腾讯云可观测平台

Prometheus 监控

产品文档

腾讯云

文档目录

Prometheus 监控
Terraform
Terraform 概述

最近更新时间：2023-12-29 16:05:46

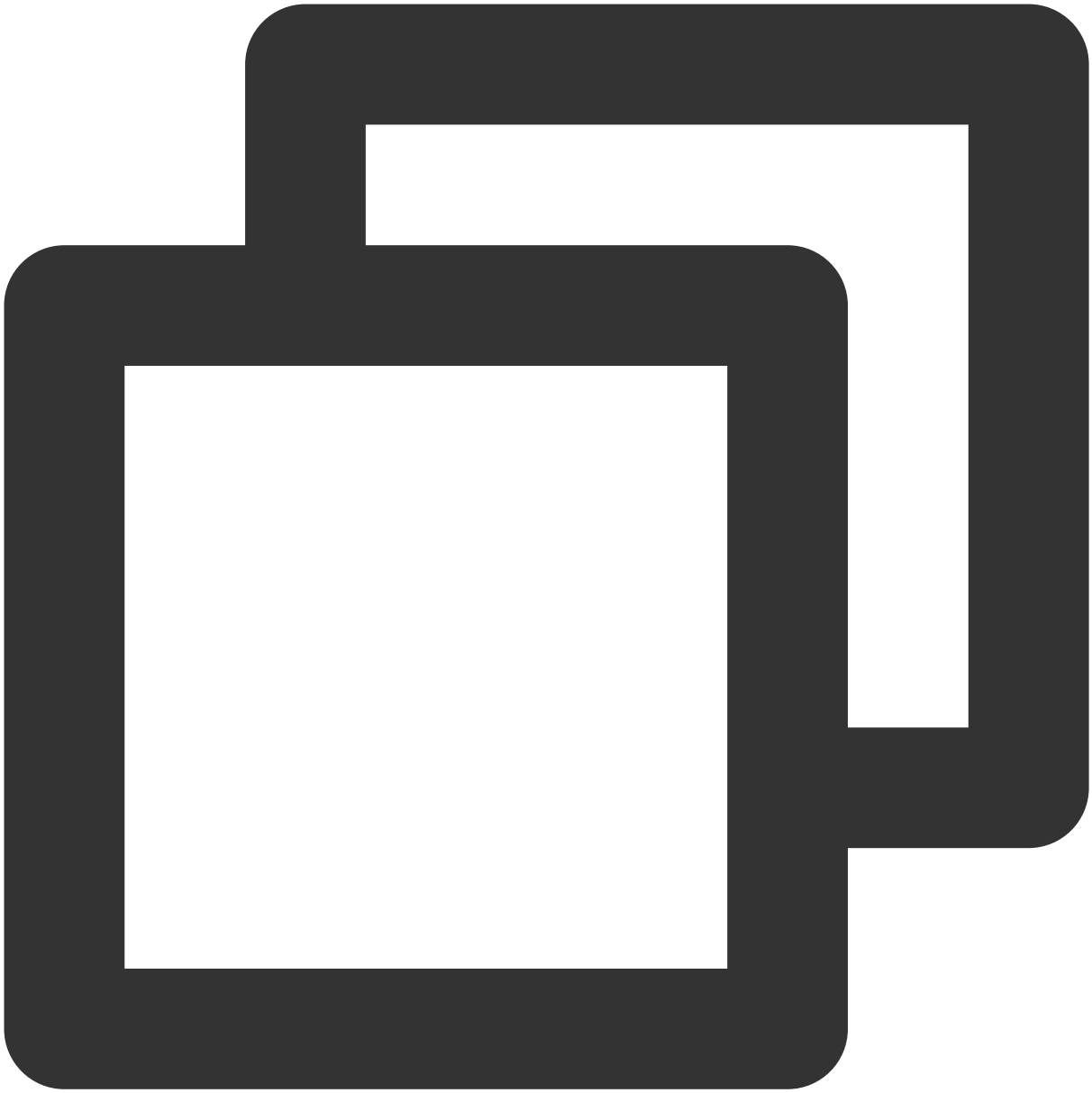
Terraform 是一种基础设施即代码（Infrastructure as Code）工具，使用 Terraform 可以编写可重复的、可维护的基础架构代码，并通过代码进行管理和更新，简化了云基础架构的部署和管理。Terraform 提供了丰富的资源星性和功模块，以及灵活的定制化配置选项，通过 Terraform，您可以轻松创建并管理云服务器、负载均衡等各种资源。

Terraform 功能与优势

Terraform 是一个功能强大的开源工具，用于管理云基础架构的自动化部署和编排。
Terraform 能够让您在腾讯云上使用简单的模板语句命令操作、构建、定义、部署和预览云基础架构。
Terraform 非常易于使用。它使用简单的 HCL 或 JSON 配置语言来定义基础设施，几乎不需要编写代码。
Terraform 的变量、模块和数据源特性，使得配置文件易于使用、重用和维护。
Terraform 可以在不同的阶段输出计划报告，方便用户了解实际执行的操作。
Terraform 可以通过使用状态文件记录基础设施的实际配置和状态信息，以及支持回滚操作。
说明：有关 Terraform 应用场景的具体介绍，请参见 应用场景。

Terraform 资源

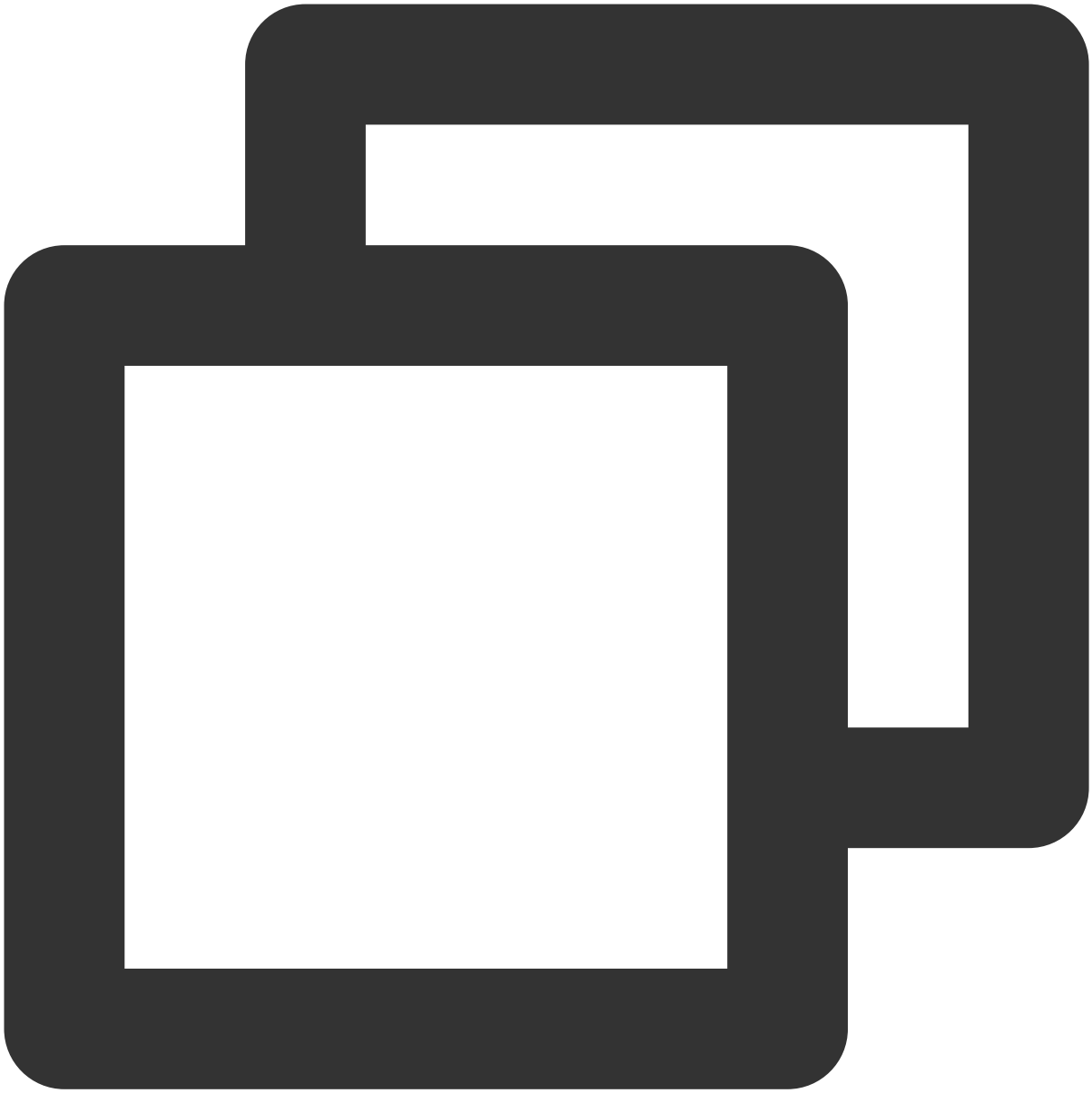
在 Terraform 中，资源主要分为资源（Resource）和数据源（Data Source）两类：
资源（Resource）
资源用于创建和管理新的云基础设施组件。通过资源定义，Terraform 可以创建、修改和删除云服务提供商支持的各种资源。例如，创建一个新的 cvm 实例。



```
#  cvm chc_assist_vpc 的资源。
# 参考文档：https://registry.terraform.io/providers/tencentcloudstack/tencentcloud/la
resource "tencentcloud_cvm_chc_assist_vpc" "chc_assist_vpc" {
  chc_id = "chc-xxxxx"
}
```

数据源（Data Source）

数据源用于查询和获取已存在的资源信息，并将其作为输入或属性在配置中使用。数据源可以检索已经存在的云基础设施中的配置和属性信息，以便在 Terraform 配置文件中进行引用和使用。例如，查询一个已经创建 cvm chc_denied_actions 的示例如下：



```
# 使用该数据源查询 cvm chc_denied_actions 的详细信息。
# 参考文档：https://registry.terraform.io/providers/tencentcloudstack/tencentcloud/la
data "tencentcloud_cvm_chc_denied_actions" "chc_denied_actions" {
  chc_ids = ["chc-xxxxx"]
}
```

关于可观测监控 Prometheus 版的 Resource 的相关信息，请参见通过 Terraform 使用。

Terraform 工具

Terraform CLI（命令行工具）：

Terraform CLI 是用于创建、管理和操作基础设施的命令行工具。它提供了一组命令和选项，让用户能够执行各种操作，如初始化配置、创建计划、应用变更等。Terraform CLI 还包括一些子命令，用于与 Terraform 相关的其他功能，如验证配置、格式化代码等。通过使用 Terraform CLI，用户可以与 Terraform 配置文件交互，并对基础设施进行管理。

Terraform Provider（提供者）：

每个云厂商都提供了自己的 Terraform Provider。提供者是一个插件，用于将云供应商的功能集成到 Terraform 中。每个 Terraform Provider 负责与相应的云服务提供商的 API 进行交互，并定义可用的资源和数据源，以及执行创建、修改和删除这些资源的操作，通过配置正确的提供者，在 Terraform 配置中可以直接使用云供应商的资源和功能。

说明：

Terraform CLI 是用于管理 Terraform 的命令行工具，而 Terraform Provider 则是用于将不同云供应商的功能集成到 Terraform 中的插件。这两个部分共同协作，使得用户能够使用 Terraform 来创建和管理基础设施。关于 Terraform 的更多信息，请参见 Terraform 使用文档。

使用优势

多云方案：Terraform 适用于多云方案，能够将类似的基础架构部署到 腾讯云 和 本地数据中心。由于 Terraform 具有可移植性，所以开发人员可以使用相同的工具和配置文件同时管理不同云提供商的资源。这种灵活性允许用户在任何时候切换到其他云平台，而不需要更改其配置文件。

自动化管理：Terraform 可以创建配置文件的模板，以可重复、可预测的方式定义、预配和配置资源，减少因人为因素导致的部署和管理错误。通过使用 Terraform，用户能够多次部署同一模板，创建相同的开发、测试和生产环境。此外，Terraform 还能自动化处理基础设施的创建和管理过程，提高了部署速度和效率。

基础设施即代码：Terraform 的基础设施即代码（IaC）方法允许用代码来管理资源，提供了版本控制、协作和重用代码的便利性。使用 Terraform，您可以将基础设施的状态保存下来，以便跟踪对系统进行的更改，并与其他人共享这些配置。这种可代码化的管理方式不仅使部署简单化，还能确保一致性、安全性和可维护性。

可视化界面：Terraform 图形界面能够查看和管理基础架构资源。这使得用户可以更加直观地查看和理解他们的基础设施，以及它们之间的相关性。通过这个可视化界面，用户可以更快、更容易地定位并修复潜在的配置问题。

社区支持：Terraform 是由 HashiCorp 公司开发的，并有一个庞大的社区支持。这个社区覆盖了各种行业和经验水平的专业人士，他们能够分享最佳实践和解决方案，提供论坛和邮件列表等支持。

通过 Terraform 使用腾讯云可观测平台 Prometheus 托管服务

Prometheus 支持通过 Terraform 管理以下 Resource：

名称	描述
tencentcloud_monitor_tmp_instance	Prometheus 实例

tencentcloud_monitor_tmp_alert_rule	Prometheus 告警规则
tencentcloud_monitor_tmp_recording_rule	Prometheus 预聚合规则
tencentcloud_monitor_tmp_cvm_agent	Prometheus Cvm Agent
tencentcloud_monitor_tmp_scrape_job	Prometheus cvm_agent 抓取任务
tencentcloud_monitor_tmp_exporter_integration	Prometheus 集成中心资源
tencentcloud_monitor_tmp_manage_grafana_attachment	Prometheus 绑定 grafana
tencentcloud_monitor_tmp_tke_cluster_agent	Prometheus 关联容器
tencentcloud_monitor_tmp_tke_config	Prometheus 集群采集配置

使用 Terraform 管理 Prometheus 实例

最近更新时间：2023-12-29 16:08:00

前提条件

安装 Terraform

关于安装 Terraform 的具体操作，请参见在 本地安装和配置 Terraform。

说明：

Terraform 安装版本不得低于 v1.6.3，您可通过 terraform --version 命令查看安装的 Terraform 版本。

配置腾讯云账号信息

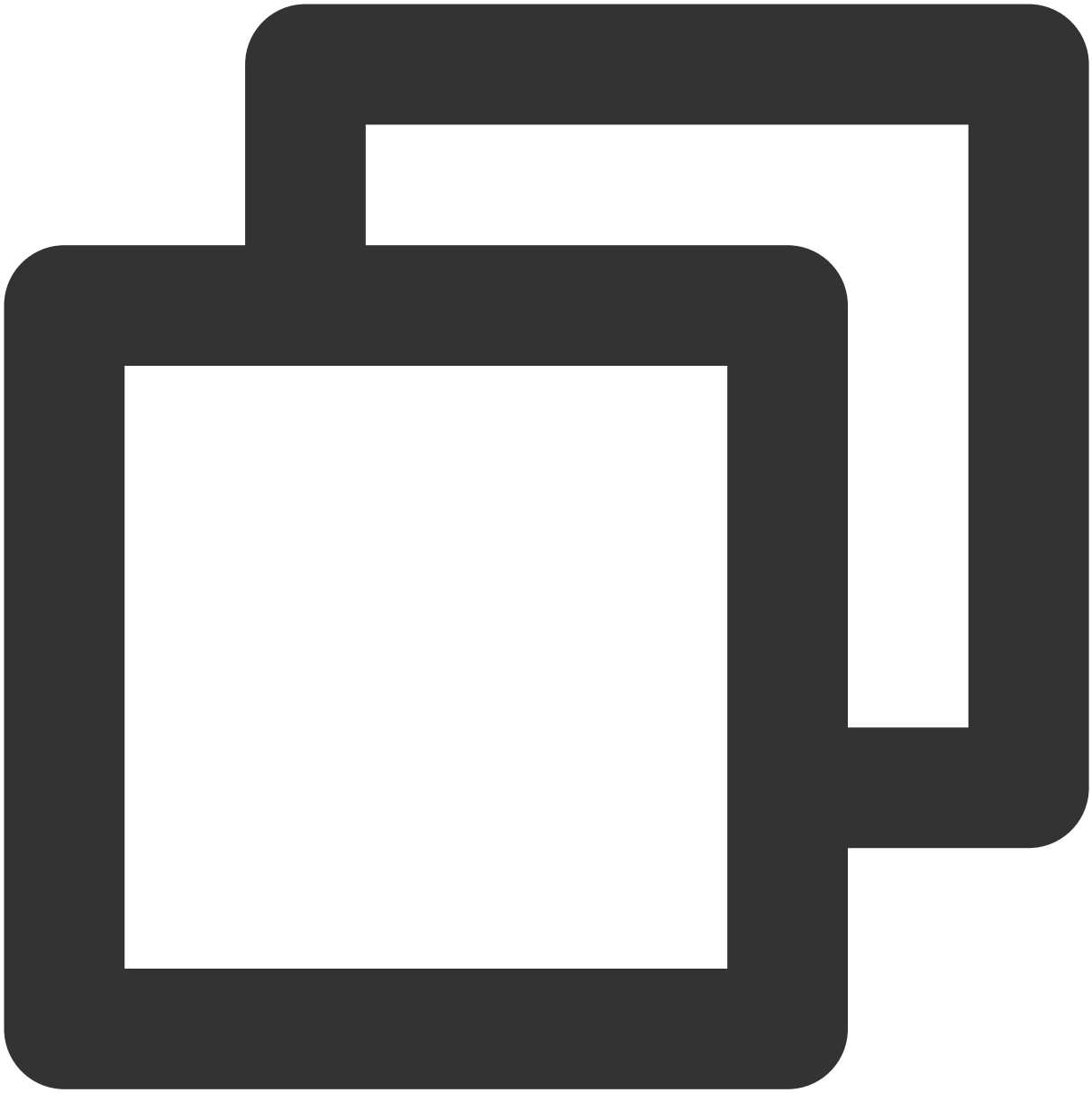
在首次使用 Terraform 之前，请前往 云 API 密钥 申请安全凭证 SecretId 和 SecretKey。如果已有可使用的安全凭证，则跳过该步骤。

1. 登录 访问管理控制台，在左侧导航栏，进入**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。

配置腾讯云账号信息，有以下两种方式：

静态凭证鉴权

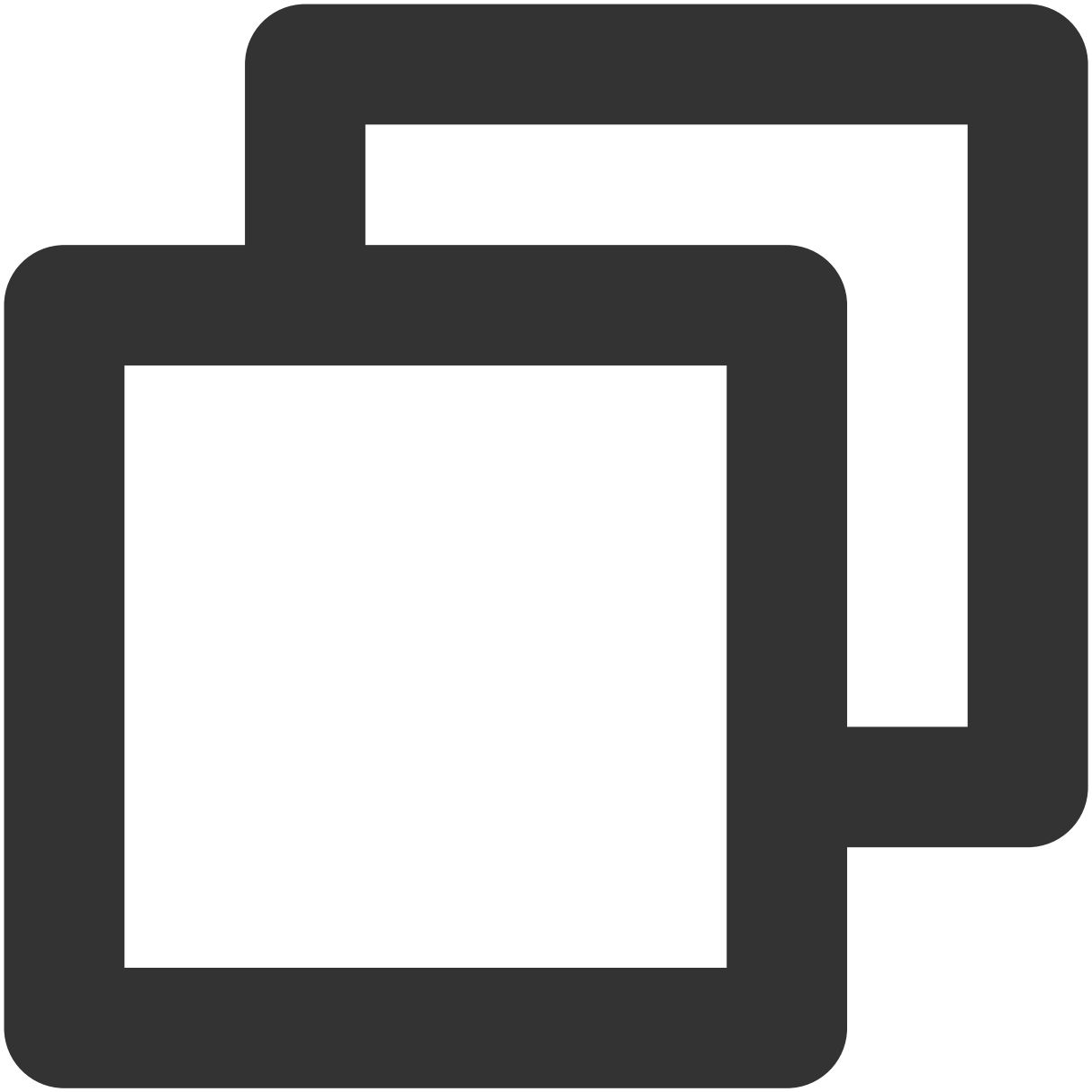
在用户目录下创建 `provider.tf` 文件，输入如下内容。其中 `my-secret-id` 和 `my-secret-key` 需替换为密钥 SecretId 和 SecretKey。



```
provider "tencentcloud" {
  secret_id = "my-secret-id"
  secret_key = "my-secret-key"
}
```

环境变量鉴权

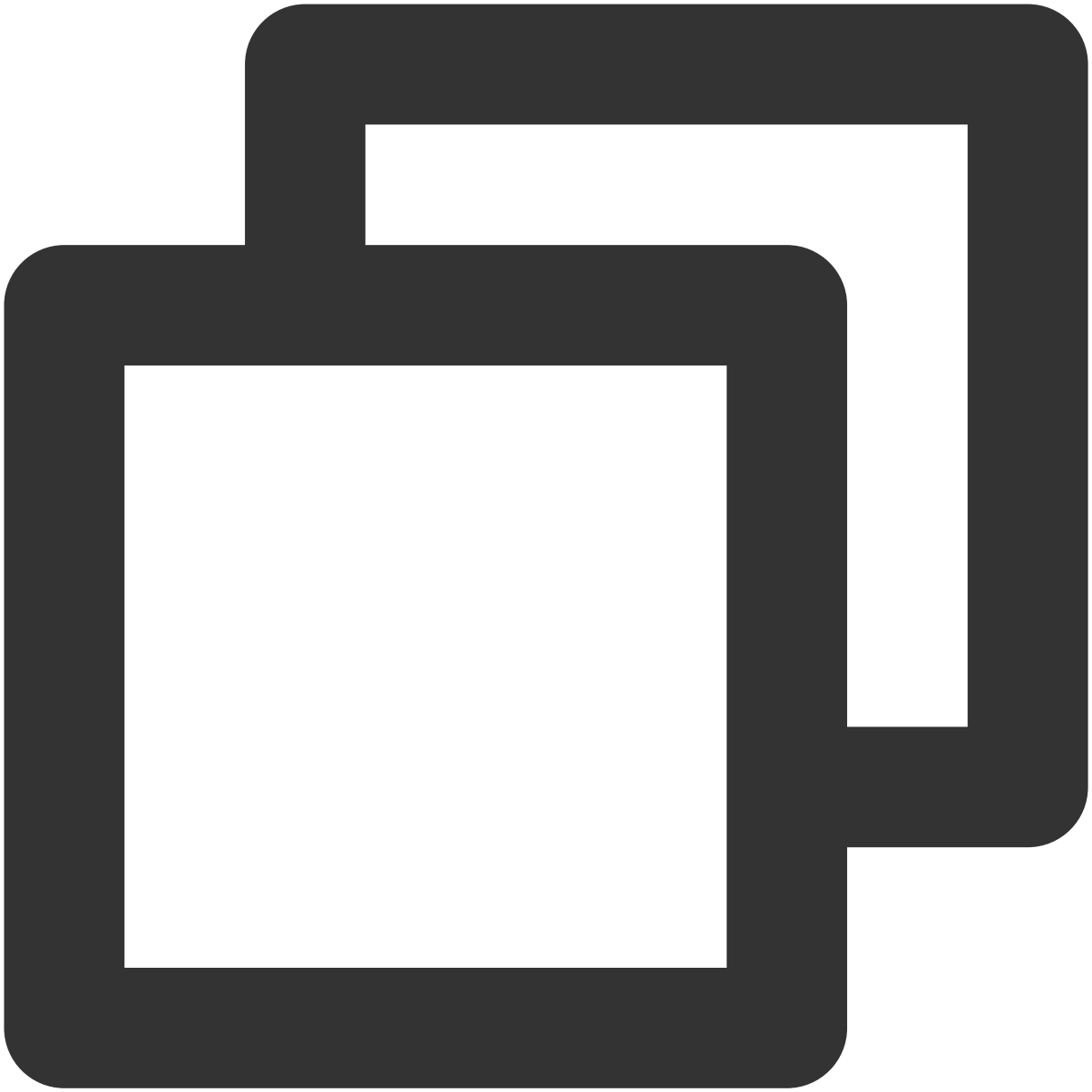
配置电脑环境变量或云端环境变量，请执行以下命令。其中 `YOUR_SECRET_ID` 和 `YOUR_SECRET_KEY` 需替换为密钥 SecretId 和 SecretKey。



```
export TENCENTCLOUD_SECRET_ID=YOUR_SECRET_ID
export TENCENTCLOUD_SECRET_KEY=YOUR_SECRET_KEY
```

