云资源自动化 for Terraform

最佳实践

产品文档

腾讯云

文档目录

最佳实践
部署云原生服务

最近更新时间：2023-04-20 17:44:06

本文介绍如何使用 Terraform 创建腾讯云 TKE 标准集群并结合 Terraform Kubernetes Provider 部署一个简单的 Nginx 应用。

前置条件

- Terraform >=0.14.0
- 注册 腾讯云账号。
- 获取凭证，在 API密钥管理 页面中创建并复制 SecretId 和 SecretKey。
- 进入 容器服务控制台，按照界面提示为容器服务授权。

创建 TKE 相关资源

创建任意空目录，如 `tf-tke-example` 。目录创建后，按照以下步骤声明腾讯云资源。

配置基础网络

创建 network.tf 文件，配置私有网络 VPC 、子网和安全组。代码如下：

```
# Networks
variable "vpc_name" {
default = "example-vpc"
}

variable "subnet_name" {
default = "example-subnet"
}

variable "security_group_name" {
default = "example-security-group"
}
```

```
variable "network_cidr" {
default = "10.0.0.0/16"
}

variable "security_ingress_rules" {
default = [
"ACCEPT#10.0.0.0/16#ALL#ALL",
"ACCEPT#172.16.0.0/22#ALL#ALL",
"DROP#0.0.0.0/0#ALL#ALL"
]
}

resource "tencentcloud_vpc" "vpc" {
cidr_block = var.network_cidr
name = var.vpc_name
tags = var.tags
}

resource "tencentcloud_subnet" "subnet" {
availability_zone = var.available_zone
cidr_block = var.network_cidr
name = var.subnet_name
vpc_id = tencentcloud_vpc.vpc.id
tags = var.tags
}

resource "tencentcloud_security_group" "sg" {
name = var.security_group_name
description = "example security groups for kubernetes networks"
tags = var.tags
}

resource "tencentcloud_security_group_lite_rule" "sg_rules" {
security_group_id = tencentcloud_security_group.sg.id
ingress = var.security_ingress_rules
egress = [
"ACCEPT#0.0.0.0/0#ALL#ALL"
]
}
```

配置集群

创建 cluster.tf 文件，配置 TKE 集群。代码如下：

```
# TKE
variable "cluster_name" {
```

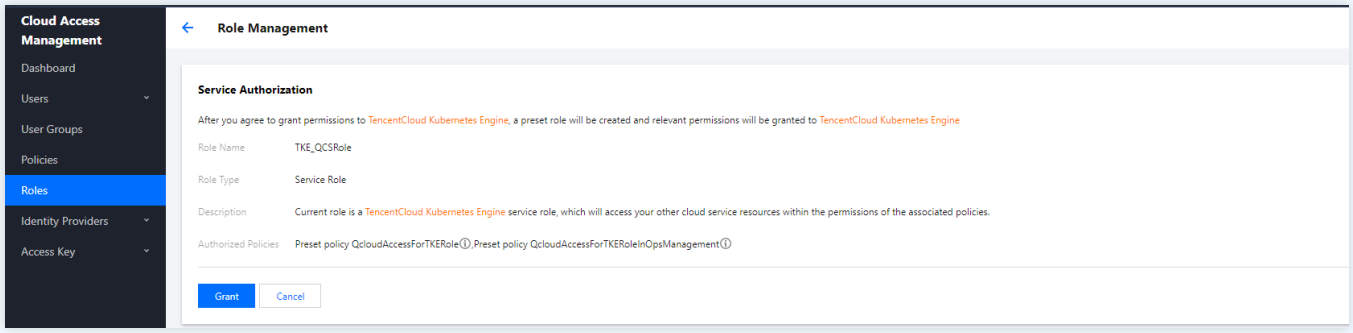
```
default = "example-cluster"
}

variable "cluster_version" {
default = "1.22.5"
}

variable "cluster_cidr" {
default = "172.16.0.0/22"
}

variable "cluster_os" {
default = "tlinux2.2(tkernel3)x86_64"
}

variable "cluster_public_access" {
default = true
}

variable "cluster_private_access" {
default = true
}

variable "worker_count" {
default = 1
}

variable "worker_instance_type" {
default = "S5.MEDIUM2"
}

variable "available_zone" {
default = "ap-guangzhou-3"
}

variable "tags" {
default = {
terraform = "example"
}
}

resource "random_password" "worker_pwd" {
length = 12
min_numeric = 1
min_special = 1
min_upper = 1
override_special = "!#$%&*()-_=+[]{}<>:?"
```

```
}

resource "tencentcloud_kubernetes_cluster" "cluster" {
cluster_name = var.cluster_name
cluster_version = var.cluster_version
cluster_cidr = var.cluster_cidr
cluster_os = var.cluster_os
cluster_internet = var.cluster_public_access
cluster_internet_security_group = var.cluster_public_access ? tencentcloud_securi
ty_group.sg.id : null
cluster_intranet = var.cluster_private_access
cluster_intranet_subnet_id = var.cluster_private_access ? tencentcloud_subnet.sub
net.id : null
vpc_id = tencentcloud_vpc.vpc.id

worker_config {
availability_zone = var.available_zone
count = var.worker_count
instance_type = var.worker_instance_type
subnet_id = tencentcloud_subnet.subnet.id
security_group_ids = [tencentcloud_security_group.sg.id]
password = random_password.worker_pwd.result
}

tags = var.tags
}
```

配置 CAM 角色（可选）

容器服务 TKE 需要访问其他资源的权限，需要 `TKE_QCSRole` 角色并授予其预设策略 `TF_QcloudAccessForTKERole`，`TF_QcloudAccessForTKERoleInOpsManagement` 。

> 注意：
>
> 如果您已经在控制台完成授权（如下图所示），则不需要创建这个文件。



创建 cam.tf 文件，配置 CAM 角色并关联策略。代码如下：

```
resource "tencentcloud_cam_role" "TKE_QCSRole" {
name = "TKE_QCSRole"
document = <<EOF
{
"statement": [
{
"action":"name/sts:AssumeRole",
"effect":"allow",
"principal":{
"service":"ccs.qcloud.com"
}
}
],
"version":"2.0"
}
EOF
description = "The TKE service role."
}

data "tencentcloud_cam_policies" "ops_mgr" {
name = "QcloudAccessForTKERoleInOpsManagement"
}

data "tencentcloud_cam_policies" "qca" {
name = "QcloudAccessForTKERole"
}

locals {
ops_policy_id = data.tencentcloud_cam_policies.ops_mgr.policy_list.0.policy_id
qca_policy_id = data.tencentcloud_cam_policies.qca.policy_list.0.policy_id
}
```

```
resource "tencentcloud_cam_role_policy_attachment" "QCS_OpsMgr" {
role_id = lookup(tencentcloud_cam_role.TKE_QCSRole, "id")
policy_id = local.ops_policy_id
}

resource "tencentcloud_cam_role_policy_attachment" "QCS_QCA" {
role_id = lookup(tencentcloud_cam_role.TKE_QCSRole, "id")
policy_id = local.qca_policy_id
}
```

封装为 Module（可选）

您可以把这些 .tf 文件组织起来，以 Module 的形式使用可以减少关注内部的实现，或者直接参考现有的 Module terraform-tencentcloud-tke ，欢迎使用、 提 Issue 和 Pull Request。

配置 Kubernetes

通过上文的配置，我们可以创建出一个基本的 TKE 托管集群，接下来介绍如何把 Kubernetes 结合 TKE 部署一个简单的 Nginx 应用。

配置 K8s Provider

通过上文编写的 TF 文件，我们可以获取集群的外网访问地址，CA 证书和用户凭证。
以上文 Modules 为例，将集群的主机和凭证填入 Kubernetes Provider 中。代码如下：

```
terraform {
required_providers {
kubernetes = {
source = "hashicorp/kubernetes"
version = ">= 2.0.0"
}
tencentcloud = {
source = "tencentcloudstack/tencentcloud"
version = ">=1.77.7"
}
}
}

provider "tencentcloud" {
region = "ap-hongkong"
}

module "tencentcloud_tke" {
source = "github.com/terraform-tencentcloud-modules/terraform-tencentcloud-tke"
```

```
available_zone = "ap-hongkong-3" # Available zone must belongs to the region.
}

provider "kubernetes" {
host = module.tencentcloud_tke.cluster_endpoint
cluster_ca_certificate = module.tencentcloud_tke.cluster_ca_certificate
client_key = base64decode(module.tencentcloud_tke.client_key)
client_certificate = base64decode(module.tencentcloud_tke.client_certificate)
}
```

配置安全组外网访问

我们不推荐完全放通外网访问，默认的安全组仅放通 `10.0.0.0/16` 、 `172.16.0.0/22` 网段，如要测试集群的外网访问，您需要额外添加期望放通的规则。

修改上文的 `module` 块，传入指定的规则。代码如下：

```
module "tencentcloud_tke" {
source = "../../"
available_zone = var.available_zone # Available zone must belongs to the region.
create_cam_strategy = false
security_ingress_rules = [
"ACCEPT#10.0.0.0/16#ALL#ALL",
"ACCEPT#172.16.0.0/22#ALL#ALL",
"ACCEPT#(改成你的 IP 地址，括号去掉)#ALL#ALL",
"DROP#0.0.0.0/0#ALL#ALL"
]
}
```

配置 resources

在 Terraform 中，我们可以使用 HCL 替代原来的 yaml 声明 Namespace、Deployment 和 Service。代码如下：

```
resource "kubernetes_namespace" "test" {
metadata {
name = "nginx"
}
}

resource "kubernetes_deployment" "test" {
metadata {
name = "nginx"
namespace = kubernetes_namespace.test.metadata.0.name
}
spec {
```

```
replicas = 2
selector {
match_labels = {
app = "MyTestApp"
}
}
template {
metadata {
labels = {
app = "MyTestApp"
}
}
spec {
container {
image = "nginx"
name = "nginx-container"
port {
container_port = 80
}
}
}
}
}
}

resource "kubernetes_service" "test" {
metadata {
name = "nginx"
namespace = kubernetes_namespace.test.metadata.0.name
}
spec {
selector = {
app = kubernetes_deployment.test.spec.0.template.0.metadata.0.labels.app
}
type = "NodePort"
port {
node_port = 30201
port = 80
target_port = 80
}
}
}
```

配置 Ingress

这里介绍如何配置 Ingress 关联负载均衡 (CLB)，以实现公网访问，我们先创建一个 CLB 实例。代码如下：

```
locals {
lb_vpc = module.tencentcloud_tke.vpc_id
lb_sg = module.tencentcloud_tke.security_group_id
}

resource "tencentcloud_clb_instance" "ingress-lb" {
address_ip_version = "ipv4"
clb_name = "example-lb"
internet_bandwidth_max_out = 1
internet_charge_type = "BANDWIDTH_POSTPAID_BY_HOUR"
load_balancer_pass_to_target = true
network_type = "OPEN"
security_groups = [local.lb_sg]
vpc_id = local.lb_vpc
}
```

配置 Ingress，指定刚才创建的 CLB ID。代码如下：

```
resource "kubernetes_ingress_v1" "test" {
metadata {
name = "test-ingress"
namespace = "nginx"
annotations = {
"ingress.cloud.tencent.com/direct-access" = "false"
"kubernetes.io/ingress.class" = "qcloud"
"kubernetes.io/ingress.existLbId" = tencentcloud_clb_instance.ingress-lb.id
"kubernetes.io/ingress.extensiveParameters" = "{\"AddressIPVersion\": \"IPV4\"}"
"kubernetes.io/ingress.http-rules" = "[{\"path\":\"/\",\"backend\":{\"serviceName
\":\"nginx\",\"servicePort\":\"80\"}}]"
"kubernetes.io/ingress.https-rules" = "null"
"kubernetes.io/ingress.qcloud-loadbalance-id" = tencentcloud_clb_instance.ingress
-lb.id
"kubernetes.io/ingress.rule-mix" = "false"
}
}
spec {
rule {
http {
path {
backend {
service {
name = kubernetes_service.test.metadata.0.name
port {
number = 80
}
}
```

```
}
path = "/"
}
}
}
}
}
```

如果需要获取 CLB 的 IP 地址，可以创建一个 output 变量。代码如下：

```
output "load_balancer_ip" {
value = kubernetes_ingress_v1.test.status.0.load_balancer.0.ingress.0.ip
}
```

