云资源自动化 for Terraform

最佳实践

腾讯云

文档目录

最佳实践
部署云原生服务

最近更新时间：2024-03-06 15:59:21

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

为了使用 Terraform 部署 TKE 资源，您需要满足以下条件：

- Terraform 版本大于等于1.2.9。
- 注册腾讯云账号，并完成 实名认证 。
- 获取凭证，在 API 密钥管理 页面中创建并复制 SecretId 和 SecretKey。
- 登录 容器服务控制台，按照界面提示为腾讯云容器服务授权。

完成授权后，请参考下述步骤，创建一个 TKE 资源，并基于该资源配置 Provider。

创建 TKE 相关资源

您可以通过以下两种方式创建 TKE 资源：

方式1：使用 TKE Module 创建实例

您可以直接使用 TKE Module(terraform-tencentcloud-tke) 来创建一个实例。然后，跳过本节内容，开始 配置 Kubernetes 。

方式2：逐步创建 TKE 集群

如果您想了解如何逐步创建一个 TKE 集群，可以按照以下步骤进行操作：

步骤1：创建目录

在您的本地终端创建任意空目录，例如 `tf-tke-example` 。按照本节内容，您将创建以下文件：

```
tf-tke-example
├── cluster.tf # 用于管理和配置集群资源
├── network.tf # 用于配置网络和安全组信息
└── provider.tf # 用于设置terraform provider和地域
```

步骤2：配置 Provider

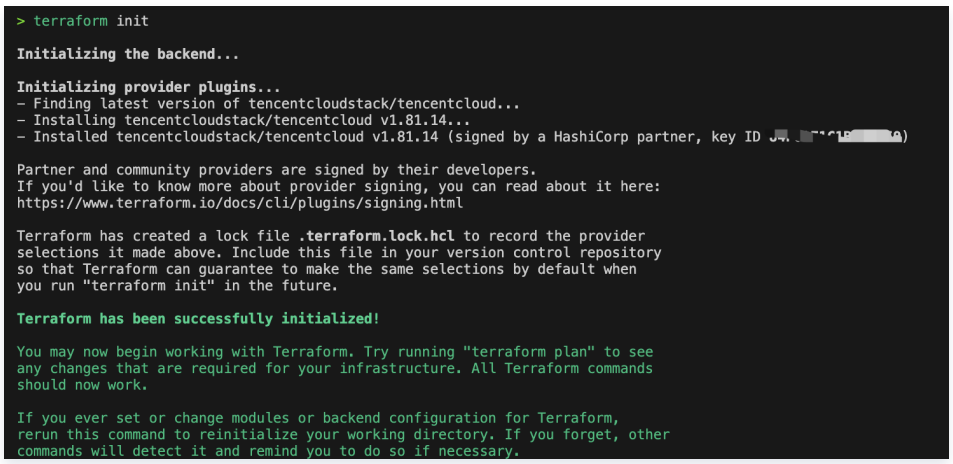
1. 在创建 TKE 资源之前，您需要在 provider.tf 文件中设置所需的 Terraform Provider 和资源所在地域。以在广州地域配置腾讯云 Provider 为例，请创建 provider.tf 文件，参考代码如下：

```
terraform {
  required_providers { #该block用于配置terraform provider，支持配置多个provider
    tencentcloud = {
      source = "tencentcloudstack/tencentcloud" #用于指定源位置，默认无需修改，会从terraform中下载
      #version = ">=1.81.14" # 用于指定版本，缺省时使用latest版本
    }
    #kubernetes = {
    #  source  = "hashicorp/kubernetes"
    #  version = ">= 2.0.0"
    #}
  }
}

provider "tencentcloud" {
  region = "ap-guangzhou" #在这里替换您的地域
```

```
}
```

> **注意：**
> 由于国内网络环境的一些限制，下载可能无法完成或者下载速度很慢，出现这种问题时，请使用 TencentCloud 镜像 。

2. 设置完 Terraform Provider 后，在命令行终端执行 `terraform init` 命令，如果显示以下信息，则说明 Terraform Provider 已经完成初始化。