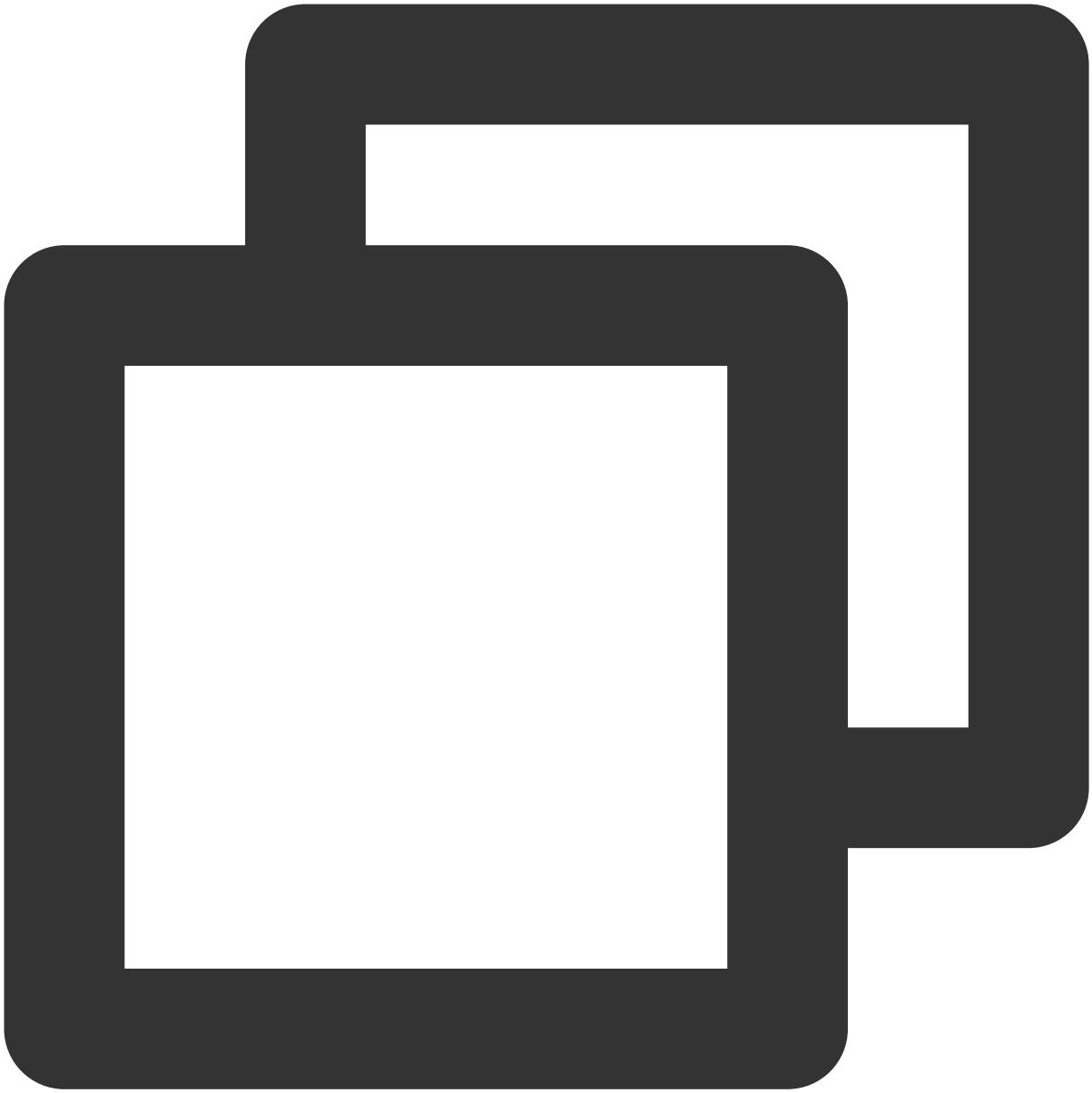
创建 prometheus 实例

1. 创建 Terraform 的配置文件。在项目目录中创建一个名为 main.tf 的文件，并使用合适的编辑器打开，在 main.tf 文件或者 provider.tf 文件中添加腾讯云提供者配置：



```
provider "tencentcloud" {
  region     = "your_region"
  secret_id  = "your_secret_id"
  secret_key = "your_secret_key"
}
```

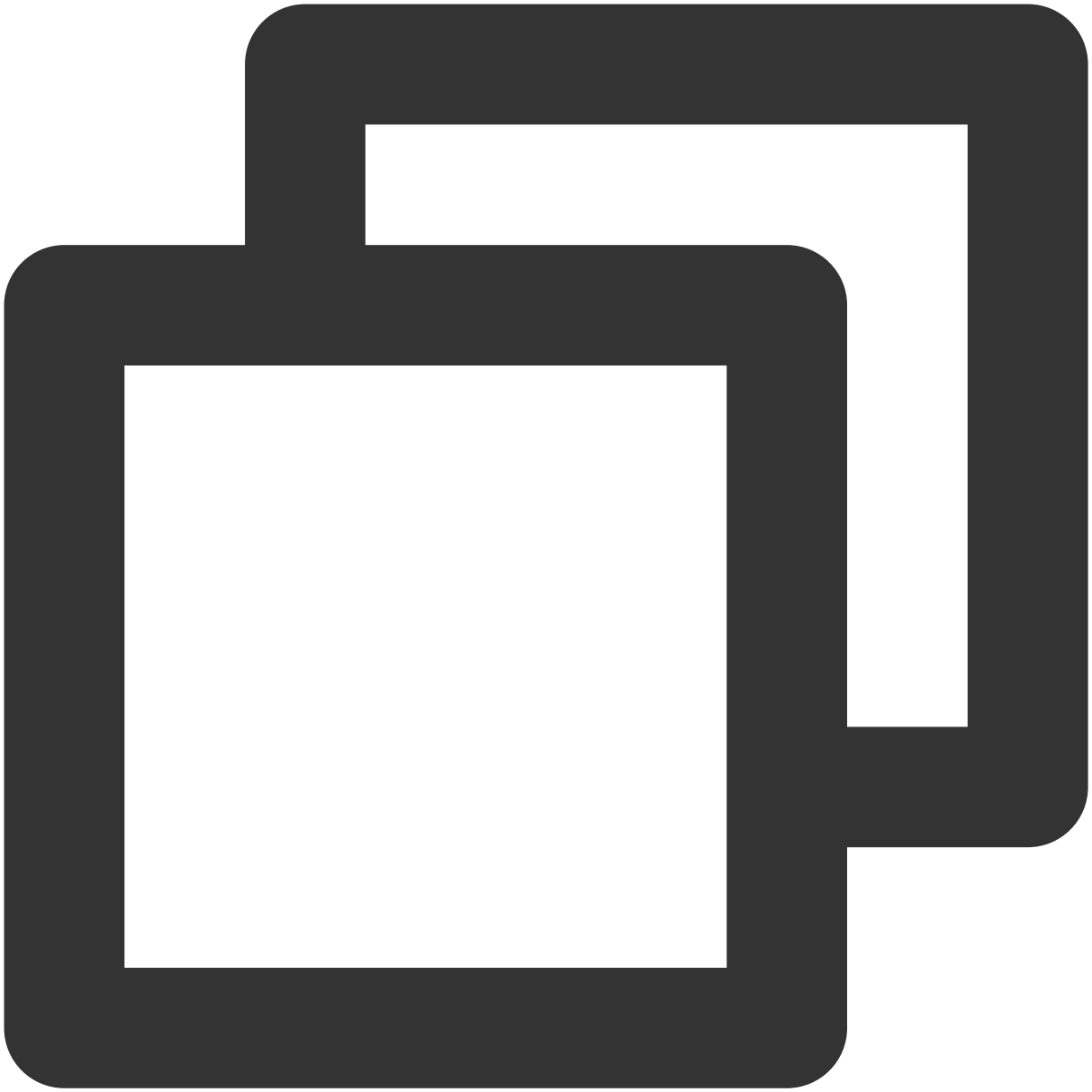
2. 定义腾讯云资源：在 main.tf 文件中使用 Terraform 腾讯云提供的资源来定义您想要创建和管理的资源。



```
 # 指定 provider 配置信息

 terraform {
  required_providers {
    tencentcloud = {
      source = "tencentcloudstack/tencentcloud"
    }
  }
}

 # 创建孟买地区prometheus实例
```

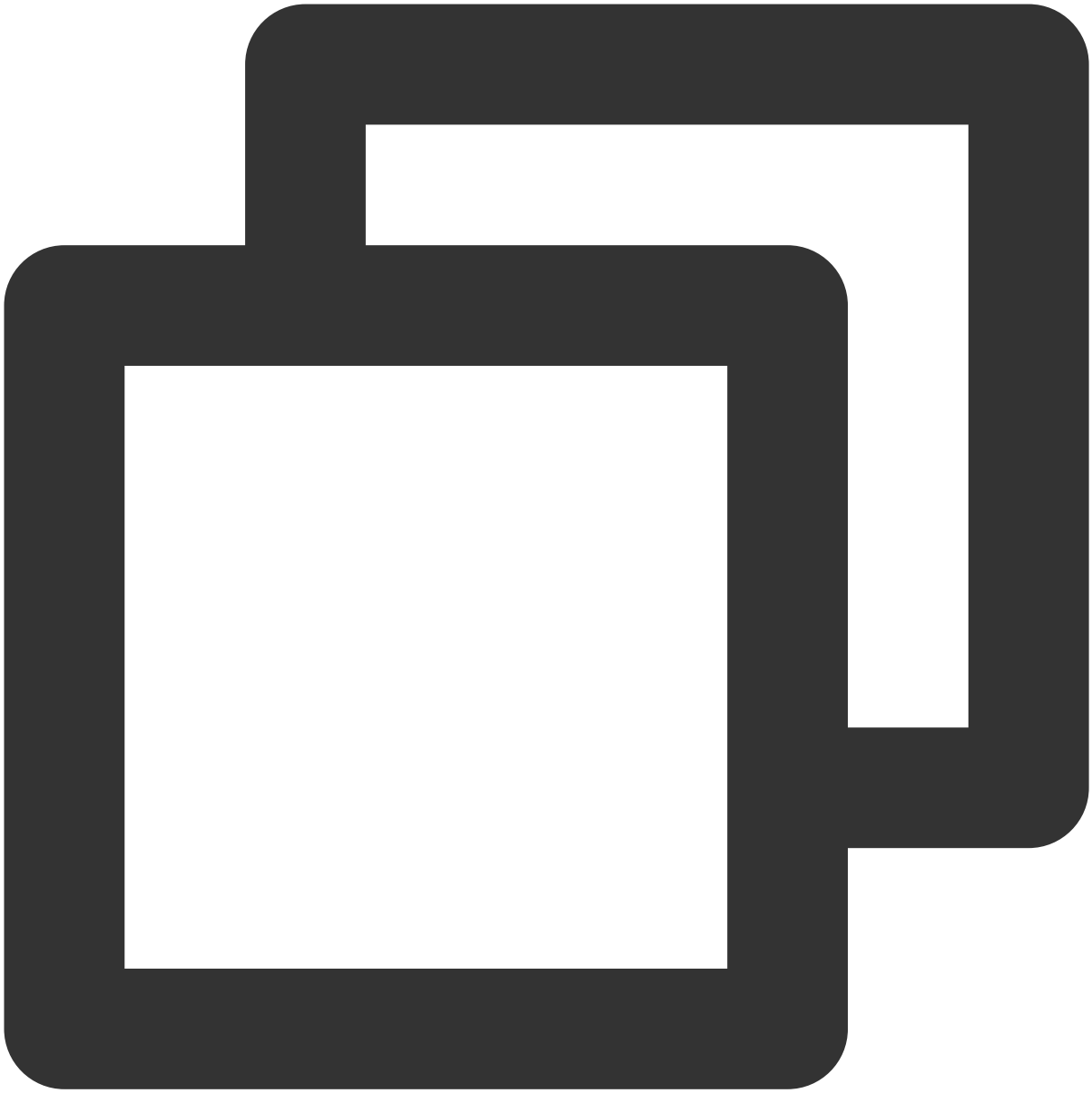
```
resource "tencentcloud_monitor_tmp_instance" "foo" {
  instance_name       = "tf-tmp-instance-sjtest"
  vpc_id              = "vpc-0n42dxzs"
  subnet_id           = "subnet-es8rv1kx"
  data_retention_time = 30
  zone                = "ap-mumbai-1"
  tags = {
    "createdBy" = "terraform"
  }
}

# 创建孟买地区grafana实例（选填）

resource "tencentcloud_monitor_grafana_instance" "foo" {
  instance_name         = "tf-grfana-cstest"
  vpc_id                = "vpc-0n42dxzs"
  subnet_ids            = ["subnet-es8rv1kx"]
  grafana_init_password = "1234567890"
  enable_internet       = false
  is_destroy            = true

  tags = {
    "createdBy" = "test"
  }
}

# grafana与Prometheus实例做绑定（选填）
resource "tencentcloud_monitor_tmp_manage_grafana_attachment" "foo" {
  grafana_id  = tencentcloud_monitor_grafana_instance.foo.id
  instance_id = tencentcloud_monitor_tmp_instance.foo.id
}
```

3. 初始化 Terraform 运行环境，执行命令如下：



```
terraform init
```

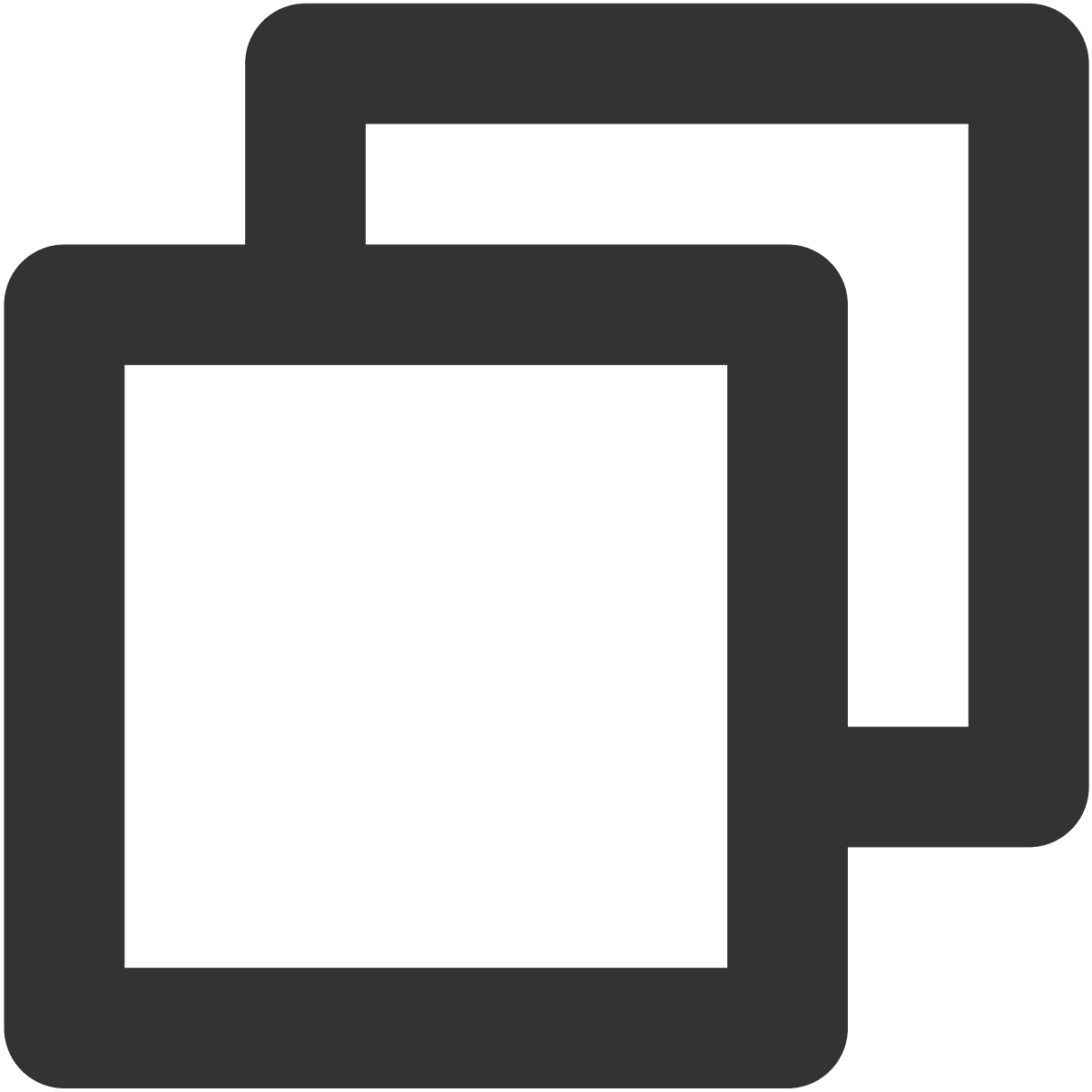
预期输出信息：



```
  Initializing the backend...

Initializing provider plugins...
- Reusing previous version of tencentcloudstack/tencentcloud from the dependency lo
- Using previously-installed tencentcloudstack/tencentcloud v1.81.32

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
```

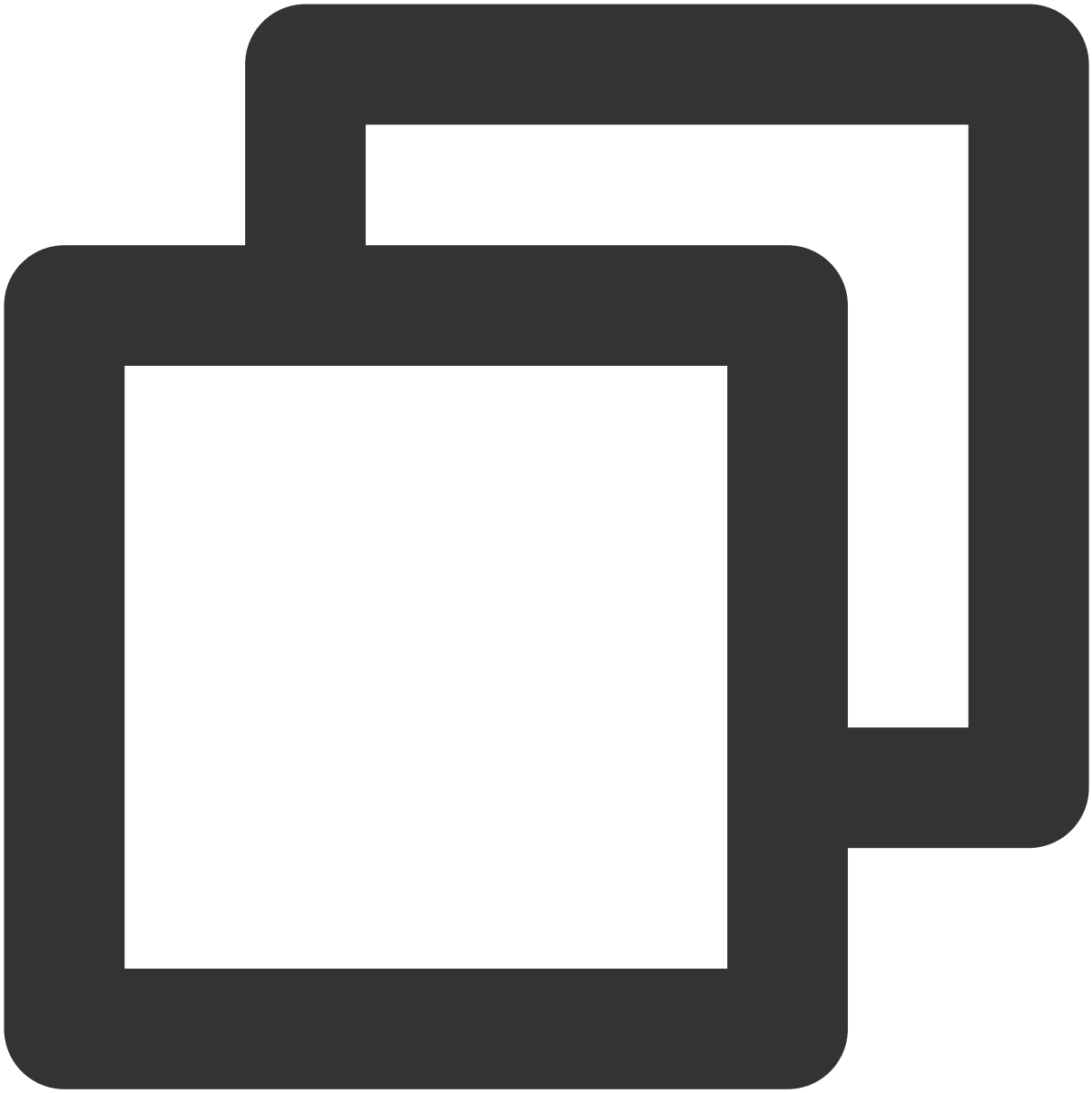
```
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

4. 生成资源规划，执行命令如下：



```
terraform plan
```

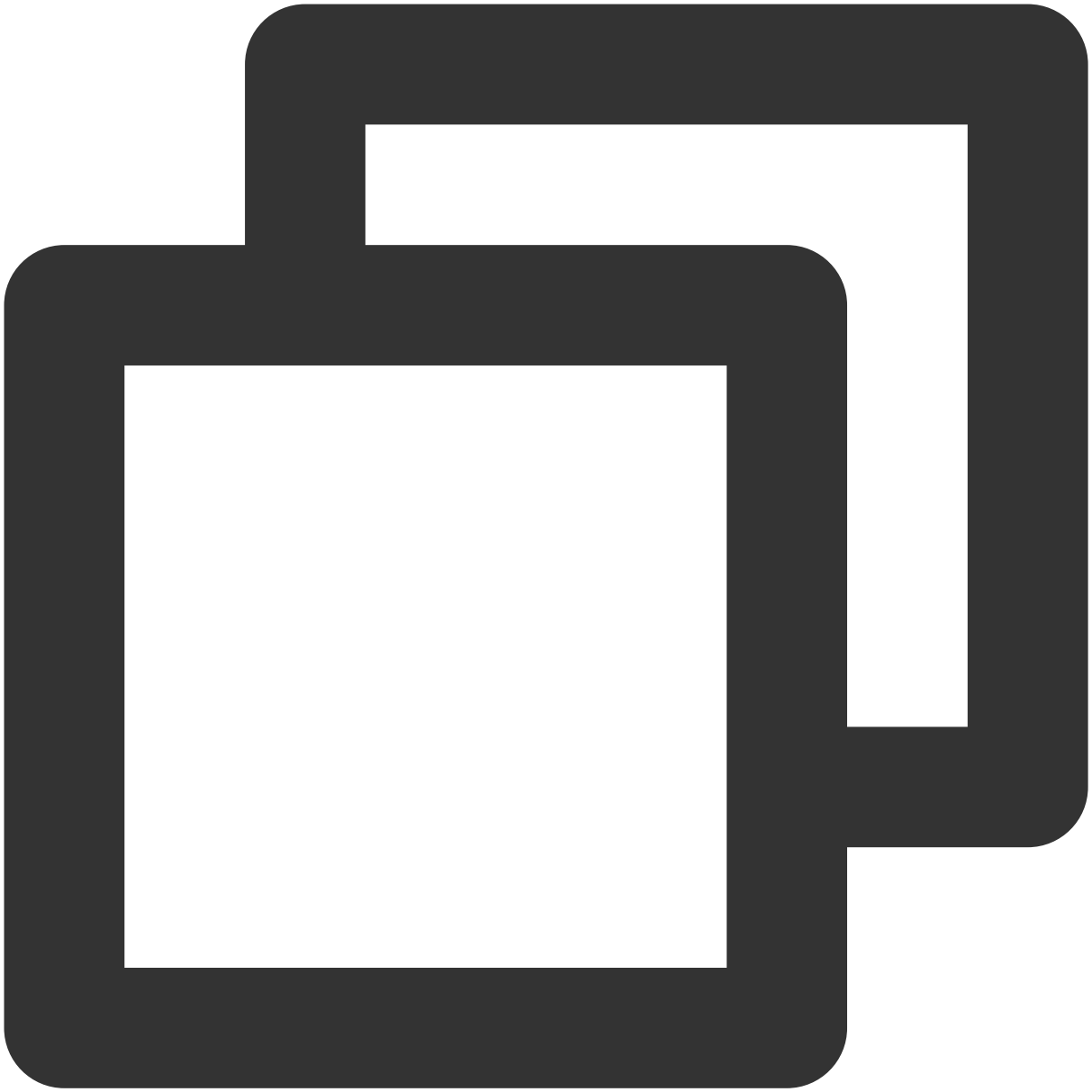
预期输出信息：



```
  tencentcloud_vpc.vpc: Refreshing state... [id=vpc-3csjp7k8]
tencentcloud_monitor_tmp_instance.foo: Refreshing state... [id=prom-4wcvt7p1]

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

Terraform will perform the following actions:

  # tencentcloud_monitor_tmp_instance.foo will be created
  + resource "tencentcloud_monitor_tmp_instance" "foo" {
```

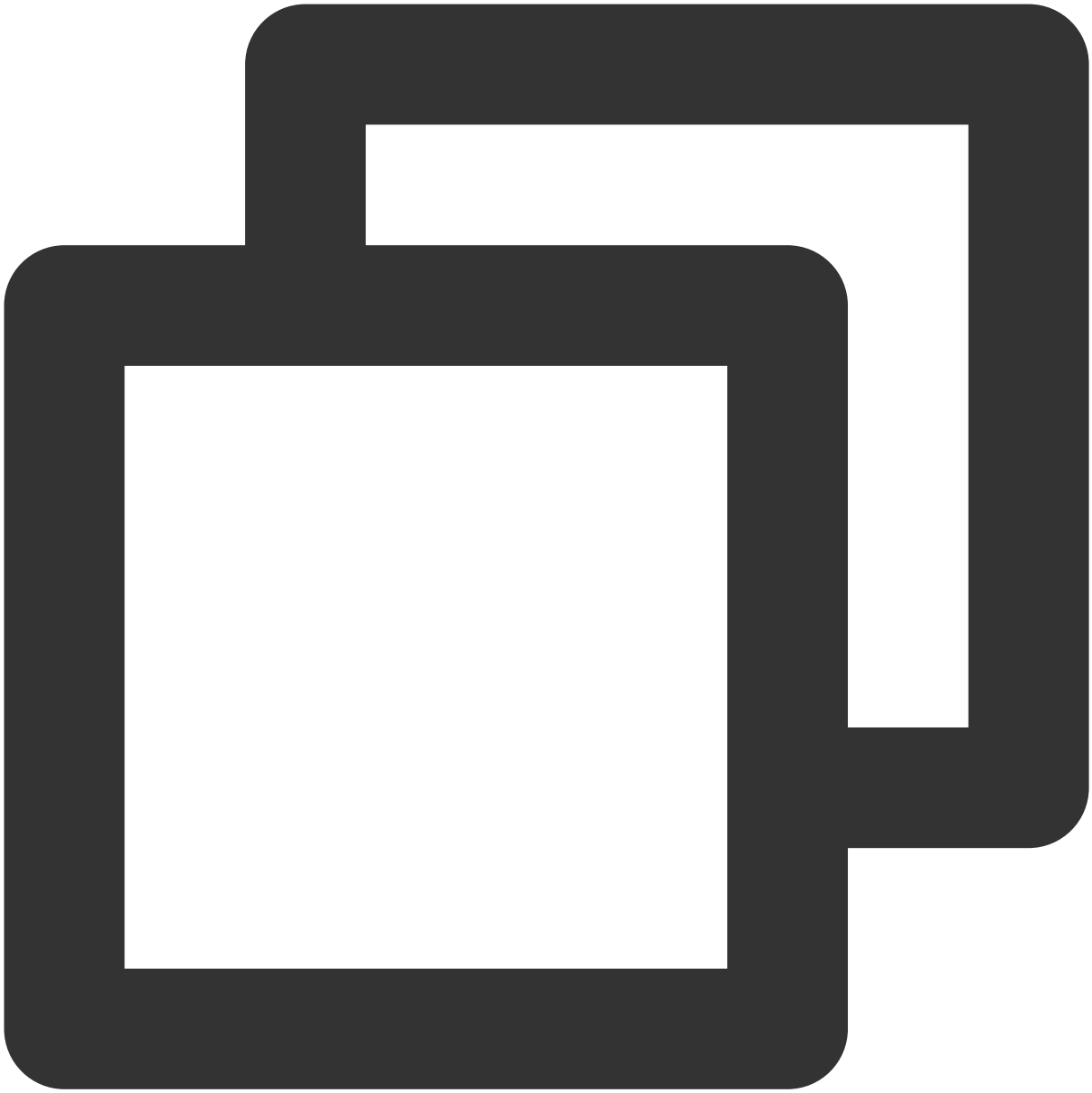
```
      + api_root_path       = (known after apply)
      + data_retention_time = 30
      + id                  = (known after apply)
      + instance_name       = "tf-tmp-instance-sjtest"
      + ipv4_address        = (known after apply)
      + proxy_address       = (known after apply)
      + remote_write        = (known after apply)
      + subnet_id           = "subnet-es8rv1kx"
      + tags                = {
          + "createdBy" = "terraform"
        }
      + vpc_id              = "vpc-0n42dxzs"
      + zone                = "ap-mumbai-1"
    }