执行创建

所有 .tf 文件编写好后，按次序执行如下命令：

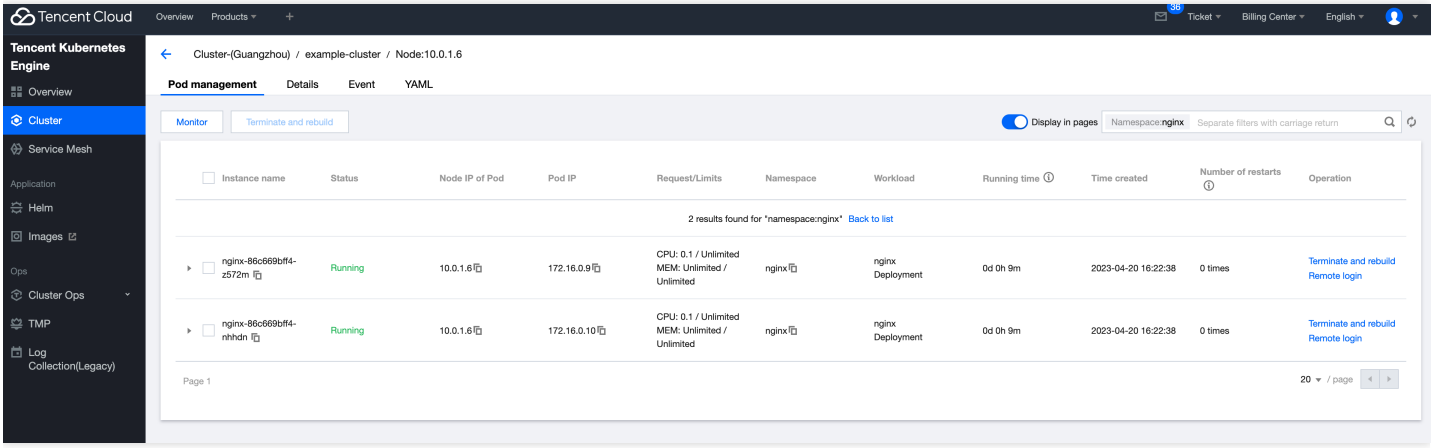
```
$ terraform init
$ terraform plan
$ terraform apply
```

创建成功后，控制台输出上文的 output 信息：

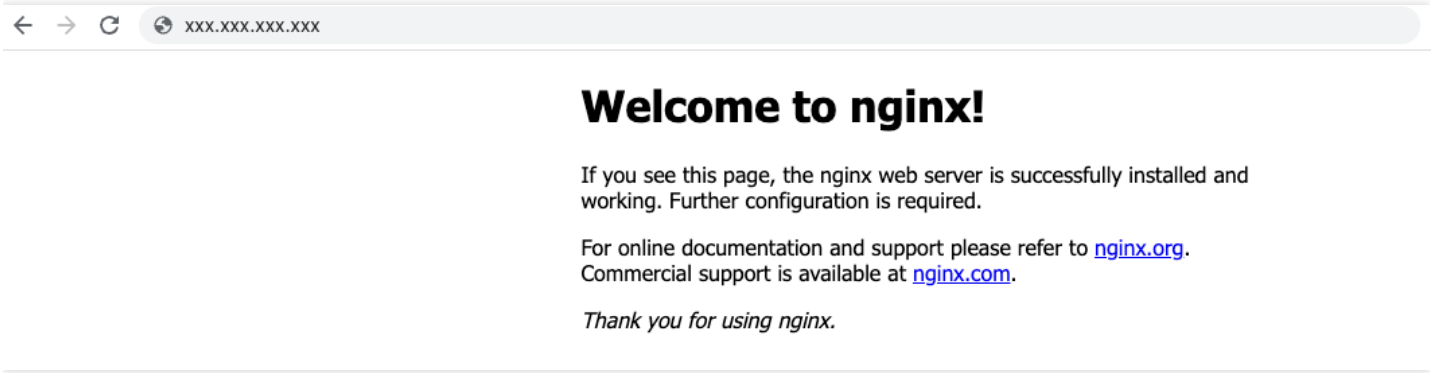
```
Apply complete! Resources: 16 added, 0 changed, 0 destroyed.

Outputs:

load_balancer_ip = "xxx.xxx.xxx.xxx"
```

验证部署

登录腾讯云控制台，访问容器服务 -> example-cluster 集群，可以看到 Nginx 相关的 Pod 已经 Running：



访问 `load_balancer_ip` 显示的地址，页面显示 Welcome To Nginx 则说明应用部署成功！



资源跨地域复制

最近更新时间：2023-05-29 16:04:10

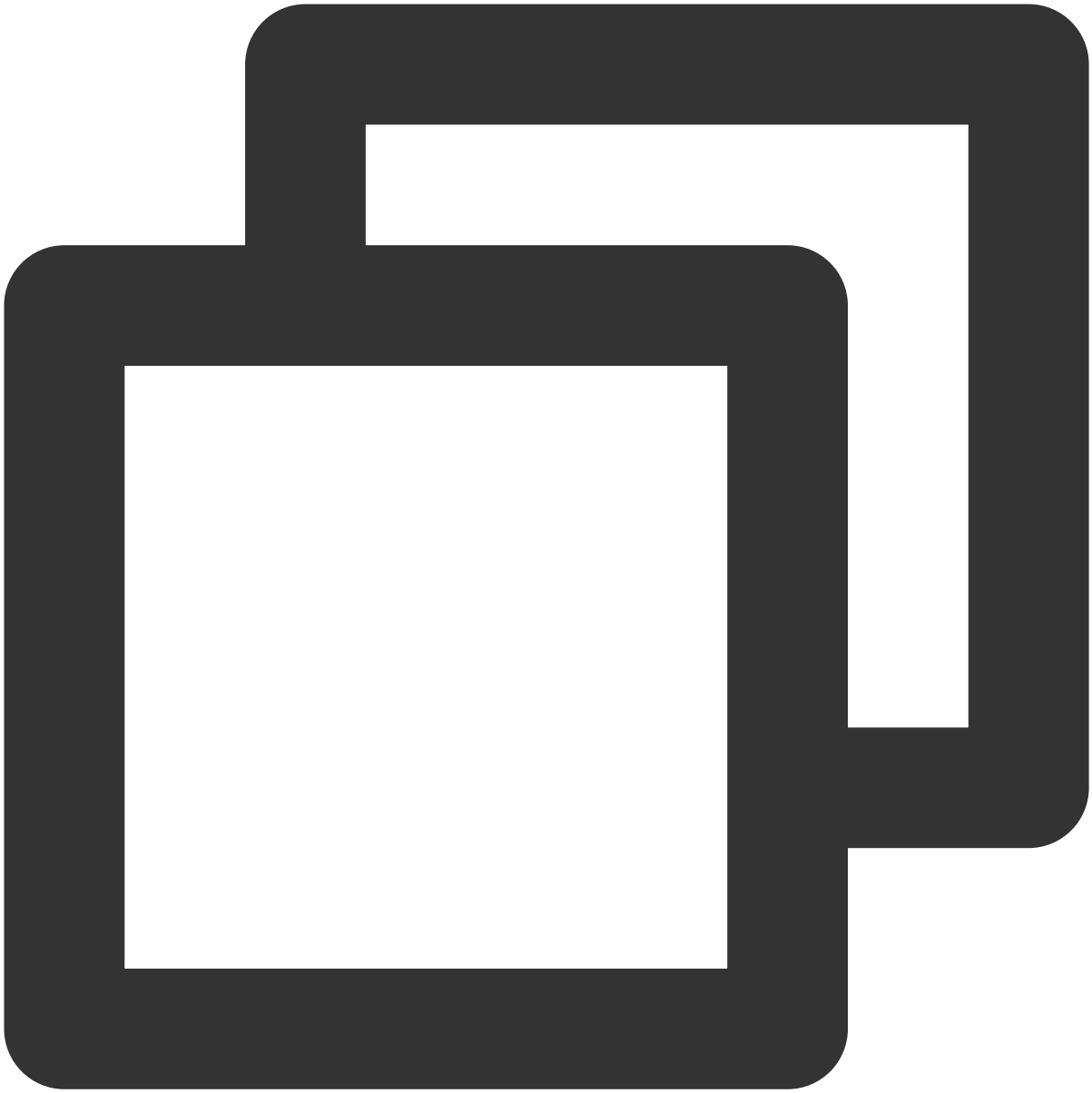
本文介绍如何将已有资源导入 Terraform，以及如何通过复制文件的方式创建新资源，完成跨地域复制。

已有资源导入到 Terraform

大部分刚接触 Terraform 的用户，可能在云上已经存在资源并期望将他们导入到 Terraform 中管理，Terraform 支持单个资源的导入，但是碰到多资源多实例的导入，则需要借助开源工具实现，以下介绍这两种场景的导入方法。

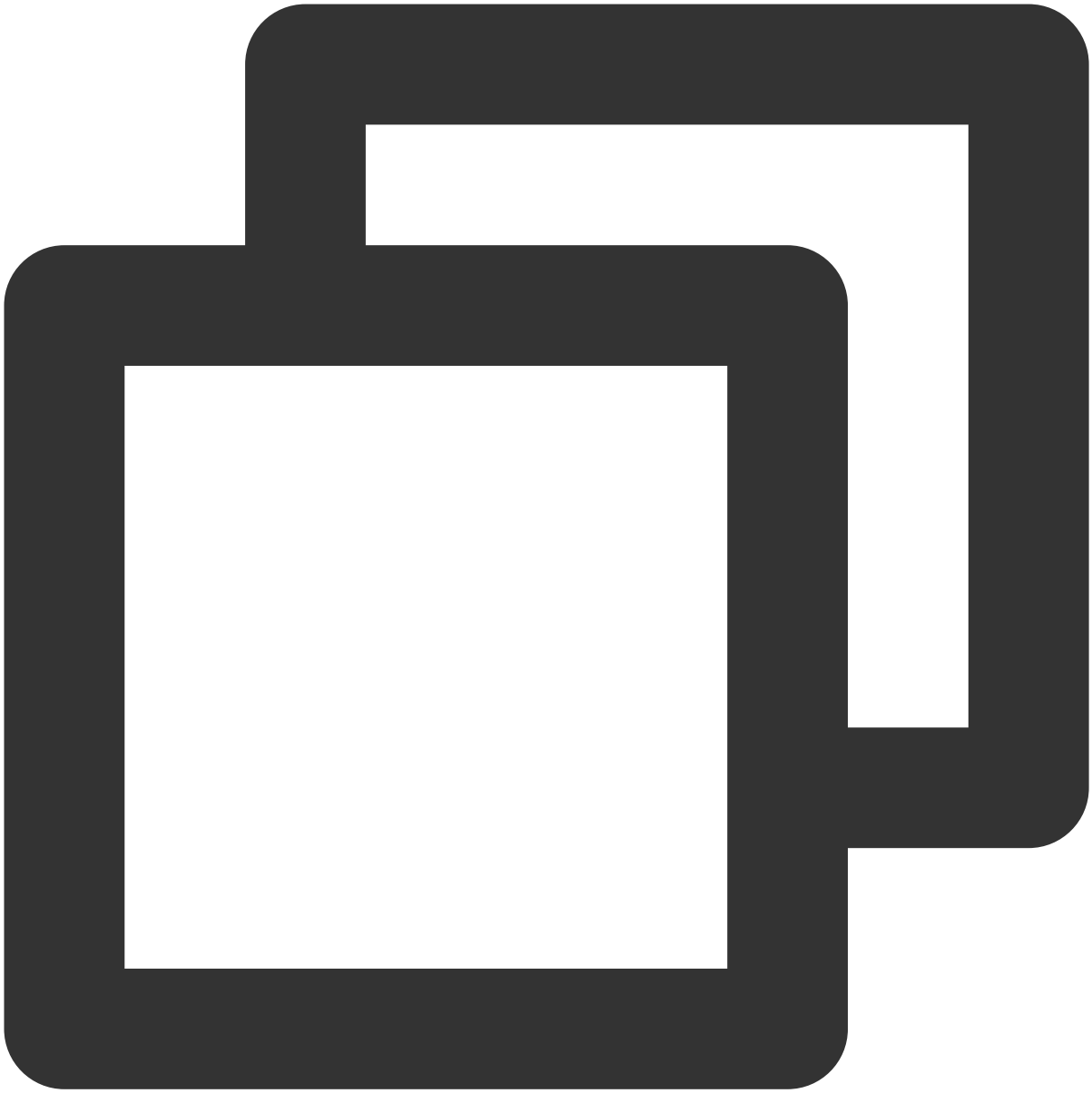
导入单个资源

Terraform 支持 Import 命令导入单个资源，格式 `terraform import [资源类型].[名称] [入参]` 。名称可以自定义，入参则是查询资源必要的字符串（一般为 ID，部分资源是名字或者多字段组合）。以云服务器实例为例，通过查询 CVM Resource 文档，可知导入命令为 ：



```
$ terraform import tencentcloud_instance.ins ins-jvu2hiw2 -allow-missing-config
```