步骤3：配置基础网络

1. 在创建 TKE 资源之前，您需要先配置私有网络 VPC、子网和安全组。请创建 network.tf 文件，并参考以下示例代码配置文件：

> **注意：**
> 请根据您的实际业务需求替换代码中相应的配置。

```
# Networks
variable "vpc_name" {
  default     = "example-vpc"
  description = "Specify the name of vpc."
}

variable "subnet_name" {
  default     = "example-subnet"
  description = "Specify the name of subnet."
}

variable "security_group_name" {
  default     = "example-security-group"
  description = "Specify the name of security group."
}

variable "network_cidr" {
  default     = "10.0.0.0/16"
  description = "Specify the cidr of vpc and subnet."
}

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"
  ]
```

```
  description = "Define the ingress rules of security group."
}

variable "available_zone" {
  default     = "ap-guangzhou-3" # 在这里指定您的可用区
  description = "Specify the available zone for your network."
}

variable "tags" {
  default = {
    terraform = "example"
  }
  description = "Specify the resource tags for your network."
}

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

# 基于上面的vpc资源，创建一个subnet资源
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
}

# 配置安全组资源
resource "tencentcloud_security_group" "sg" {
  name        = var.security_group_name
  description = "example security groups for kubernetes networks"
  tags        = var.tags
}

# 为安全组配置入站和出站规则
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"
  ]
}
```

2. 文件配置完成后，执行 `terraform plan` 命令，如果提示将会创建上述4个资源，则说明配置正确。

```
  # tencentcloud_vpc.vpc will be created
  + resource "tencentcloud_vpc" "vpc" {
      + assistant_cidrs        = (known after apply)
      + cidr_block             = "10.0.0.0/16"
      + create_time            = (known after apply)
      + default_route_table_id = (known after apply)
      + dns_servers            = (known after apply)
      + docker_assistant_cidrs = (known after apply)
      + id                     = (known after apply)
      + is_default             = (known after apply)
      + is_multicast           = true
      + name                   = "example-vpc"
      + tags                   = {
          + "terraform" = "example"
        }
    }

Plan: 4 to add, 0 to change, 0 to destroy.
```

3. 执行 terraform apply 命令，确认创建信息无误后，键入 **yes**，开始创建。创建需要一些时间，请您耐心等待。

```
Plan: 4 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_vpc.vpc: Creating...
tencentcloud_security_group.sg: Creating...
tencentcloud_security_group.sg: Creation complete after 1s [id=sg-nmdoyhhp]
tencentcloud_security_group_lite_rule.sg_rules: Creating...
tencentcloud_security_group_lite_rule.sg_rules: Creation complete after 1s [id=sg-nmdoyhhp]
tencentcloud_vpc.vpc: Creation complete after 9s [id=vpc-0ficxfu5]
tencentcloud_subnet.subnet: Creating...
tencentcloud_subnet.subnet: Creation complete after 1s [id=subnet-qwsfd0lu]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
```

4. 创建完成后，登录 私有网络控制台，分别查看已经创建的资源。如下图所示：
   - 私有网络：

   - 子网：

   - 安全组：

步骤4：配置集群

1. 在网络配置完成后，您可以创建并配置一个 TKE 集群。请创建 cluster.tf 文件，并参考以下示例代码配置文件：

> **注意：**
> 请根据您的实际业务需求替换代码中相应的配置。

```
# TKE
variable "cluster_name" {
  default     = "example-cluster"
  description = "Specify the name of your cluster."
}

variable "cluster_version" {
  default     = "1.22.5" # 在这里指定您的集群版本
  description = "Specify the name of your cluster."
}

variable "cluster_cidr" {
  default     = "172.16.0.0/22"
  description = "Specify the cidr of your cluster."
}

variable "cluster_os" {
  default     = "tlinux2.2(tkernel3)x86_64"
  description = "Specify the kind of OS for your cluster."
}

variable "cluster_public_access" {
  default     = false # 在这里指定是否打开外网访问，true为打开，false为关闭（这里暂时关闭以避免安全风险）
  description = "Specify the switch of public access of your cluster. false means closed public access, and true
means open it."
}

variable "cluster_private_access" {
  default     = true
  description = "Specify the switch of private access of your cluster. false means closed the private access, and
true means open it."
}

variable "worker_count" {
  default     = 1
  description = "Specify the initial count of the worker nodes."
}

variable "worker_instance_type" {
  default     = "S5.MEDIUM2" # 在这里指定您的worker版本
  description = "Specify the type of the worker nodes."
}

variable "cluster_available_zone" {
  default     = "ap-guangzhou-3" # 在这里指定您的可用区
  description = "Specify the available zone for your cluster."
}

variable "cluster_tags" {
  default = {
    terraform = "example"
  }
  description = "Specify the resource tags for your cluster."
}

# 定义随机密码
resource "random_password" "worker_pwd" {
  length          = 12
```

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

# 基于以上variables, 定义一个TKE集群
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_security_group.sg.id : null
  cluster_intranet            = var.cluster_private_access
  cluster_intranet_subnet_id  = var.cluster_private_access ? tencentcloud_subnet.subnet.id : null
  vpc_id                      = tencentcloud_vpc.vpc.id

  # 工作节点配置
  worker_config {
    availability_zone  = var.cluster_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.cluster_tags
}
```

2. 文件配置完成后，由于引入一个新的 Provider（random），您需要在命令行终端执行 `terraform init -upgrade` 命令，更新 Terraform Provider。

```
> terraform init -upgrade

Initializing the backend...