Plan: 1 to add, 0 to change, 0 to destroy.
```

5. 创建实例，执行命令如下：



```
terraform apply
```

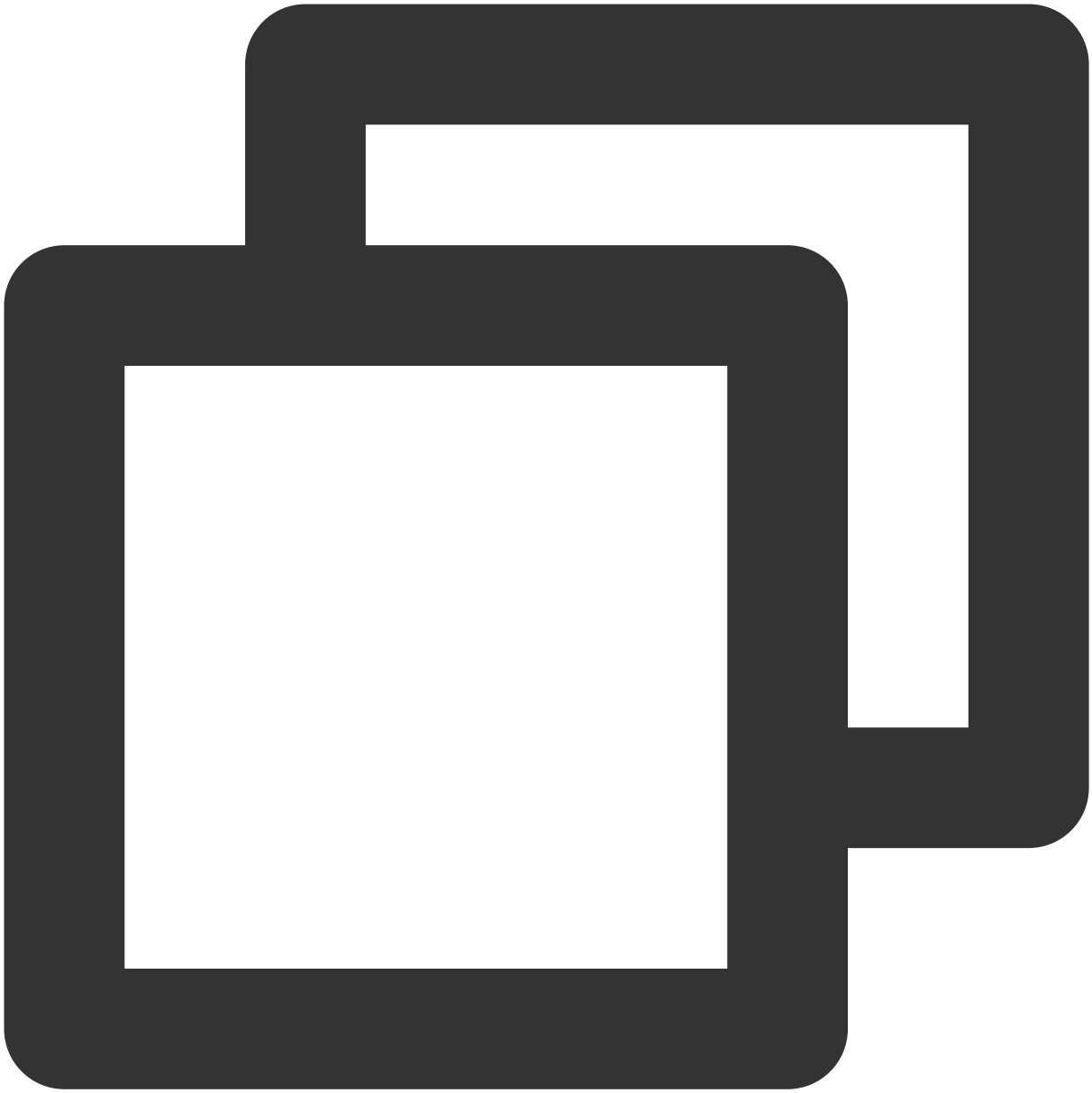
预期输出信息：



```
tencentcloud_vpc.vpc: Refreshing state... [id=vpc-3csjp7k8]
tencentcloud_monitor_tmp_instance.foo: Refreshing state... [id=prom-4wcvt7p1]

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

Terraform will perform the following actions:

  # tencentcloud_monitor_tmp_instance.foo will be created
  + resource "tencentcloud_monitor_tmp_instance" "foo" {
```

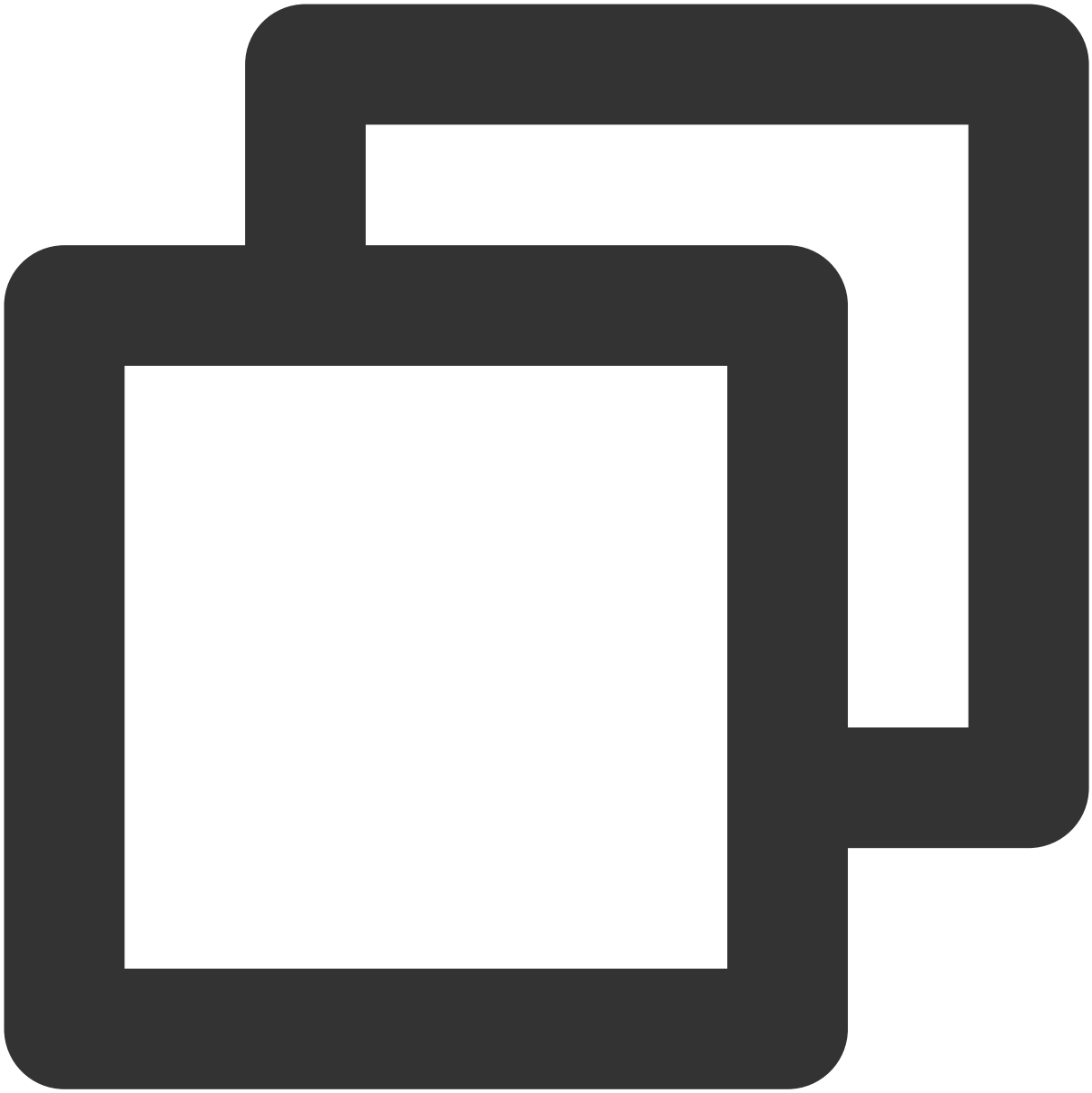
```
      + api_root_path       = (known after apply)
      + data_retention_time = 30
      + id                  = (known after apply)
      + instance_name       = "tf-tmp-instance-sjtest"
      + ipv4_address        = (known after apply)
      + proxy_address       = (known after apply)
      + remote_write        = (known after apply)
      + subnet_id           = "subnet-es8rv1kx"
      + tags                = {
          + "createdBy" = "terraform"
        }
      + vpc_id              = "vpc-0n42dxzs"
      + zone                = "ap-mumbai-1"
    }

Plan: 1 to add, 0 to change, 0 to destroy.
```

以下信息填入 “ yes ” 继续操作：



```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value:
```

若出现以下信息，表示您已创建成功：



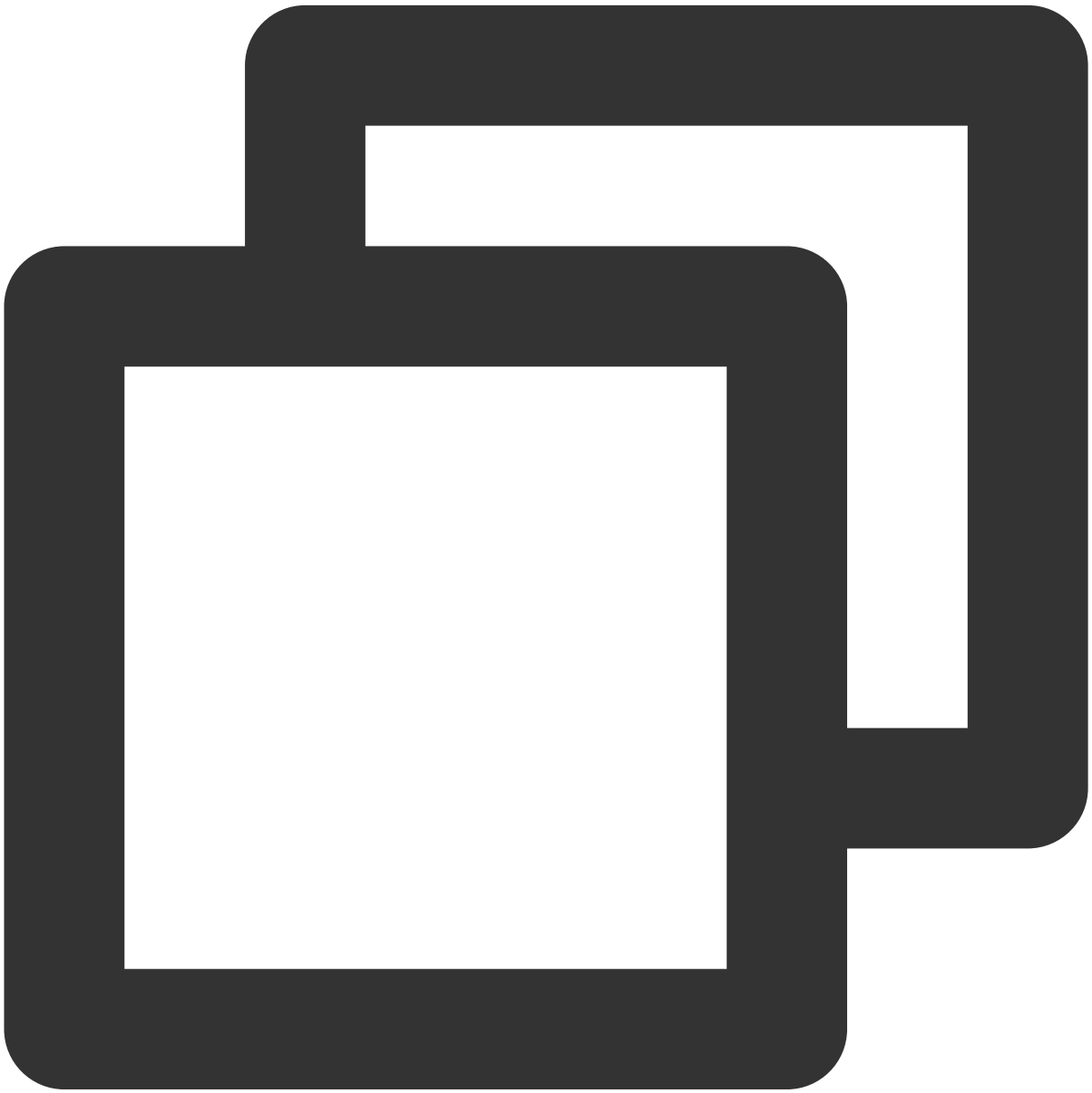
```
tencentcloud_monitor_tmp_instance.foo: Creating...
tencentcloud_monitor_tmp_instance.foo: Still creating... [10s elapsed]
tencentcloud_monitor_tmp_instance.foo: Creation complete after 12s [id=prom-8dyb6in

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

查看 Prometheus 实例状态

登录 腾讯云可观测平台，在左侧导航栏选择 **prometheus 监控**，可在 prometheus 实例列表中看到存在的实例。

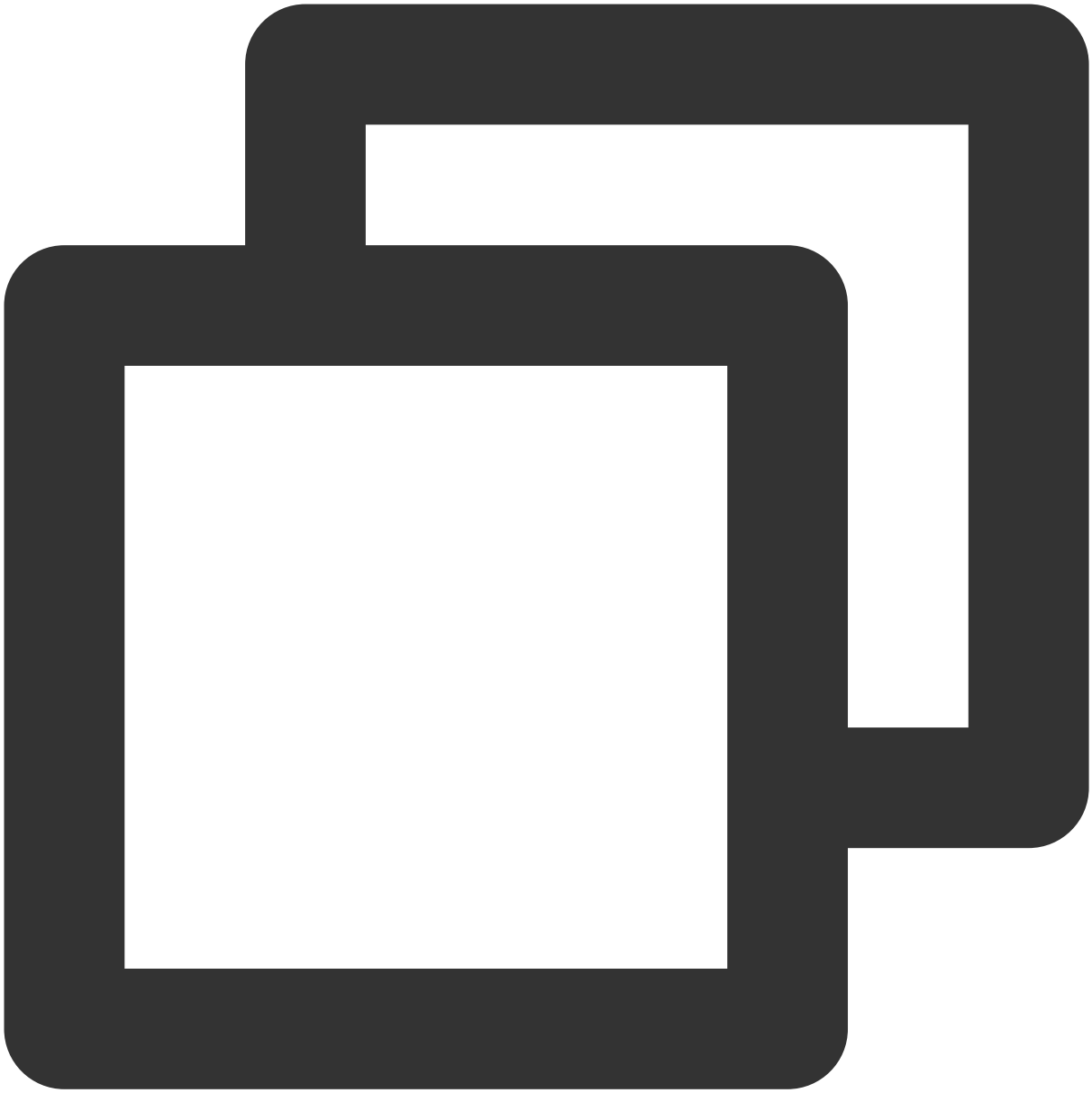
删除 Prometheus 实例

删除 Prometheus 实例，执行命令如下：



```
terraform destroy
```

出现以下信息填入提示您填入“yes”确认：



```
tencentcloud_monitor_tmp_instance.foo: Refreshing state... [id=prom-8dyb6iny]

Terraform used the selected providers to generate the following execution plan. Res
following symbols:
  - destroy

Terraform will perform the following actions:

  # tencentcloud_monitor_tmp_instance.foo will be destroyed
  - resource "tencentcloud_monitor_tmp_instance" "foo" {
      - api_root_path       = "http://10.0.0.34:9090/api/v1" -> null
```

```
      - data_retention_time = 30 -> null
      - id                  = "prom-8dyb6iny" -> null
      - instance_name       = "tf-tmp-instance-sjtest" -> null
      - ipv4_address        = "10.0.0.34" -> null
      - proxy_address       = "10.0.0.34:9090" -> null
      - remote_write        = "http://10.0.0.34:9090/api/v1/prom/write" -> null
      - subnet_id           = "subnet-es8rv1kx" -> null
      - tags                = {
          - "createdBy" = "terraform"
        } -> null
      - vpc_id              = "vpc-0n42dxzs" -> null
      - zone                = "ap-mumbai-1" -> null
    }

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

tencentcloud_monitor_tmp_instance.foo: Destroying... [id=prom-8dyb6iny]
tencentcloud_monitor_tmp_instance.foo: Destruction complete after 6s

Destroy complete! Resources: 1 destroyed.
```

说明：

当出现 `Destroy complete! Resources: 1 destroyed.` 表示您已删除该实例。

使用 Terraform 管理 Prometheus 实例的集成中心

最近更新时间：2023-12-29 16:09:03

前提条件

安装 Terraform

关于安装 Terraform 的具体操作，请参见在 本地安装和配置 Terraform。

说明：

Terraform 安装版本不得低于 v1.6.3，您可通过 terraform --version 命令查看安装的 Terraform 版本。

配置腾讯云账号信息

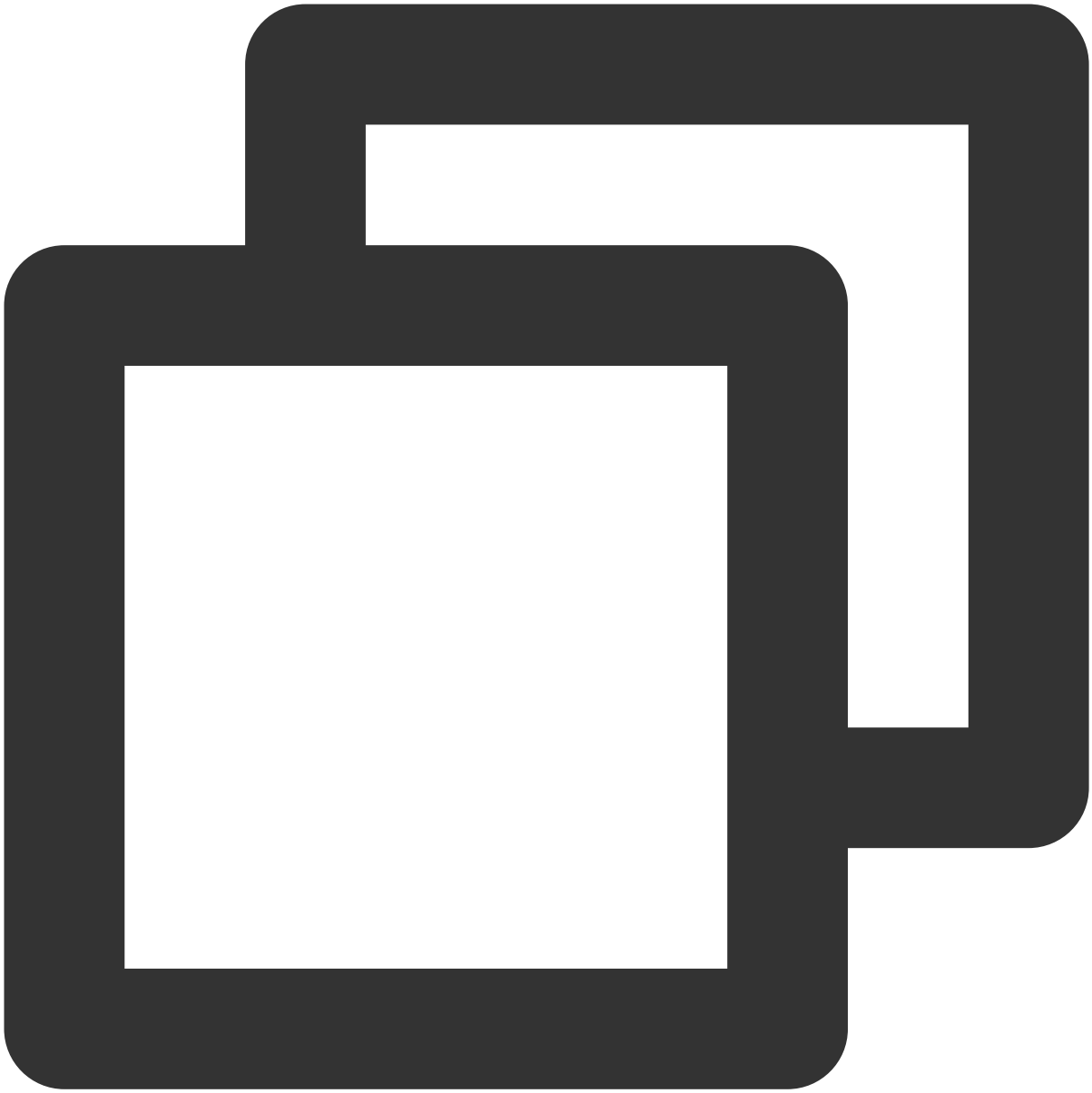
在首次使用 Terraform 之前，请前往 云 API 密钥 申请安全凭证 SecretId 和 SecretKey。若已有可使用的安全凭证，则跳过该步骤。

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。

配置腾讯云账号信息，有以下两种方式：

静态凭证鉴权

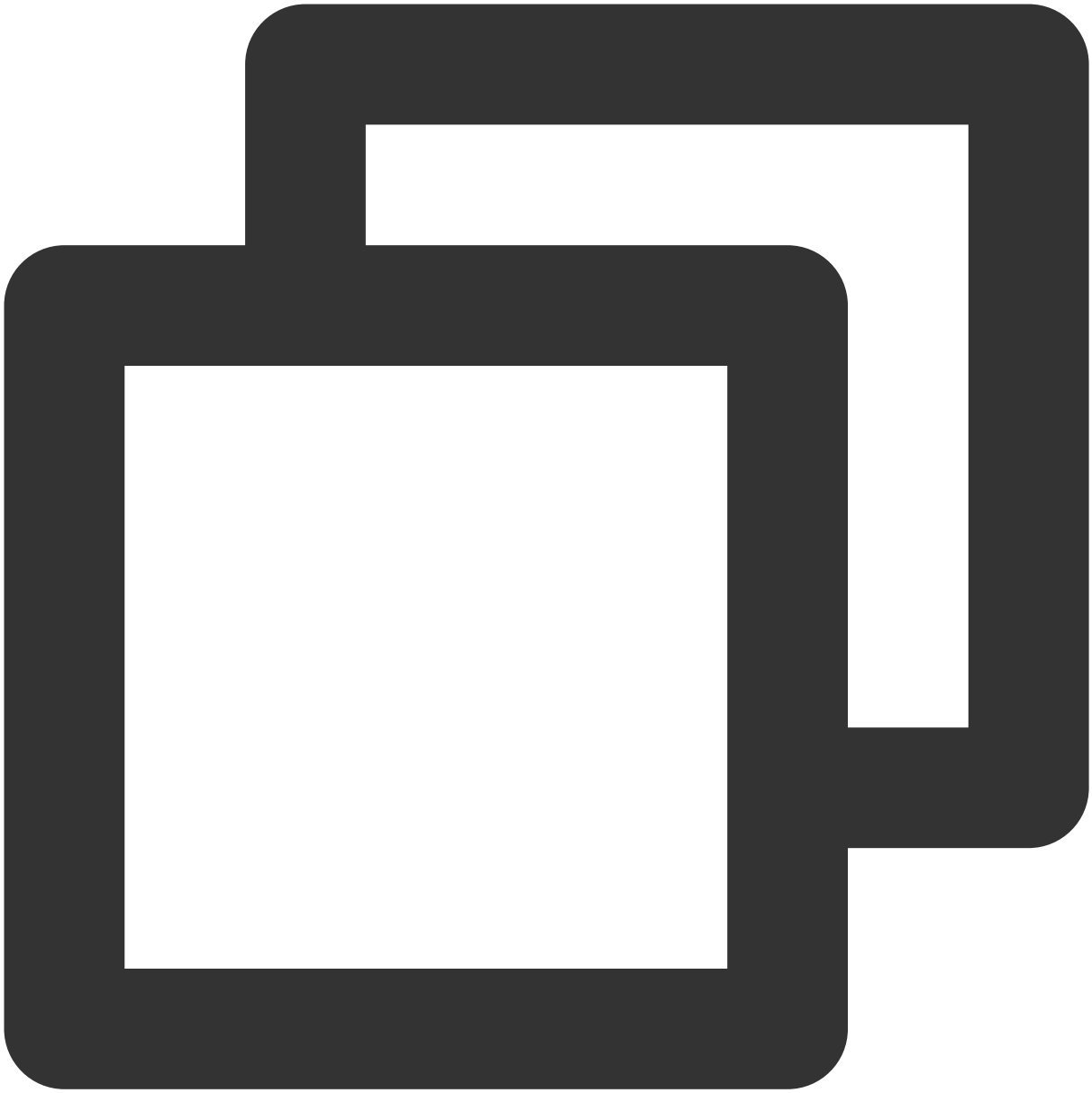
在用户目录下创建 `provider.tf` 文件，输入如下内容。其中 `my-secret-id` 和 `my-secret-key` 需替换为密钥 SecretId 和 SecretKey。



```
provider "tencentcloud" {
  secret_id = "my-secret-id"
  secret_key = "my-secret-key"
}
```