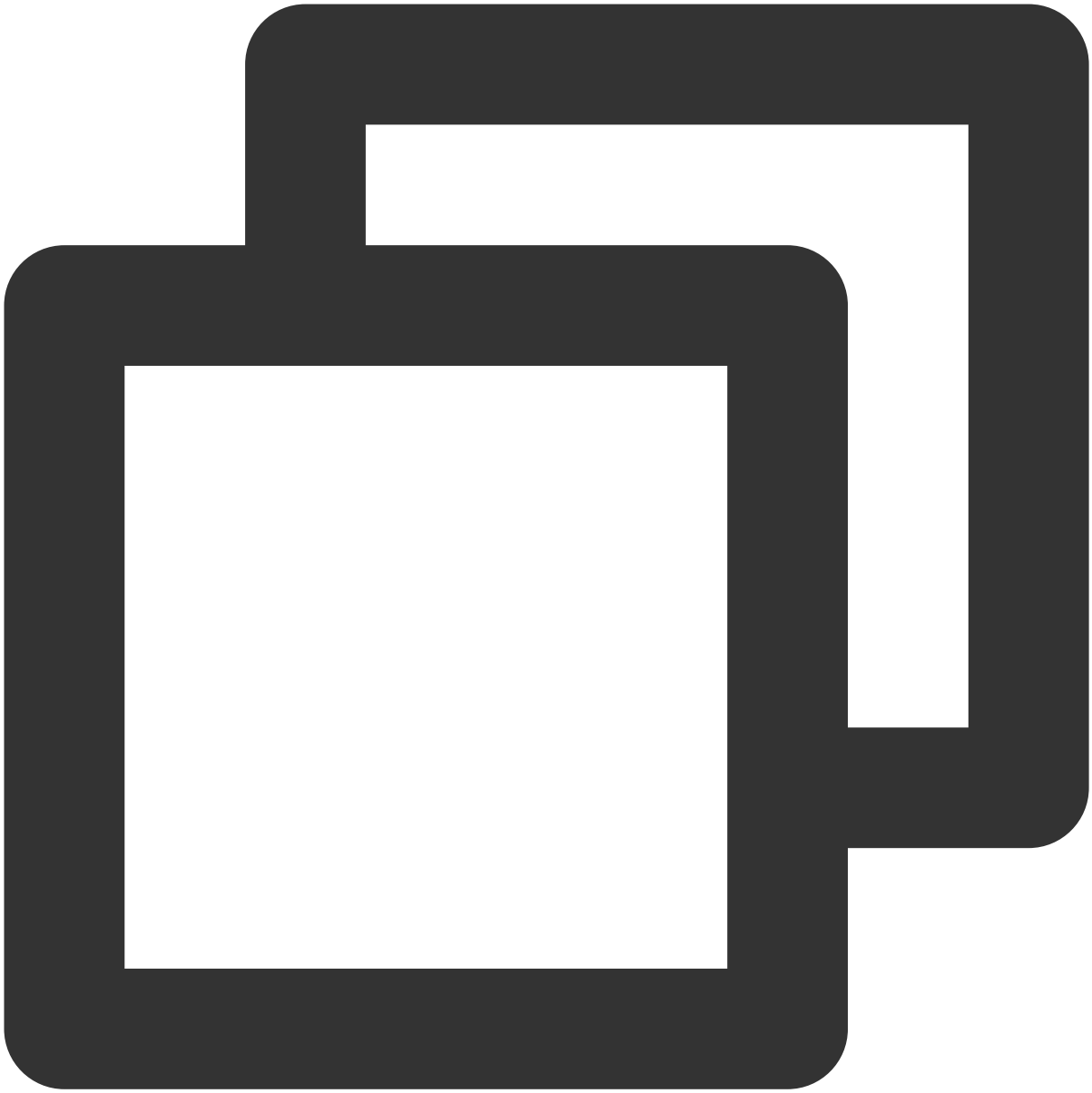
其中 `-allow-missing-config` 表示允许本地不需要预先声明 block，否则需要在文件中预先写一段 `resource [资源类型].[名称] {}` 这样的空块导入完成后，字段**不会**写入 TF 文件中，需要执行 `terraform show` 查看导入的资源代码：



```
# tencentcloud_instance.ins:
resource "tencentcloud_instance" "ins" {
    allocate_public_ip         = true
    availability_zone          = "ap-guangzhou-3"
    create_time                = "2022-01-01T01:11:11Z"
    id                         = "ins-xxxxxxxx"
    image_id                   = "img-xxxxxxxx"
    instance_charge_type       = "POSTPAID_BY_HOUR"
    instance_name              = "xxxxxxxx"
    instance_status            = "RUNNING"
    instance_type              = "S3.MEDIUM2"
```

```
    internet_charge_type        = "TRAFFIC_POSTPAID_BY_HOUR"
    internet_max_bandwidth_out  = 1
    key_name                    = "skey-xxxxxxxx"
    private_ip                  = "10.0.1.1"
    project_id                  = 0
    public_ip                   = "1.1.1.1"
    running_flag                = true
    security_groups             = [
        "sg-xxxxxxxx",
    ]
    subnet_id                   = "subnet-xxxxxxxx"
    system_disk_id              = "disk-xxxxxxxx"
    system_disk_size            = 50
    system_disk_type            = "CLOUD_PREMIUM"
    tags                        = {}
    vpc_id                      = "vpc-xxxxxxxx"
}
```

将这段代码填入您的 TF 文件中，还需要去掉只读字段，通过 Attribute Reference 可知，需要去掉 `id`，`create_time`，`public_ip` 后完成导入。



```
resource "tencentcloud_instance" "ins" {
    allocate_public_ip         = true
    availability_zone          = "ap-guangzhou-3"
#    create_time                = "2022-01-01T01:11:11Z"
#    id                         = "ins-xxxxxxxx"
    image_id                   = "img-xxxxxxxx"
    instance_charge_type       = "POSTPAID_BY_HOUR"
    instance_name              = "xxxxxxxx"
#    instance_status            = "RUNNING"
    instance_type              = "S3.MEDIUM2"
    internet_charge_type       = "TRAFFIC_POSTPAID_BY_HOUR"
```

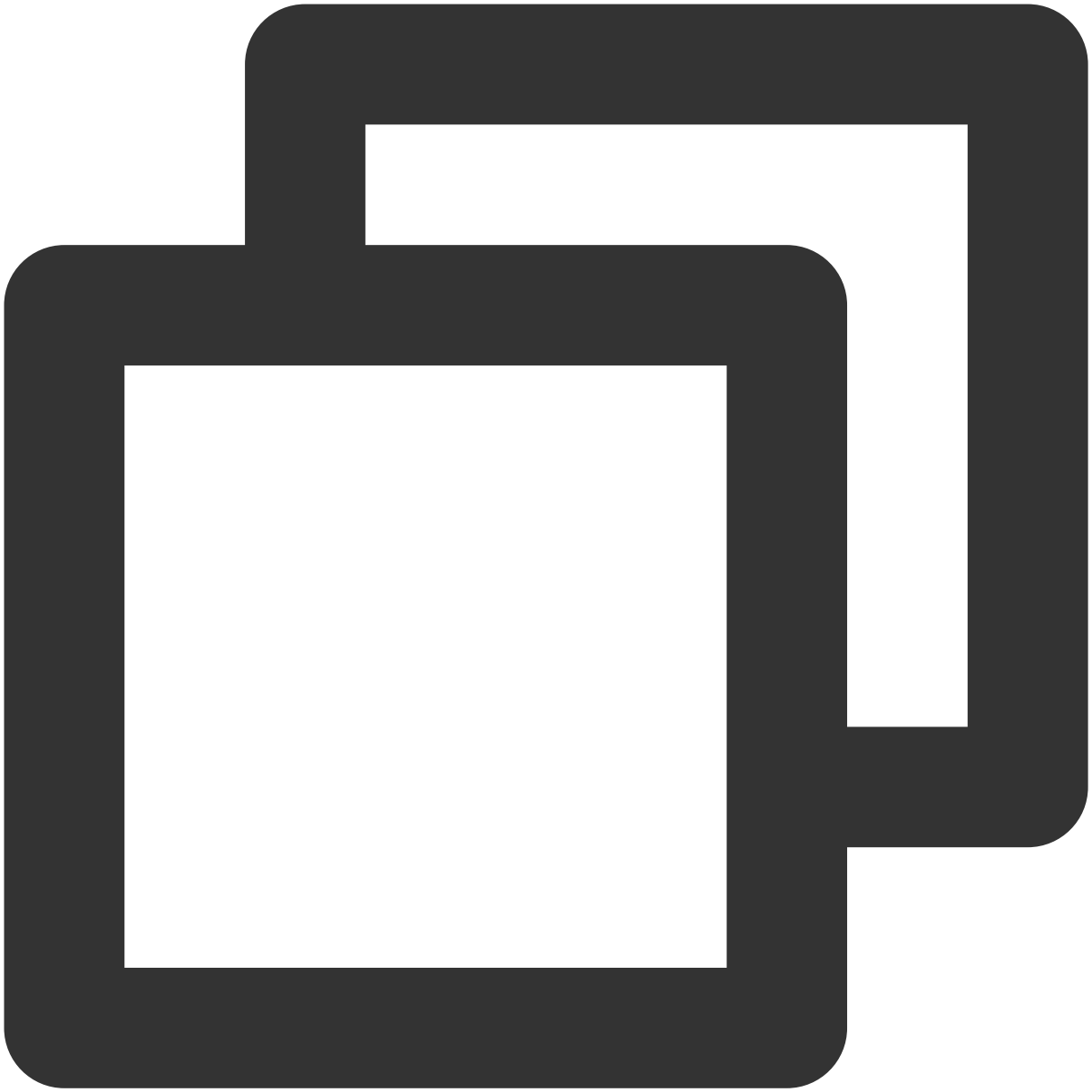
```
    internet_max_bandwidth_out = 1
    key_name                   = "skey-xxxxxxxx"
    private_ip                 = "10.0.1.1"
    project_id                 = 0
#    public_ip                  = "1.1.1.1"
    running_flag               = true
    security_groups            = [
        "sg-xxxxxxxx",
    ]
    subnet_id                  = "subnet-xxxxxxxx"
    system_disk_id             = "disk-xxxxxxxx"
    system_disk_size           = 50
    system_disk_type           = "CLOUD_PREMIUM"
    tags                       = {}
    vpc_id                     = "vpc-xxxxxxxx"
}
```