Initializing provider plugins...
- Finding latest version of tencentcloudstack/tencentcloud...
- Finding latest version of hashicorp/random...
- Installing hashicorp/random v3.5.1...
- Installed hashicorp/random v3.5.1 (signed by HashiCorp)
- Using previously-installed tencentcloudstack/tencentcloud v1.81.14

Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

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. 与配置网络类似，执行 `terraform plan` 命令后，如果提示将会创建2个资源，则说明配置正确。
4. 执行 `terraform apply` 命令，确认创建信息无误后，键入**yes**，开始创建。创建需要一些时间，请您耐心等待。

```
tencentcloud_kubernetes_cluster.cluster: Creating...
tencentcloud_kubernetes_cluster.cluster: Still creating... [10s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [20s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [30s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [40s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [50s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [1m0s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [1m10s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [1m20s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [1m30s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [1m40s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [1m50s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [2m0s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [2m10s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [2m20s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [2m30s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [2m40s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [2m50s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [3m0s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [3m10s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [3m20s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [3m30s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [3m40s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [3m50s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [4m0s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [4m10s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [4m20s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [4m30s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [4m40s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [4m50s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [5m0s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [5m10s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [5m20s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [5m30s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [5m40s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [5m50s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [6m0s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [6m10s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [6m20s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [6m30s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [6m40s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [6m50s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [7m0s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [7m10s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [7m20s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [7m30s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [7m40s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [7m50s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [8m0s elapsed]
tencentcloud_kubernetes_cluster.cluster: Still creating... [8m10s elapsed]
tencentcloud_kubernetes_cluster.cluster: Creation complete after 8m16s [id=cls-arsp848s]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

5. 登录 容器服务控制台，可以看到您已经创建的资源。如下图所示：
   - 在集群列表中，可查看已创建的集群：

   - 单击集群名称，在**节点管理 > Worker 节点**中，选择**节点**页面，可查看已经创建的节点：

步骤5：（可选）配置 CAM 角色

1. 您的账号需要创建 **TKE_QCSRole** 角色并授予其预设策略 `TF_QcloudAccessForTKERole`，`TF_QcloudAccessForTKERoleInOpsManagement` ，用于访问其它的云资源。（如果您已经拥有该角色权限，请跳过本小节）。

说明：
如果您已经在控制台完成授权（如下图所示），则不需要创建 cam.tf 文件。

2. 创建 cam.tf 文件，配置 CAM 角色并关联策略。代码如下：

```
# 配置TKE_QCSRole角色，该角色用于TKE访问您的相关云服务资源
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."
}

# 设置角色ops_mgr策略：该权限用于腾讯云容器服务（TKE）对云资源的访问，包含日志服务(CLS)的访问
data "tencentcloud_cam_policies" "ops_mgr" {
  name = "TF_QcloudAccessForTKERoleInOpsManagement"
}

# 设置角色qca策略：该权限用于腾讯云容器服务（TKE）对云资源的访问
data "tencentcloud_cam_policies" "qca" {
  name = "TF_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
}

# 角色与策略ops_mgr关联
resource "tencentcloud_cam_role_policy_attachment" "QCS_OpsMgr" {
```

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

# 角色与策略qca关联
resource "tencentcloud_cam_role_policy_attachment" "QCS_QCA" {
  role_id   = lookup(tencentcloud_cam_role.TKE_QCSRole, "id")
  policy_id = local.qca_policy_id
}
```