环境变量鉴权

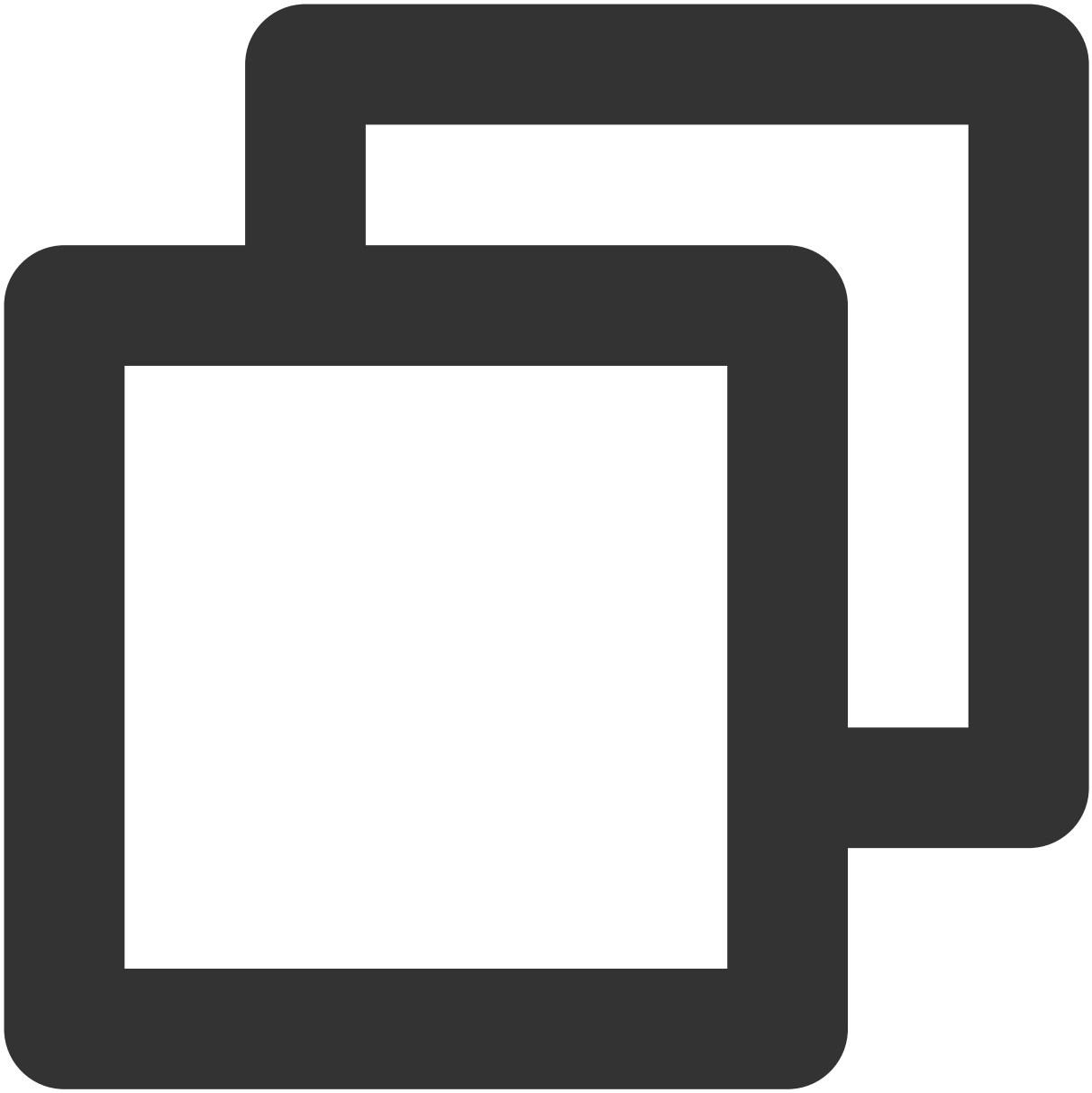
配置电脑环境变量或云端环境变量，请执行以下命令。其中 `YOUR_SECRET_ID` 和 `YOUR_SECRET_KEY` 需替换为密钥 SecretId 和 SecretKey。



```
export TENCENTCLOUD_SECRET_ID=YOUR_SECRET_ID
export TENCENTCLOUD_SECRET_KEY=YOUR_SECRET_KEY
```

增加 Prometheus 实例的集成中心组件集成

1. 创建一个新的 Terraform 配置文件：创建一个新的目录，并在该目录下创建一个 main.tf 的文件，配置如下信息：



```
# 指定 provider 配置信息

terraform {
  required_providers {
    tencentcloud = {
      source = "tencentcloudstack/tencentcloud"
    }
  }
}

# prometheus管理云监控集成
```

```
## black-box 集成
resource "tencentcloud_monitor_tmp_exporter_integration" "tmpExporterIntegration" {
  instance_id = tencentcloud_monitor_tmp_instance.foo.id
  kind        = "blackbox-exporter"
  content     = "{\\"name\\":\\"balck-box-tf-test\\",\\"kind\\":\\"blackbox-exporte
  kube_type   = 1
  cluster_id  = ""
}

## 云监控插件集成
resource "tencentcloud_monitor_tmp_exporter_integration" "tmpExporterMointor" {
  instance_id = tencentcloud_monitor_tmp_instance.foo.id
  kind        = "qcloud-exporter"
  content     = "{\\"name\\":\\"tf-test-cjtest\\",\\"kind\\":\\"qcloud-exporter\\",
  kube_type   = 1
  cluster_id  = ""
}
```

说明：

创建 Prometheus 实例的集成中心组件时配置的字段如下：

cluster_id：（必填，字符串）集群 ID。

content：（必填，字符串）集成配置（content 内部参数可自行根据需要更换）。

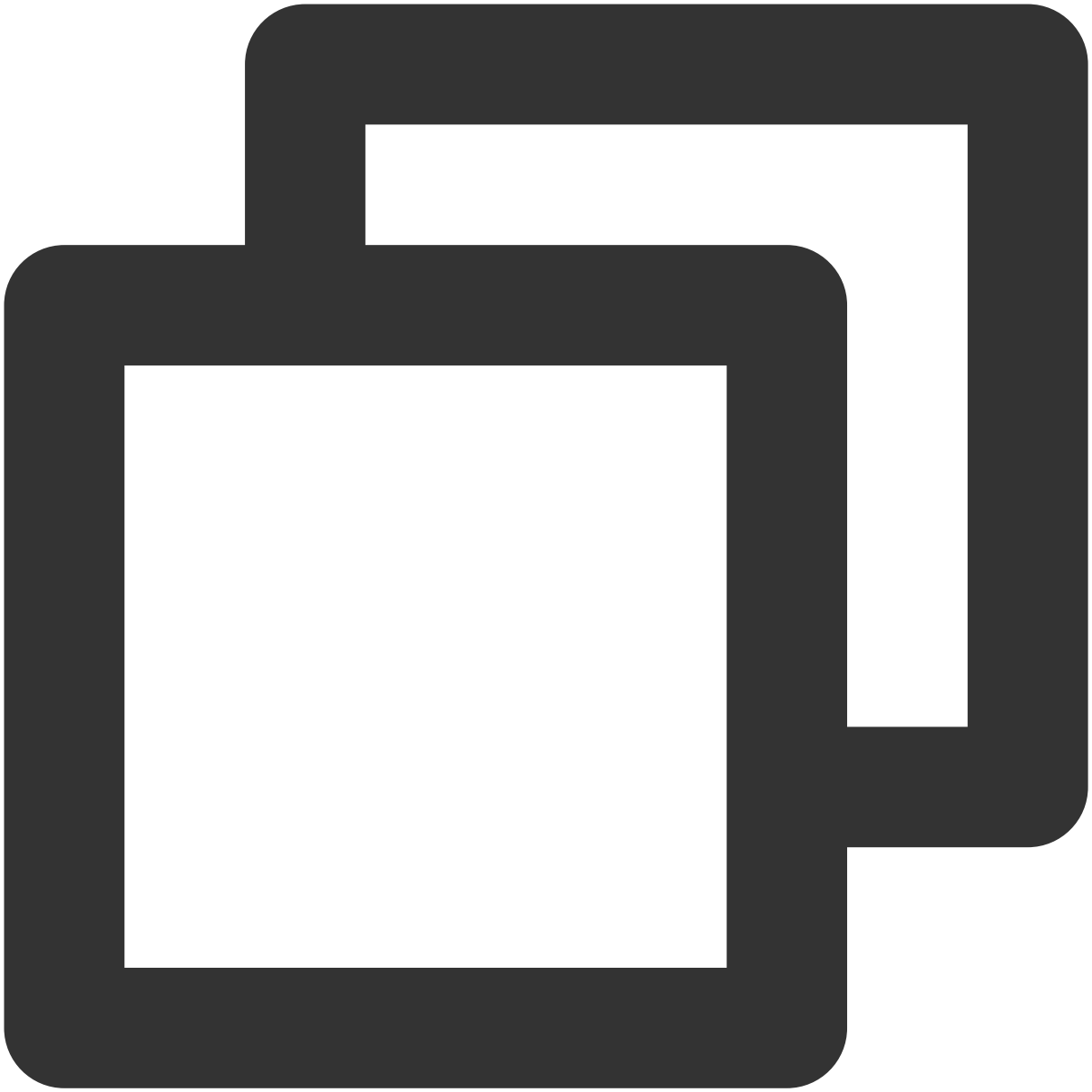
instance_id：（必填，字符串）实例 ID。

kind：（必填，字符串）类型。

kube_type：（必填，整数）集成配置。

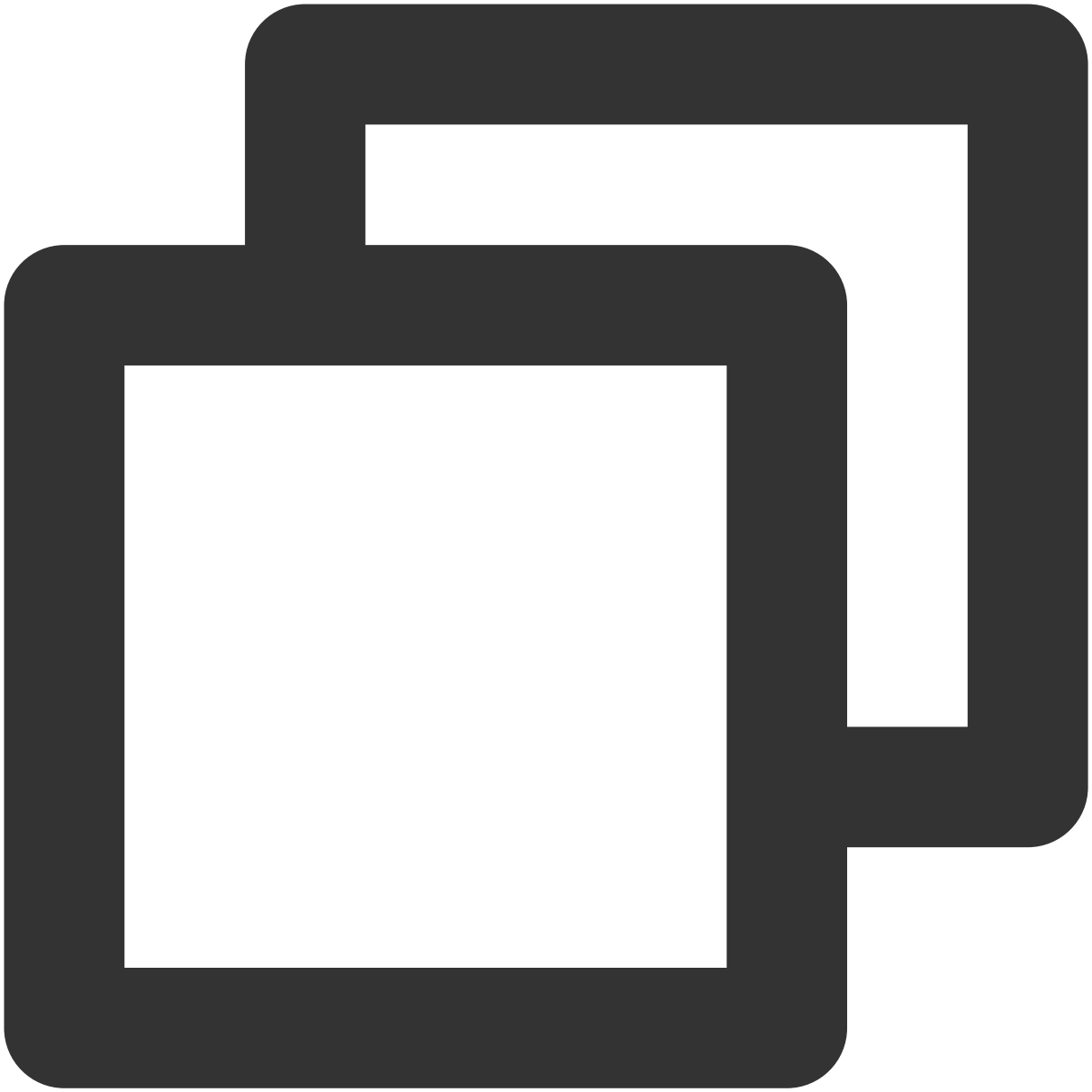
您若需要获取更多详细参数，请参考 腾讯接入云集成接入 github，也可以通过 Terraform 腾讯云提供商 了解更多。

2. 初始化 Terraform 运行环境，执行命令如下：



```
terraform init
```

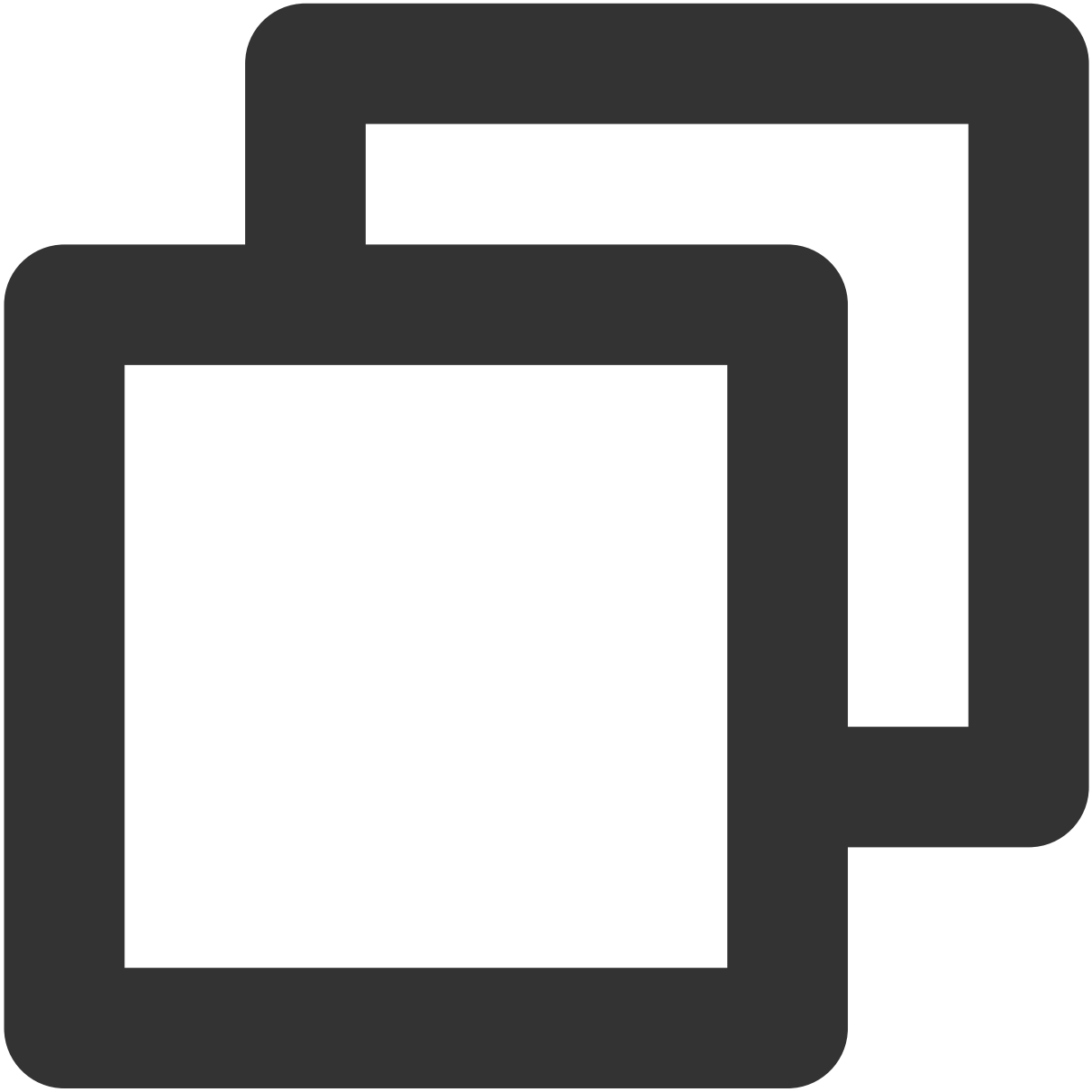
预期输出信息：



```
Initializing the backend...

Initializing provider plugins...
- Reusing previous version of tencentcloudstack/tencentcloud from the dependency lo
- Using previously-installed tencentcloudstack/tencentcloud v1.81.32

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
```

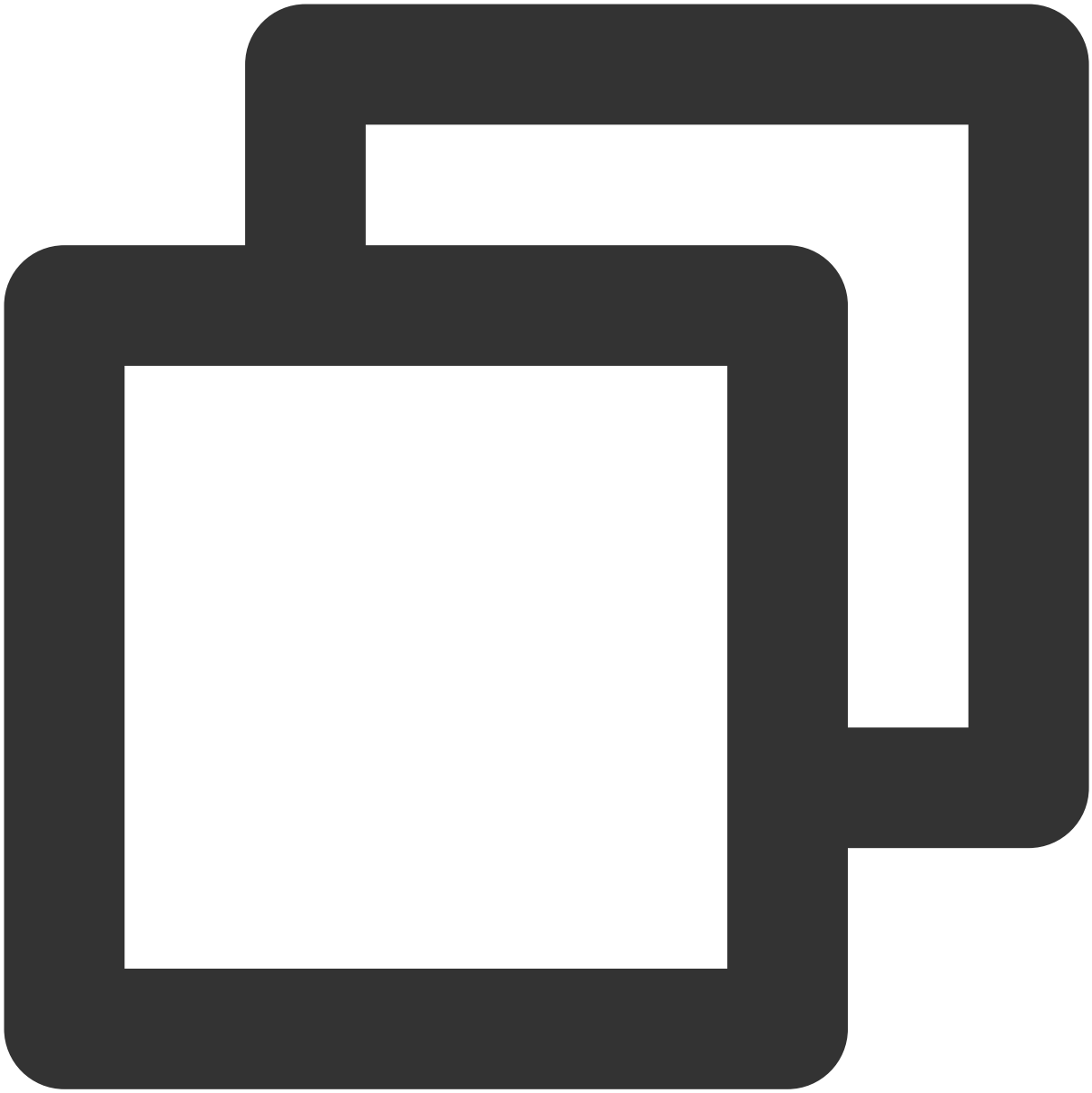
```
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

3. 生成资源规划，执行命令如下：



```
terraform plan
```

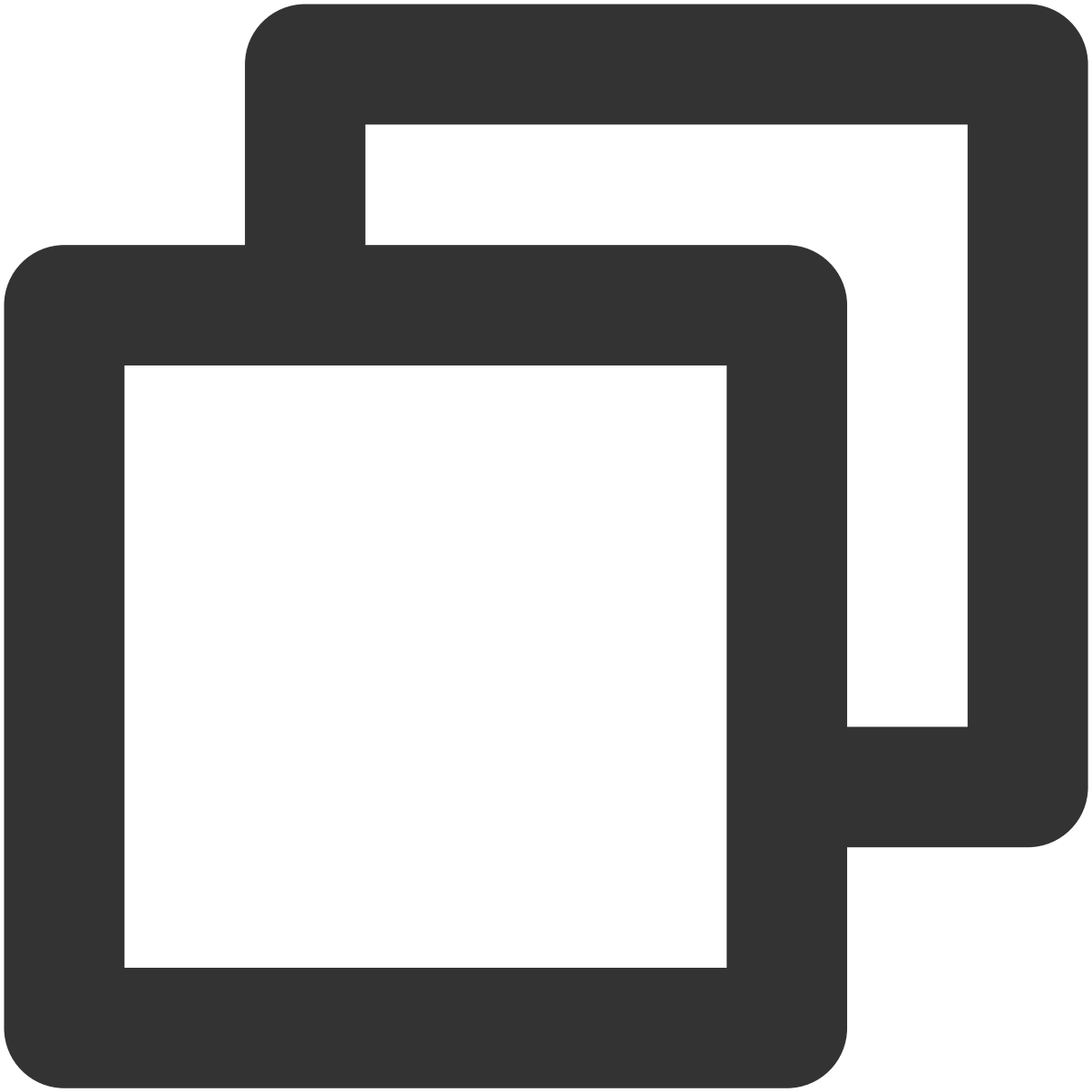
预期输出信息：



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

Terraform will perform the following actions:

  # tencentcloud_monitor_tmp_exporter_integration.tmpExporterIntegration will be cr
  + resource "tencentcloud_monitor_tmp_exporter_integration" "tmpExporterIntegratio
      + content     = jsonencode(
          XXX...
        )
```

