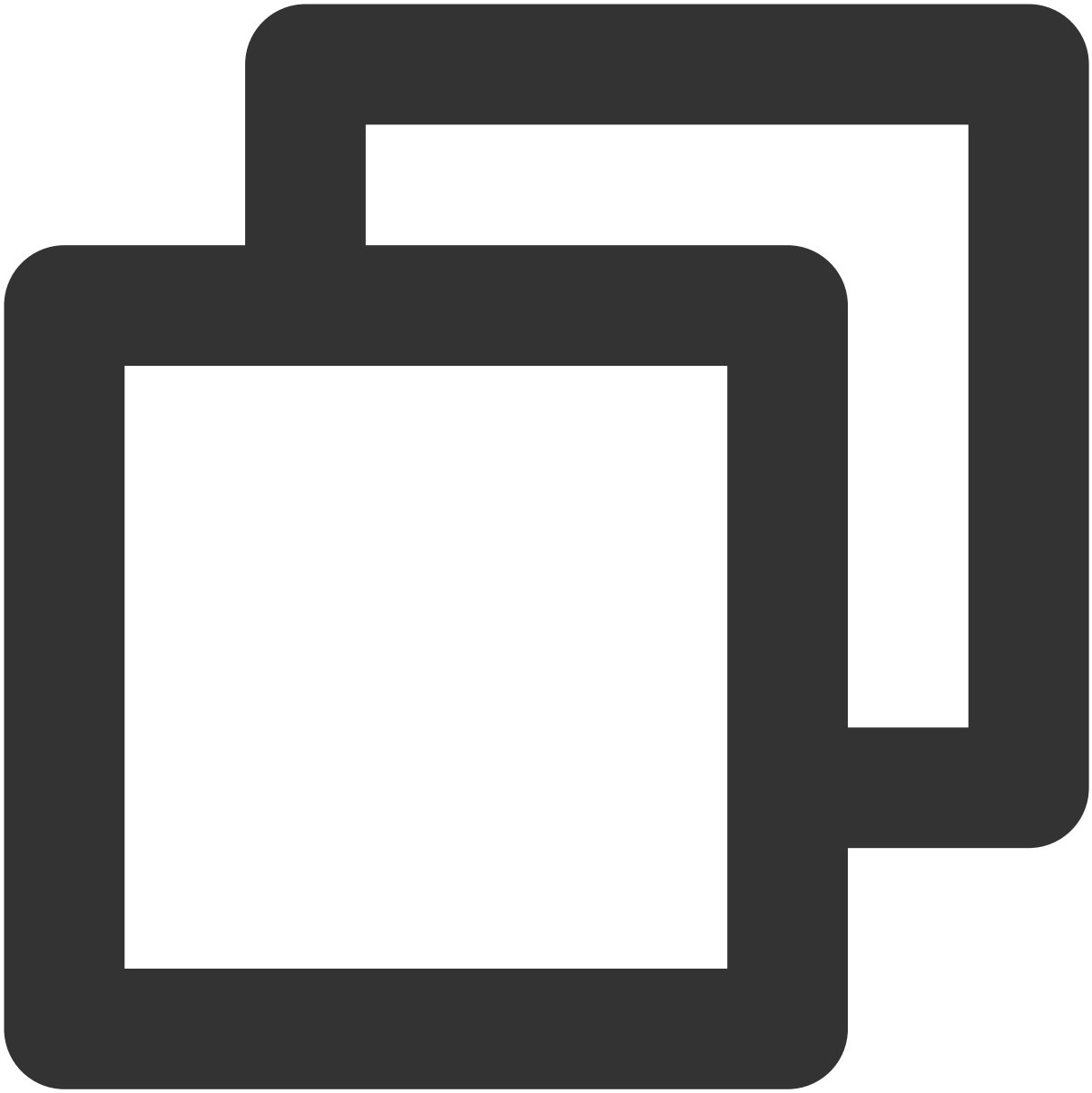
5. 单击**确定**即可。

3. 执行以下命令，查看是否安装成功。



```
terraform  -version
```

返回信息如下所示（版本号可能存在差异），则表示安装成功。

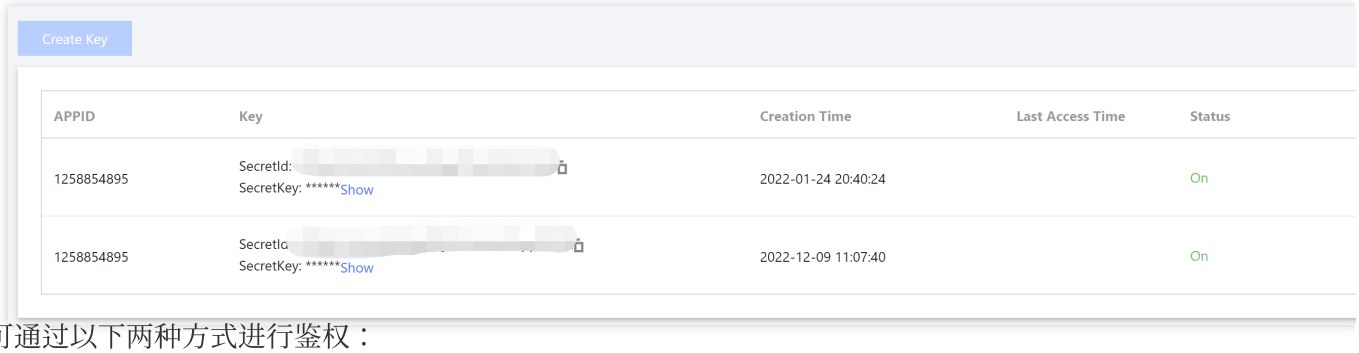


```
> Terraform v1.0.10
> on darwin_amd64
> Your version of Terraform is out of date! The latest version
> is 1.1.0. You can update by downloading from https://www.terraform.io/downloads.h
```

步骤2：配置安全凭证

在首次使用 Terraform 之前，请前往 云 API 密钥页面 申请安全凭证 SecretId 和 SecretKey。若已有可使用的安全凭证，则跳至第3步。

1. 登录 访问管理控制台，在左侧导航栏，选择**访问密钥 > API 密钥管理**。
2. 在 API 密钥管理页面，单击 **新建密钥**，即可以创建一对 SecretId/SecretKey。