3. 与配置网络类似，执行 `terraform plan` 命令后，如果提示将会创建3个资源，则说明配置正确。
4. 执行 `terraform apply` 命令，确认创建信息无误后，键入**yes**，开始创建。创建需要一些时间，请您耐心等待。

```
Plan: 3 to add, 0 to change, 0 to destroy.
tencentcloud_cam_role.TKE_QCSRole: Creating...
tencentcloud_cam_role.TKE_QCSRole: Still creating... [10s elapsed]
tencentcloud_cam_role.TKE_QCSRole: Creation complete after 11s [id=4611686018431022245]
tencentcloud_cam_role_policy_attachment.QCS_OpsMgr: Creating...
tencentcloud_cam_role_policy_attachment.QCS_QCA: Creating...
tencentcloud_cam_role_policy_attachment.QCS_QCA: Still creating... [10s elapsed]
tencentcloud_cam_role_policy_attachment.QCS_OpsMgr: Still creating... [10s elapsed]
tencentcloud_cam_role_policy_attachment.QCS_QCA: Creation complete after 11s [id=4611686018431022245#9087631]
tencentcloud_cam_role_policy_attachment.QCS_OpsMgr: Creation complete after 11s [id=4611686018431022245#34488263]

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

5. 登录访问管理控制台，选择 角色，查看您已经创建的角色：

查看该角色对应的权限策略：

步骤6：（可选）封装为 Module

完成上述步骤后，您已经创建了一个基础可用的 TKE 资源实例。为了更好地组织代码并减少关注内部实现，建议将上述步骤中的所有 .tf 文件组织成一个 Module 进行使用。您也可以参考现有的 TKE Module(terraform-tencentcloud-tke) 。

配置 Kubernetes

在完成 创建 TKE 相关资源 后，您已经获得了一个 TKE 集群，从而可以获取它的外网访问地址，CA 证书和用户凭证。接下来，您可以基于这个 TKE 集群或者使用 TKE Module(terraform-tencentcloud-tke) 来配置 Kubernetes，部署一个简单的 Nginx 应用。在本示例中，您将创建和修改以下文件：

```
.
├── main.tf # 用于配置module
```

```
├── clb.tf  # 用于配置负载均衡
├── kubernetes.tf # 用于配置一套kubernetes资源
├── network.tf    # 用于配置网络和安全组信息
└── provider.tf   # 用于定义tencentcloud和kubernetes两个provider
```

现在，我们以 方式1：使用 TKE Module 创建实例 为例，创建一个 TKE 集群并进行相关配置。

配置 Kubernetes Provider

1. 在您的本地终端，新建一个新的文件夹，如 `tf-tke-k8s-example` 。

> ⚠ **注意：**
> 如果您是基于上一小节 方式2：逐步创建 TKE 集群 创建的集群，那么直接修改 `tf-tke-example` 中的 provider.tf 即可。

2. 您需要执行以下操作配置 Provider：在该目录中，新建一个 provider.tf 文件，然后添加配置，具体包括：
   - 创建 provider.tf 文件，配置 `kubernetes` 和 `tencentcloud` Provider。
   - 创建 main.tf 文件，配置 `tencentcloud_tke` Module。

   provider.tf 文件示例代码如下：

```
terraform {
  required_providers {
    kubernetes = { # 添加一个kubernetes provider
      source  = "hashicorp/kubernetes"
    }
    tencentcloud = {
      source  = "tencentcloudstack/tencentcloud"
    }
  }
}

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

# 对kubernetes provider进行配置，关联已有的TKE集群
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)
}
```

   main.tf 文件示例代码如下：

```
# 引用module
module "tencentcloud_tke" {
  source         = "github.com/terraform-tencentcloud-modules/terraform-tencentcloud-tke"
  #available_zone = "ap-guangzhou-3" # Available zone must belongs to the region.
}
```

3. 在命令行终端执行 `terraform init` 命令，Terraform 会下载 TKE Module，以及 TencentCloud、Kubernetes 和 Random 三个 Provider。

```
> terraform init
Initializing modules...
Downloading git::https://github.com/terraform-tencentcloud-modules/terraform-tencentcloud-tke.git for tencentcloud_tke...
- tencentcloud_tke in .terraform/modules/tencentcloud_tke

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/kubernetes...
- Finding tencentcloudstack/tencentcloud versions matching ">= 1.79.2"...
- Finding latest version of hashicorp/random...
- Installing hashicorp/kubernetes v2.22.0...
- Installed hashicorp/kubernetes v2.22.0 (signed by HashiCorp)
- Installing tencentcloudstack/tencentcloud v1.81.14...
- Installed tencentcloudstack/tencentcloud v1.81.14 (signed by a HashiCorp partner, key ID 84F69E1C1BECF459)
- Installing hashicorp/random v3.5.1...
- Installed hashicorp/random v3.5.1 (signed by HashiCorp)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
```

配置 TKE Module

您需要执行以下操作配置 TKE Module：

- 创建 network.tf 文件，配置网络和安全组。
- 修改 main.tf 文件，完善 Module 中的参数项。

配置安全组外网访问

请创建 network.tf 文件。以下是一个 network.tf 的示例代码：

```
variable "accept_ip" {
  type    = string
  default = "x.x.x.x" #  设置为您的IP
}

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

variable "region" {
  default = "ap-guangzhou"
}

resource "tencentcloud_security_group" "this" {
  name = "tke-security-group"
}

resource "tencentcloud_security_group_lite_rule" "this" {
  security_group_id = tencentcloud_security_group.this.id

  ingress = [
    "ACCEPT#10.0.0.0/16#ALL#ALL",
    "ACCEPT#172.16.0.0/22#ALL#ALL",
```