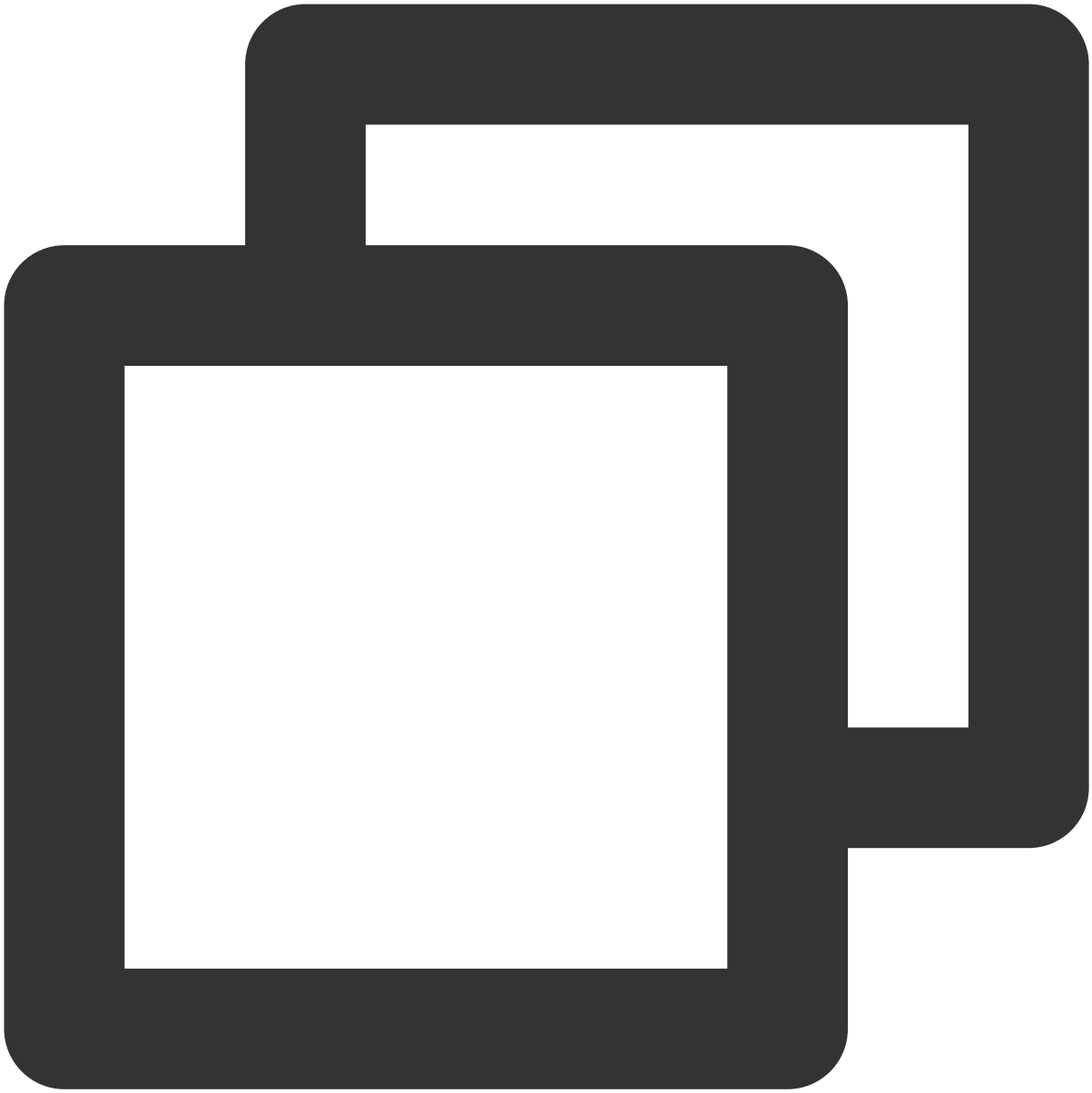
```
      + id          = (known after apply)
      + instance_id = (known after apply)
      + kind        = "blackbox-exporter"
      + kube_type   = 1
    }

  # tencentcloud_monitor_tmp_exporter_integration.tmpExporterMointor will be create
  + resource "tencentcloud_monitor_tmp_exporter_integration" "tmpExporterMointor" {
      + content     = jsonencode(
            XXX...
        )
      + id          = (known after apply)
      + instance_id = (known after apply)
      + kind        = "qcloud-exporter"
      + kube_type   = 1
    }

  # tencentcloud_monitor_tmp_instance.foo will be created
  + resource "tencentcloud_monitor_tmp_instance" "foo" {
      + api_root_path       = (known after apply)
      + data_retention_time = 30
      + id                  = (known after apply)
      + instance_name       = "tf-tmp-instance-sjtest"
      + ipv4_address        = (known after apply)
      + proxy_address       = (known after apply)
      + remote_write        = (known after apply)
      + subnet_id           = "subnet-es8rv1kx"
      + tags                = {
          + "createdBy" = "terraform"
        }
      + vpc_id              = "vpc-0n42dxzs"
      + zone                = "ap-mumbai-1"
    }

Plan: 3 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
you run "terraform apply" now.
```

4. 创建集成中心组件集成，执行命令如下：



```
terraform apply
```

预期输出信息：



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

Terraform will perform the following actions:
XXX...


Plan: 3 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_monitor_tmp_instance.foo: Creating...
tencentcloud_monitor_tmp_instance.foo: Still creating... [10s elapsed]
。。。

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
```

注意：

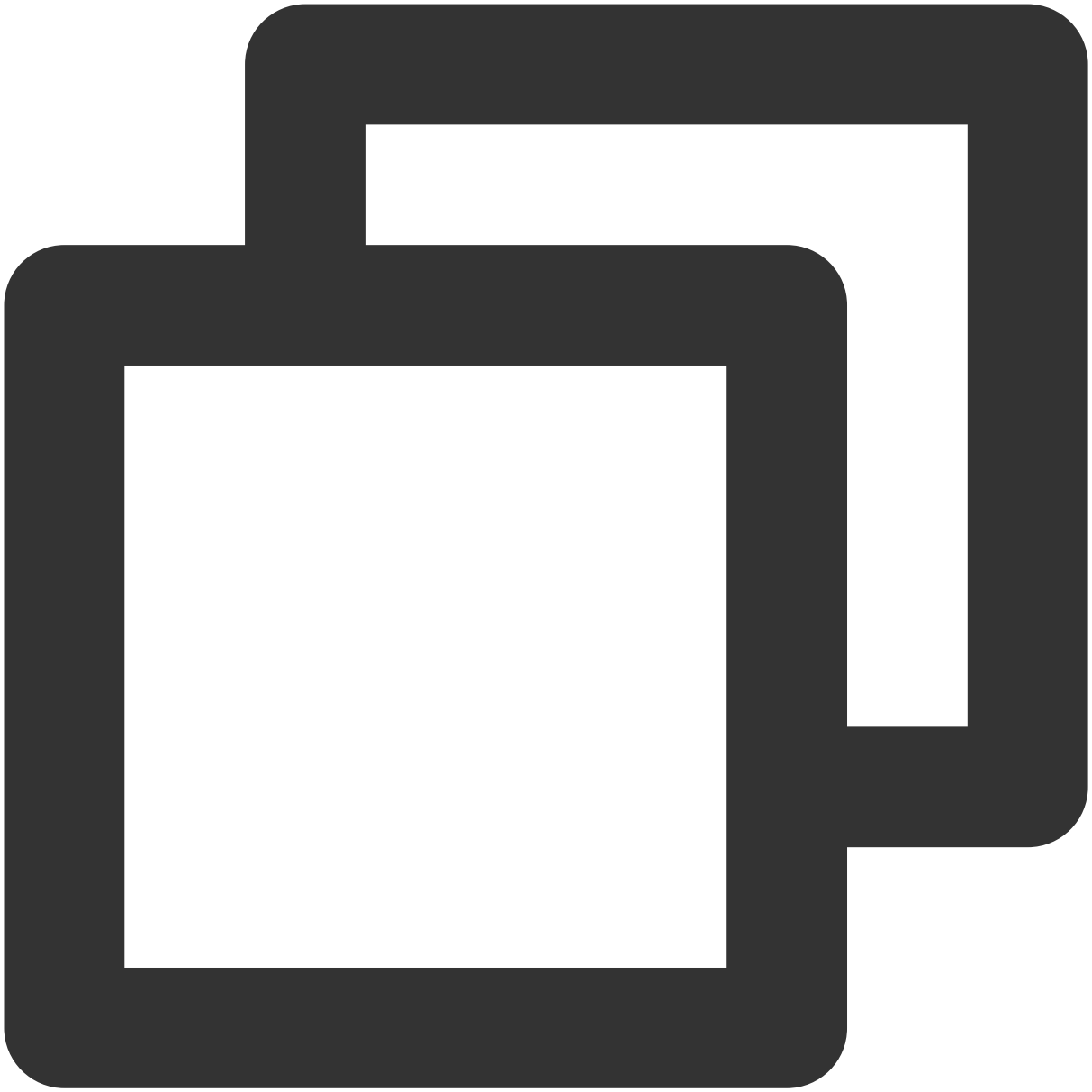
当出现 Apply complete! Resources: 3 added, 0 changed, 0 destroyed. 表示您已创建成功。

查看 Prometheus 实例状态

登录 腾讯云可观测平台，在左侧导航栏，选择 **prometheus 监控，**可在 prometheus 实例列表中看到存在的实例。

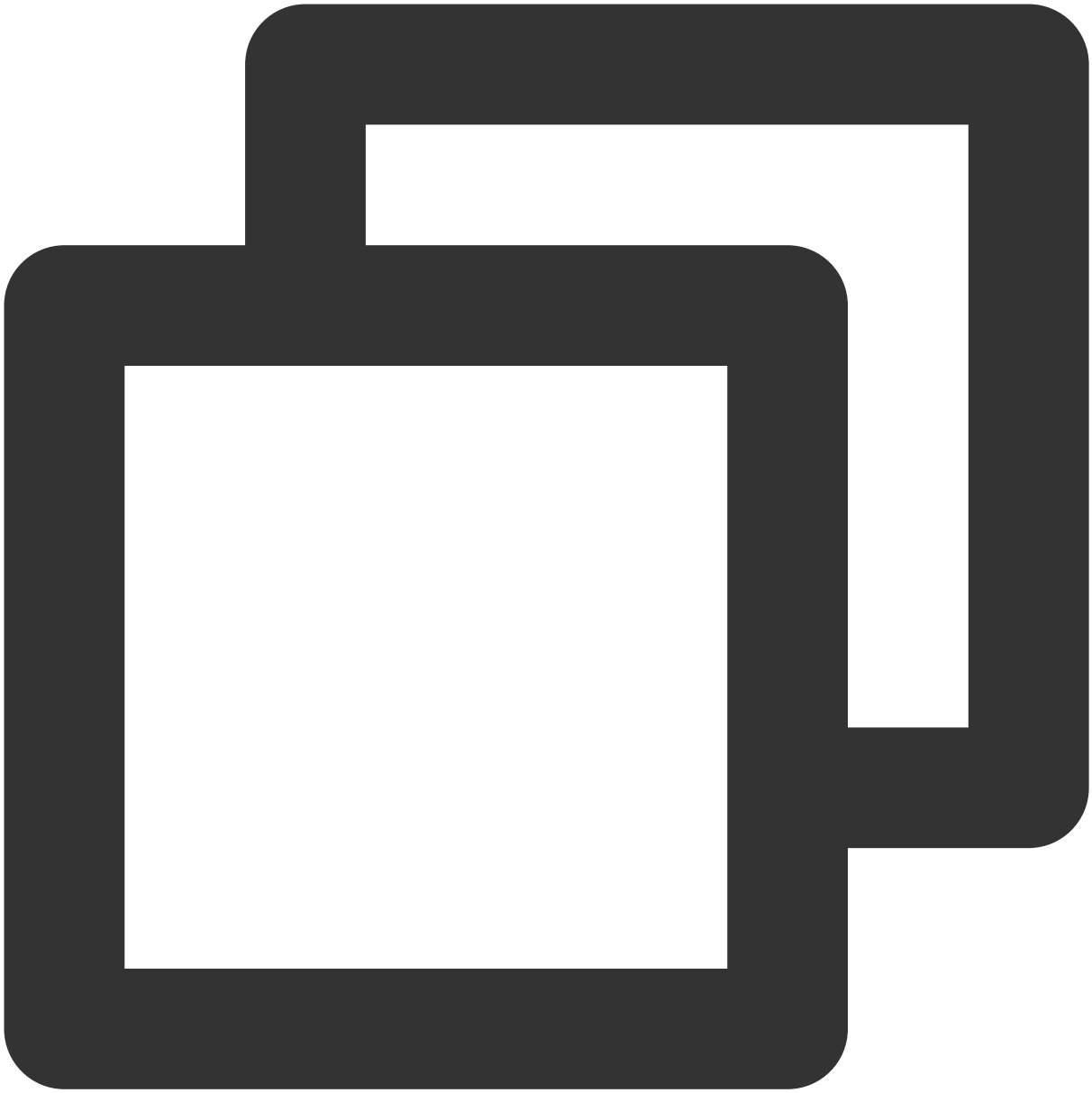
删除 Prometheus 实例集成中心组件集成

销毁资源，执行命令如下：



```
terraform destroy
```

预期输出信息：



```
tencentcloud_monitor_tmp_instance.foo: Refreshing state... [id=prom-8dyb6iny]

Terraform used the selected providers to generate the following execution plan. Res
following symbols:
  - destroy

Terraform will perform the following actions:
XXX...

Plan: 0 to add, 0 to change, 1 to destroy.
```

```
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

tencentcloud_monitor_tmp_instance.foo: Destroying... [id=prom-8dyb6iny]
tencentcloud_monitor_tmp_instance.foo: Destruction complete after 6s

Destroy complete! Resources: 1 destroyed.
```

注意：

当出现 `Destroy complete! Resources: 1 destroyed.` 表示您已删除该实例。

使用 Terraform 采集容器监控数据

最近更新时间：2023-12-29 16:10:11

前提条件

安装 Terraform

关于安装 Terraform 的具体操作，请参见在 本地安装和配置 Terraform。

说明：

Terraform 安装版本不得低于 v1.6.3，可通过 terraform --version 命令查看安装后 Terraform 版本。

配置腾讯云账号信息

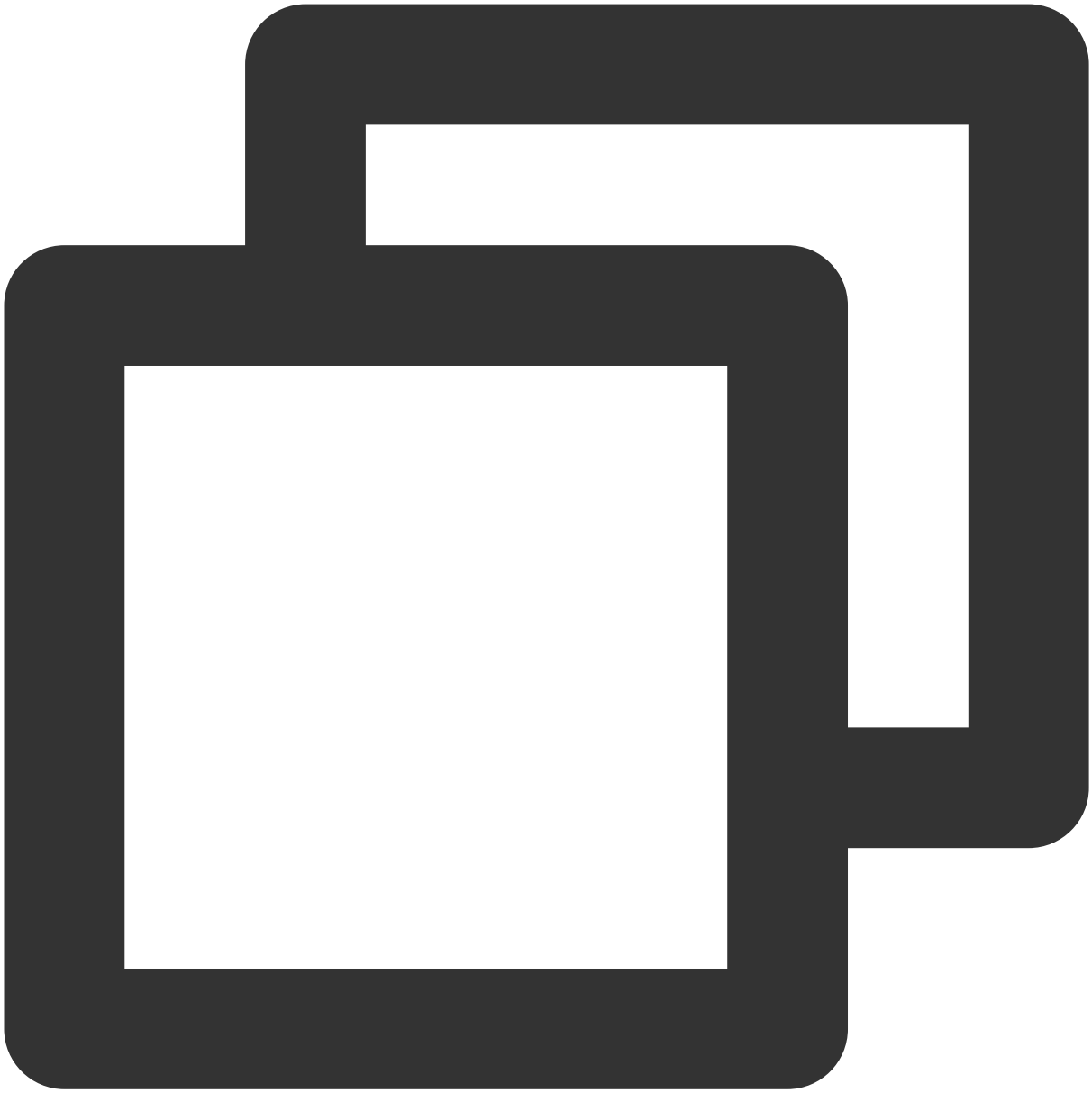
在首次使用 Terraform 之前，请前往 云 API 密钥页面 申请安全凭证 SecretId 和 SecretKey。若已有可使用的安全凭证，则跳过该步骤。

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。

配置腾讯云账号信息，有以下两种方式：

静态凭证鉴权

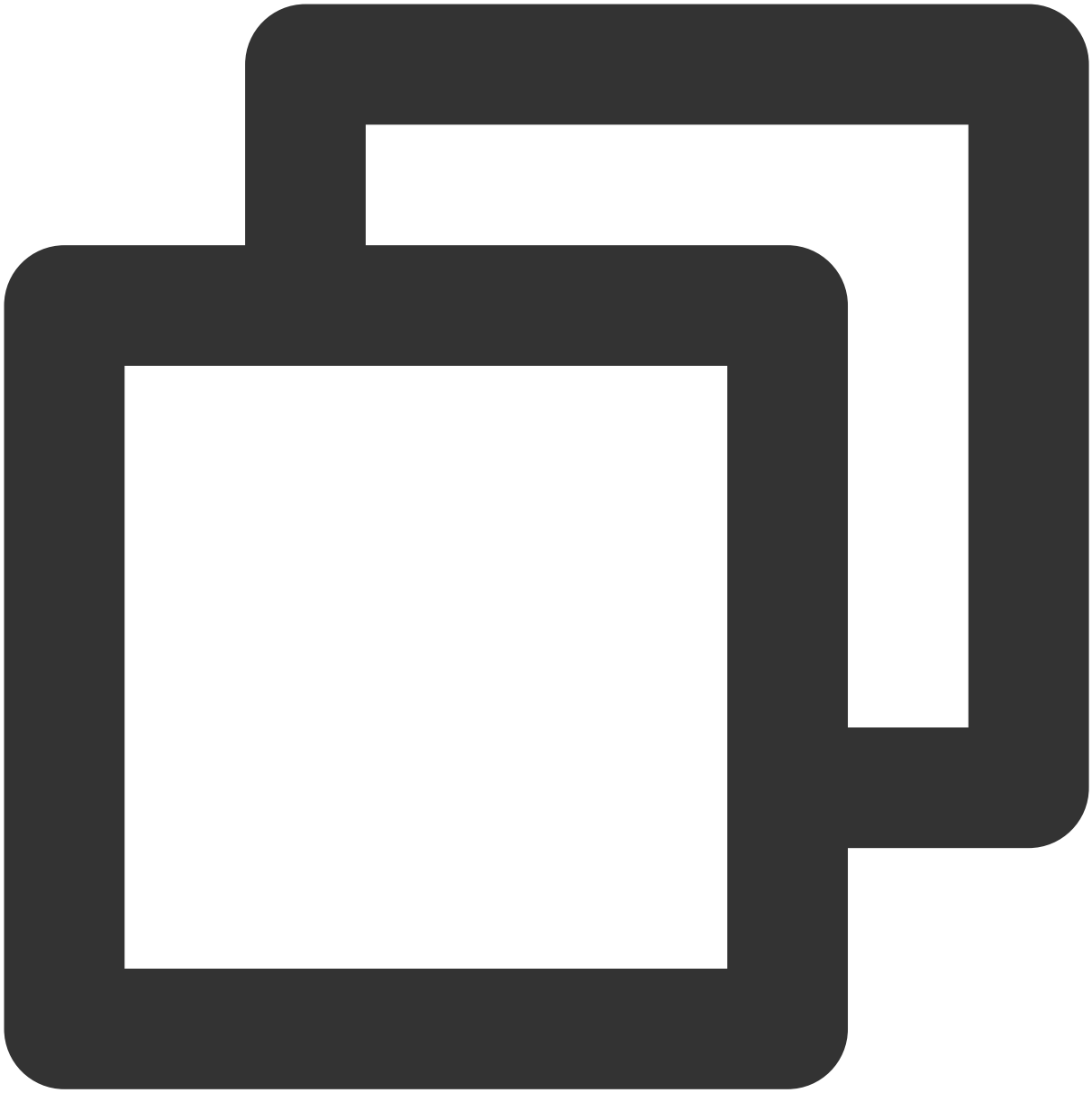
在用户目录下创建 `provider.tf` 文件，输入如下内容。其中 `my-secret-id` 和 `my-secret-key` 需替换为密钥 SecretId 和 SecretKey。



```
provider "tencentcloud" {
  secret_id = "my-secret-id"
  secret_key = "my-secret-key"
}
```

环境变量鉴权

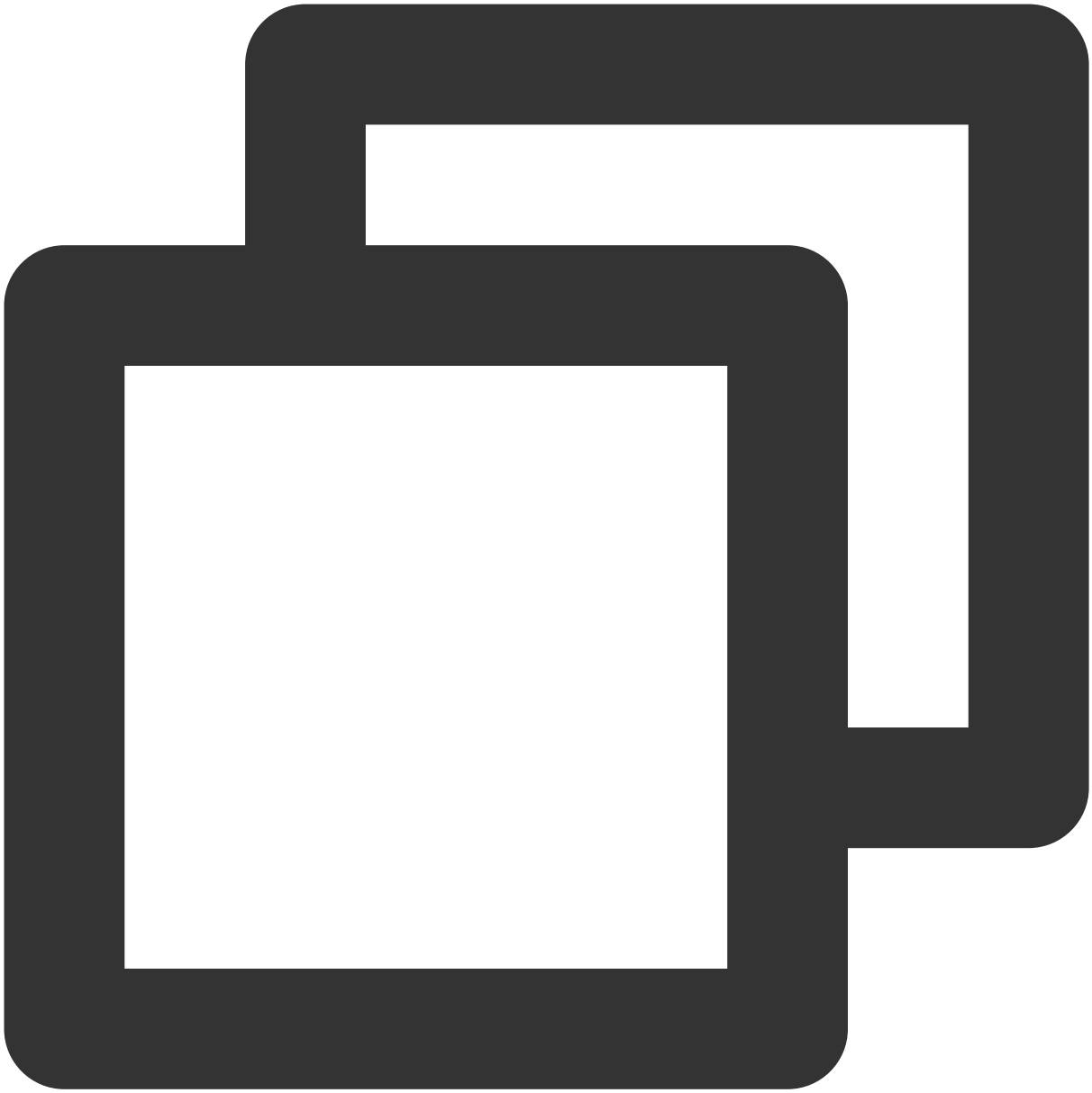
配置电脑环境变量或云端环境变量，请执行以下命令。其中 `YOUR_SECRET_ID` 和 `YOUR_SECRET_KEY` 需替换为密钥 SecretId 和 SecretKey。



```
export TENCENTCLOUD_SECRET_ID=YOUR_SECRET_ID
export TENCENTCLOUD_SECRET_KEY=YOUR_SECRET_KEY
```

增加 Prometheus 实例的采集容器监控数据

1. 创建一个新的 Terraform 配置文件：创建一个新目录，并在该目录下创建一个名为 main.tf 的文件，配置如下信息：



```
# 指定 provider 配置信息

terraform {
  required_providers {
    tencentcloud = {
      source = "tencentcloudstack/tencentcloud"
    }
  }
}
# prometheus管理云监控集成
```