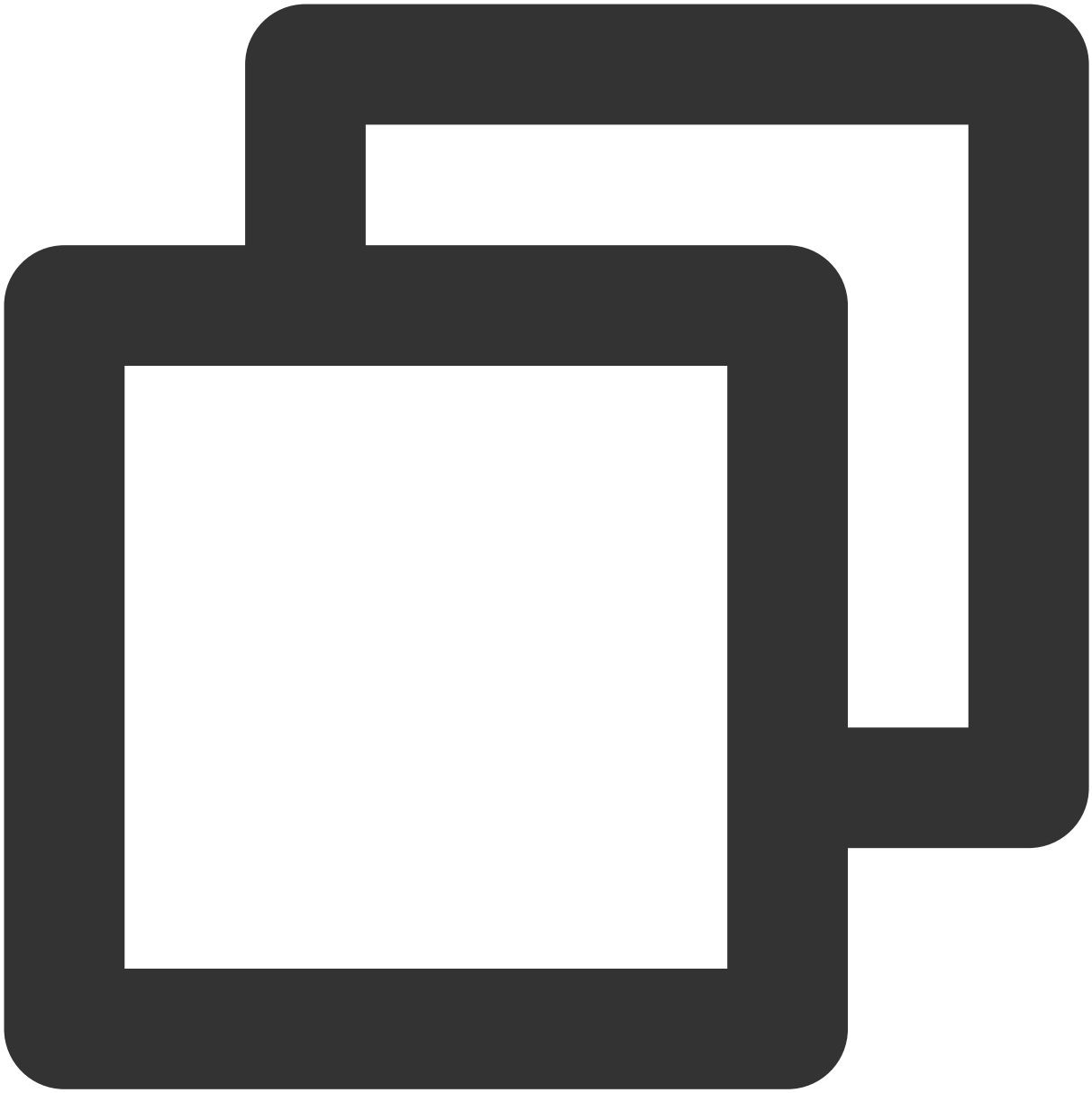
3. 您可通过以下两种方式进行鉴权：

环境变量鉴权

静态凭证鉴权

请将如下信息添加至环境变量配置：

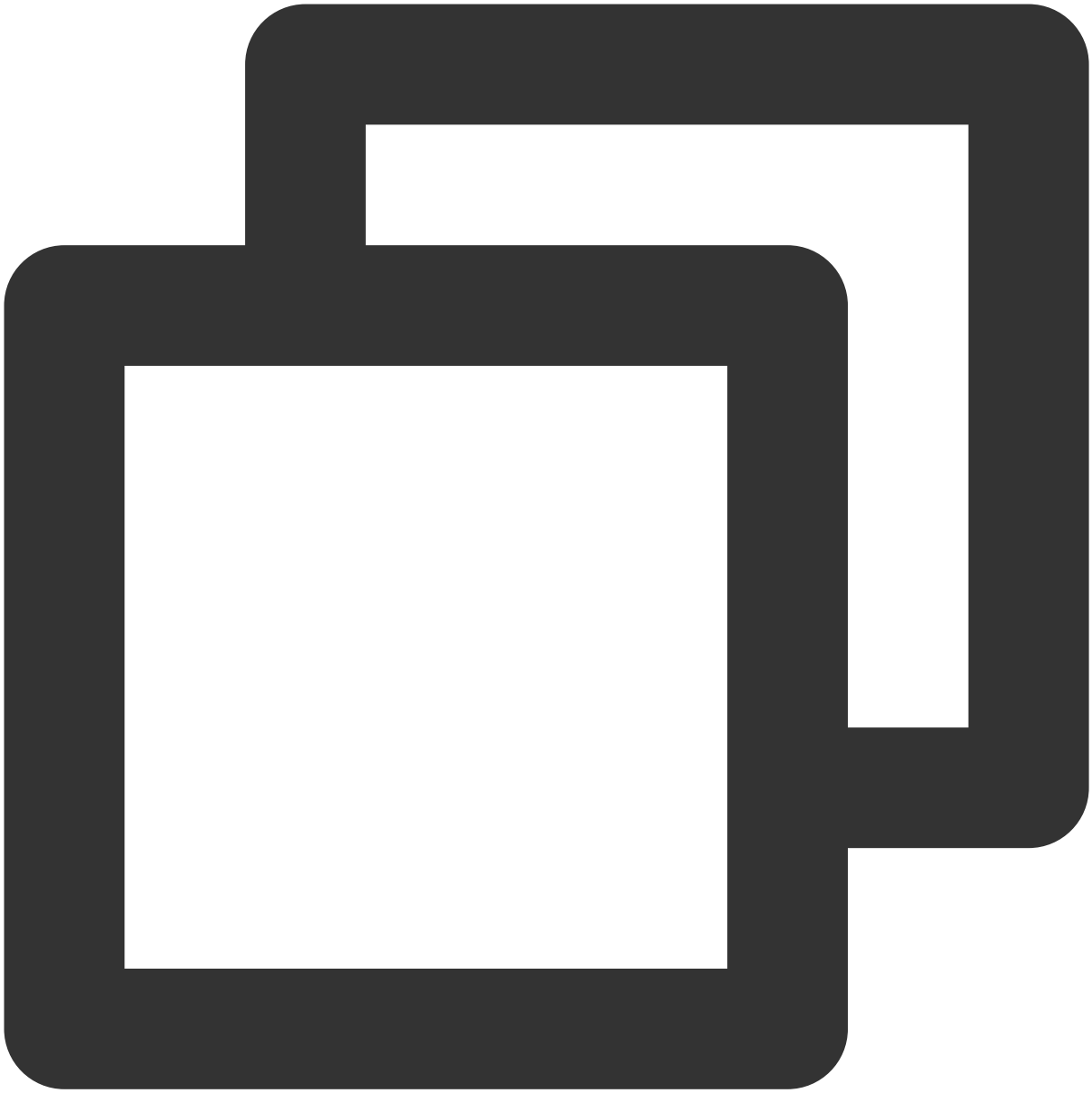
`<your-secret-id>` 及 `<your-secret-key>` 请替换为 获取凭证 中的 `SecretId` 和 `SecretKey` 。



```
export TENCENTCLOUD_SECRET_ID=<your-secret-id>
export TENCENTCLOUD_SECRET_KEY=<your-secret-key>
```

在用户目录下创建 `provider.tf` 文件，输入如下内容：

`<your-secret-id>` 及 `<your-secret-key>` 请替换为 获取凭证 中的 `SecretId` 和 `SecretKey` 。



```
provider "tencentcloud" {
    secret_id = "<your-secret-id>"
    secret_key = "<your-secret-key>"
}
```

4. 到此已完成 Terraform 安装和环境变量配置，可进行下一步：通过 Terraform 创建站点。

通过 Terraform 配置站点加速

最近更新时间：2024-01-25 11:20:13

功能简介

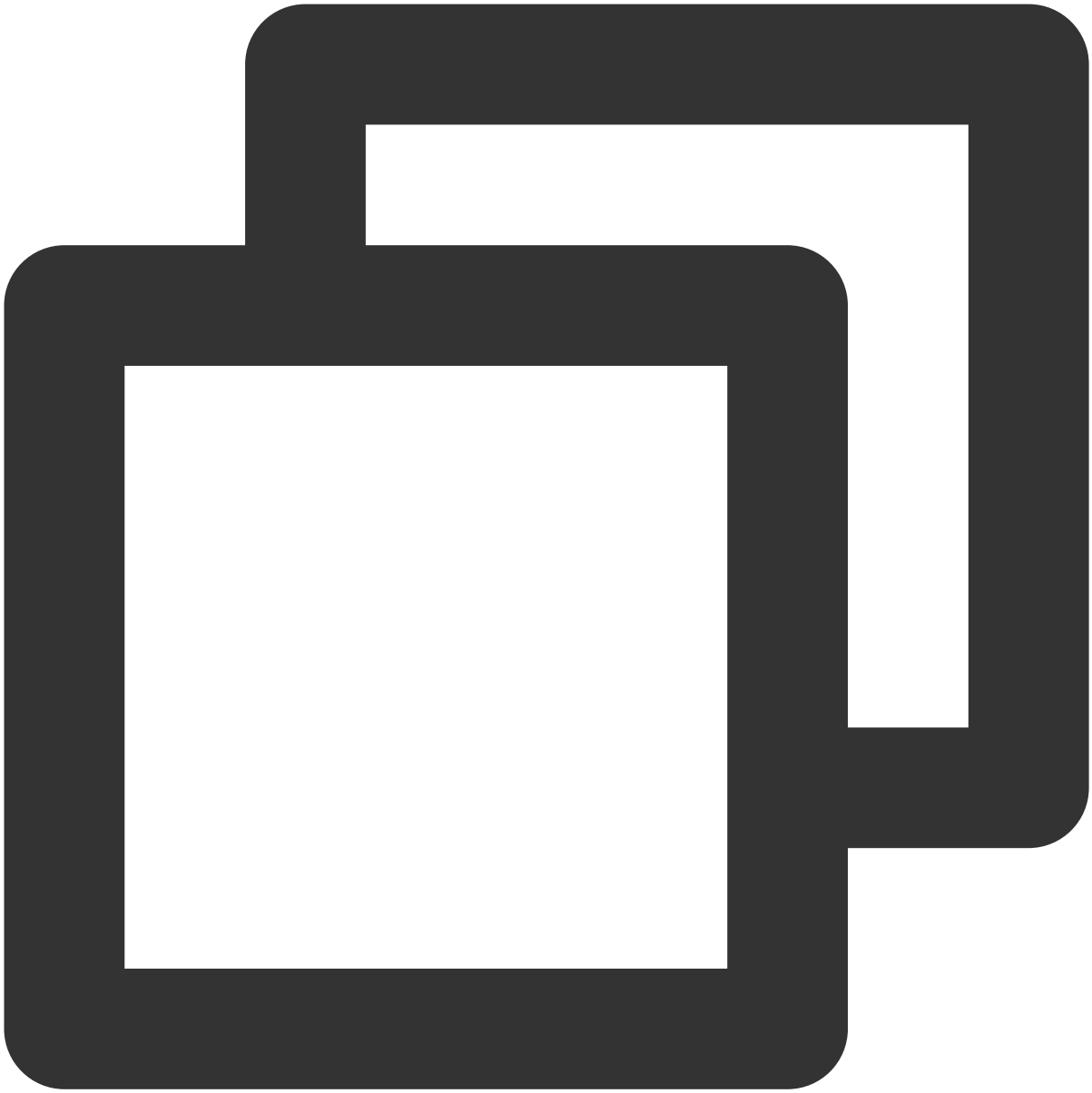
腾讯云边缘安全加速平台（TencentCloud EdgeOne，下文简称为 EdgeOne）已经接入 Terraform，可以通过 Terraform 来实现快速配置。本文介绍如何使用 Terraform 来配置站点加速。站点加速相关配置的说明，详情请参见操作指南。