要获取各个资源的 Import 命令和只读字段，访问 文档 中对应的实例中可以查询。如果未填写，则说明该资源暂不支持导入。

使用 Terraformer 批量导入

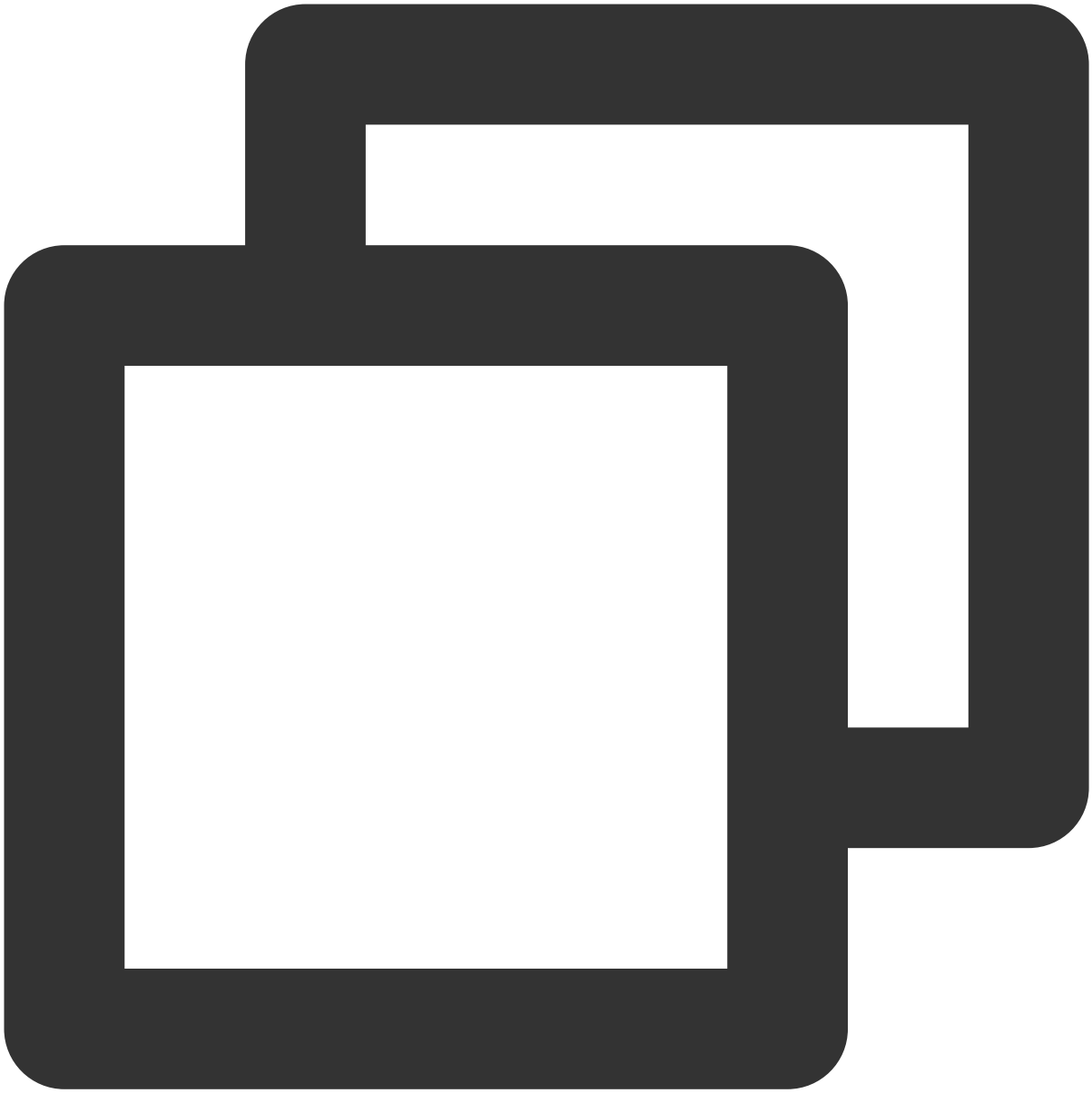
通过上文可以看到使用 Terraform 的导入相当繁琐，仅适合导入少量资源。您可能需要借助 Terraformer 进行批量导入。Terraformer 是一个属于 GoogleCloudPlatform 的命令行工具，可以把账号下大部分云资源标记并导入为 TF 文件。

1. 安装



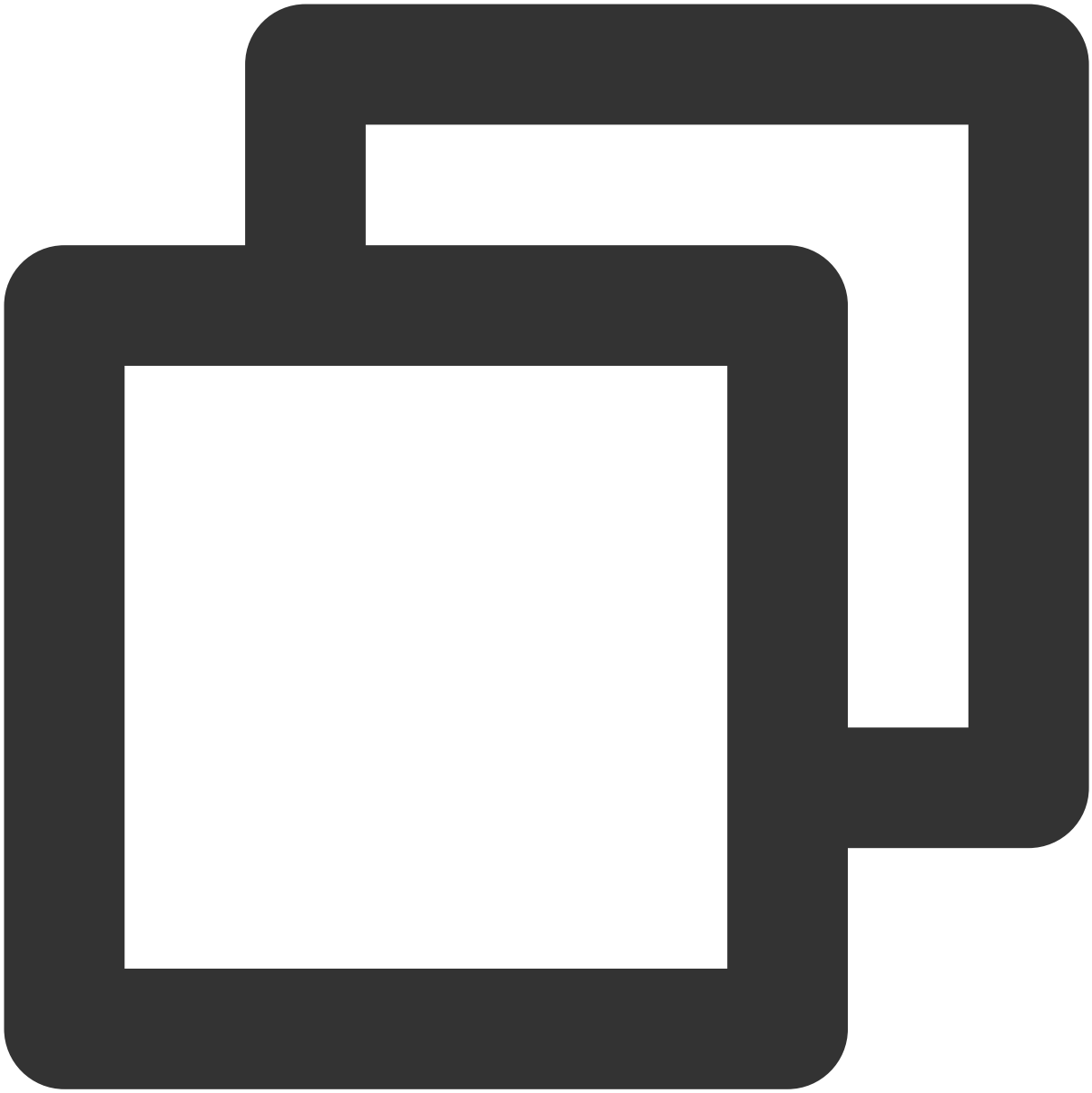
```
$ brew install terraformer
```

2. 执行导入命令，假如我想导入腾讯云广州区下所有的 CVM 和 VPC 资源，那么命令格式如下：



```
terraformer import tencentcloud --resources="vpc,cvm" --regions=ap-guangzhou
```

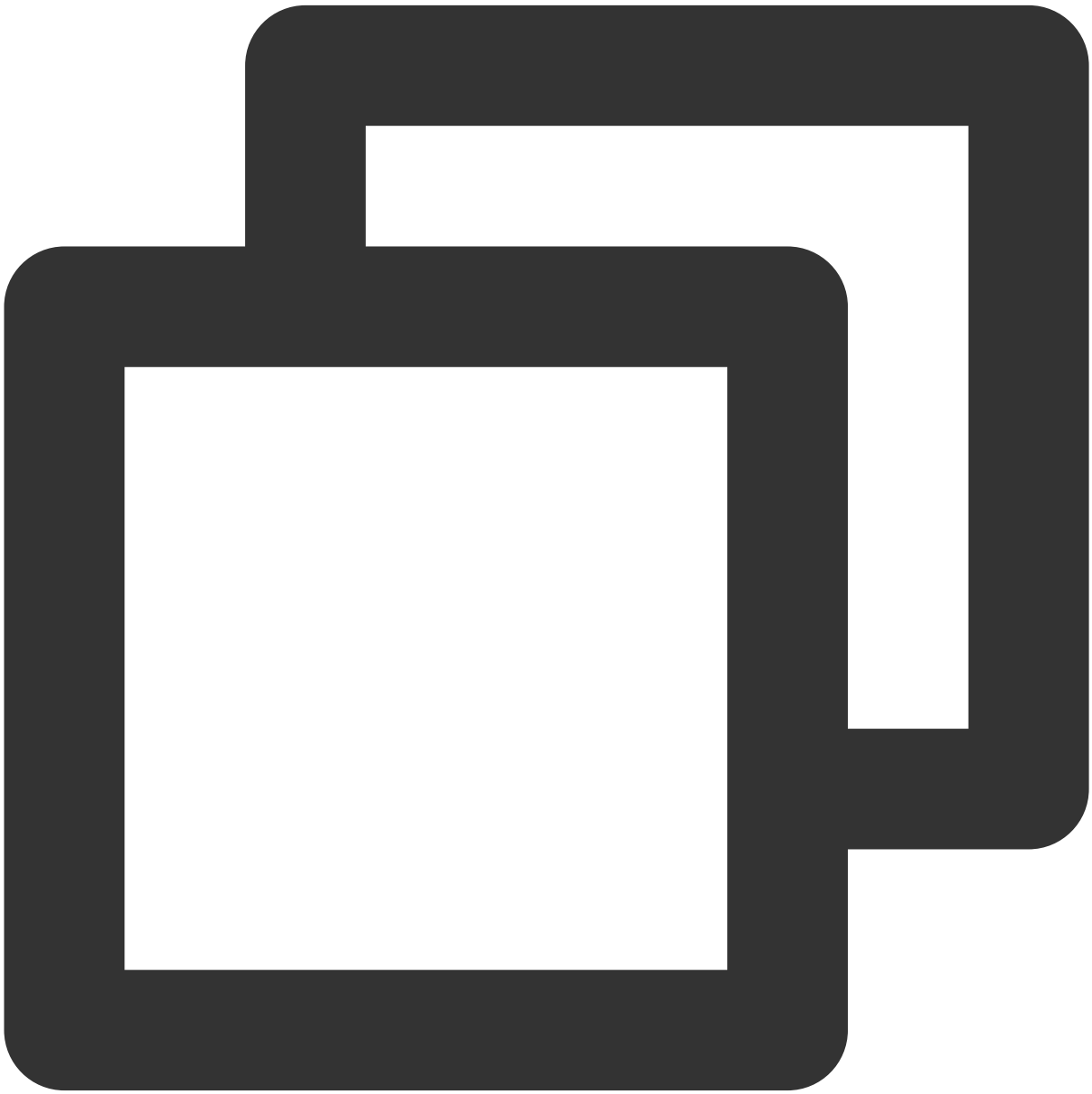
命令执行完成后，Terraformer 默认将导入的资源文件写入 `./generated` 目录，示例如下：



```
.
└── tencentcloud
    ├── cvm
    │   └── ap-guangzhou
    │       ├── instance.tf
    │       ├── key_pair.tf
    │       ├── outputs.tf
    │       ├── provider.tf
    │       ├── terraform.tfstate
    │       └── variables.tf
    └── vpc
```

```
        └── ap-guangzhou
            ├── outputs.tf
            ├── provider.tf
            ├── terraform.tfstate
            └── vpc.tf
```

3. **换源**：TencentCloudProvider 由我们腾讯云维护而非 Terraform 官方，需要在生成的 `provider.tf` 中添加 `source` 字段，值为 `tencentcloudstack/tencentcloud` 。



```
provider "tencentcloud" {
  version = "~> 1.77.11"
```