```
#prometheus 管理容器集群(采集容器)

resource "tencentcloud_monitor_tmp_tke_cluster_agent" "foo" {
  instance_id = tencentcloud_monitor_tmp_instance.foo.id
  agents {
    region          = "ap-mumbai"
    cluster_type    = "eks"
    cluster_id      = "cls-1uary7z2" #需要关联的集群id
    enable_external = false
  }
}
```

说明：

创建 Prometheus 实例的集成中心组件需配置对应参数，需配置的字段如下：

instance_id: （必填、字符串、ForceNew）实例 ID。

agents ：（必填，列表）代理列表。

cluster_id ：（必填，字符串）标识集群的 id。

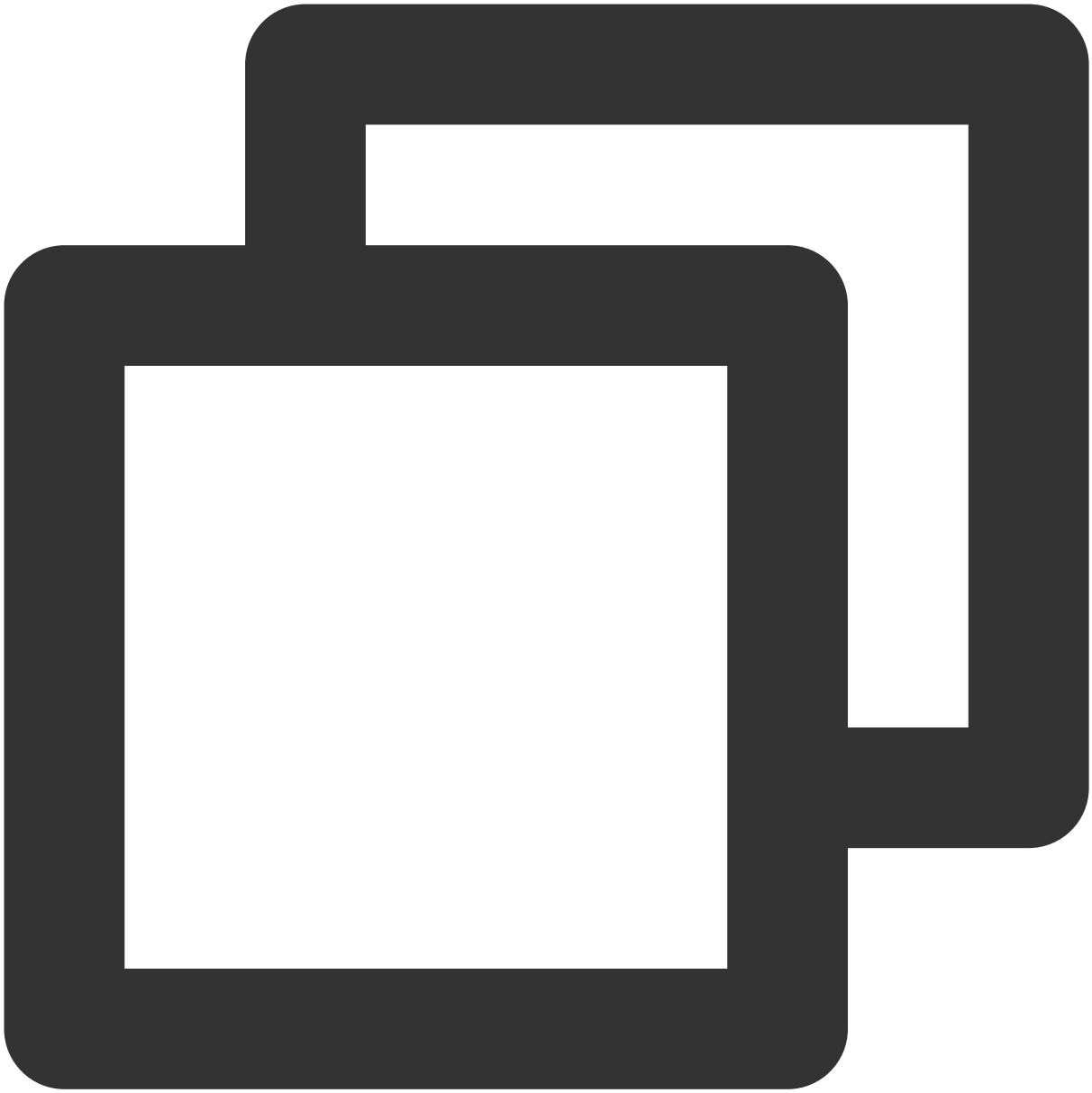
region：（必填，字符串）区域限制。

enable_external：(必填，Bool) 是否开启公网 CLB。

cluster_type:（必填，字符串）集群类型。

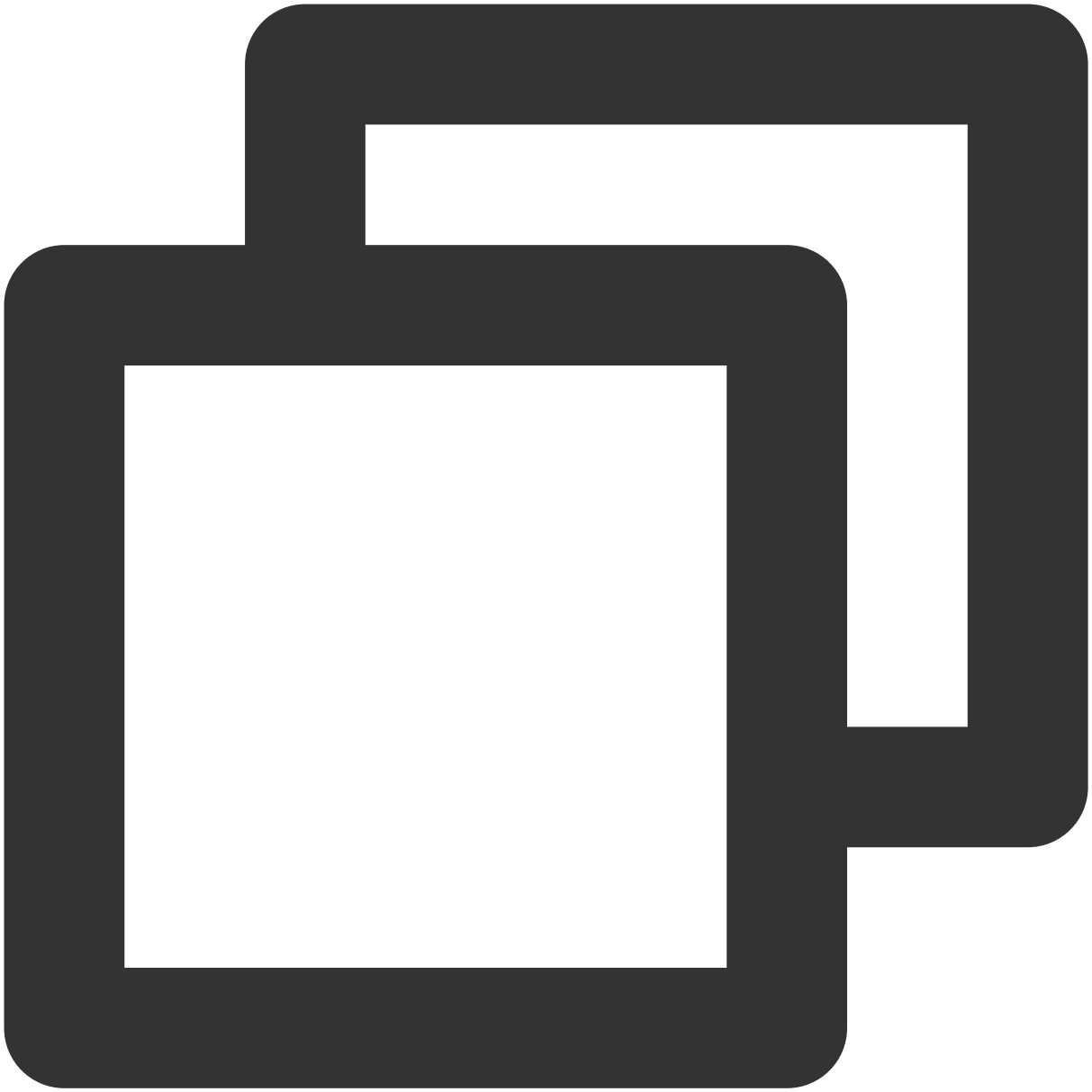
您若需要获取**更多详细参数**请参考 腾讯接入云集成接入 github，也可以通过 Terraform 腾讯云提供商 了解更多。

2. 初始化 Terraform 运行环境，执行命令如下：



```
terraform init
```

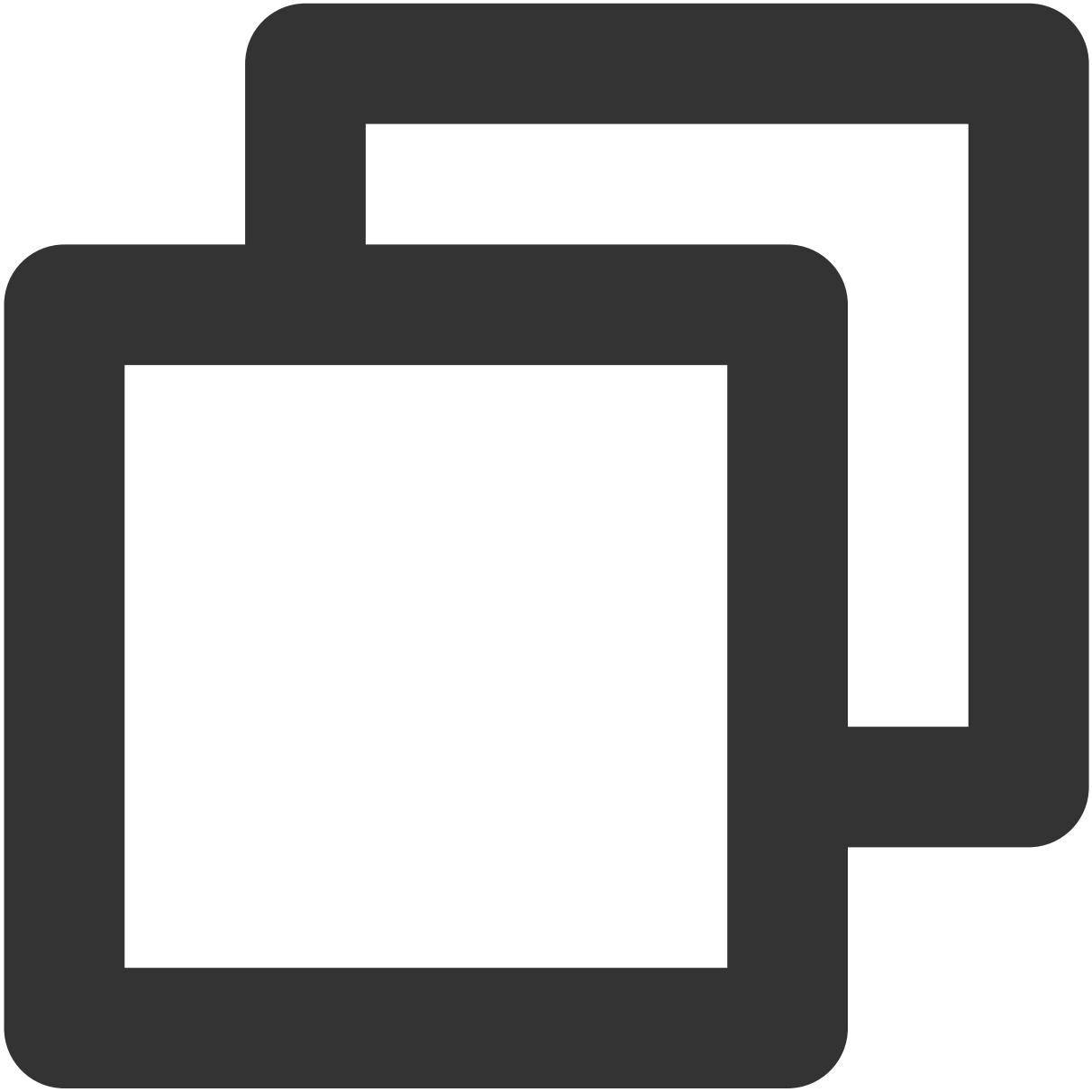
预期输出信息：



```
Initializing the backend...

Initializing provider plugins...
- Reusing previous version of tencentcloudstack/tencentcloud from the dependency lo
- Using previously-installed tencentcloudstack/tencentcloud v1.81.32

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
```

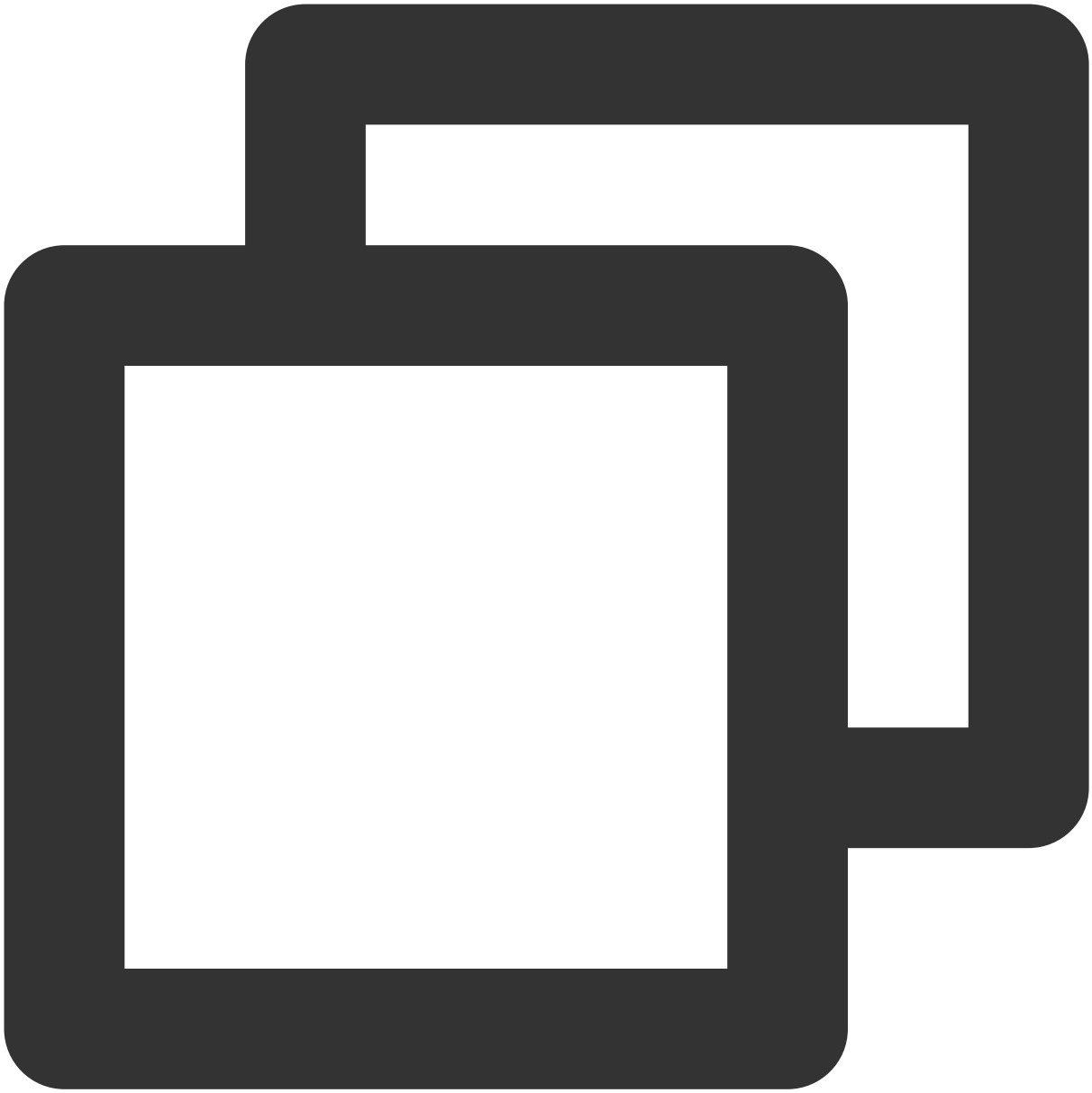
```
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

3. 生成资源规划，执行命令如下：



```
terraform plan
```

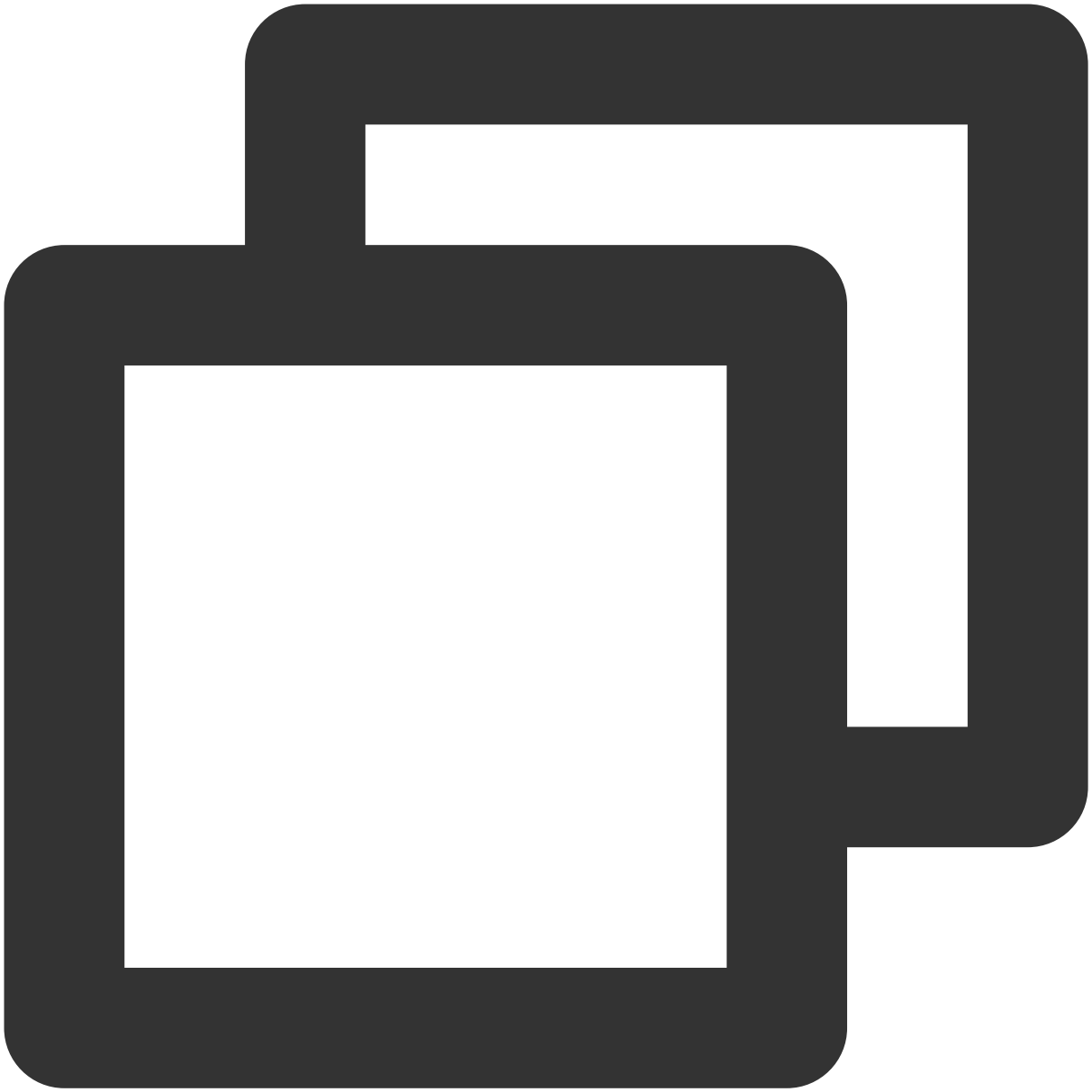
预期输出信息：



```
tencentcloud_monitor_tmp_instance.foo: Refreshing state... [id=prom-jh0zntj2]
tencentcloud_monitor_tmp_exporter_integration.tmpExporterIntegration: Refreshing st
tencentcloud_monitor_tmp_exporter_integration.tmpExporterMointor: Refreshing state.

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

Terraform will perform the following actions:

  # tencentcloud_monitor_tmp_tke_cluster_agent.foo will be created
```

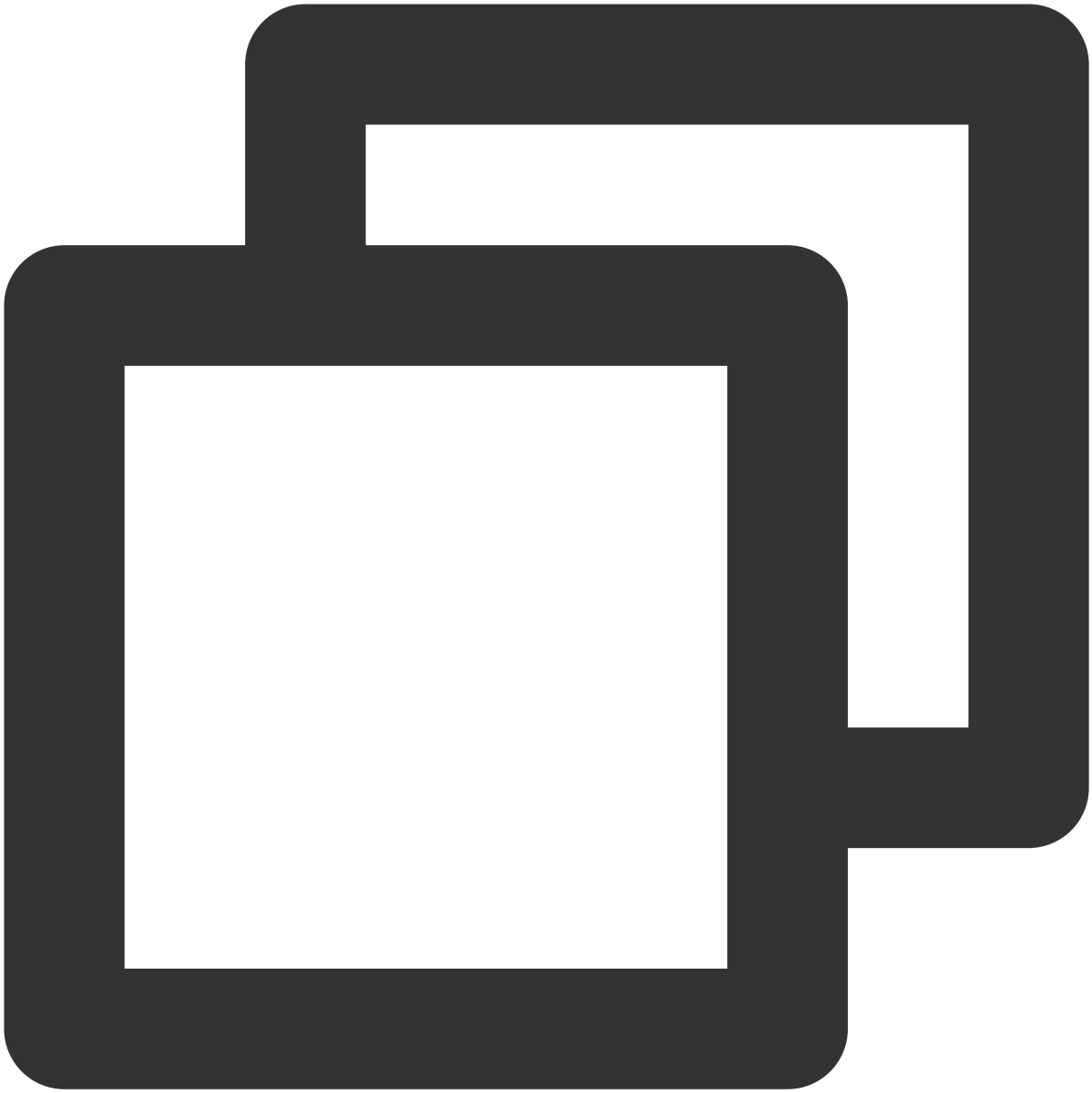
```
  + resource "tencentcloud_monitor_tmp_tke_cluster_agent" "foo" {
      + id          = (known after apply)
      + instance_id = "prom-jh0zntj2"

      + agents {
          + cluster_id      = "cls-1uary7z2"
          + cluster_name    = (known after apply)
          + cluster_type    = "eks"
          + enable_external = false
          + region          = "ap-mumbai"
          + status          = (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
you run "terraform apply" now.
```

4. 创建集成中心组件集成，执行命令如下：



```
terraform apply
```

预期输出信息：



```
tencentcloud_monitor_tmp_instance.foo: Refreshing state... [id=prom-jh0zntj2]
tencentcloud_monitor_tmp_exporter_integration.tmpExporterIntegration: Refreshing st
tencentcloud_monitor_tmp_exporter_integration.tmpExporterMointor: Refreshing state.

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

Terraform will perform the following actions:

  # tencentcloud_monitor_tmp_tke_cluster_agent.foo will be created
```