前提条件

1. 已完成 Terraform 的安装与配置，操作步骤请参见 安装和配置 Terraform。
2. 已通过 Terraform 接入了站点，操作步骤请参见 通过 Terraform 创建站点。

操作步骤

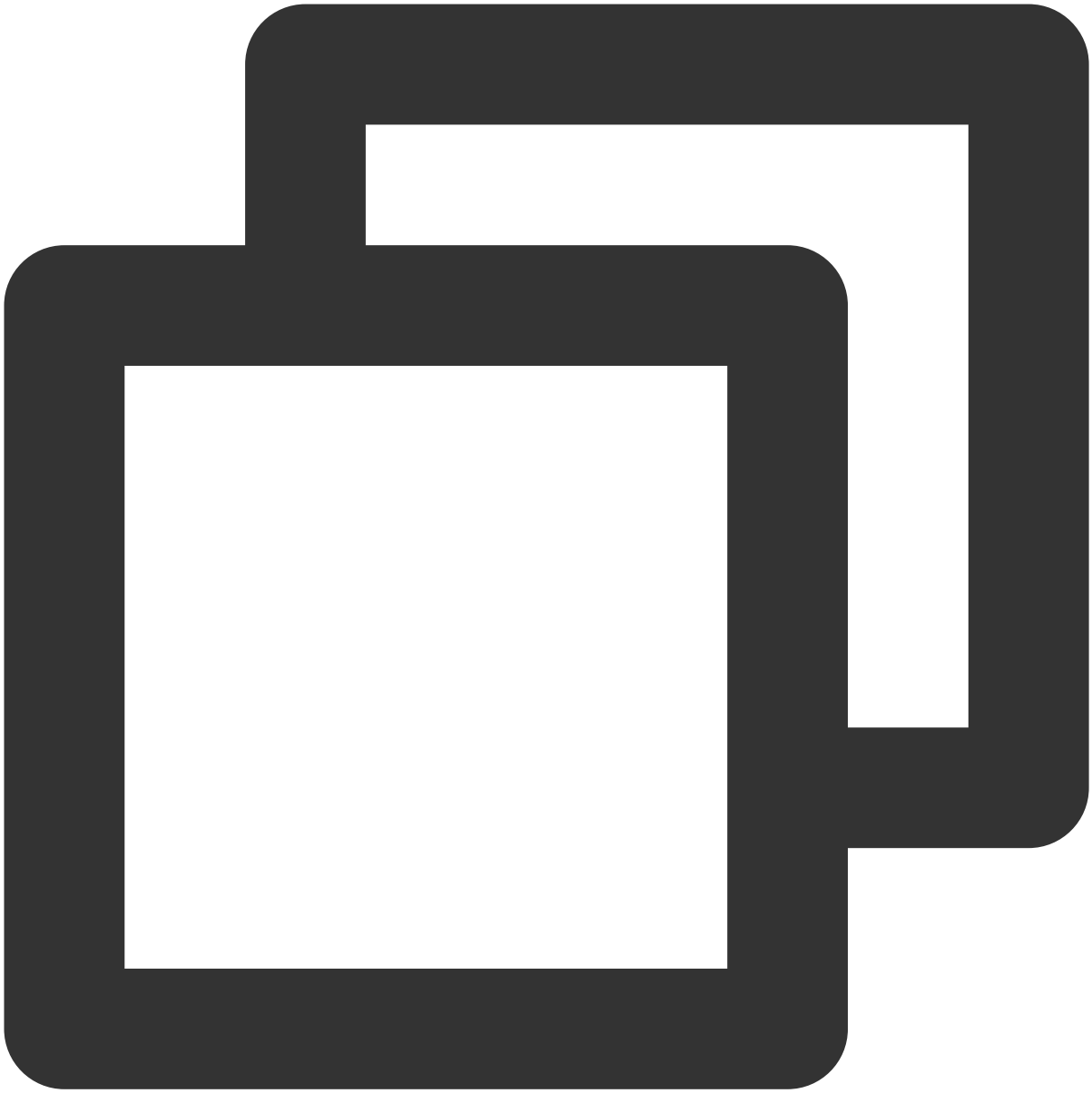
1. 修改 Terraform 配置文件，添加站点加速配置的资源定义。
您可以在 Terraform Provider 文档页面上查看 站点加速配置 的参数定义。以下为示例配置文件 tencent_teo.tf 的内容：



```
terraform {
  required_providers {
    tencentcloud = {
      source  = "tencentcloudstack/tencentcloud"
      version = ">= 1.78.5"
    }
  }
}
provider "tencentcloud" {
  secret_id  = "<your-secret-id>"
  secret_key = "<your-secret-key>"
```

```
  region     = "ap-guangzhou"
}
resource "tencentcloud_teo_zone" "example" {
  zone_name = "example.com"
  plan_type = "ent"
  tags = {
    "createdBy" = "terraform"
  }
}
resource "tencentcloud_teo_zone_setting" "example" {
  zone_id = tencentcloud_teo_zone.example.id
  # 缓存规则配置
  cache {
    follow_origin {
      switch = "on" # 遵循源站
    }
  }
  # 缓存键配置
  cache_key {
    full_url_cache = "off" # 不开启全路径缓存
    ignore_case    = "on" # 忽略大小写
    query_string {
      switch = "on"
      action = "includeCustom" # 仅使用指定的 URL 参数
      value  = ["param0", "param1"]
    }
  }
  # 域名 https 加速配置
  https {
    ocsp_stapling = "on" # OCSP 配置开启
    tls_version   = ["TLSv1.2", "TLSv1.3"] # 支持的 TLS 协议版本
  }
  # 智能压缩配置
  compression {
    switch     = "on"
    algorithms = ["brotli", "gzip"]
  }
  # 回源时携带客户端IP所属地域信息
  client_ip_header {
    switch      = "on"
    header_name = "EO-Client-IPCountry"
  }
}
```

2. 执行 `terraform plan` 命令预览配置，可以校验配置是否正确。



```
PS tf-doc> terraform.exe plan
tencentcloud_teo_zone.example: Refreshing state... [id=zone-2ag9gej58j36]
Terraform used the selected providers to generate the following execution plan. Res
  + create
Terraform will perform the following actions:
  # tencentcloud_teo_zone_setting.example will be created
  + resource "tencentcloud_teo_zone_setting" "example" {
      + area    = (known after apply)
      + id      = (known after apply)
      + zone_id = "zone-2ag9gej58j36"
      + cache {
```