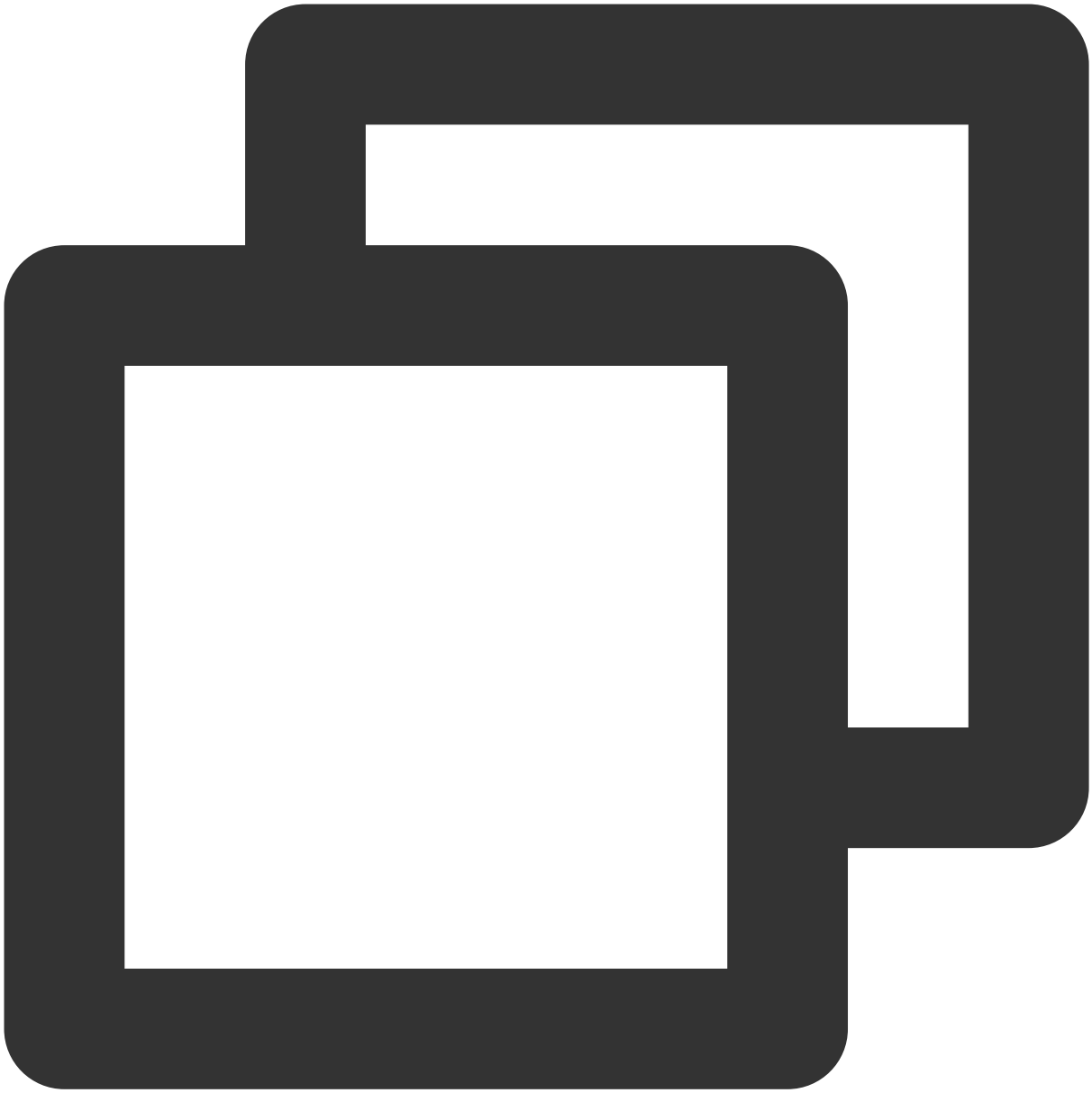
```
}

terraform {
  required_providers {
    tencentcloud = {
      source  = "tencentcloudstack/tencentcloud" # 添加 source 以指定命名空间
      version = "~> 1.77.11"
    }
  }
}
```

当然，不是所有的腾讯云资源 Terraformer 都支持导入，查看已支持导入的资源参考 Terraformer 源码。

跨地域复制

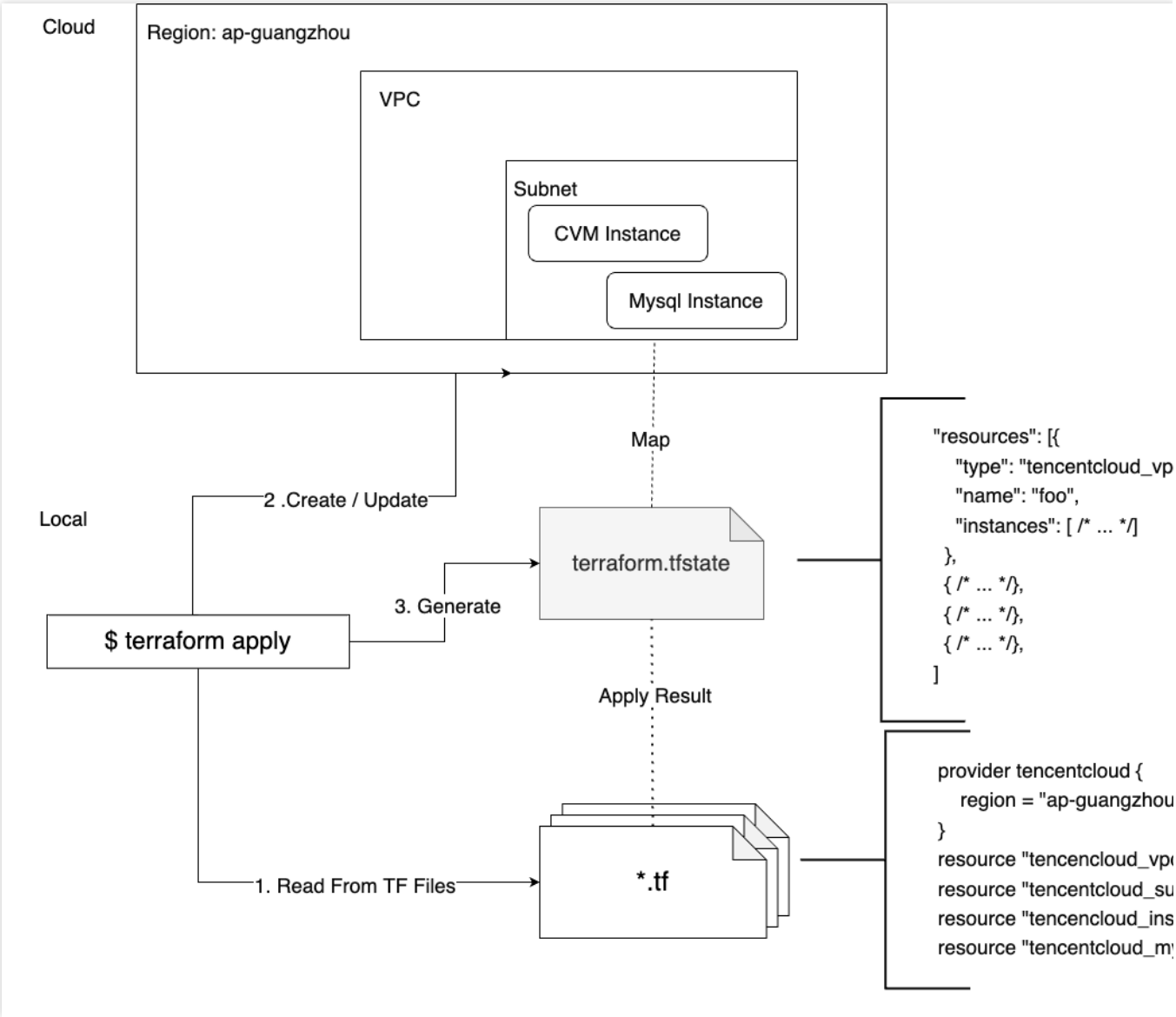
实现原理

一个简单的 Terraform 工作目录结构如下：



```
.
├── .terraform
│   └── providers # 引用到的 Provider
├── .terraform.lock.hcl # Provider 锁版本
├── main.tf             # TF 文件
├── vars.tf             # TF 文件
├── outputs.tf          # TF 文件
└── terraform.tfstate   # 状态文件
```

而本地的工作目录跟腾讯云资源映射结构如下图所示：

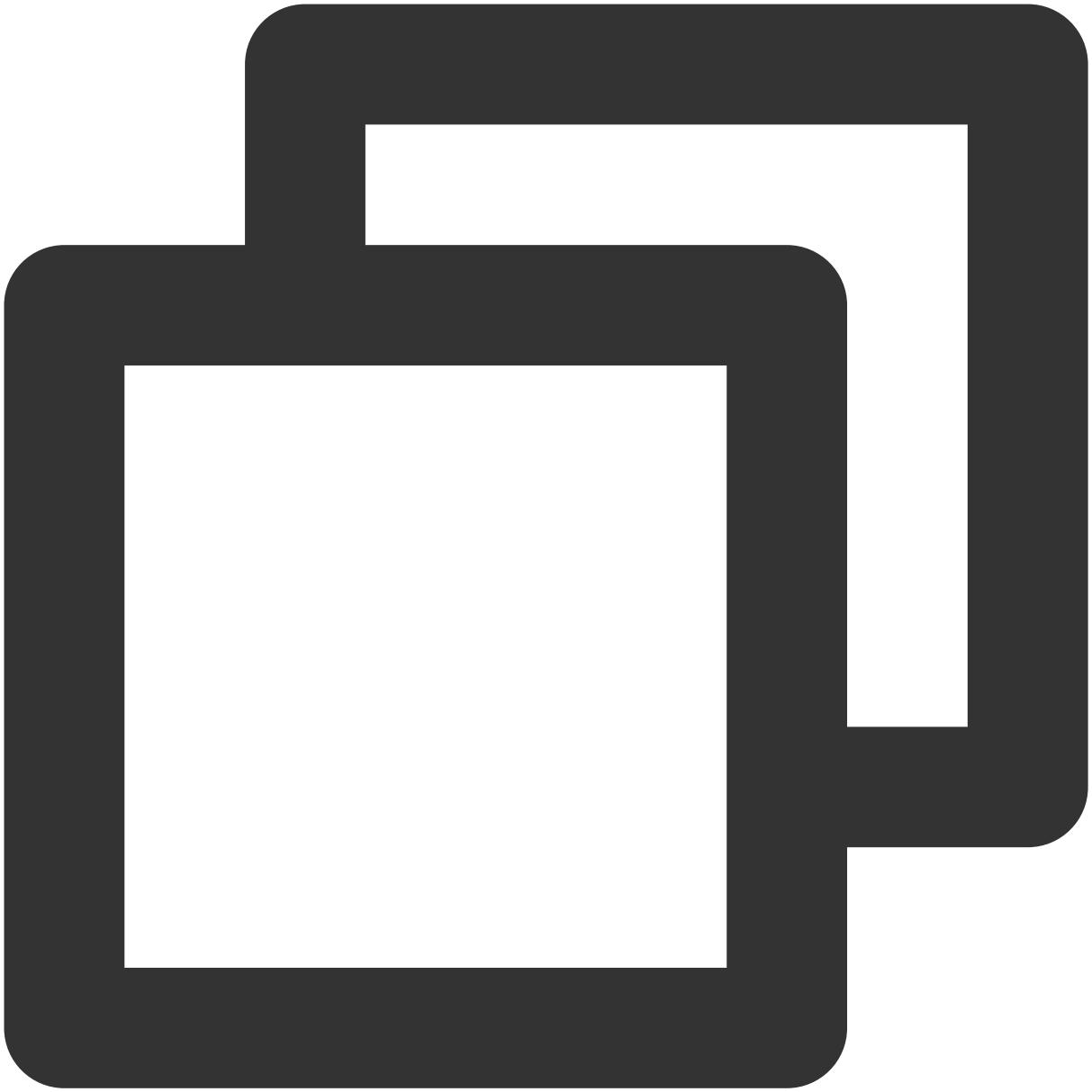


当用户执行 `terraform apply` 命令且部署完成后。会生成 `terraform.tfstate` 文件。它是一份 JSON 格式的文件，默认存储在本地，或者配置在远端存储桶中（需要配置 Backend ）用来描述 TF 声明的资源和真实云资源的映射关系。如果本地目录或 Backend 中不存在 `terraform.tfstate` ，或者该文件没有写入云资源数据，Terraform 就会认为资源没有被部署，执行 `apply` 会进行资源创建操作。