```
    # "ACCEPT#${var.accept_ip}#ALL#ALL", # 添加您的IP
    "DROP#0.0.0.0/0#ALL#ALL",
  ]

  egress = [
    "ACCEPT#172.16.0.0/22#ALL#ALL",
  ]
}

resource "tencentcloud_vpc" "this" {
  cidr_block = "10.0.0.0/16"
  name       = "tke-test"
}

resource "tencentcloud_subnet" "intranet" {
  cidr_block        = "10.0.1.0/24"
  name              = "tke-subnet"
  availability_zone = var.available_zone
  vpc_id            = tencentcloud_vpc.this.id
}
```

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。如果您需要测试集群的外网访问，您需要将自己的 IP 添加到安全组的规则中。

配置 TKE 资源

我们将通过 节点池 来创建节点，并使用 `tencentcloud_kubernetes_cluster_endpoint` 资源来开启网络访问。
请修改 main.tf 文件中的 `tencentcloud_tke` module 块，以下是一个可用的参数项示例：

> **注意：**
> 请根据您的实际业务需求替换代码中相应的配置。

```
module "tencentcloud_tke" {
  source                = "github.com/terraform-tencentcloud-modules/terraform-tencentcloud-tke"
  available_zone        = var.available_zone # Available zone must belongs to the region.
  create_cam_strategy   = false
  enhanced_monitor_service = true


  create_endpoint_with_cluster    = false # 使用 endpoint 资源开启网络访问
  create_workers_with_cluster     = false # 使用 节点池 资源创建节点
  cluster_public_access           = true  # 开启公网
  cluster_private_access          = true  # 开启内网
  cluster_security_group_id       = tencentcloud_security_group.this.id
  node_security_group_id          = tencentcloud_security_group.this.id
  cluster_private_access_subnet_id = tencentcloud_subnet.intranet.id
  vpc_id                          = tencentcloud_vpc.this.id
  intranet_subnet_id              = tencentcloud_subnet.intranet.id


  enable_cluster_audit_log = true
  worker_bandwidth_out     = 100


  tags = {
    module = "tke"
  }
```

```
# 配置标准节点池
self_managed_node_groups = {
 test = {
  max_size              = 4
  min_size              = 1
  subnet_ids            = [tencentcloud_subnet.intranet.id]
  retry_policy          = "INCREMENTAL_INTERVALS"
  desired_capacity      = 2
  enable_auto_scale     = true
  multi_zone_subnet_policy = "EQUALITY"


  auto_scaling_config = [{
   instance_type           = "S5.MEDIUM2"
   system_disk_type        = "CLOUD_PREMIUM"
   system_disk_size        = 50
   orderly_security_group_ids = [tencentcloud_security_group.this.id]


   data_disk = [{
    disk_type = "CLOUD_PREMIUM"
    disk_size = 50
   }]


   internet_charge_type      = "TRAFFIC_POSTPAID_BY_HOUR"
   internet_max_bandwidth_out = 10
   public_ip_assigned        = true
   enhanced_security_service  = false
   enhanced_monitor_service   = false
   host_name               = "12.123.0.0"
   host_name_style         = "ORIGINAL"


  }]


  labels = {
   "test1" = "test1",
   "test2" = "test2",
  }


  taints = [{
   key    = "test_taint"
   value  = "taint_value"
   effect = "PreferNoSchedule"
   },
   {
    key    = "test_taint2"
    value  = "taint_value2"
    effect = "PreferNoSchedule"
  }]


  node_config = [{
   extra_args = ["root-dir=/var/lib/kubelet"]
  }]
```

```
    }
  }
}
```

配置 Kubernetes resources

在 Terraform 中，您可以使用 HCL 替代原来的 yaml 声明 Namespace、Deployment 和 Service。通过使用 Kubernetes Provider 的 `kubernetes_namespace` 、 `kubernetes_deployment` 、 `kubernetes_service` 资源，可以完成上述配置。
请创建 kubernetes.tf 文件，并参考以下示例代码：

```
# 配置k8 namespace
resource "kubernetes_namespace" "test" {
  metadata {
    name = "nginx"
  }
}