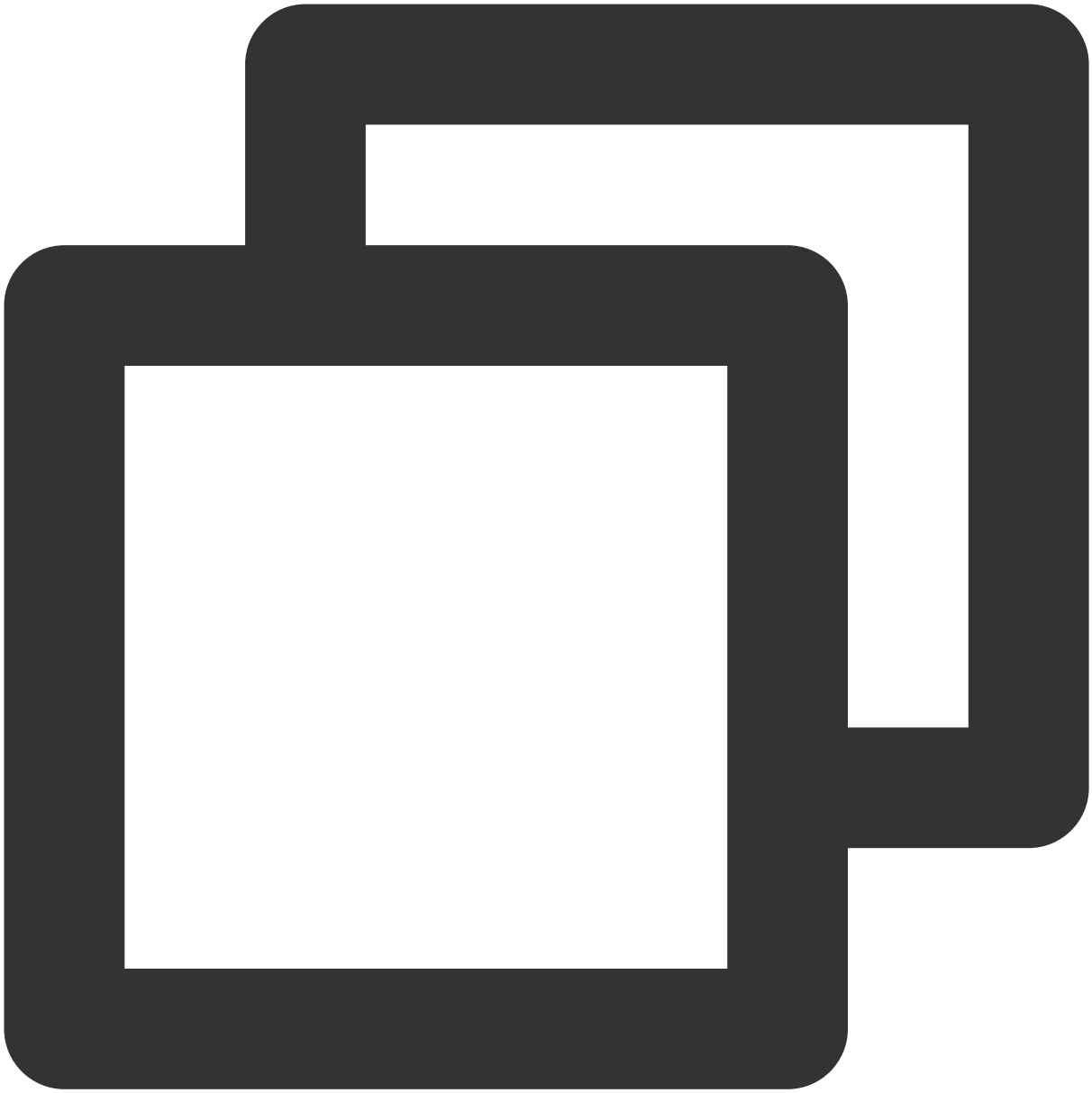
```
      + cache {
          + cache_time           = (known after apply)
          + ignore_cache_control = (known after apply)
          + switch               = (known after apply)
        }
      + follow_origin {
          + switch = "on"
        }
      + no_cache {
          + switch = (known after apply)
        }
    }
  + cache_key {
      + full_url_cache = "off"
      + ignore_case    = "on"
      + query_string {
          + action = "includeCustom"
          + switch = "on"
          + value  = [
              + "param0",
              + "param1",
            ]
        }
    }
  + cache_prefresh {
      + percent = (known after apply)
      + switch  = (known after apply)
    }
  + client_ip_header {
      + header_name = "EO-Client-IPCountry"
      + switch      = "on"
    }
  + compression {
      + algorithms = [
          + "brotli",
          + "gzip",
        ]
      + switch     = "on"
    }
  + force_redirect {
      + redirect_status_code = (known after apply)
      + switch               = (known after apply)
    }
  + https {
      + ocsp_stapling = "on"
      + tls_version   = [
          + "TLSv1.2",
```

```
                + "TLSv1.3",
              ]
        }
      + ipv6 {
          + switch = (known after apply)
        }
      + max_age {
          + follow_origin = (known after apply)
          + max_age_time  = (known after apply)
        }
      + offline_cache {
          + switch = (known after apply)
        }
      + origin {
          + backup_origins       = (known after apply)
          + cos_private_access   = (known after apply)
          + origin_pull_protocol = (known after apply)
          + origins              = (known after apply)
        }
      + post_max_size {
          + max_size = (known after apply)
          + switch   = (known after apply)
        }
      + quic {
          + switch = (known after apply)
        }
      + smart_routing {
          + switch = (known after apply)
        }
      + upstream_http2 {
          + switch = (known after apply)
        }
      + web_socket {
          + switch  = (known after apply)
          + timeout = (known after apply)
        }
    }
Plan: 1 to add, 0 to change, 0 to destroy.
------------------------------------------------------------------------

Note: You didn't use the -out option to save this plan, so Terraform can't guarante
```

3. 执行 `terraform apply` 创建站点加速配置。

执行 apply 命令后 Terraform 会让您再次确认将要执行的动作。在确认无误后输入 `yes` 二次确认，然后等待命令执行完成。



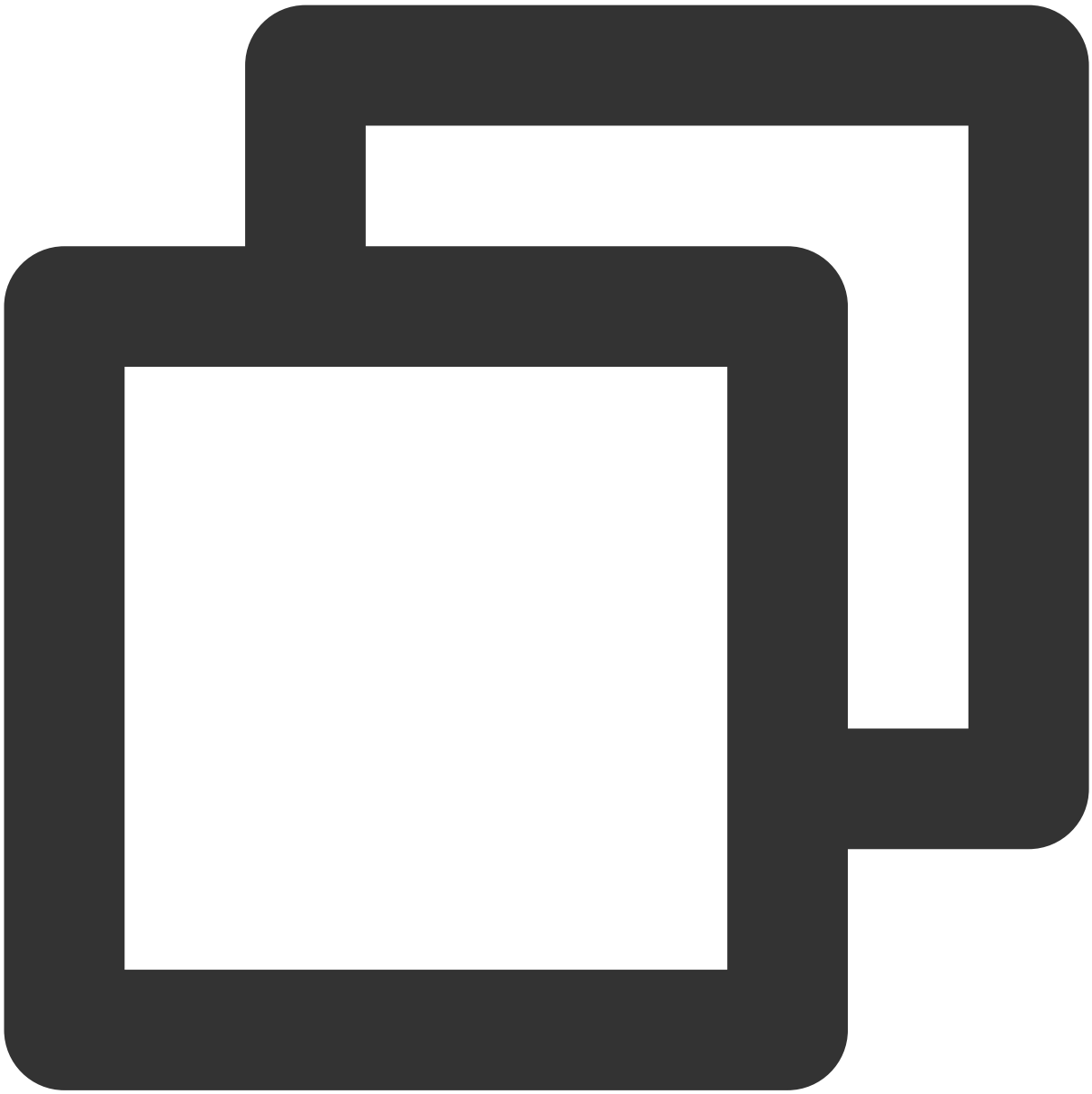
```
PS tf-doc> terraform.exe apply
tencentcloud_teo_zone.example: Refreshing state... [id=zone-2ag9gej58j36]
Terraform used the selected providers to generate the following execution plan. Res
  + create
Terraform will perform the following actions:
  # tencentcloud_teo_zone_setting.example will be created
  + resource "tencentcloud_teo_zone_setting" "example" {
      + area    = (known after apply)
      + id      = (known after apply)
      + zone_id = "zone-2ag9gej58j36"
      + cache {
```

```
      + cache {
          + cache_time           = (known after apply)
          + ignore_cache_control = (known after apply)
          + switch               = (known after apply)
        }
      + follow_origin {
          + switch = "on"
        }
      + no_cache {
          + switch = (known after apply)
        }
    }
  + cache_key {
      + full_url_cache = "off"
      + ignore_case    = "on"
      + query_string {
          + action = "includeCustom"
          + switch = "on"
          + value  = [
              + "param0",
              + "param1",
            ]
        }
    }
  + cache_prefresh {
      + percent = (known after apply)
      + switch  = (known after apply)
    }
  + client_ip_header {
      + header_name = "EO-Client-IPCountry"
      + switch      = "on"
    }
  + compression {
      + algorithms = [
          + "brotli",
          + "gzip",
        ]
      + switch     = "on"
    }
  + force_redirect {
      + redirect_status_code = (known after apply)
      + switch               = (known after apply)
    }
  + https {
      + ocsp_stapling = "on"
      + tls_version   = [
          + "TLSv1.2",
```

```
                + "TLSv1.3",
              ]
        }
      + ipv6 {
          + switch = (known after apply)
        }
      + max_age {
          + follow_origin = (known after apply)
          + max_age_time  = (known after apply)
        }
      + offline_cache {
          + switch = (known after apply)
        }
      + origin {
          + backup_origins       = (known after apply)
          + cos_private_access   = (known after apply)
          + origin_pull_protocol = (known after apply)
          + origins              = (known after apply)
        }
      + post_max_size {
          + max_size = (known after apply)
          + switch   = (known after apply)
        }
      + quic {
          + switch = (known after apply)
        }
      + smart_routing {
          + switch = (known after apply)
        }
      + upstream_http2 {
          + switch = (known after apply)
        }
      + web_socket {
          + switch  = (known after apply)
          + timeout = (known after apply)
        }
    }
Plan: 1 to add, 0 to change, 0 to destroy.
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
  Enter a value: yes
tencentcloud_teo_zone_setting.example: Creating...
tencentcloud_teo_zone_setting.example: Creation complete after 1s [id=zone-2ag9gej5
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

4. 检查命令执行结果。

您可以通过 `terraform show` 命令检查站点加速配置是否生效，也可以登录 边缘安全加速平台控制台 来确认。