示例：TKE Serverless 集群跨地域复制

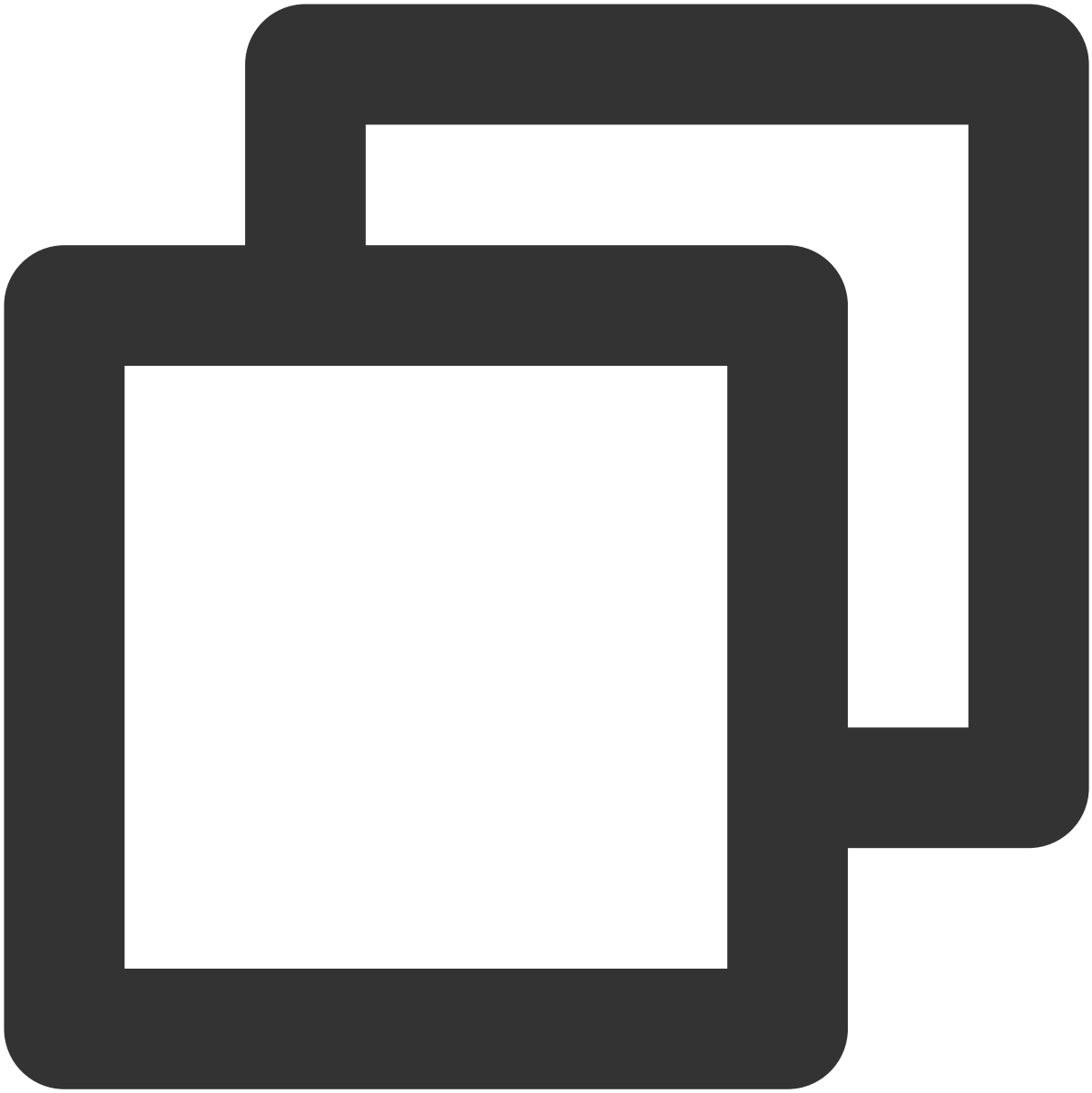
只要没有 `tfstate` 映射的资源声明都视为创建。基于这一思路，我们通过复制文件并修改地域的方法，再执行 `apply` 即可完成资源跨地域复制。

假设我们已经通过 Terraform 在广州部署了一套基于 Serverless 集群服务的应用，目录如下：



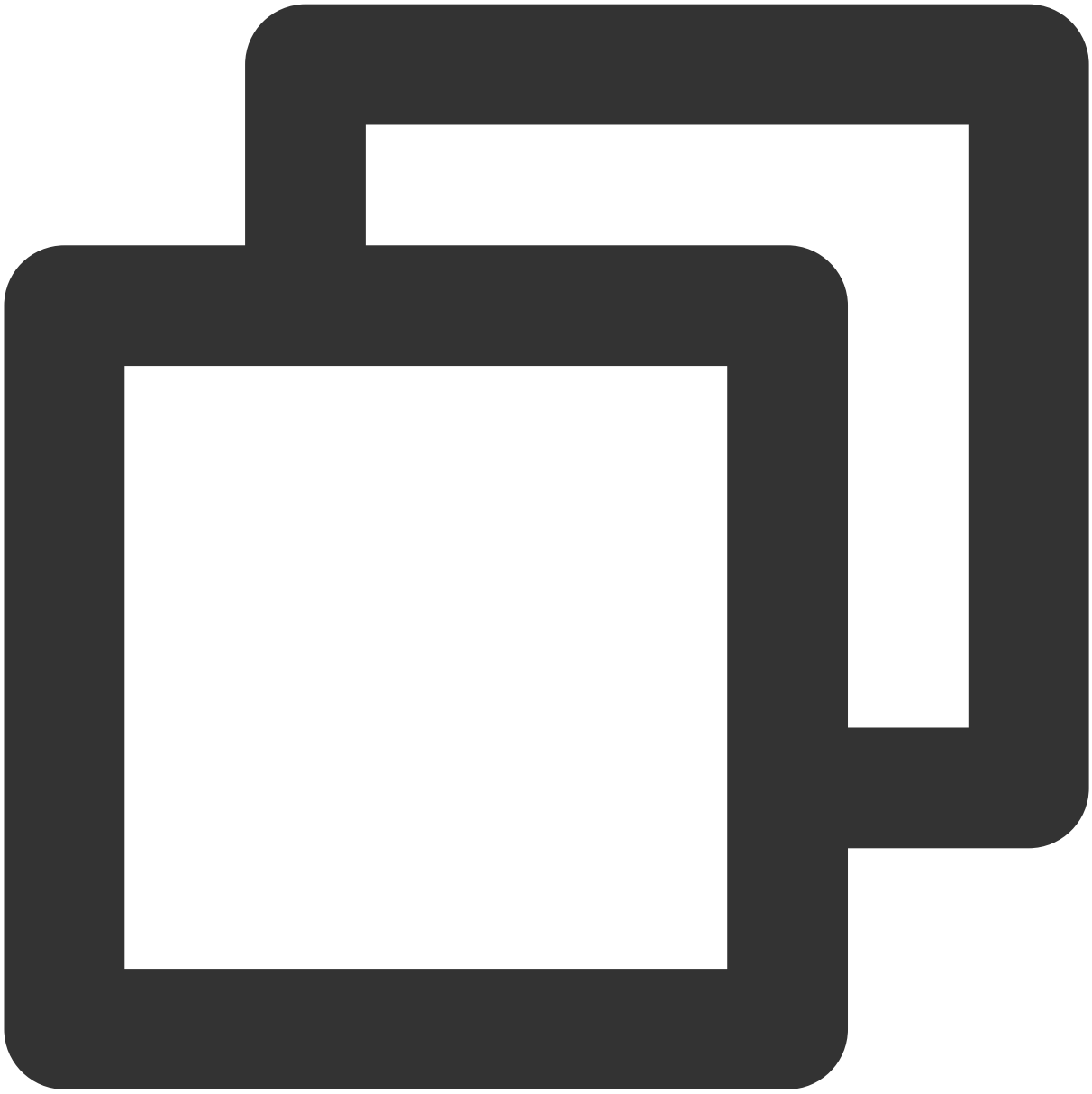
```
eks-app-guangzhou
├── crds.tf
├── infra.tf
├── main.tf
├── terraform.log
└── terraform.tfstate
```

其中 `main.tf` 指定 Terraform 和 Provider 的元信息。代码如下：



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

provider "tencentcloud" {
  region = "ap-guangzhou"
}
```

`infra.tf` 指定 TKE Serverless 集群和所需的资源：VPC、子网、安全组、TKE Serverless 集群、负载均衡。代码如下：

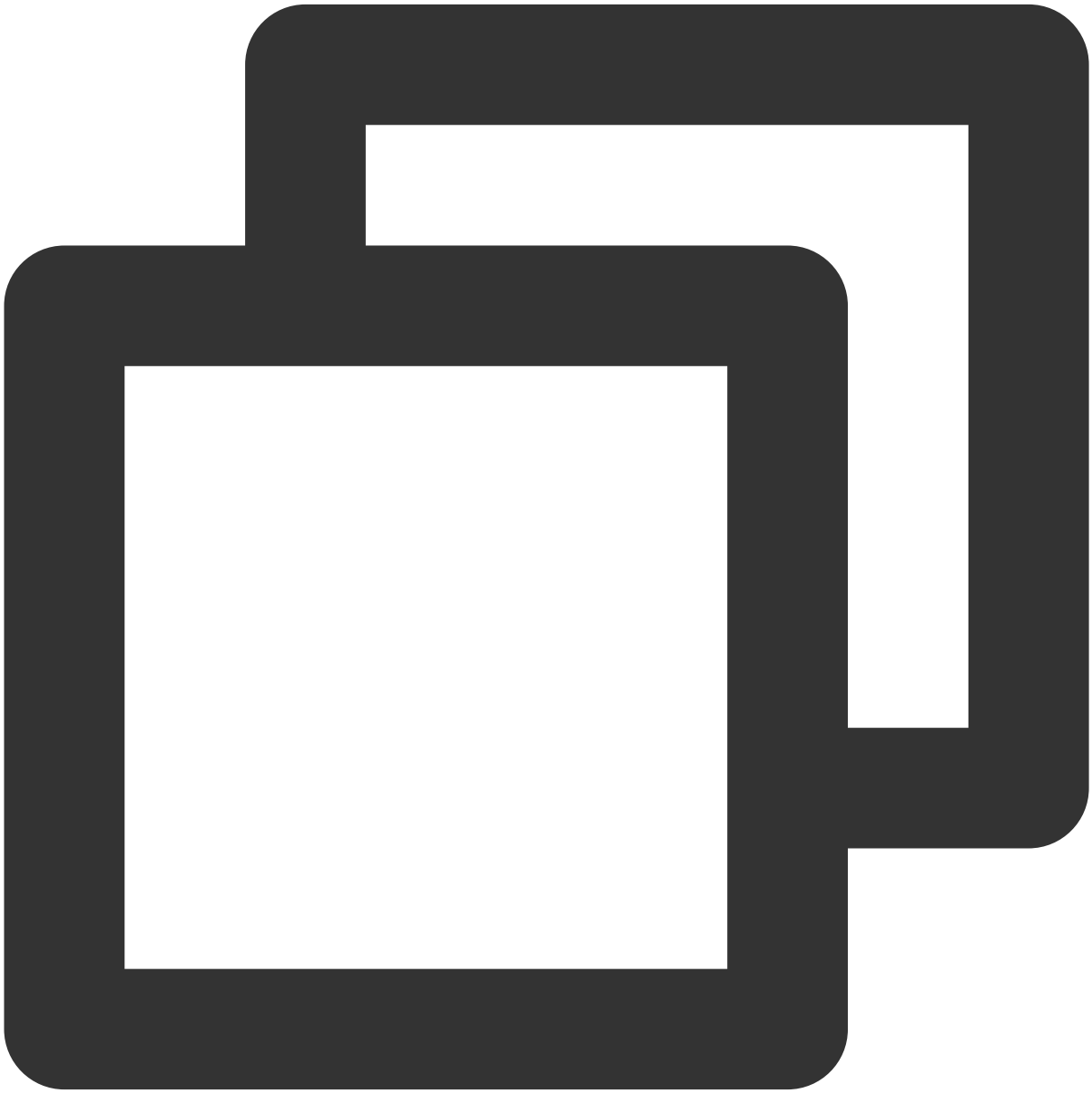


```
# 服务对外放通测试 IP 地址
variable "accept_ip" {
   description = "Use EnvVar: $TF_VAR_accept_ip instead"
}

resource "tencentcloud_vpc" "vpc" {
  name       = "eks-vpc"
```

```
  cidr_block = "10.2.0.0/16"
}

resource "tencentcloud_subnet" "sub" {
  vpc_id            = tencentcloud_vpc.vpc.id
  name              = "eks-subnet"
  cidr_block        = "10.2.0.0/20"
  availability_zone = "ap-guangzhou-3"
}

resource "tencentcloud_security_group" "sg" {
  name = "eks-sg"
}

resource "tencentcloud_security_group_lite_rule" "sgr" {
  security_group_id = tencentcloud_security_group.sg.id
  ingress = [
    "ACCEPT#10.2.0.0/16#ALL#ALL",
    "ACCEPT#${var.accept_ip}#ALL#ALL"
  ]
}

resource "tencentcloud_eks_cluster" "foo" {
  cluster_name = "tf-test-eks"
  k8s_version = "1.20.6"
  vpc_id = tencentcloud_vpc.vpc.id
  subnet_ids = [
    tencentcloud_subnet.sub.id,
  ]
  cluster_desc = "test eks cluster created by terraform"
  service_subnet_id = tencentcloud_subnet.sub.id
  enable_vpc_core_dns = true
  need_delete_cbs = true
  public_lb {
    enabled = true
    security_policies = [var.accept_ip]
  }
  internal_lb {
    enabled = true
    subnet_id = tencentcloud_subnet.sub.id
  }
}

resource "tencentcloud_clb_instance" "ingress-lb" {
  address_ip_version           = "ipv4"
  clb_name                     = "example-lb"
  internet_bandwidth_max_out   = 1
```

```
  internet_charge_type          = "BANDWIDTH_POSTPAID_BY_HOUR"
  load_balancer_pass_to_target = true
  network_type                  = "OPEN"
  security_groups               = [tencentcloud_security_group.sg.id]
  vpc_id                        = tencentcloud_vpc.vpc.id
}
```

`crds.tf` 指定基于 TKE Serverless 集群的 CRD。代码如下：