# 定义了一个k8s Deployment资源，把两个Pod部署到Kubernetes集群中
resource "kubernetes_deployment" "test" {
  metadata {
    name      = "nginx"
    namespace = kubernetes_namespace.test.metadata.0.name
  }
  spec {
    replicas = 2
    selector {
      match_labels = {
        app = "MyTestApp"
      }
    }
    # 设置nginx的container和对应镜像
    template {
      metadata {
        labels = {
          app = "MyTestApp"
        }
      }
      # 使用Nginx镜像
      spec {
        container {
          image = "nginx"
          name  = "nginx-container"
          port {
            container_port = 80
          }
        }
      }
    }
  }
}

# 创建名为nginx的k8 service
resource "kubernetes_service" "test" {
  metadata {
    name      = "nginx"
    namespace = kubernetes_namespace.test.metadata.0.name
  }
  # 将NodePort类型的服务映射到nginx的Pod上
  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

您需要配置 Ingress 关联负载均衡 CLB，以便通过 CLB 提供的访问地址访问 Nginx 服务，从而实现公网访问。

1. 创建一个 CLB 实例。请创建 clb.tf 文件，并参考以下示例代码：

```
locals {
  lb_vpc = tencentcloud_vpc.this.id
  lb_sg  = tencentcloud_security_group.this.id
}

# 配置clb，而后通过clb提供的访问地址访问nginx服务
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
}
```

2. 在 kubernetes.tf 文件中配置 Ingress，并指定刚才创建的 CLB ID。为此，您需要定义一个 Kubernetes Ingress 资源，将外部流量路由到 Nginx 服务上。示例代码如下：

```
# 定义一个k8s Ingress资源，将外部流量路由到名为"test-ingress"的Ingress中的Nginx服务上。
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 = "/"
      }
    }
  }
 }
}
```

3. 为了方便观察调试结果，我们可以获取 CLB 的 IP 地址并输出。请创建一个 output 变量，示例代码如下：

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

（可选）配置 PVC

您可以通过指定 PVC，为 TKE 集群挂载存储资源。您需要执行以下操作：

- 创建 cbs.tf，配置磁盘资源。
- 修改 kubernetes.tf 文件，添加 PV 与对应 PVC 配置。

请创建 cbs.tf 文件。以下是一个 cbs.tf 的示例代码：

```
# 配置CBS, 而后用于k8s PV存储资源
resource "tencentcloud_cbs_storage" "stroage" {
  storage_name      = "example-cbs"
  storage_type      = "CLOUD_SSD"
  storage_size      = 100
  availability_zone = var.available_zone
  project_id        = 0
  encrypt           = false

}
```

修改 kubernetes.tf 文件。以下是一个 kubernetes.tf 的示例代码：

```
# 定义一个PVC资源
resource "kubernetes_persistent_volume_claim" "test" {
  metadata {
    name = "example-pv-claim"
    namespace = kubernetes_namespace.test.metadata.0.name
  }
  spec {
    storage_class_name = "my-storage"
    access_modes = ["ReadWriteMany"]
    resources {
      requests = {
        storage = "5Gi"
      }
    }
    volume_name = "${kubernetes_persistent_volume.test.metadata.0.name}"
  }
```

```
}


# 定义一个PV资源
resource "kubernetes_persistent_volume" "test" {
 metadata {
  name = "example-pv"
 }
 spec {
  capacity = {
   storage = "10Gi"
  }
  storage_class_name = "my-storage"
  access_modes = ["ReadWriteMany"]
  persistent_volume_source {
   # 配置cbs
   csi {
    driver = "com.tencent.cloud.csi.cbs"
    volume_handle = tencentcloud_cbs_storage.stroage.id
    fs_type = "ext4"
   }
  }
 }
}
```