```
PS tf-doc> terraform state show tencentcloud_teo_zone_setting.example
# tencentcloud_teo_zone_setting.example:
resource "tencentcloud_teo_zone_setting" "example" {
    area    = "overseas"
    id      = "zone-2ag9gej58j36"
    zone_id = "zone-2ag9gej58j36"
    cache {
        follow_origin {
```

```
            switch = "on"
        }
        no_cache {
            switch = "off"
        }
    }
    cache_key {
        full_url_cache = "off"
        ignore_case    = "on"
        query_string {
            action = "includeCustom"
            switch = "on"
            value  = [
                "param0",
                "param1",
            ]
        }
    }
    cache_prefresh {
        percent = 90
        switch  = "off"
    }
    client_ip_header {
        header_name = "EO-Client-IPCountry"
        switch      = "on"
    }
    compression {
        algorithms = [
            "brotli",
            "gzip",
        ]
        switch     = "on"
    }
    force_redirect {
        redirect_status_code = 302
        switch               = "off"
    }
    https {
        http2         = "on"
        ocsp_stapling = "on"
        tls_version   = [
            "TLSv1.2",
            "TLSv1.3",
        ]
        hsts {
            include_sub_domains = "off"
            max_age             = 0
```

```
            preload             = "off"
            switch              = "off"
        }
    }
    ipv6 {
        switch = "off"
    }
    max_age {
        follow_origin = "on"
        max_age_time  = 600
    }
    offline_cache {
        switch = "on"
    }
    origin {
        backup_origins       = []
        origin_pull_protocol = "follow"
        origins              = []
    }
    post_max_size {
        max_size = 524288000
        switch   = "on"
    }
    quic {
        switch = "off"
    }
    smart_routing {
        switch = "off"
    }
    upstream_http2 {
        switch = "off"
    }
    web_socket {
        switch  = "off"
        timeout = 30
    }
}
```

通过 Terraform 配置规则引擎

最近更新时间：2024-01-25 11:20:13

功能简介

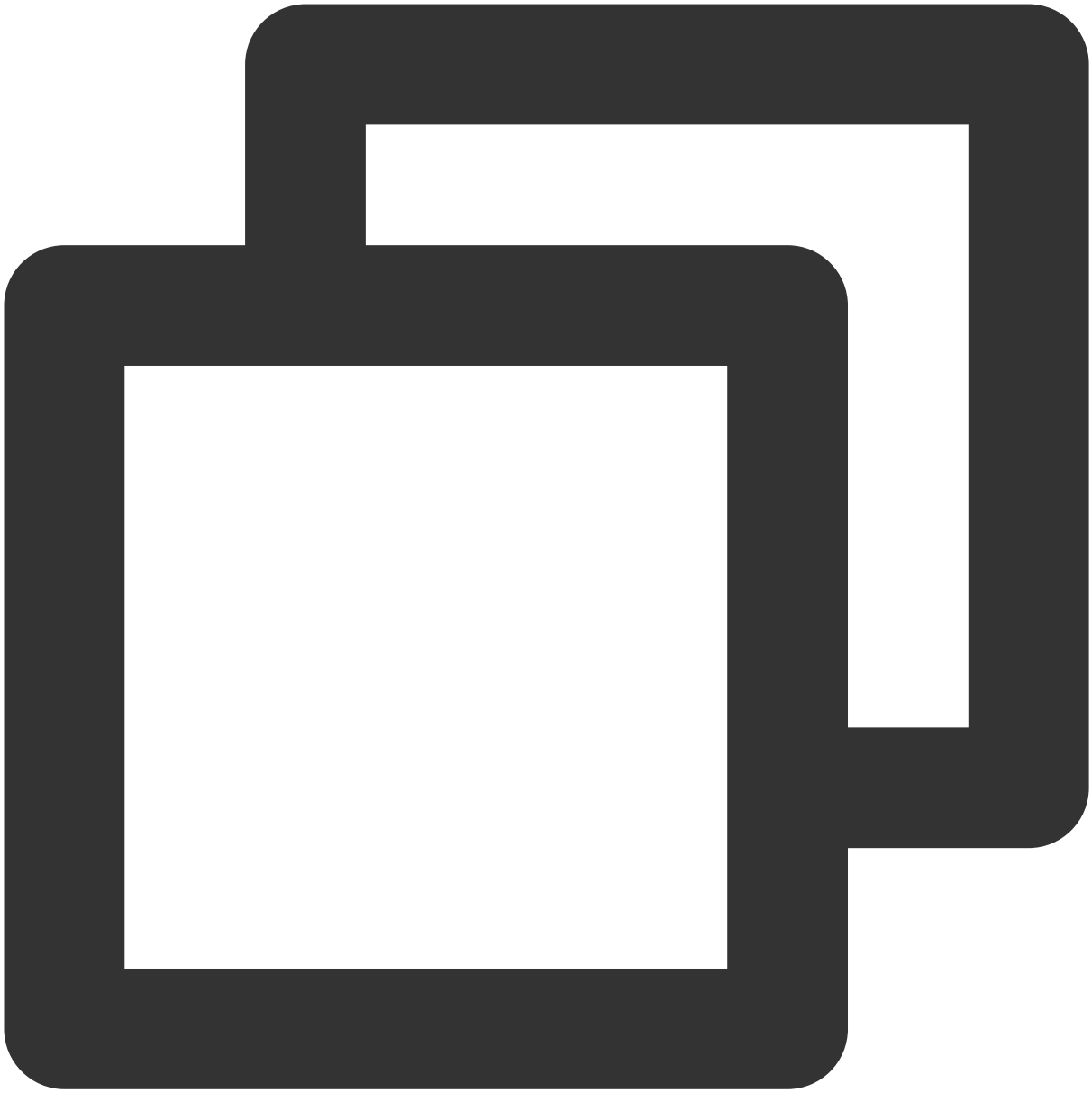
腾讯云边缘安全加速平台（TencentCloud EdgeOne，下文简称为 EdgeOne）已经接入 Terraform，可以通过 Terraform 来实现快速配置。本文介绍如何使用 Terraform 来配置 EdgeOne 站点的规则引擎，实现子域名差异化配置。

前置条件

1. 已完成 Terraform 的安装与配置，操作步骤请参见 安装和配置 Terraform。
2. 已通过 Terraform 接入了站点，操作步骤请参见 通过 Terraform 创建站点。

操作步骤

1. 修改 Terraform 配置文件，添加子域名 DNS 记录和规则引擎的资源定义。
您可以在 Terraform Provider 文档页面上查看 DNS 记录 和 规则引擎 的参数定义。以下为示例配置文件 tencent_teo.tf 的内容：

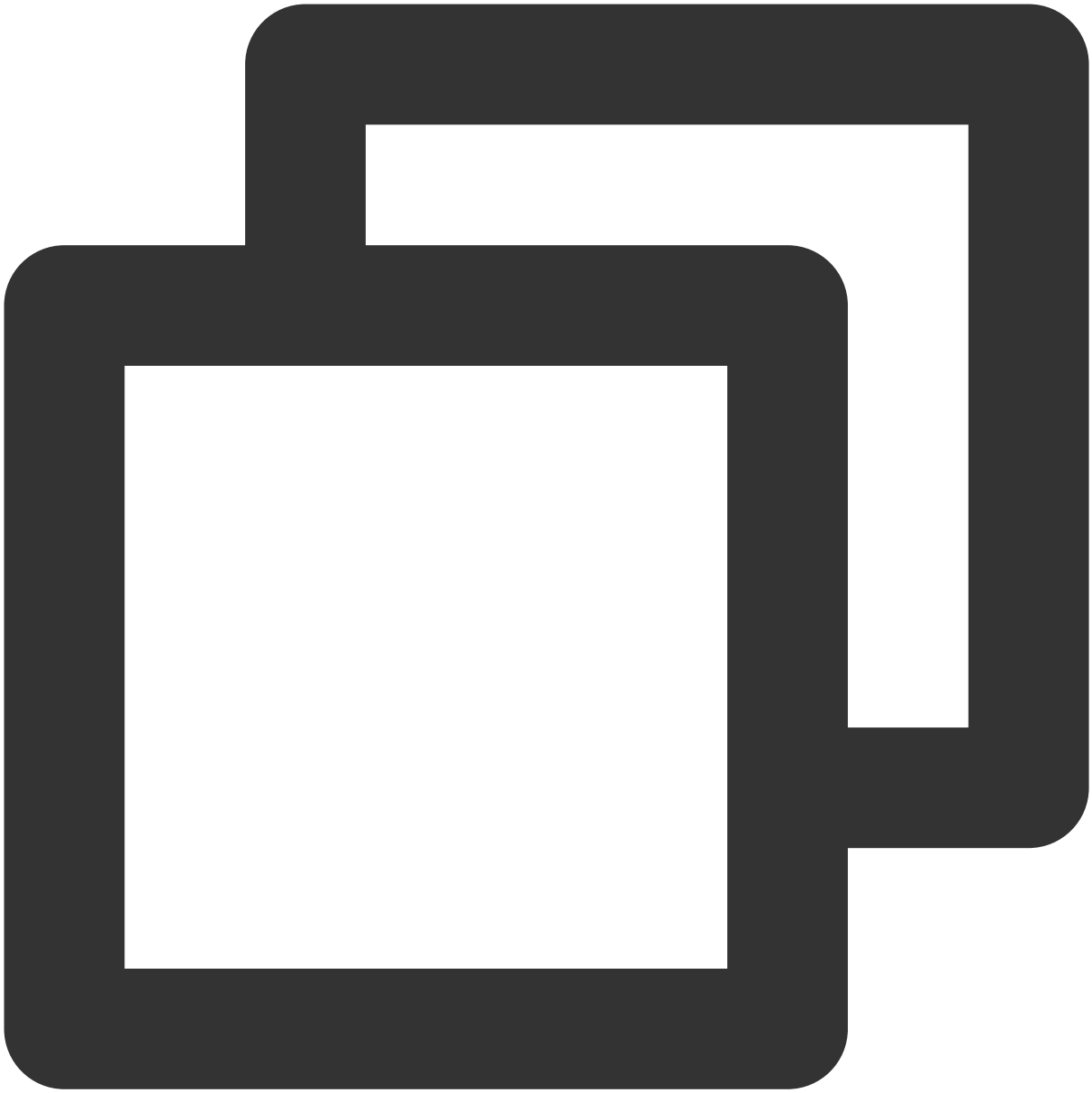