```
locals {
```

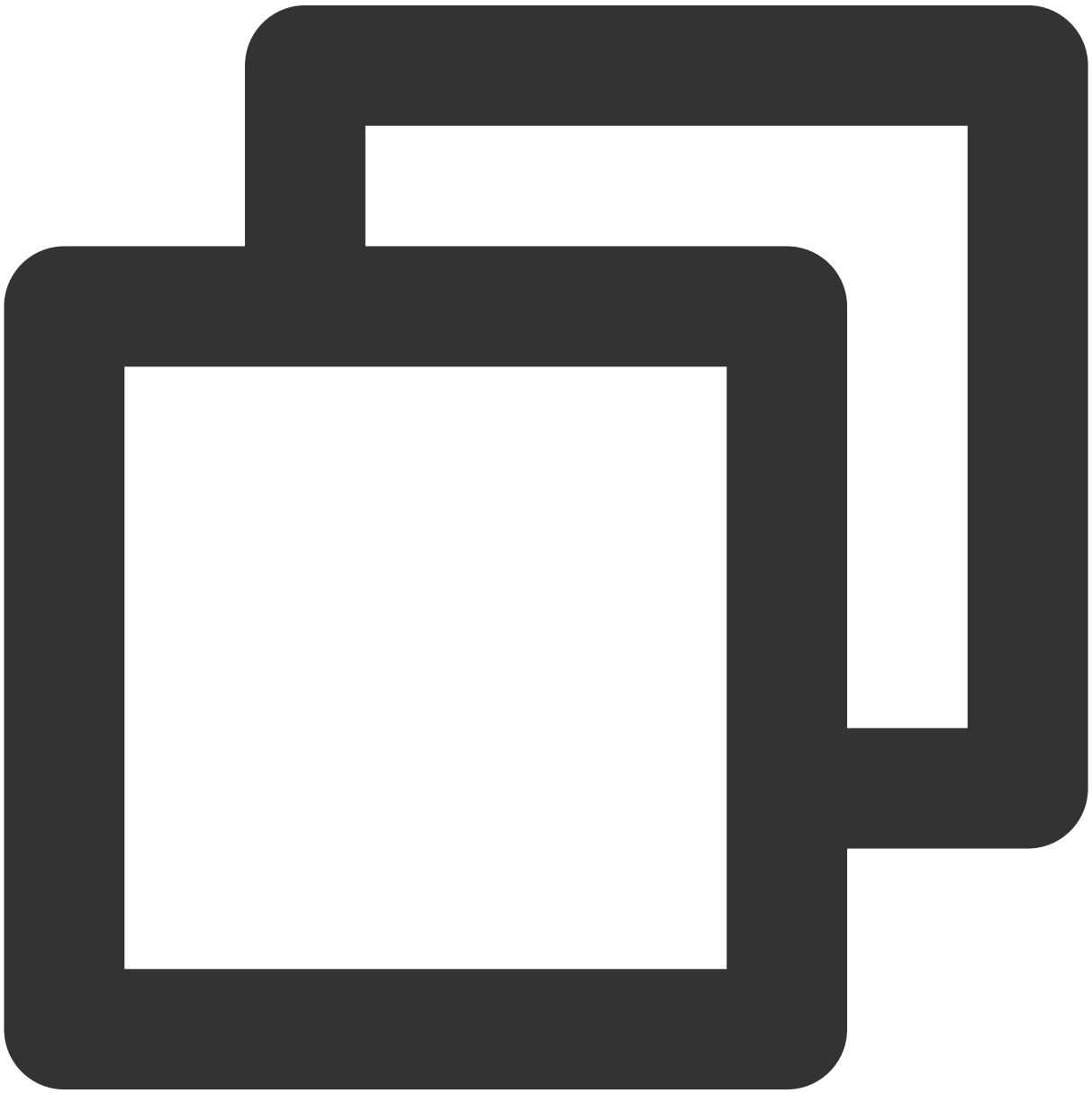
```
  kubeconfig = yamldecode(tencentcloud_eks_cluster.foo.kube_config)
}

provider "kubernetes" {
  host                   = local.kubeconfig.clusters[0].cluster.server
  cluster_ca_certificate = base64decode(local.kubeconfig.clusters[0].cluster["certi
  client_key             = base64decode(local.kubeconfig.users[0].user["client-key-
  client_certificate     = base64decode(local.kubeconfig.users[0].user["client-cert
}

resource "kubernetes_namespace" "test" {
  metadata {
    name = "nginx"
  }
}

resource "kubernetes_deployment" "test" {
  metadata {
    name      = "nginx"
    namespace = kubernetes_namespace.test.metadata.0.name
  }
  spec {
    replicas = 2
    selector {
      match_labels = {
        app = "MyTestApp"
      }
    }
    template {
      metadata {
        labels = {
          app = "MyTestApp"
        }
      }
      spec {
        container {
          image = "nginx"
          name  = "nginx-container"
          port {
            container_port = 80
          }
        }
      }
    }
  }
}
```

```
resource "kubernetes_service" "test" {
  metadata {
    name      = "nginx"
    namespace = kubernetes_namespace.test.metadata.0.name
  }
  spec {
    selector = {
      app = kubernetes_deployment.test.spec.0.template.0.metadata.0.labels.app
    }
    type = "NodePort"
    port {
      node_port   = 30201
      port        = 80
      target_port = 80
    }
  }
}

resource "kubernetes_ingress_v1" "test" {
  metadata {
    name      = "test-ingress"
    namespace = "nginx"
    annotations = {
      "ingress.cloud.tencent.com/direct-access"     = "false"
      "kubernetes.io/ingress.class"                 = "qcloud"
      "kubernetes.io/ingress.existLbId"             = tencentcloud_clb_instance.ing
      "kubernetes.io/ingress.extensiveParameters"   = "{\\"AddressIPVersion\\": \\"
      "kubernetes.io/ingress.http-rules"            = "[{\\"path\\":\\"/\\",\\"back
      "kubernetes.io/ingress.https-rules"           = "null"
      "kubernetes.io/ingress.qcloud-loadbalance-id" = tencentcloud_clb_instance.ing
      "kubernetes.io/ingress.rule-mix"              = "false"
    }
    #    selfLink = "/apis/networking.k8s.io/v1/namespaces/nginx/ingresses/test-ing
  }
  spec {
    rule {
      http {
        path {
          backend {
            service {
              name = kubernetes_service.test.metadata.0.name
              port {
                number = 80
              }
            }
          }
          path = "/"
```

```
          }
        }
      }
    }
}
```

如果想要将这些资源复制一份到其他地域（以新加坡为例），那么可以执行以下步骤：

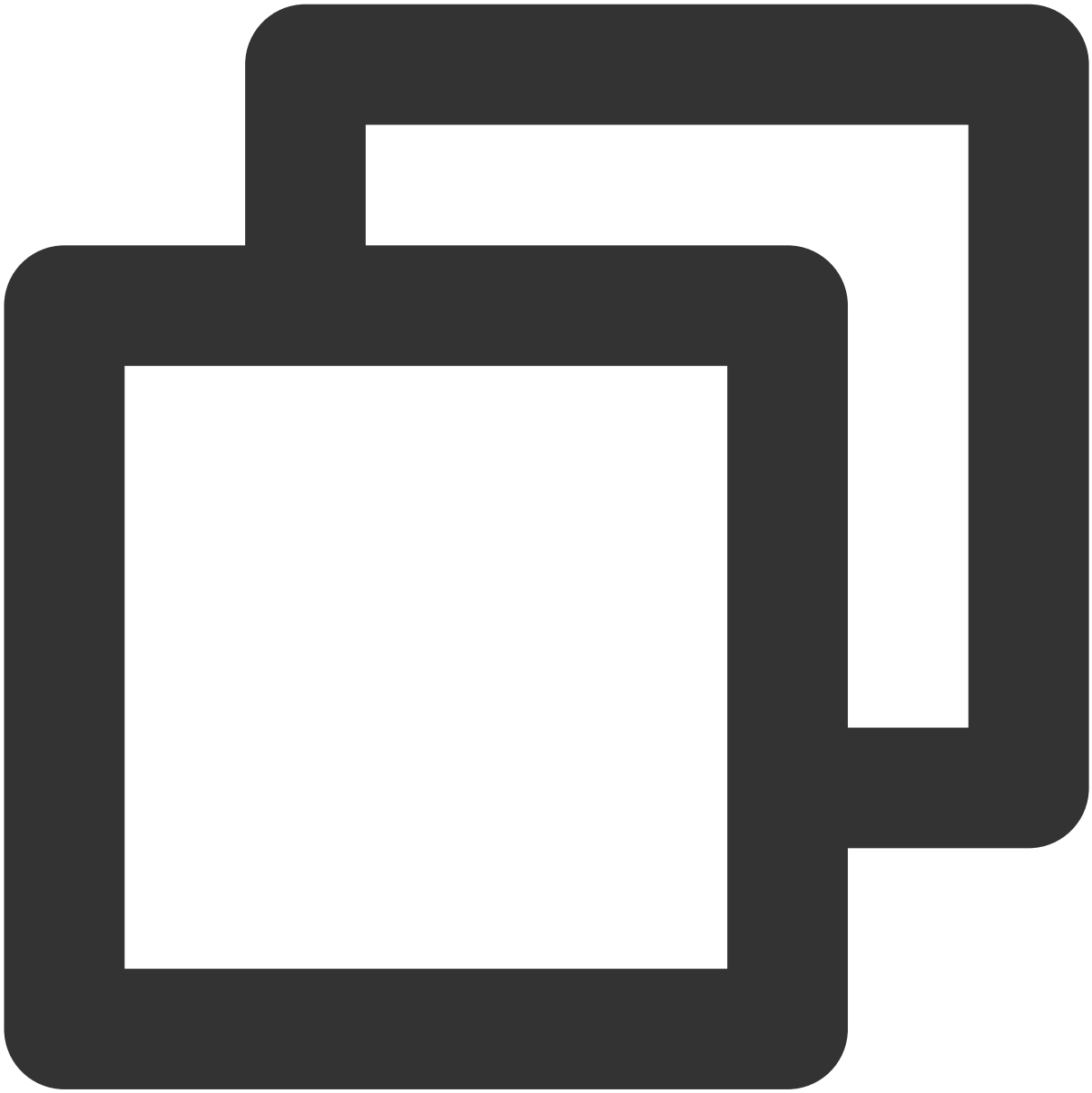
1. 复制该目录下的所有 .tf 文件到新的目录下，如 `eks-app-singapore`，断开原目录的 `tfstate` 引用：