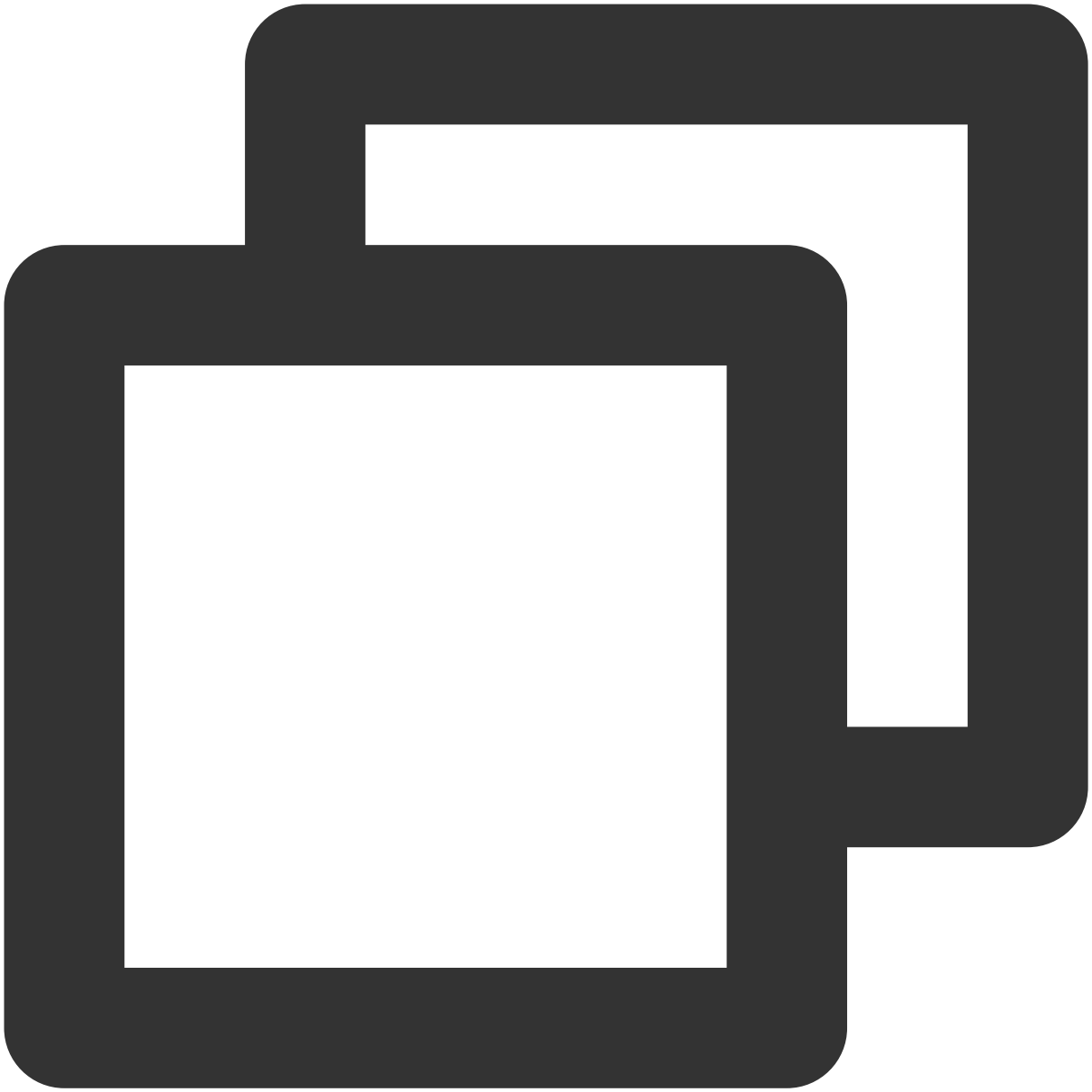
```
  + resource "tencentcloud_monitor_tmp_tke_cluster_agent" "foo" {
      + id          = (known after apply)
      + instance_id = "prom-jh0zntj2"

      + agents {
          + cluster_id      = "cls-1uary7z2"
          + cluster_name    = (known after apply)
          + cluster_type    = "eks"
          + enable_external = false
          + region          = "ap-mumbai"
          + status          = (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_monitor_tmp_tke_cluster_agent.foo: Creating...
实例内容。。。

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

查看 Prometheus 实例状态

登录 腾讯云可观测平台，在左侧导航栏，选择 **prometheus 监控**，可在 prometheus 实例列表中看到存在的实例。

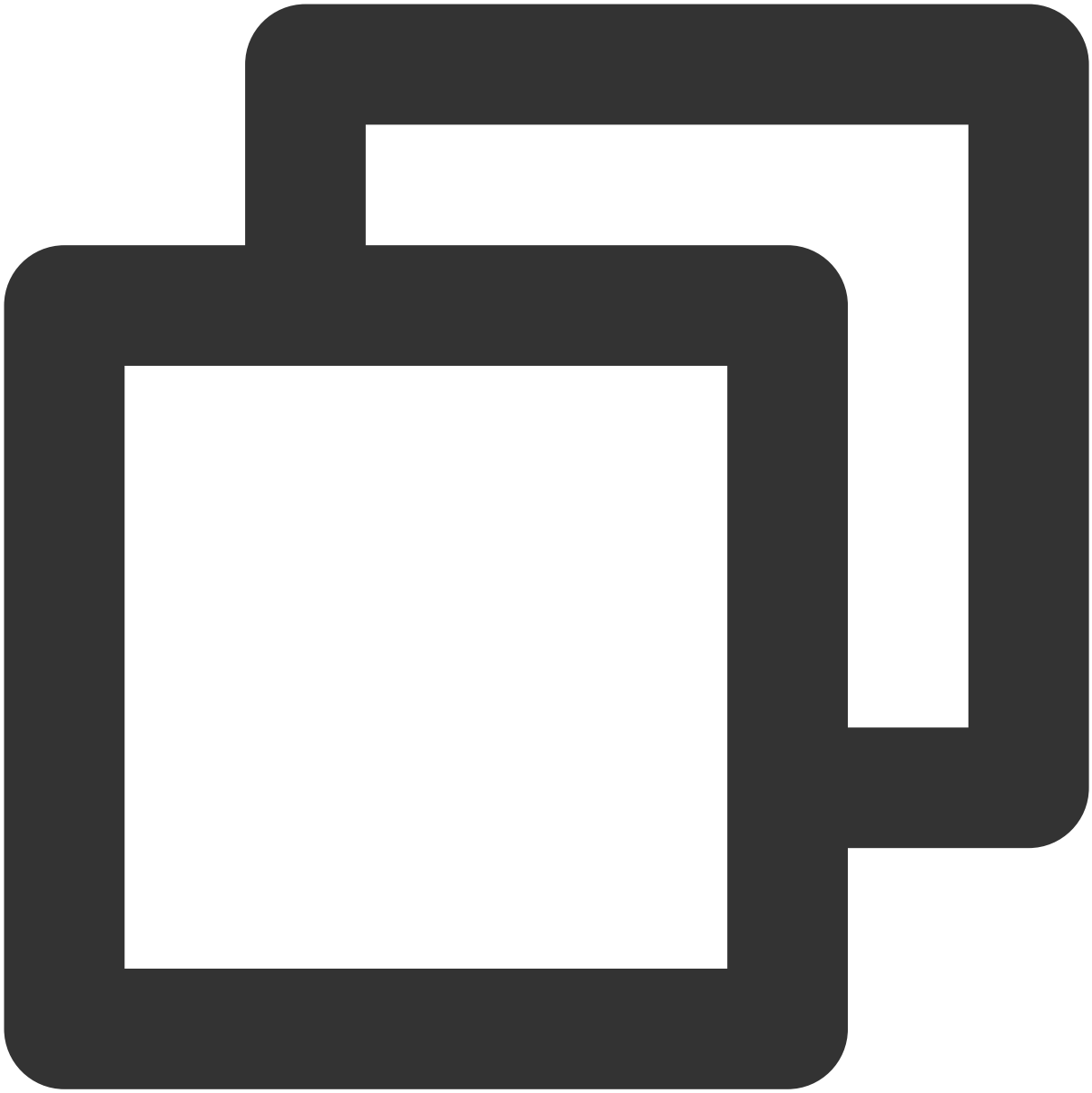
删除 Prometheus 实例集成中心组件集成

销毁资源，执行命令如下：



```
terraform destroy
```

预期输出信息：



```
tencentcloud_monitor_tmp_instance.foo: Refreshing state... [id=prom-8dyb6iny]

Terraform used the selected providers to generate the following execution plan. Res
following symbols:
  - destroy

Terraform will perform the following actions:
实例内容。。。

Plan: 0 to add, 0 to change, 1 to destroy.
```

```
Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

tencentcloud_monitor_tmp_instance.foo: Destroying... [id=prom-8dyb6iny]
tencentcloud_monitor_tmp_instance.foo: Destruction complete after 6s

Destroy complete! Resources: 1 destroyed.
```

注意：

当出现 `Destroy complete! Resources: （存在的实例个数） destroyed.` 表示您已删除该实例。

使用 Terraform 配置告警策略

最近更新时间：2023-12-29 16:11:00

前提条件

安装 Terraform

关于安装 Terraform 的具体操作，请参见在 本地安装和配置 Terraform。

说明：

Terraform 安装版本不得低于 v1.6.3，您可通过 terraform --version 命令查看安装的 Terraform 版本。

配置腾讯云账号信息

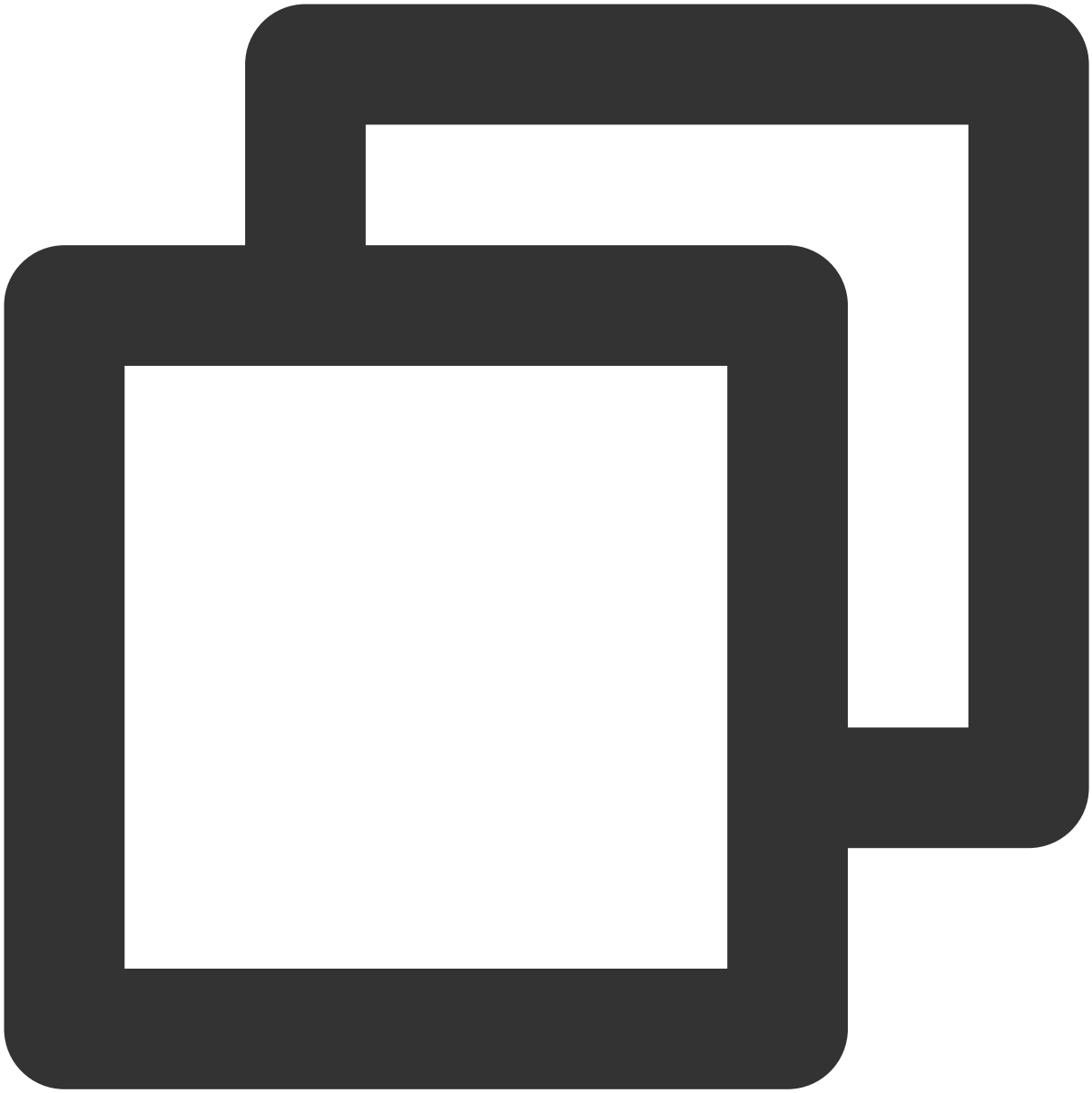
在首次使用 Terraform 之前，请前往 云 API 密钥 申请安全凭证 SecretId 和 SecretKey。如果已有可使用的安全凭证，则跳过该步骤。

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。

配置腾讯云账号信息，有以下两种方式：

静态凭证鉴权

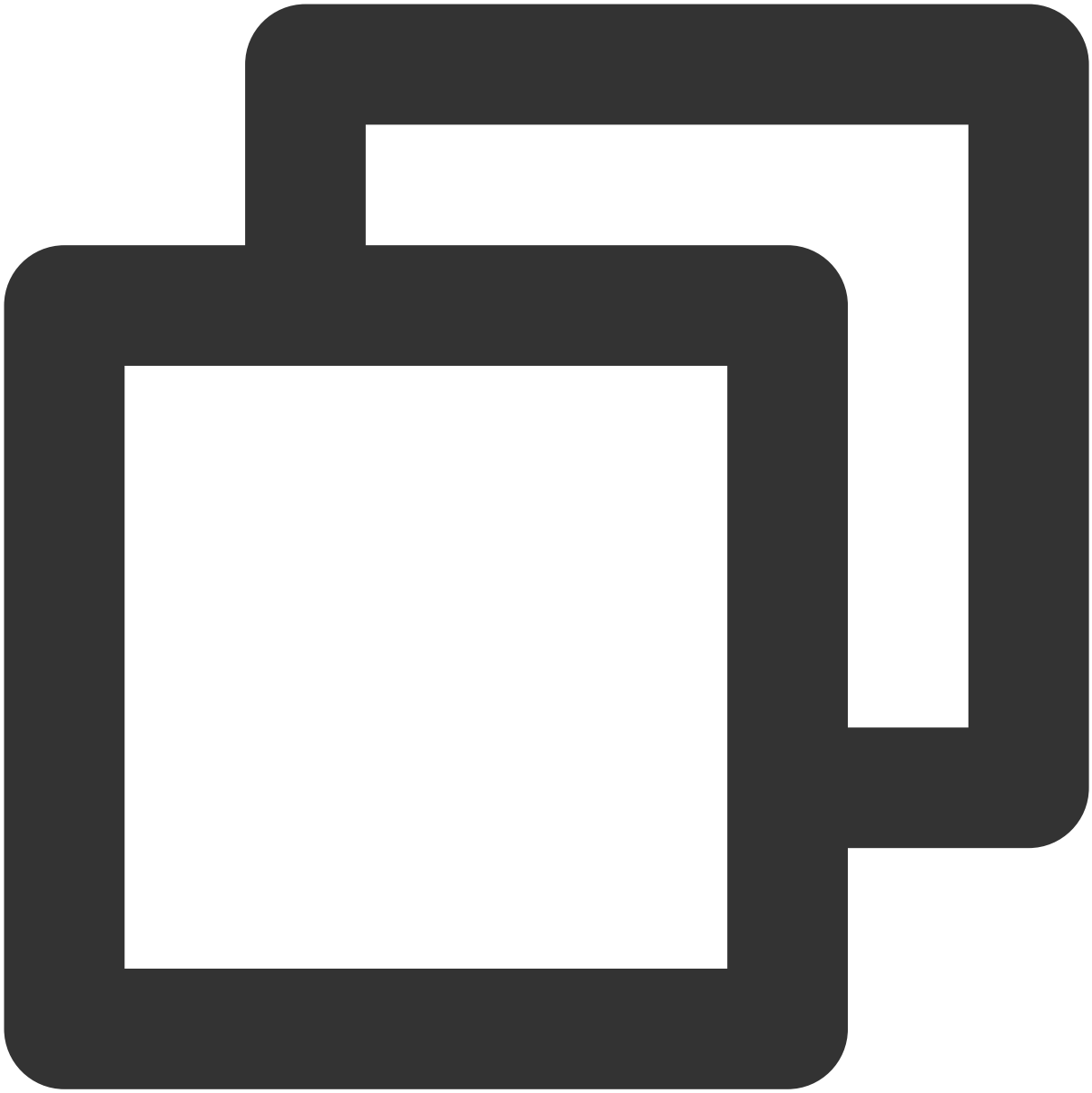
在用户目录下创建 `provider.tf` 文件，输入如下内容。其中 `my-secret-id` 和 `my-secret-key` 需替换为密钥 SecretId 和 SecretKey。



```
provider "tencentcloud" {
  secret_id = "my-secret-id"
  secret_key = "my-secret-key"
}
```

环境变量鉴权

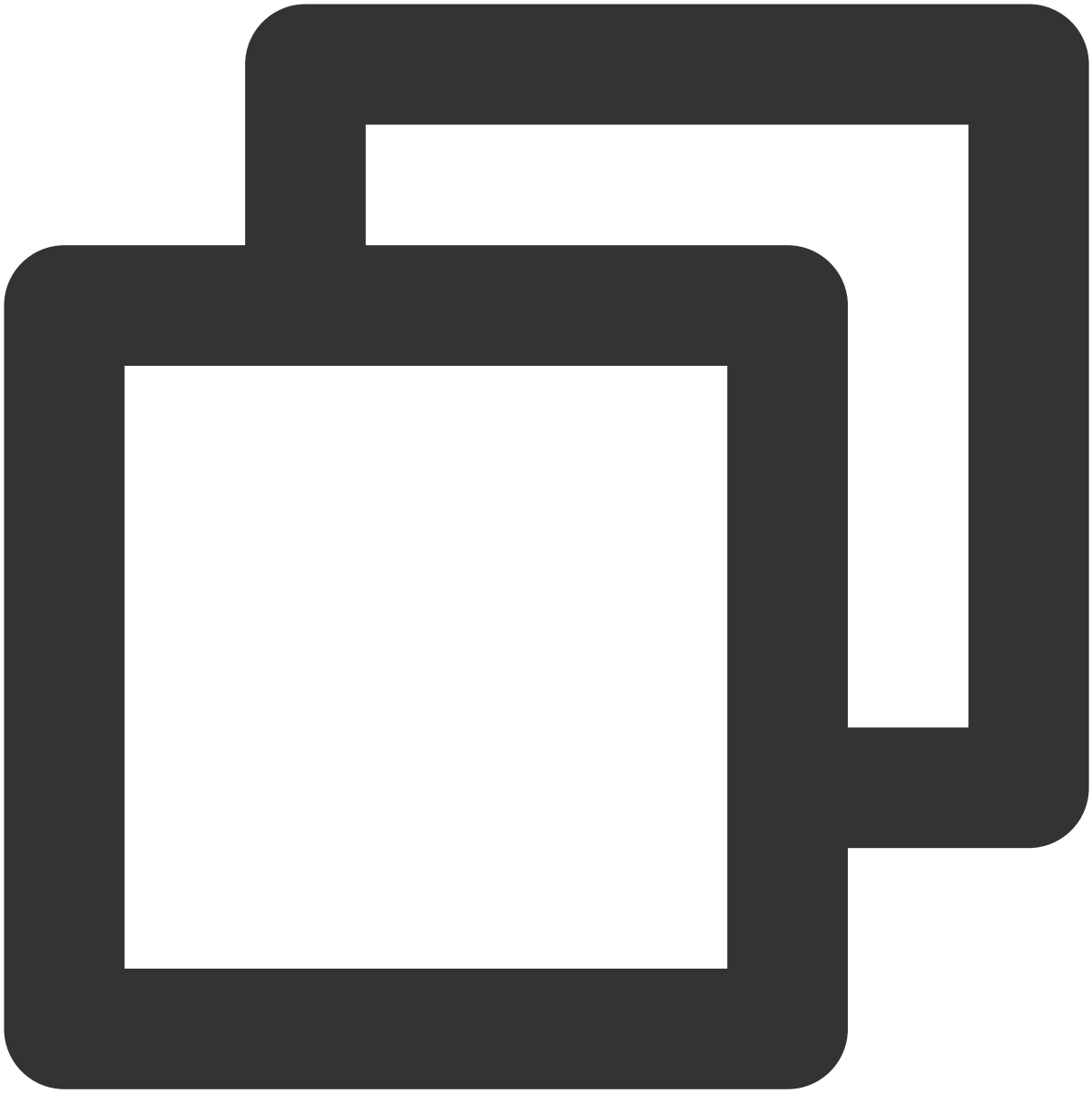
配置电脑环境变量或云端环境变量，请执行以下命令。其中 `YOUR_SECRET_ID` 和 `YOUR_SECRET_KEY` 需替换为密钥 SecretId 和 SecretKey。



```
export TENCENTCLOUD_SECRET_ID=YOUR_SECRET_ID
export TENCENTCLOUD_SECRET_KEY=YOUR_SECRET_KEY
```

增加 Prometheus 实例的 Terraform 配置告警策略

1. 创建一个新的 Terraform 配置文件。创建一个新的目录，并在该目录下创建一个 main.tf 文件，配置如下信息：



```
# 指定 provider 配置信息

terraform {
  required_providers {
    tencentcloud = {
      source = "tencentcloudstack/tencentcloud"
    }
  }
}

# promethues 配置告警 （云监控侧配置）
```

```
resource "tencentcloud_monitor_tmp_alert_rule" "foo" {
  duration    = "2m"
  expr        = "avg by (instance) (mysql_global_status_threads_connected) / avg by
  instance_id = tencentcloud_monitor_tmp_instance.foo.id
  receivers   = ["notice-zmjsavnp"] # 此处可通过云监控的tf创建通知模板
  rule_name   = "MySQL 连接数过多--tf-云监控test"
  rule_state  = 2
  type        = "MySQL/MySQL 连接数过多"

  annotations {
    key   = "description"
    value = "MySQL 连接数过多，实例：{{$labels.instance}}，当前值：{{ $value | humanize
  }
  annotations {
    key   = "summary"
    value = "MySQL 连接数过多(>80%)"
  }

  labels {
    key   = "severity"
    value = "warning"
  }
}
```

说明：

配置的字段如下：

duration：规则持续时间。

expr：报警表达式。

instance_id：实例 ID。

receivers：报警接收人列表。

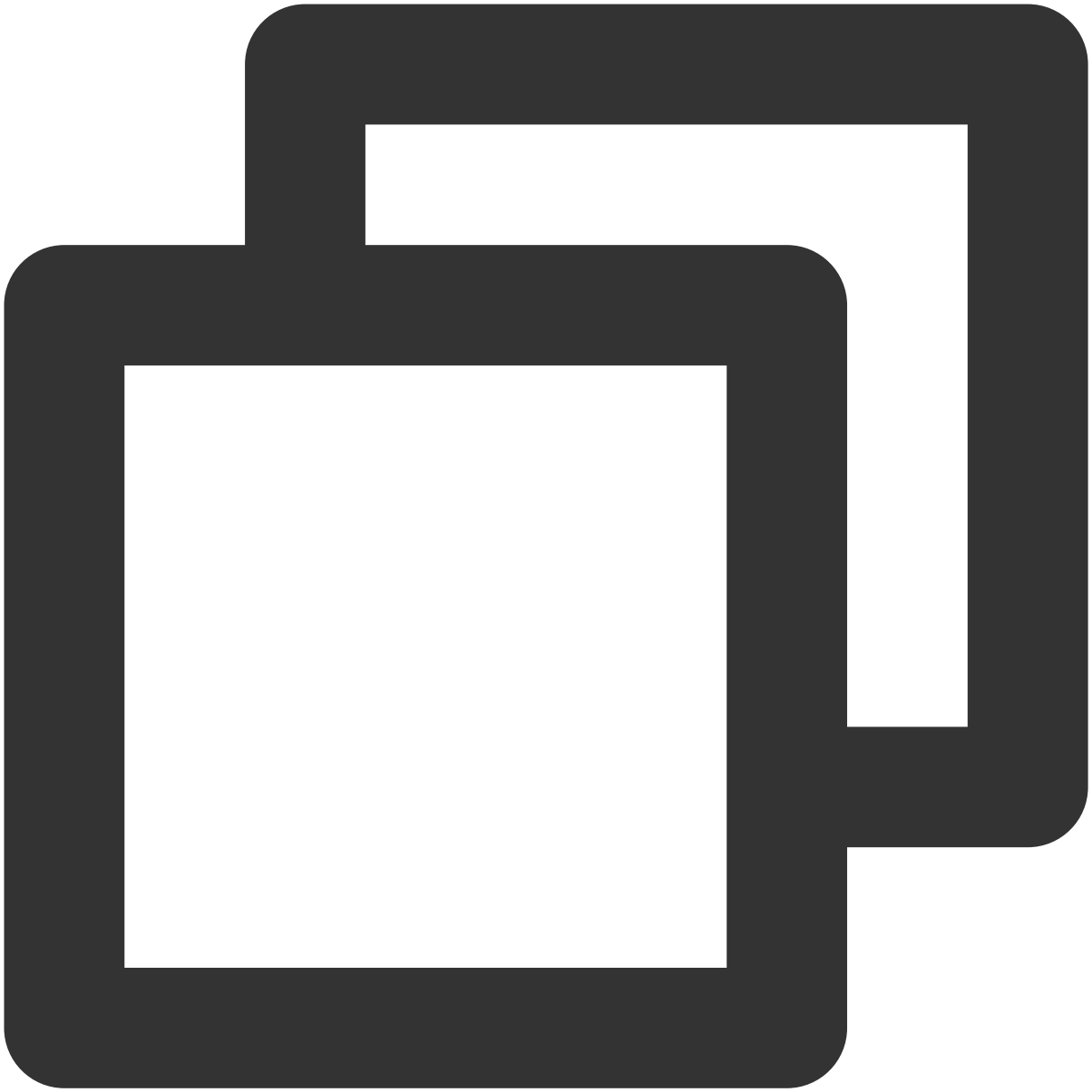
rule_name：报警规则名称。

rule_state：报警规则状态。

type：报警规则类型。

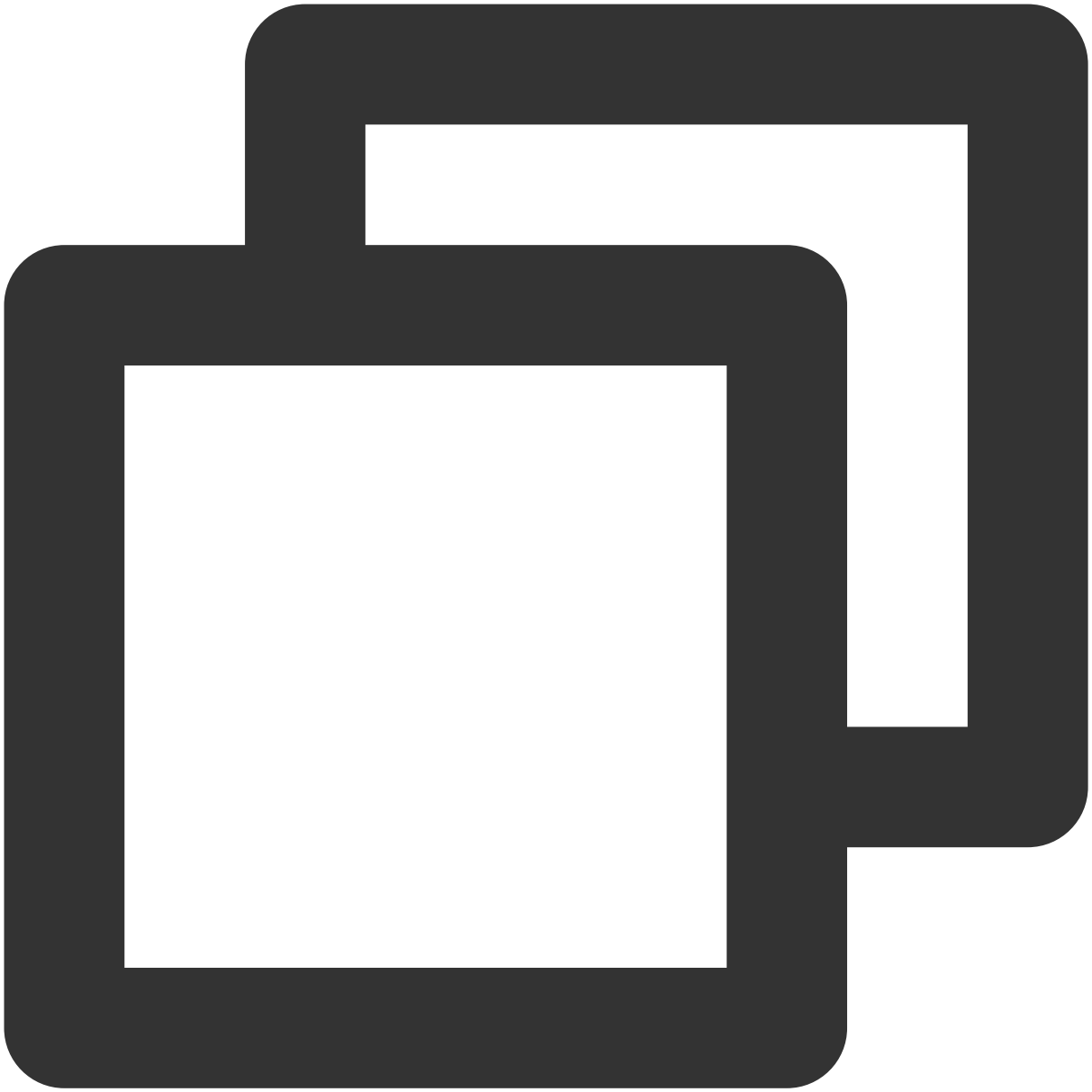
您若需要获取更多详细参数请参考 腾讯云集成接入 github，也可以通过 Terraform 腾讯云提供商 了解更多。

2. 初始化 Terraform 运行环境，执行命令如下：