```
terraform {
  required_providers {
    tencentcloud = {
      source  = "tencentcloudstack/tencentcloud"
      version = ">= 1.78.5"
    }
  }
}
provider "tencentcloud" {
  secret_id  = "<your-secret-id>"
  secret_key = "<your-secret-key>"
```

```
  region     = "ap-guangzhou"
}
resource "tencentcloud_teo_zone" "example" {
  zone_name = "example.com"
  plan_type = "ent"
  tags = {
    "createdBy" = "terraform"
  }
}
# 子域名 DNS 记录
resource "tencentcloud_teo_dns_record" "rule_record" {
  zone_id = tencentcloud_teo_zone.example.id
  type    = "A"
  name    = "rule.example.com"
  content = "<your-backend-ip>"
  mode    = "proxied"
  ttl     = 300
}
# 子域名差异化配置
resource "tencentcloud_teo_rule_engine" "rule_example" {
  zone_id   = tencentcloud_teo_zone.example.id
  rule_name = "example_rule"
  status    = "enable" # 启用该规则
  rules {
    # 针对 rule.example.com 且文件后缀为 mp3、mp4 的请求
    or {
      and {
        target   = "host"
        operator = "equal"
        values   = [tencentcloud_teo_dns_record.rule_record.name]
      }
      and {
        target   = "extension"
        operator = "equal"
        values   = ["mp4", "mp3"]
      }
    }
    actions {
      # 使用指定 CacheKey
      normal_action {
        action = "CacheKey"
        # CacheKey是大小写敏感的
        parameters {
          name   = "Type"
          values = ["IgnoreCase"]
        }
        parameters {
```

```
            name   = "Switch"
            values = ["off"]
          }
          # CacheKey 包含 User-Agent 头
          parameters {
            name   = "Type"
            Values = ["Header"]
          }
          parameters {
            name   = "Switch"
            values = ["on"]
          }
          parameters {
            name   = "Value"
            values = "User-Agent"
          }
        }
      }
      # 增加指定响应头
      actions {
        rewrite_action {
          action = "ResponseHeader"
          parameters {
            action = "add"
            name   = "Added-Header"
            values = ["Added-Value"]
          }
        }
      }
    }
}
```

2. 执行 `terraform plan` 命令预览配置，可以校验配置是否正确。



```
PS tf-doc> terraform plan
tencentcloud_teo_zone.example: Refreshing state... [id=zone-2ag9gej58j36]
Terraform used the selected providers to generate the following execution plan. Res
  + create
Terraform will perform the following actions:
  # tencentcloud_teo_dns_record.rule_record will be created
  + resource "tencentcloud_teo_dns_record" "rule_record" {
      + cname         = (known after apply)
      + content       = "<your-backend-ip>"
      + created_on    = (known after apply)
      + dns_record_id = (known after apply)
```

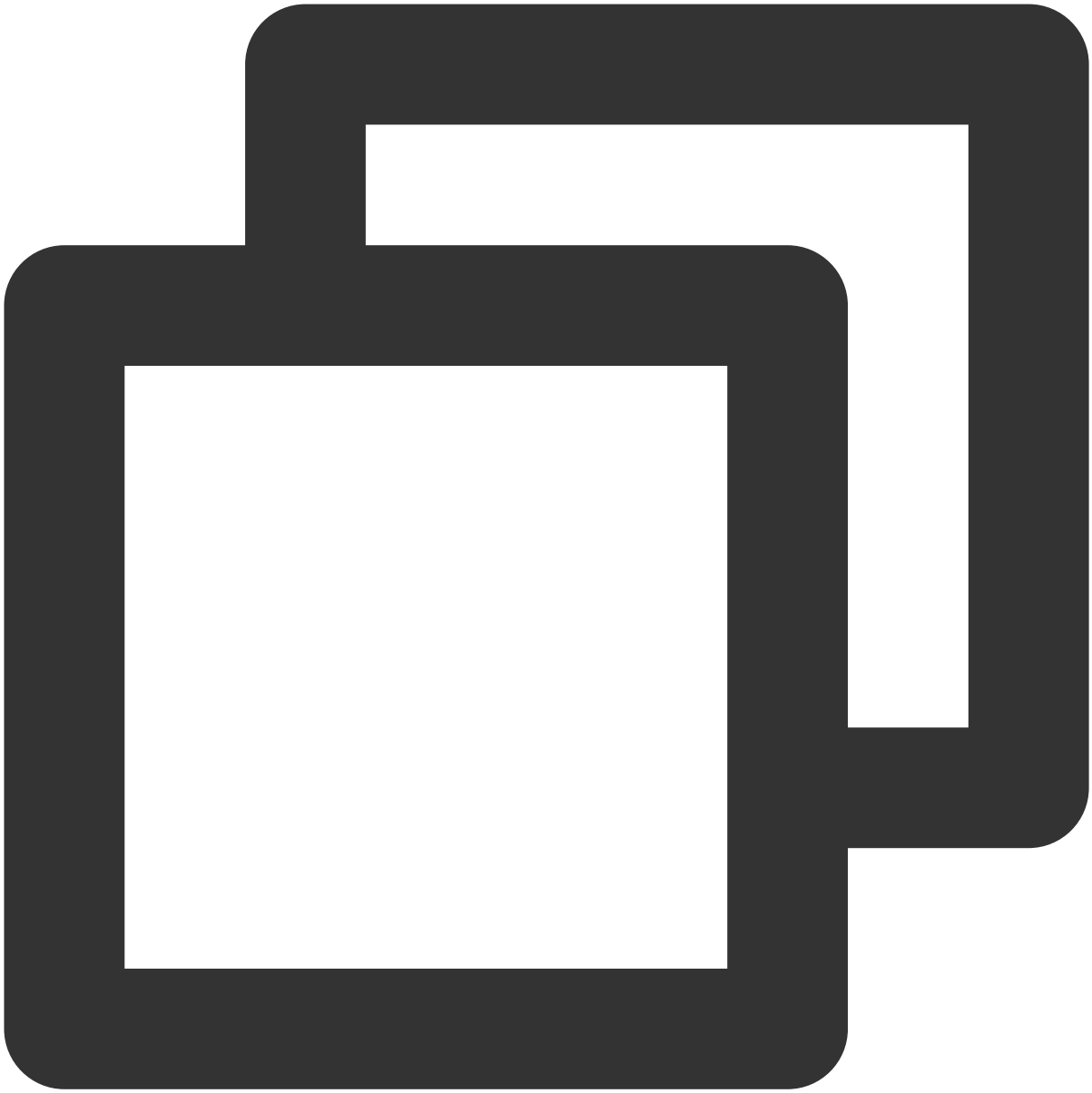
```
      + domain_status = (known after apply)
      + id            = (known after apply)
      + locked        = (known after apply)
      + mode          = "proxied"
      + modified_on   = (known after apply)
      + name          = "rule.example.com"
      + priority      = (known after apply)
      + status        = (known after apply)
      + ttl           = 300
      + type          = "A"
      + zone_id       = "zone-2ag9gej58j36"
    }
  # tencentcloud_teo_rule_engine.rule_example will be created
  + resource "tencentcloud_teo_rule_engine" "rule_example" {
      + id        = (known after apply)
      + rule_id   = (known after apply)
      + rule_name = "example_rule"
      + status    = "enable"
      + zone_id   = "zone-2ag9gej58j36"
      + rules {
          + actions {
              + normal_action {
                  + action = "CacheKey"
                  + parameters {
                      + name   = "Type"
                      + values = [
                          + "IgnoreCase",
                        ]
                    }
                  + parameters {
                      + name   = "Switch"
                      + values = [
                          + "off",
                        ]
                    }
                  + parameters {
                      + name   = "Type"
                      + values = [
                          + "Header",
                        ]
                    }
                  + parameters {
                      + name   = "Switch"
                      + values = [
                          + "on",
                        ]
                    }
```