执行创建

1. 在编写完所有的 .tf 文件后，按照以下顺序执行命令：

   初始化：

   ```
   $ terraform init
   ```

   预览并确认将创建的资源：

   ```
   $ terraform plan
   ```

   执行资源创建：

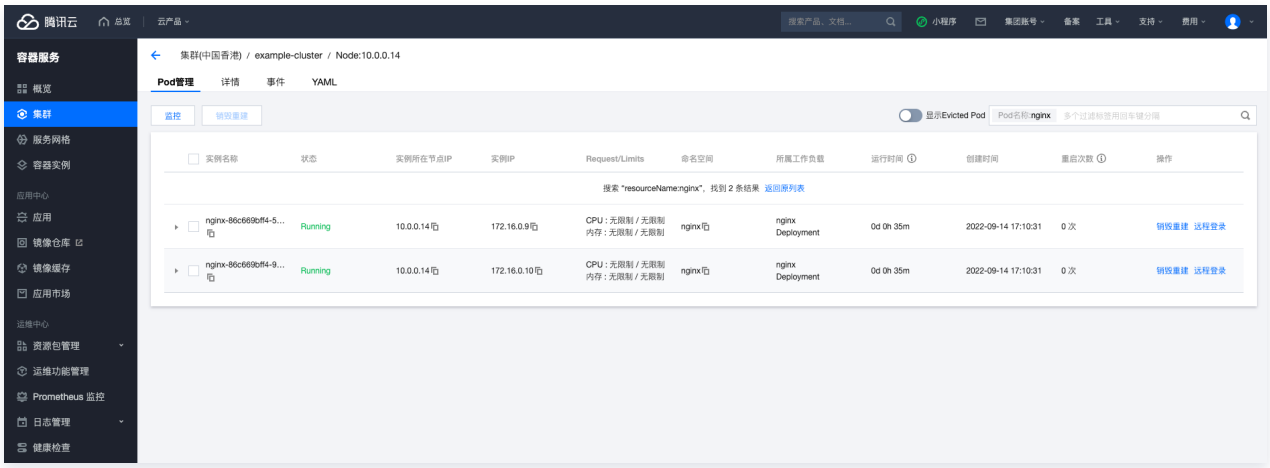
   ```
   $ terraform apply
   ```

2. 创建成功后，控制台将输出上文中定义的 output 信息。

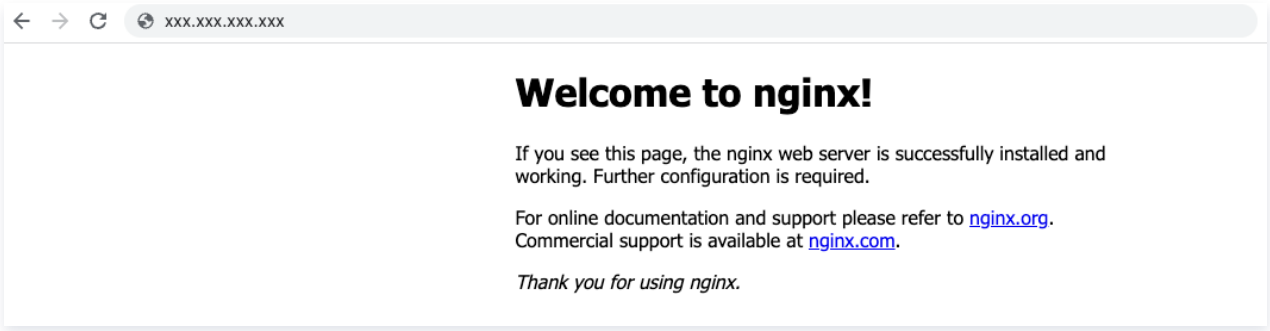
   ```
   Apply complete! Resources: 16 added, 0 changed, 0 destroyed.

   Outputs:

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

验证部署

1. 登录 容器服务控制台，选择 example-cluster 集群，进入节点页面，在 Pod 管理页，可以查看 Nginx 相关的 Pod 是否已经处于 Running 状态。如果相关 Pod 已经处于 Running 状态，则说明应用已经成功部署。



2. 访问 load_balancer_ip 显示的地址，如果页面显示 Welcome To Nginx，则说明应用已经成功部署并可以通过 CLB 访问。

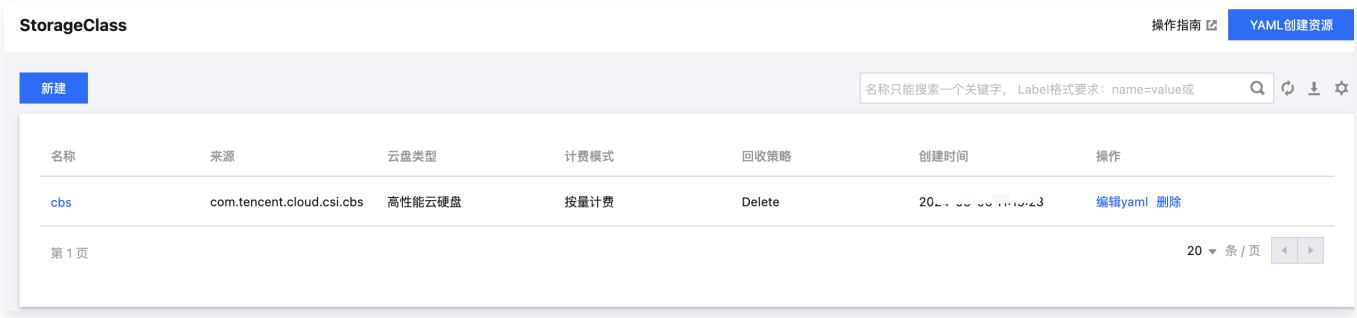


3. 如果您执行过（可选）配置 PVC，可以在 example-cluster 集群的存储管理页中，查看刚才配置的 PV、PVC 和 CBS 信息。

PV 列表：

PVC 列表：

StorageClass 列表：



资源跨地域复制

最近更新时间：2023-09-21 20:57:32