```
$ mkdir ../eks-app-singapore
```

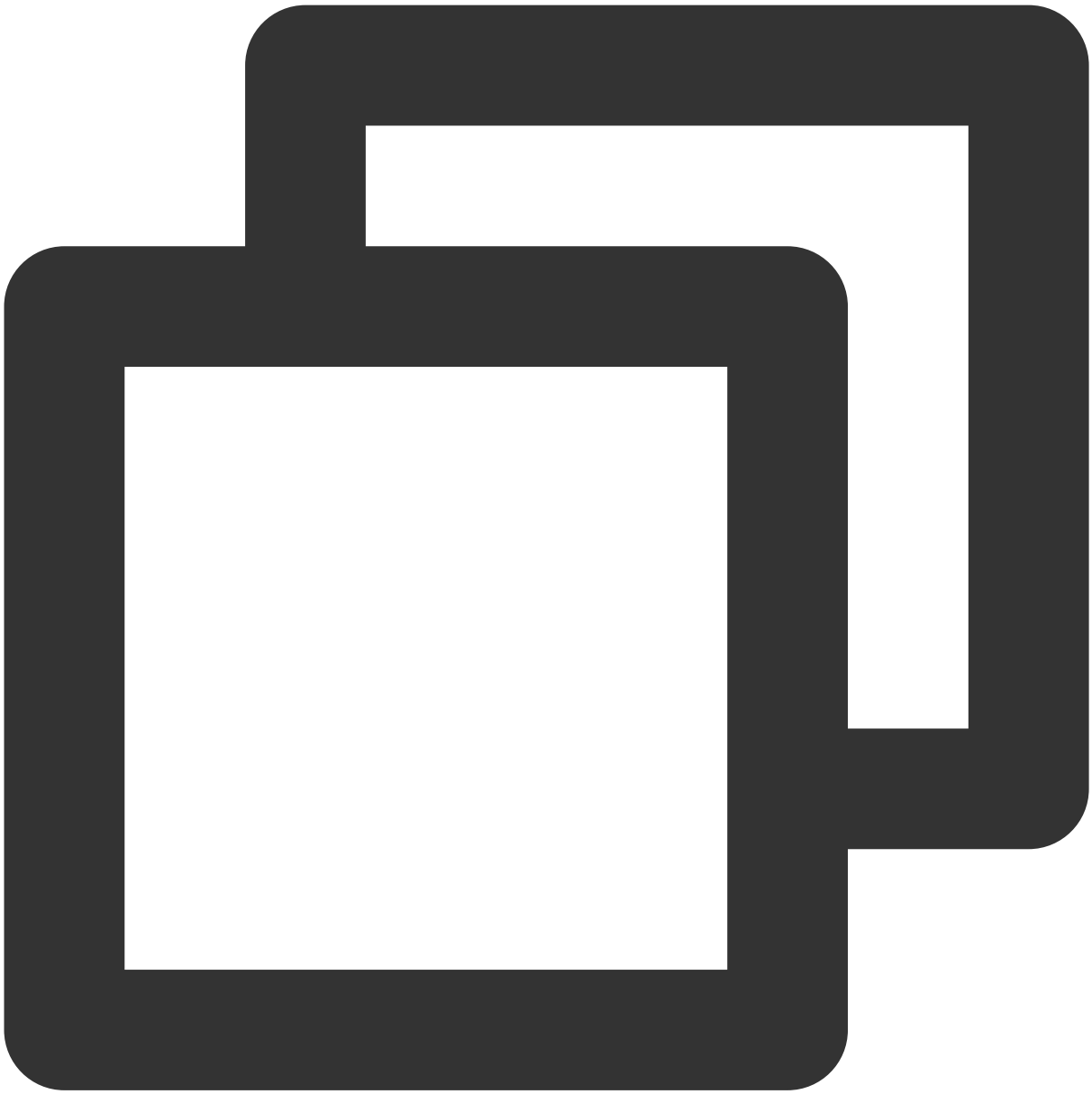
```
$ cp *.tf ../eks-app-singapore
$ cd ../eks-app-singapore
```

2. 修改 TencentCloud Provider 的地域。代码如下：



```
provider "tencentcloud" {
  # - replace
  # region = "ap-guangzhou"
  # + to
  region = "ap-singapore"
}
```

3. 在新目录 `eks-app-singapore` 下执行 `terraform init` 和 `terraform plan` 。由于没有 `tfstate` 文件， plan 提示即将创建新的资源：



```
Plan: 11 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
```

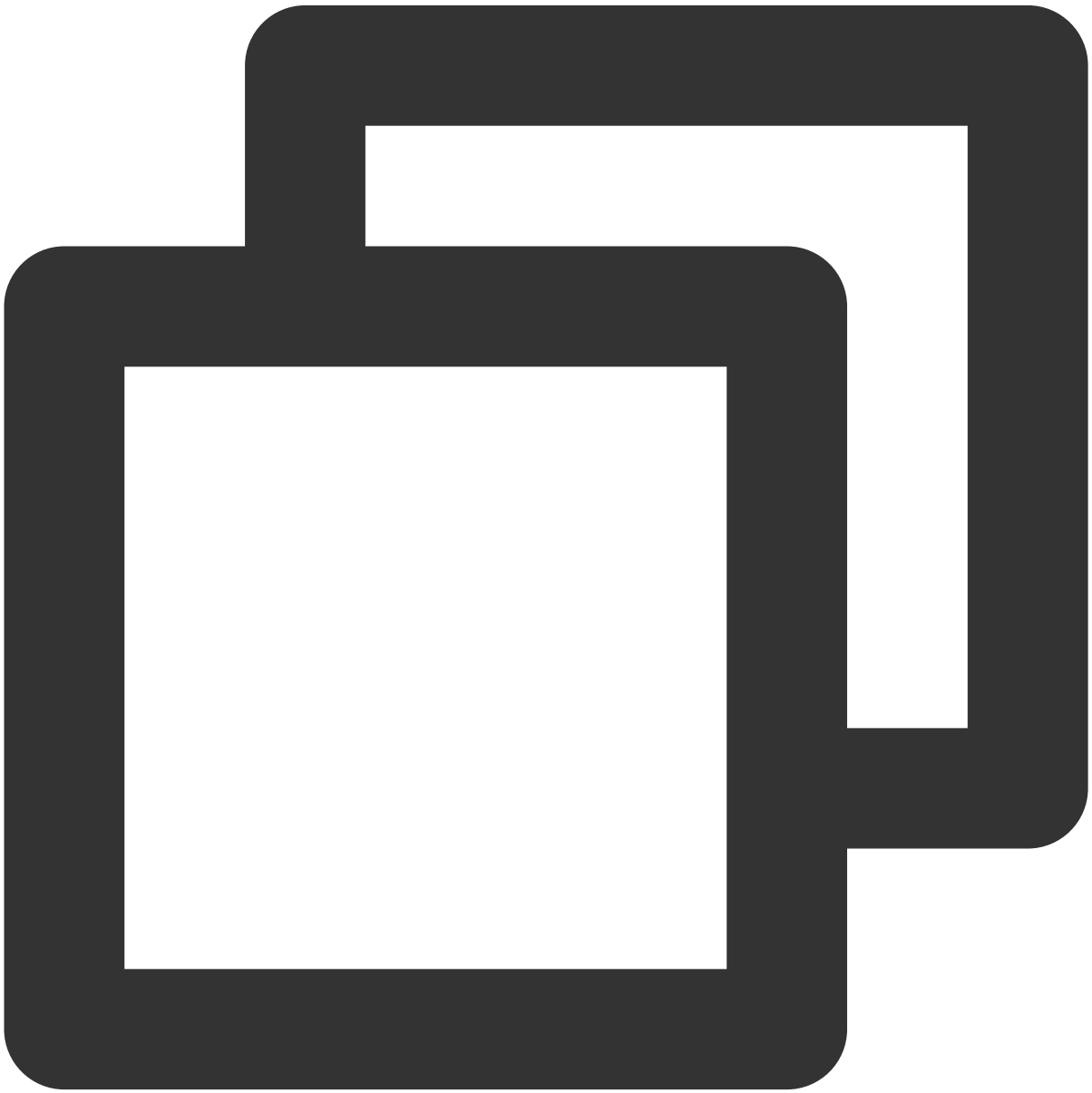
4. 确认无误后，执行 `terraform apply` 即可配置新目录下的云资源在新地域管理。

局限性

不同产品的业务形态和逻辑差异较大，导致跨地域复制也是一个比较繁琐的操作。主要限制如下：

实例规格和库存限制

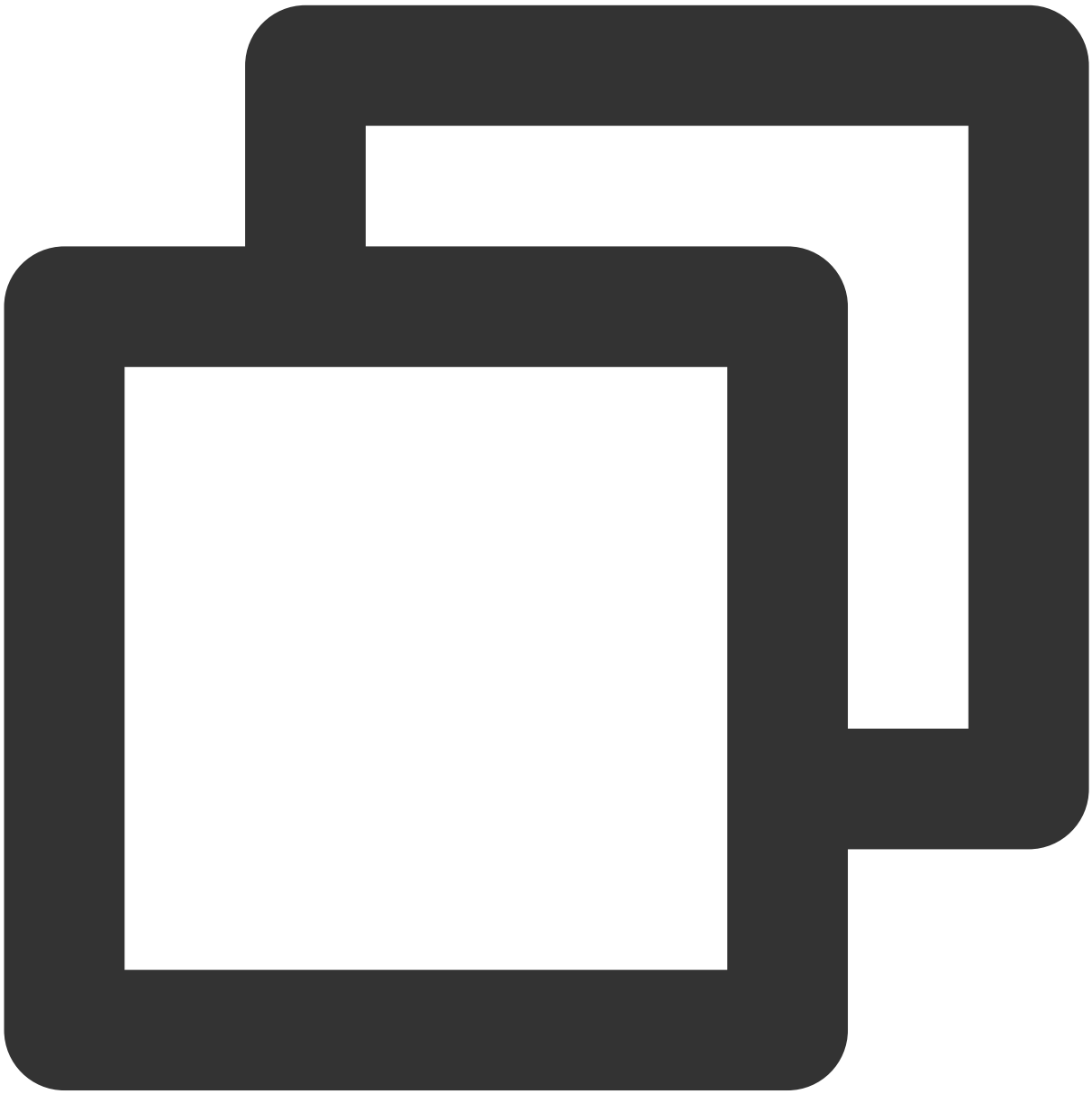
如云服务器、云硬盘、云数据库等实例的资源，各个可用区的实例规格和库存差异较大，很可能出现当前实例规格在其他区域售罄或者不可用的情况，建议使用动态的实例类型，即查询各个资源的 datasource 查询可用的实例规格而非硬编码在文件中，例如：
在上海四区购买 2 核 2G 的 CVM 实例。代码如下：



```
resource "tencentcloud_instance" "cvm" {
```

```
  name              = "my-instance"
  availability_zone = "ap-shanghai-4"
  image_id          = "local.cvm_img_id"
  instance_type     = "S5.MEDIUM2"
}
```

切换到广州地域，替换成 datasource 动态获取。代码如下：



```
provider "tencentcloud" {
   region = "ap-guangzhou"
}
```

```
# 查询广州地域 CVM 有哪些可用区
data "tencentcloud_availability_zones_by_product" "cvm" {
  product = "cvm"
}

# 查询以 Tencent 开头的 CVM 镜像
data "tencentcloud_images" "img" {
  image_name_regex = "Tencent"
}

# 查询指定可用区下 2 核 2G 有哪些实例类型
data "tencentcloud_instance_types" "types" {
  availability_zone = data.tencentcloud_availability_zones_by_product.cvm.zones.0.n
  cpu_core_count = 2
  memory_size = 2
}

locals {
   # 挑选第可用区列表的第一个结果
  cvm_zone = data.tencentcloud_availability_zones_by_product.cvm.zones.0.name
   # 挑选镜像列表的第一个结果
  cvm_img_id = data.tencentcloud_images.img.images.0.image_id
   # 挑选实例类型的第一个结果
  cvm_type = data.tencentcloud_instance_types.types.instance_types.0.instance_type
}

resource "tencentcloud_instance" "cvm" {
  name              = "my-instance"
  availability_zone = local.cvm_zone
  image_id          = local.cvm_img_id
  instance_type     = local.cvm_type
}
```

资源数量限制

有些资源在各个地域有数量限制，如 TKE 集群、私有网络、对象存储桶（总量）等，复制之前请确认好目标区域有充足的存量配额，如有配额提升需求，可以通过 提交工单 申请。

不需要复制的资源

有些资源本身没有地域属性，例如 CAM 用户/角色和策略、SSL 证书、SSH 密钥等。这些资源在进行整体复制操作时需要过滤掉，避免重复创建。