```
terraform init
```

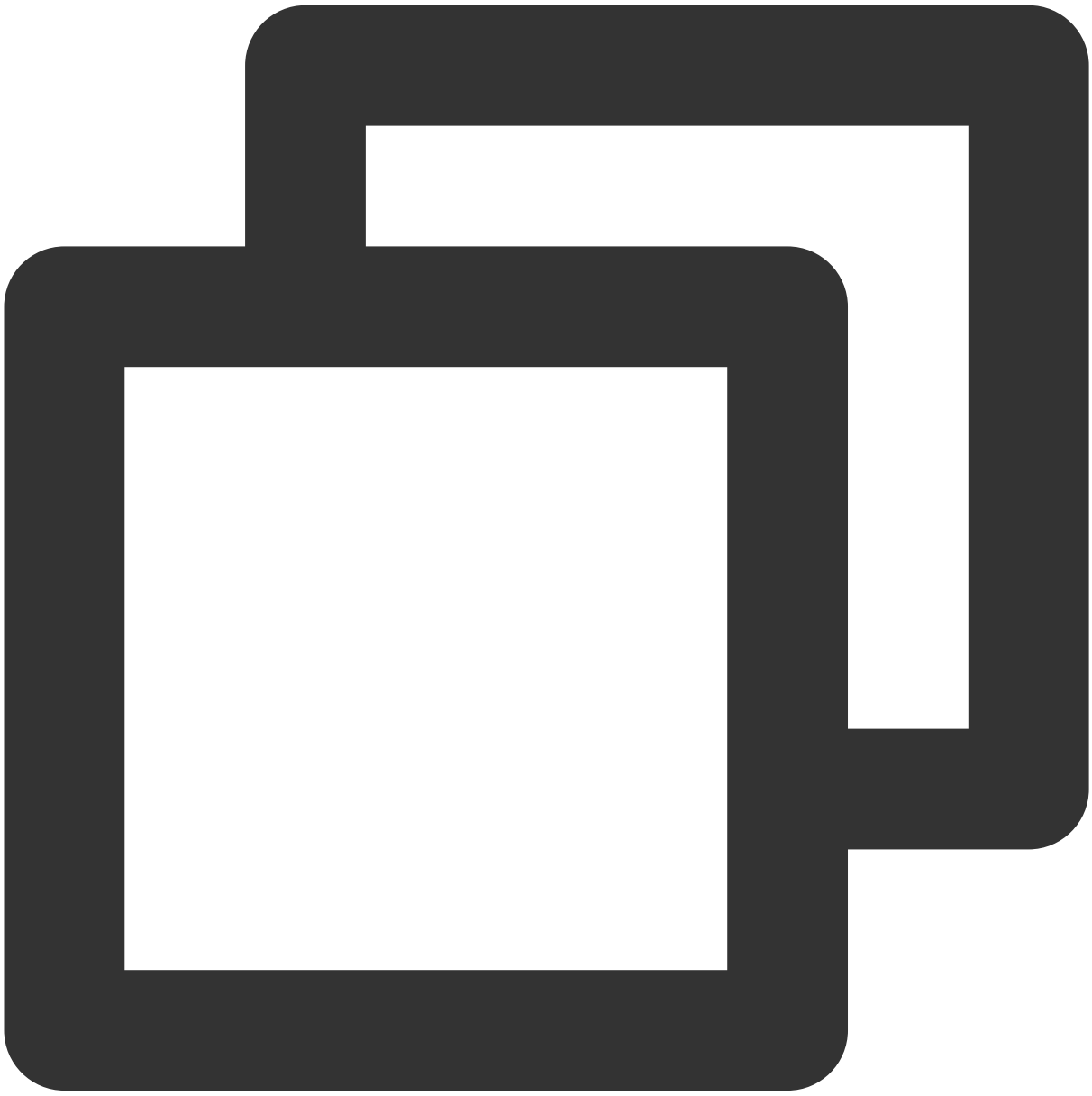
预期输出信息：



```
Initializing the backend...

Initializing provider plugins...
- Reusing previous version of tencentcloudstack/tencentcloud from the dependency lo
- Using previously-installed tencentcloudstack/tencentcloud v1.81.32

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
```

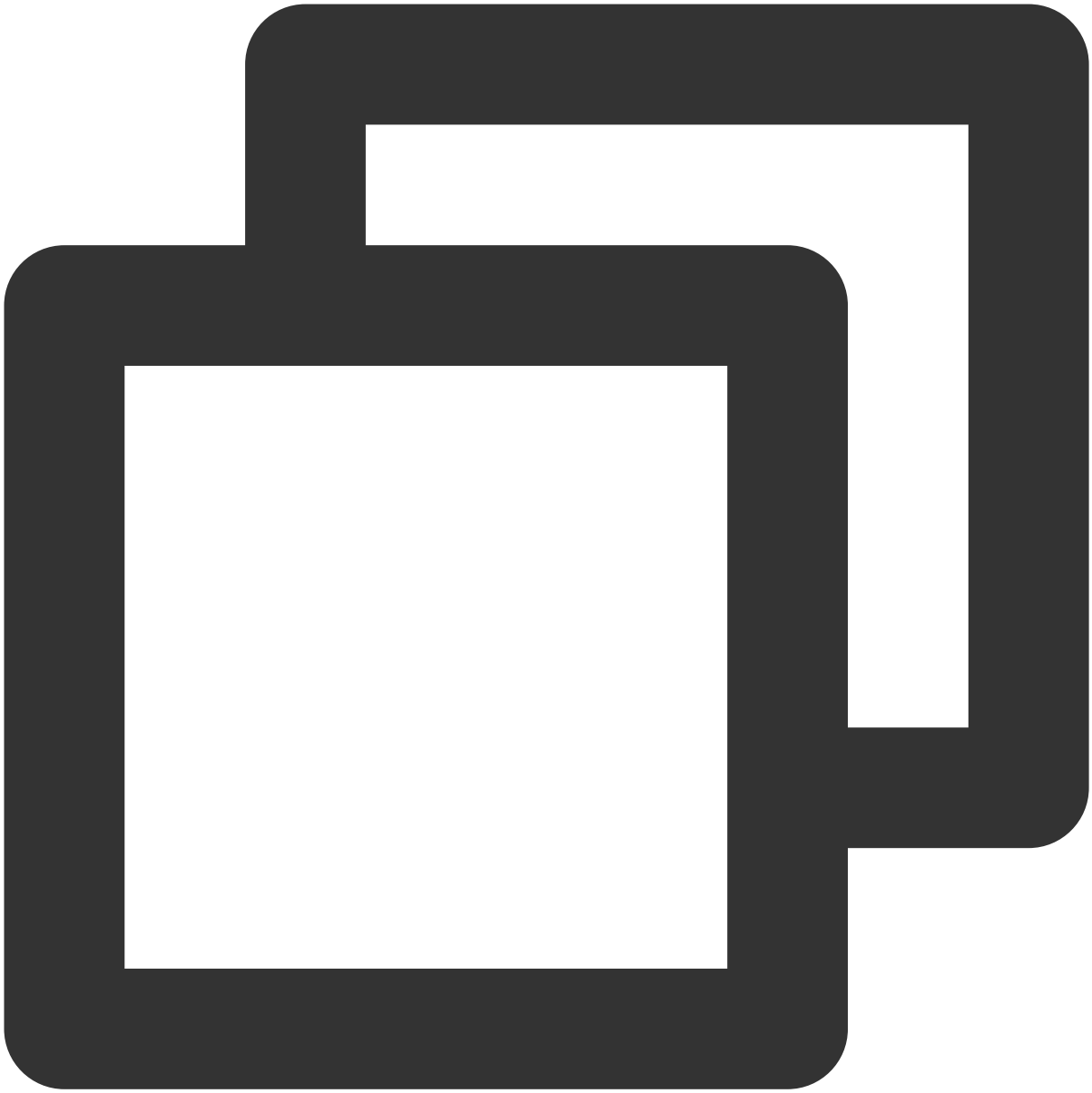
```
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

3. 生成资源规划，执行命令如下：



```
terraform plan
```

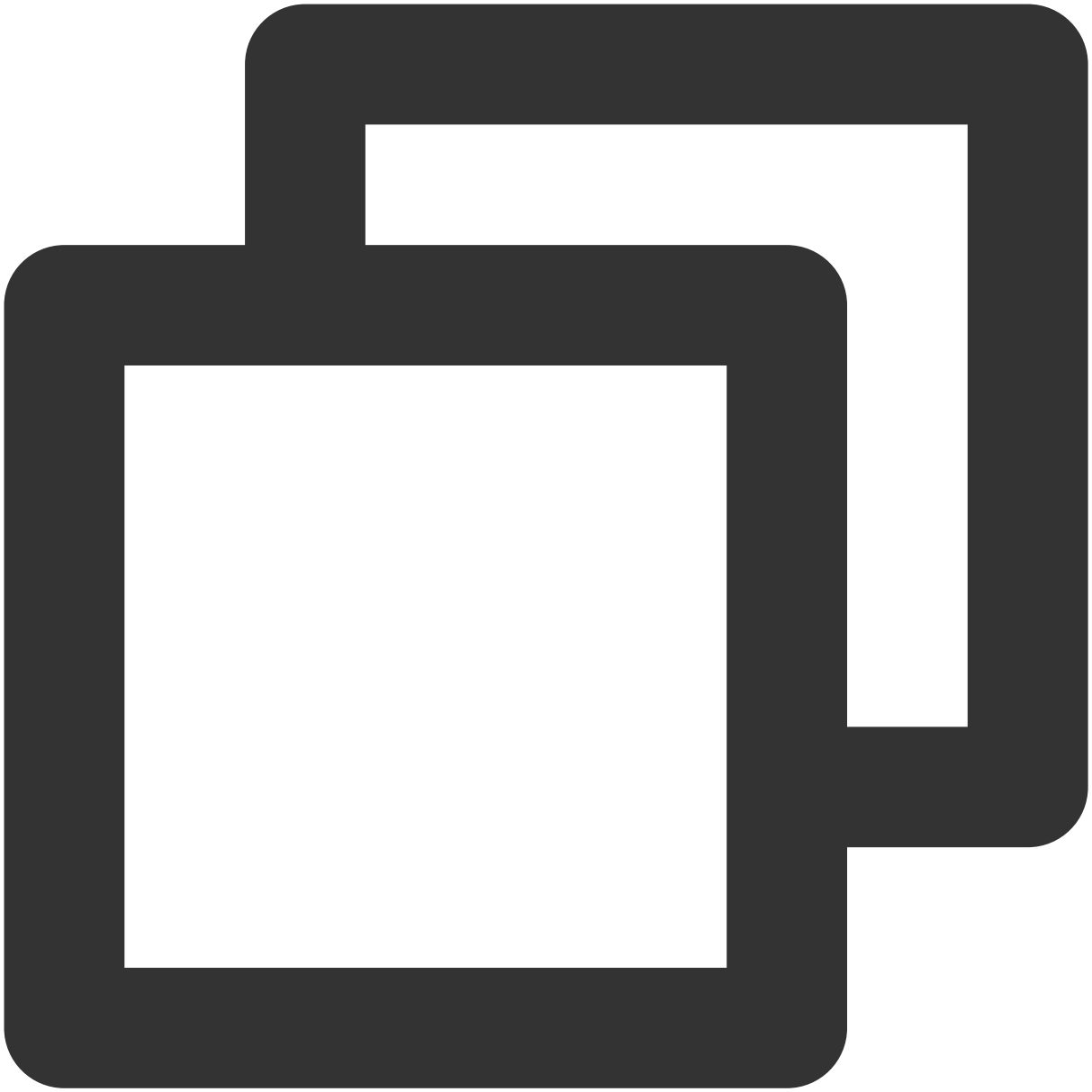
预期输出信息：



```
tencentcloud_monitor_tmp_instance.foo: Refreshing state... [id=prom-jh0zntj2]
tencentcloud_monitor_tmp_exporter_integration.tmpExporterIntegration: Refreshing st
tencentcloud_monitor_tmp_tke_cluster_agent.foo: Refreshing state... [id=prom-jh0znt
tencentcloud_monitor_tmp_exporter_integration.tmpExporterMointor: Refreshing state.

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

Terraform will perform the following actions:
```

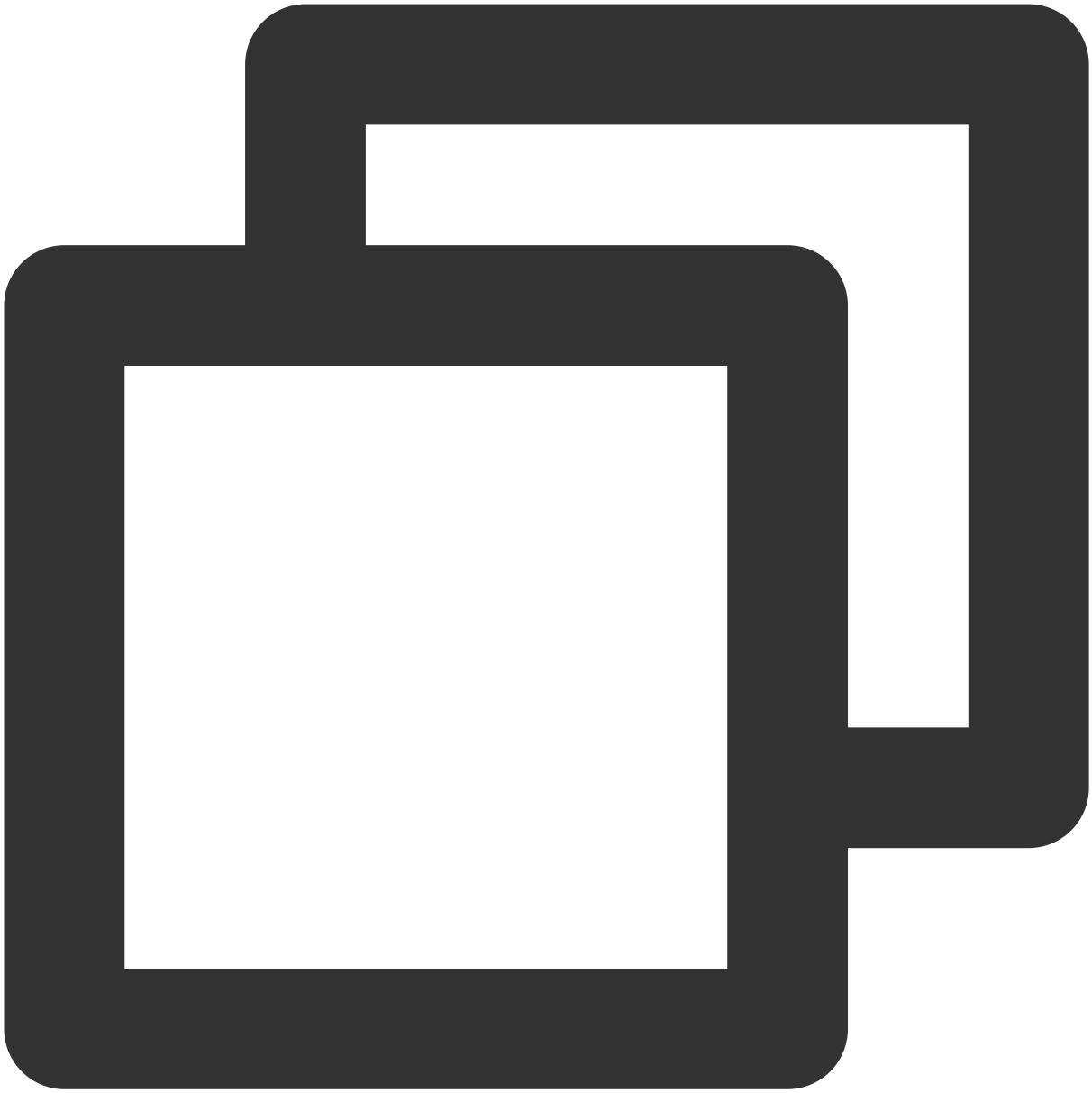
```
  # tencentcloud_monitor_tmp_alert_rule.foo will be created
  + resource "tencentcloud_monitor_tmp_alert_rule" "foo" {
      + duration    = "2m"
      + expr        = "avg by (instance) (mysql_global_status_threads_connected) /
      + id          = (known after apply)
      + instance_id = "prom-jh0zntj2"
      + receivers   = [
          + "notice-zmjsavnp",
        ]
      + rule_name   = "MySQL 连接数过多--tf-云监控test"
      + rule_state  = 2
      + type        = "MySQL/MySQL 连接数过多"

      + annotations {
          + key   = "description"
          + value = "MySQL 连接数过多, 实例: {{$labels.instance}}, 当前值: {{ $value |
        }
      + annotations {
          + key   = "summary"
          + value = "MySQL 连接数过多(>80%)"
        }

      + labels {
          + key   = "severity"
          + value = "warning"
        }
    }

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
you run "terraform apply" now.
```

4. 创建集成中心组件集成，执行命令如下：



```
terraform apply
```

预期输出信息：



```
tencentcloud_monitor_tmp_instance.foo: Refreshing state... [id=prom-jh0zntj2]
tencentcloud_monitor_tmp_exporter_integration.tmpExporterIntegration: Refreshing st
tencentcloud_monitor_tmp_exporter_integration.tmpExporterMointor: Refreshing state.
tencentcloud_monitor_tmp_tke_cluster_agent.foo: Refreshing state... [id=prom-jh0znt

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

Terraform will perform the following actions:
 实例内容。。。
```

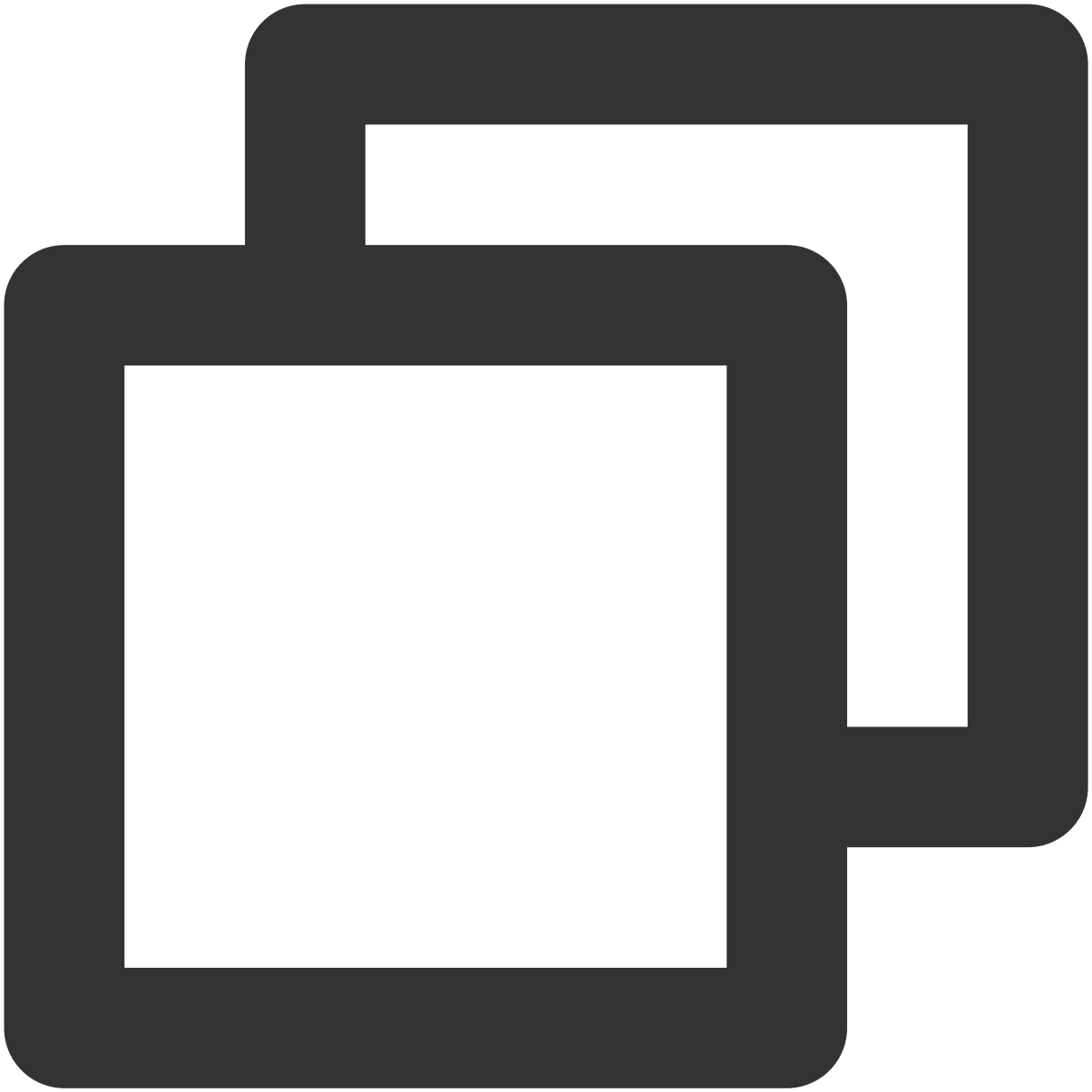
```
Plan: 1 to add, 0 to change, 0 to destroy.

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_monitor_tmp_alert_rule.foo: Creating...
tencentcloud_monitor_tmp_alert_rule.foo: Creation complete after 2s [id=prom-jh0znt

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

查看 Prometheus 实例状态

登录 腾讯云可观测平台，在左侧导航栏，选择 **prometheus 监控，**可在 prometheus 实例列表中看到存在的实例。

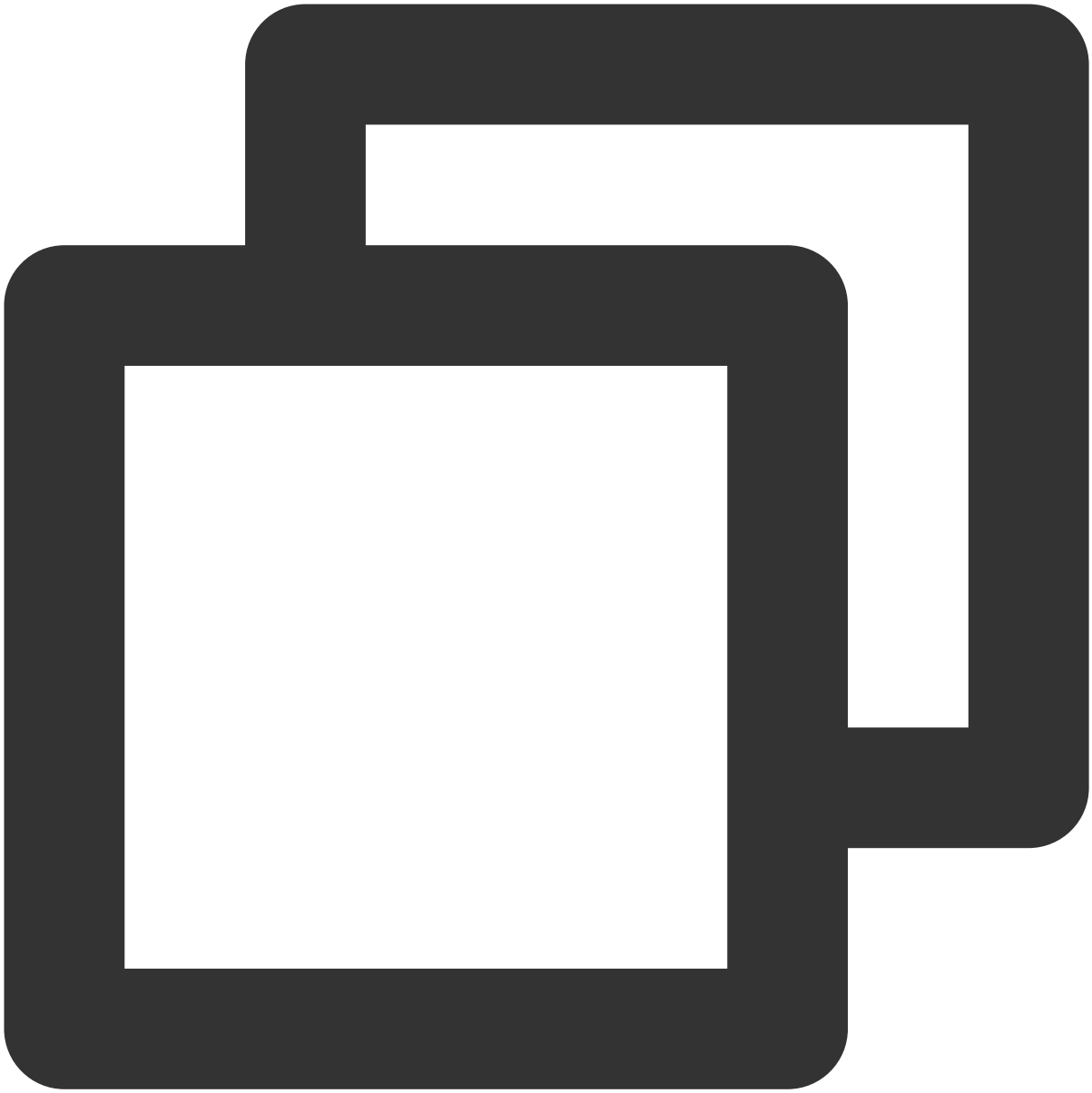
删除 Prometheus 实例集成中心组件集成

销毁资源，执行命令如下：



```
terraform destroy
```

预期输出信息：



```
tencentcloud_monitor_tmp_instance.foo: Refreshing state... [id=prom-8dyb6iny]

Terraform used the selected providers to generate the following execution plan. Res
following symbols:
  - destroy

Terraform will perform the following actions:
实例内容。。。

Do you really want to destroy all resources?
  Terraform will destroy all your managed infrastructure, as shown above.
```

```
  There is no undo. Only 'yes' will be accepted to confirm.

  Enter a value: yes

tencentcloud_monitor_tmp_instance.foo: Destroying... [id=prom-8dyb6iny]
tencentcloud_monitor_tmp_instance.foo: Destruction complete after 6s

Destroy complete! Resources: 1 destroyed.
```

注意：

当出现 `Destroy complete! Resources: （存在的实例个数） destroyed.` 表示您已删除该实例。