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。

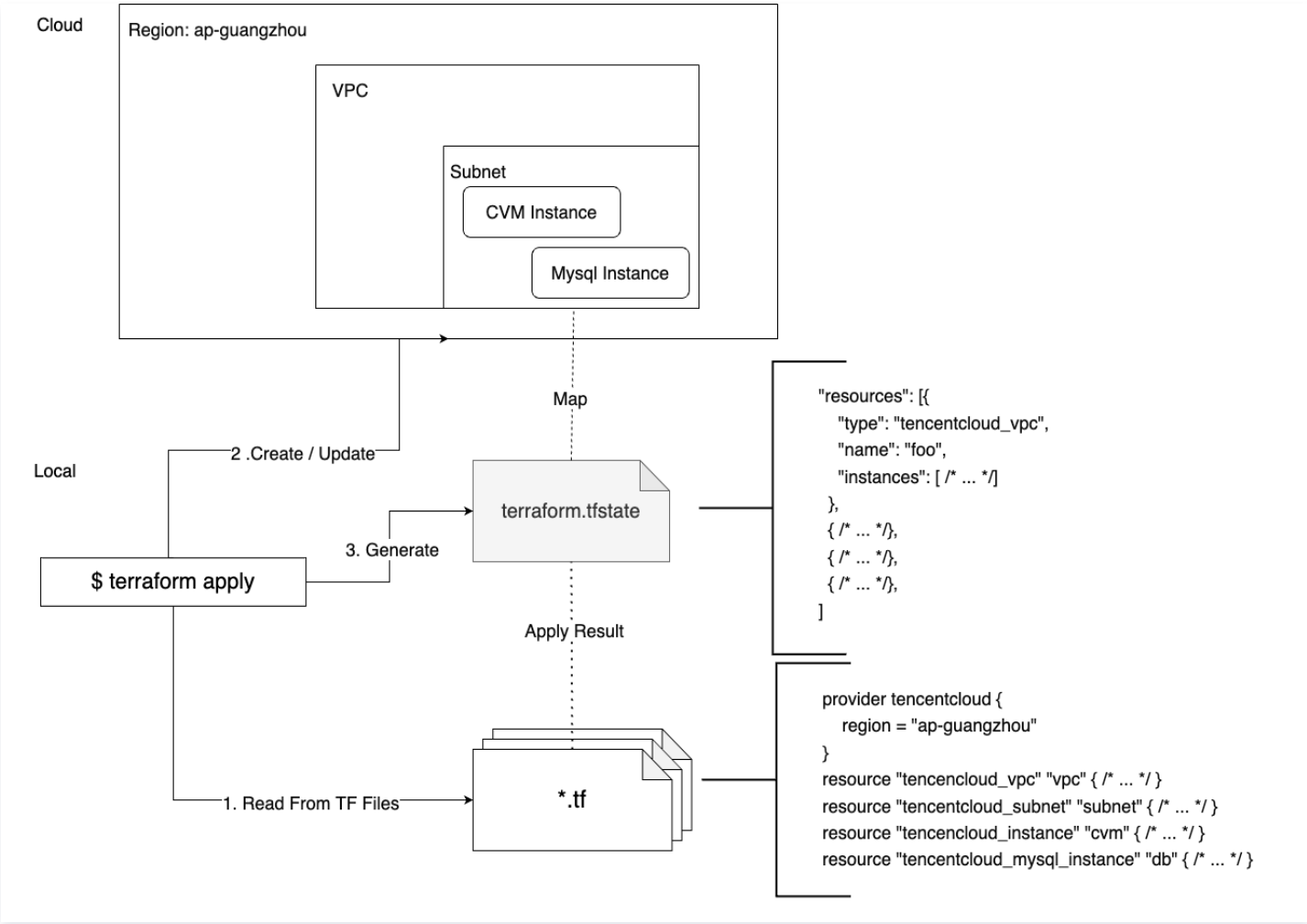
跨地域复制

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["certificate-authority-data"])
  client_key             = base64decode(local.kubeconfig.users[0].user["client-key-data"])
  client_certificate     = base64decode(local.kubeconfig.users[0].user["client-certificate-data"])
}

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.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"
    }
    #    selfLink = "/apis/networking.k8s.io/v1/namespaces/nginx/ingresses/test-ingress"
  }
  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 guarantee to take exactly these actions if
you run "terraform apply" now.
```

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.name
  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。