```
                + parameters {
                    + name   = "Value"
                    + values = [
                        + "User-Agent",
                      ]
                  }
              }
          }
        + actions {
            + rewrite_action {
                + action = "ResponseHeader"
                + parameters {
                    + action = "add"
                    + name   = "Added-Header"
                    + values = [
                        + "Added-Value",
                      ]
                  }
              }
          }
        + or {
            + and {
                + operator = "equal"
                + target   = "host"
                + values   = [
                    + "rule.example.com",
                  ]
              }
            + and {
                + operator = "equal"
                + target   = "extension"
                + values   = [
                    + "mp3",
                    + "mp4",
                  ]
              }
          }
      }
  }
Plan: 2 to add, 0 to change, 0 to destroy.
─────────────────────────────────────────────────────────────────────────────

Note: You didn't use the -out option to save this plan, so Terraform can't guarante
```

3. 执行 `terraform apply` 创建子域名 DNS 记录和规则引擎。

执行 apply 命令后 Terraform 会让您再次确认将要执行的动作。在确认无误后输入 `yes` 二次确认，然后等待命令执行完成。



```
PS tf-doc> terraform apply
tencentcloud_teo_zone.example: Refreshing state... [id=zone-2ag9gej58j36]
Terraform used the selected providers to generate the following execution plan. Res
  + create
Terraform will perform the following actions:
  # tencentcloud_teo_dns_record.rule_record will be created
  + resource "tencentcloud_teo_dns_record" "rule_record" {
      + cname         = (known after apply)
      + content       = "<your-backend-ip>"
      + created_on    = (known after apply)
      + dns_record_id = (known after apply)
```

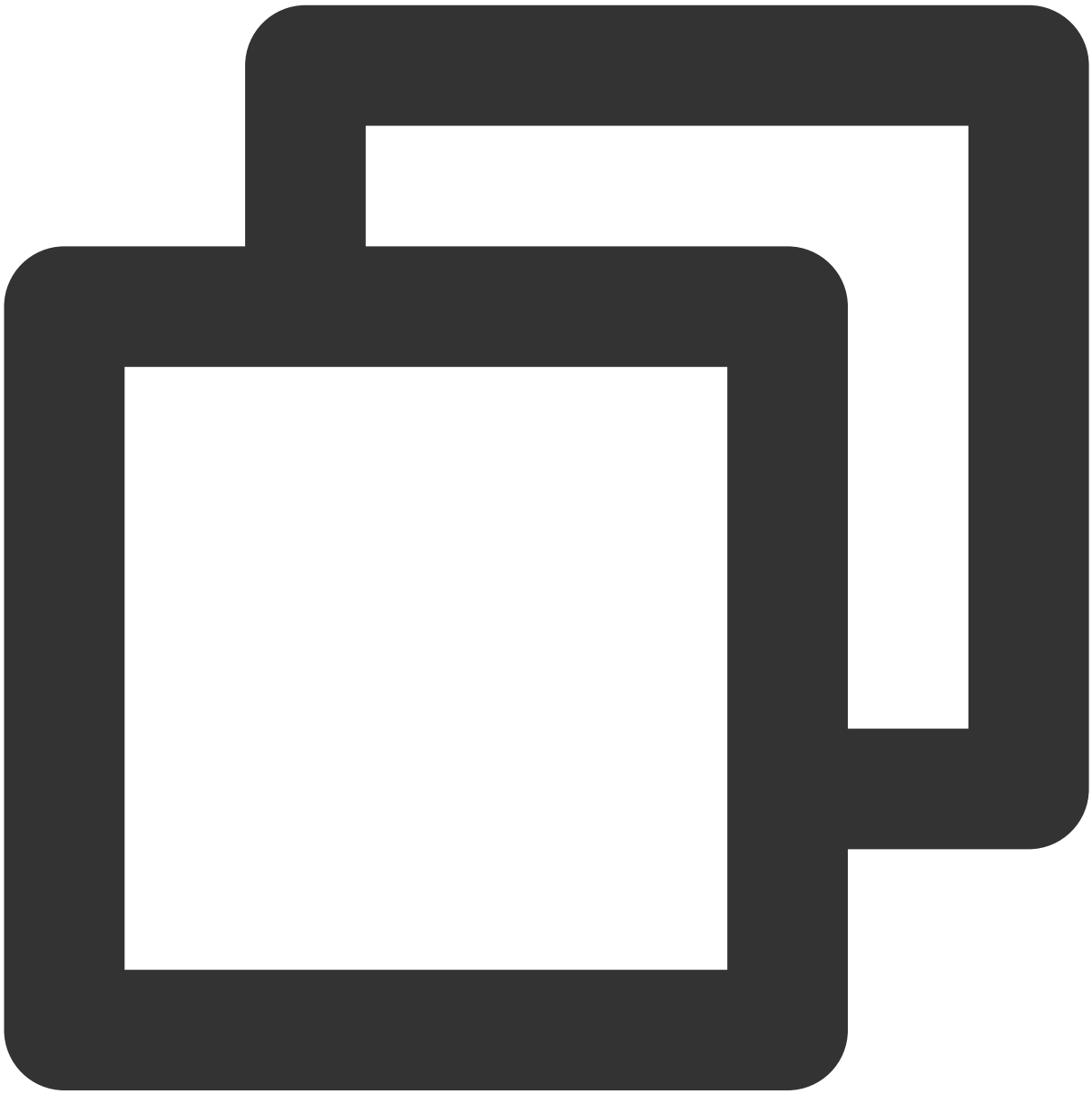
```
      + domain_status = (known after apply)
      + id            = (known after apply)
      + locked        = (known after apply)
      + mode          = "proxied"
      + modified_on   = (known after apply)
      + name          = "rule.example.com"
      + priority      = (known after apply)
      + status        = (known after apply)
      + ttl           = 300
      + type          = "A"
      + zone_id       = "zone-2ag9gej58j36"
    }
  # tencentcloud_teo_rule_engine.rule_example will be created
  + resource "tencentcloud_teo_rule_engine" "rule_example" {
      + id        = (known after apply)
      + rule_id   = (known after apply)
      + rule_name = "example_rule"
      + status    = "enable"
      + zone_id   = "zone-2ag9gej58j36"
      + rules {
          + actions {
              + normal_action {
                  + action = "CacheKey"
                  + parameters {
                      + name   = "Type"
                      + values = [
                          + "IgnoreCase",
                        ]
                    }
                  + parameters {
                      + name   = "Switch"
                      + values = [
                          + "off",
                        ]
                    }
                  + parameters {
                      + name   = "Type"
                      + values = [
                          + "Header",
                        ]
                    }
                  + parameters {
                      + name   = "Switch"
                      + values = [
                          + "on",
                        ]
                    }
```

```
                    + parameters {
                        + name   = "Value"
                        + values = [
                            + "User-Agent",
                          ]
                      }
                  }
              }
            + actions {
                + rewrite_action {
                    + action = "ResponseHeader"
                    + parameters {
                        + action = "add"
                        + name   = "Added-Header"
                        + values = [
                            + "Added-Value",
                          ]
                      }
                  }
              }
            + or {
                + and {
                    + operator = "equal"
                    + target   = "host"
                    + values   = [
                        + "rule.example.com",
                      ]
                  }
                + and {
                    + operator = "equal"
                    + target   = "extension"
                    + values   = [
                        + "mp3",
                        + "mp4",
                      ]
                  }
              }
          }
      }
Plan: 2 to add, 0 to change, 0 to destroy.
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
  Enter a value: yes
tencentcloud_teo_dns_record.rule_record: Creating...
tencentcloud_teo_dns_record.rule_record: Creation complete after 2s [id=zone-2ag9ge
tencentcloud_teo_rule_engine.rule_example: Creating...
```

```
tencentcloud_teo_rule_engine.rule_example: Creation complete after 1s [id=zone-2ag9
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

4. 检查命令执行结果。

您可以通过 `terraform show` 命令检查站点加速配置是否生效，也可以登录 边缘安全加速平台控制台 来确认。



```
PS tf-doc> terraform state show tencentcloud_teo_dns_record.rule_record
# tencentcloud_teo_dns_record.rule_record:
resource "tencentcloud_teo_dns_record" "rule_record" {
    content       = "<your-backend-ip>"
    created_on    = "2022-10-20T06:31:38Z"
```

```
    dns_record_id = "record-2ahmb3w7ssl8"
    domain_status = [
        "security",
    ]
    id            = "zone-2ag9gej58j36#record-2ahmb3w7ssl8"
    locked        = false
    mode          = "proxied"
    modified_on   = "2022-10-20T06:31:38Z"
    name          = "rule.example.com"
    priority      = 0
    status        = "active"
    ttl           = 300
    type          = "A"
    zone_id       = "zone-2ag9gej58j36"
}
PS tf-doc> terraform state show tencentcloud_teo_rule_engine.rule_example
# tencentcloud_teo_rule_engine.rule_example:
resource "tencentcloud_teo_rule_engine" "rule_example" {
    id        = "zone-2ag9gej58j36#rule-2ahmb5dhn9qq"
    rule_id   = "rule-2ahmb5dhn9qq"
    rule_name = "example_rule"
    status    = "enable"
    zone_id   = "zone-2ag9gej58j36"
    rules {
        actions {
            normal_action {
                action = "CacheKey"
                parameters {
                    name   = "Type"
                    values = [
                        "IgnoreCase",
                    ]
                }
                parameters {
                    name   = "Switch"
                    values = [
                        "off",
                    ]
                }
                parameters {
                    name   = "Type"
                    values = [
                        "Header",
                    ]
                }
                parameters {
                    name   = "Switch"
```

```
                    values = [
                        "on",
                    ]
                }
                parameters {
                    name   = "Value"
                    values = [
                        "User-Agent",
                    ]
                }
            }
        }
        actions {
            rewrite_action {
                action = "ResponseHeader"
                parameters {
                    action = "add"
                    name   = "Added-Header"
                    values = [
                        "Added-Value",
                    ]
                }
            }
        }
        or {
            and {
                operator = "equal"
                target   = "host"
                values   = [
                    "rule.example.com",
                ]
            }
            and {
                operator = "equal"
                target   = "extension"
                values   = [
                    "mp3",
                    "mp4",
                ]
            }
        }
    }
}
```

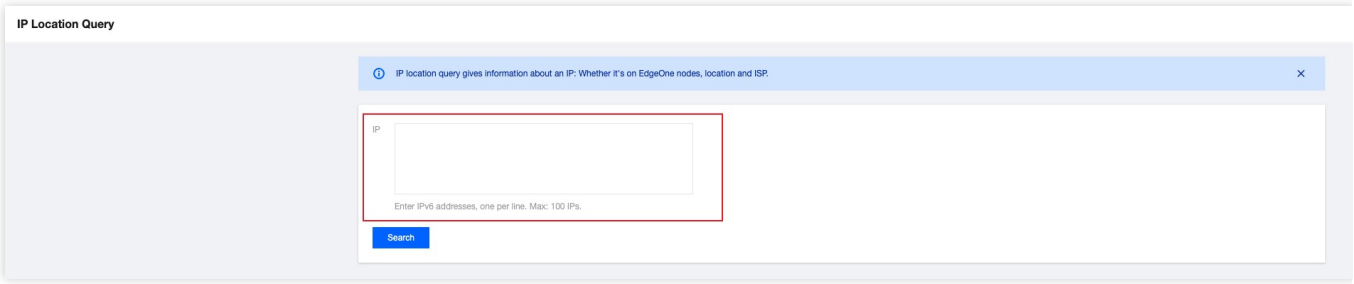
IP 归属查询

最近更新时间：2023-10-11 10:52:59

本文介绍了如何使用 IP 归属查询工具来验证 IP 是否为腾讯云 Edgeone 节点 IP，以及 IP 归属地信息。

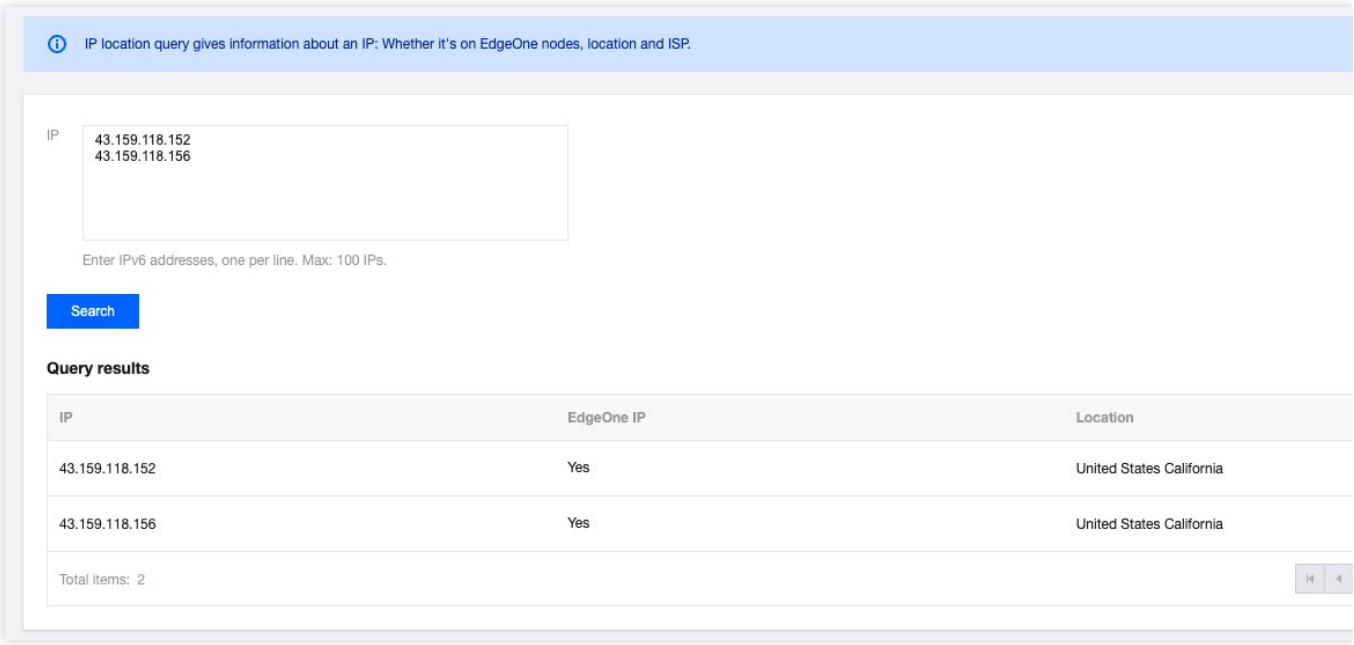
操作步骤

1. 登录 边缘安全加速平台 控制台，在左侧菜单栏中，单击 **IP 归属查询**。



2. 在 IP 归属查询页面，查询框内输入需要查询的 IP，一行一个，单次查询最多支持 100 个 IP，支持输入 IPv6 地址。
3. 单击**查询**，查询结果可展示当前查询的 IP 是否为腾讯云 Edgeone 节点 IP，以及 IP 归属地信息。单击查询结果右上角

，可导出查询结果，导出为 CSV 文件。


IP location query gives information about an IP: Whether it's on EdgeOne nodes, location and ISP.
IP
43.159.118.152
43.159.118.156
Enter IPv6 addresses, one per line. Max: 100 IPs.
Search
Query results
IP
EdgeOne IP
Location
43.159.118.152
Yes
United States California
43.159.118.156
Yes
United States California
Total items: 2