弹性微服务

Terraform 管理

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

Terraform 管理
Terraform 概述

最近更新时间：2022-11-18 16:34:37

Terraform 是一个 IT 基础架构自动化编排工具，它的口号是“Write, Plan, and Create Infrastructure as Code”，是一个“基础设施即代码”工具，通过 Terraform 您可以创建、更新和版本控制的 TEM 中的资源。了解更多关于 Terraform，请参见 Terraform 指南 。

通过 Terraform 管理弹性微服务资源

资源类型	说明
tencentcloud_tem_environment	提供 TEM 环境资源。
tencentcloud_tem_application	提供 TEM 应用资源。
tencentcloud_tem_workload	提供 TEM 应用工作负载资源。
tencentcloud_tem_gateway	提供 TEM 网关资源。
tencentcloud_tem_scale_rule	提供 TEM 弹性伸缩资源。
tencentcloud_tem_log_config	提供 TEM 日志采集规则资源。
tencentcloud_tem_app_config	提供 TEM 配置文件资源。

环境资源管理

最近更新时间：2022-09-09 14:34:54

操作场景

在弹性微服务中，环境是一组计算、网络、存储等资源的集合。TEM 提供多环境管理的功能，您可根据自身业务需要，创建开发、测试、预发、生产等多个环境，分别部署应用，达成环境隔离的目的。不同环境中的应用彼此隔离。 本文介绍如何使用 Terraform 创建、编辑和删除 TEM 环境资源。

前提条件

请参见 Terraform 快速开始 安装 Terraform 并配置腾讯云鉴权。

TEM 环境资源

Terraform 的 tencentcloud_tem_environment 资源提供了以下参数：

- 必填：environment_name 环境名称。
- 必填：vpc 指定私有网络。
- 必填：subnet_ids 指定子网，
- 可选：description 环境描述信息。
  详细信息请参见 terraform_tem_environment 。

创建环境

此处以在广州地域下创建一个命名为 `tf-test` 的环境为示例：

1. 创建 `provider.tf` 文件，指定 `provider` 配置信息。文件内容如下：

```
terraform {
  required_providers {
    tencentcloud = {
      source = "tencentcloudstack/tencentcloud"
      # 通过version指定版本
      # version = ">=1.60.18"
    }
  }
}

provider "tencentcloud" {
  region = "ap-guangzhou"
  # secret_id = "my-secret-id"
  # secret_key = "my-secret-key"
}
```

- region：填写地域 ID。
- 获取凭证：在 API 密钥管理 面中创建并复制 SecretId 和 SecretKey。

2. 创建 `main.tf` 文件，配置腾讯云 `Provider` 并创建弹性微服务环境。文件内容如下：

```
resource "tencentcloud_tem_environment" "environment" {
  environment_name = "test_tf"
  description      = "demo for terraform"
  vpc              = "my-vpc-id"
  subnet_ids       = ["my-subnet-1-id", "my-subnet-2-id"]
}
```

3. 执行以下命令，初始化工作目录并下载插件。

```
terraform init
```

返回信息如下：

```
Initializing the backend...

Initializing provider plugins...
- Finding latest version of tencentcloudstack/tencentcloud...
- Installing tencentcloudstack/tencentcloud v1.77.4...
- Installed tencentcloudstack/tencentcloud v1.77.4 (signed by a HashiCorp partner,
key ID 84F69E1C1BECF459)

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.
```

4. 执行以下命令，创建资源。

```
terraform apply
```

返回信息如下：

```
Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # tencentcloud_tem_environment.environment will be created
  + resource "tencentcloud_tem_environment" "environment" {
      + description      = "demo for terraform"
      + environment_name = "test_tf"
      + id               = (known after apply)
      + subnet_ids       = [
          + "subnet-5ob92rbm",
          + "subnet-dwpro9ni",
        ]
      + vpc              = "vpc-dd5m94px"
    }

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_tem_environment.environment: Creating...
tencentcloud_tem_environment.environment: Creation complete after 2m37s
[id=en-853agv3j]
```

5. 执行完毕后，您可以在 弹性微服务控制台 查看创建的资源。

编辑环境

以编辑上面所创建的环境名称为示例。

1. 在 `main.tf` 文件中编辑环境资源信息：

```
resource "tencentcloud_tem_environment" "environment" {
  environment_name = "test_tf_renamed"
  description      = "demo for terraform"
  vpc              = "my-vpc-id"
  subnet_ids       = ["my-subnet-1-id", "my-subnet-2-id"]
}
```

2. 执行 `terraform plan` 命令更新计划，应返回如下信息：

```
tencentcloud_tem_environment.environment: Refreshing state... [id=en-853agv3j]

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # tencentcloud_tem_environment.environment will be updated in-place
  ~ resource "tencentcloud_tem_environment" "environment" {
      ~ environment_name = "test_tf" -> "test_tf_renamed"
        id               = "my-env-id"
        # (3 unchanged attributes hidden)
    }

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

3. 执行 `terraform apply` 执行更新了的计划，返回信息如下：

```
tencentcloud_tem_environment.environment: Refreshing state... [id=en-853agv3j]

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  ~ update in-place

Terraform will perform the following actions:

  # tencentcloud_tem_environment.environment will be updated in-place
  ~ resource "tencentcloud_tem_environment" "environment" {
      ~ environment_name = "test_tf" -> "test_tf_renamed"
        id               = "en-853agv3j"
        # (3 unchanged attributes hidden)
    }
```

```
Plan: 0 to add, 1 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_tem_environment.environment: Modifying... [id=en-853agv3j]
tencentcloud_tem_environment.environment: Modifications complete after 1s
[id=en-853agv3j]

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

销毁环境

以销毁上面所创建的环境资源为例。

1. 执行以下命令。

```
terraform destroy
```

2. 销毁成功后返回以下信息。

```
Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # tencentcloud_tem_environment.environment will be destroyed
  - resource "tencentcloud_tem_environment" "environment" {
      - description      = "demo for terraform" -> null
      - environment_name = "test_tf_renamed" -> null
      - id               = "en-853agv3j" -> null
      - subnet_ids       = [
          - "subnet-5ob92rbm",
          - "subnet-dwpro9ni",
        ] -> null
      - vpc              = "vpc-dd5m94px" -> 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_tem_environment.environment: Destroying... [id=en-853agv3j]
tencentcloud_tem_environment.environment: Destruction complete after 2s

Destroy complete! Resources: 1 destroyed.
```

应用资源管理

最近更新时间：2024-02-01 10:08:21

操作场景

在弹性微服务中，为了帮助您聚焦设计与部署业务应用程序，省去对集群与服务器的考虑，抽象出了应用的概念。本文介绍如何使用 Terraform 创建应用、部署和删除应用。

前提条件

请参见 Terraform 快速开始 安装 Terraform 并配置腾讯云鉴权。

TEM应用资源

Terraform的tencentcloud_tem_application 资源提供了以下参数：

- 必填：application_name 应用名称。
- 必填：coding_language 应用的开发语言。
- 可选：repo_type 镜像仓库类型：“0”为个人版镜像仓库，“1”为企业版镜像仓库，“2”为公共镜像仓库，“4”为演示镜像。
- 可选：repo_name 镜像仓库名称。
- 可选：repo_server 镜像仓库地址。
- 可选：use_default_image_service 是否新建镜像仓库。
  详细信息请参见 tencentcloud_tem_application 。

创建应用

此处以创建一个应用，并在广州地域下的 `test_tf` 环境中部署该应用为示例。

1. 创建 `provider.tf` 文件，指定 `provider` 配置信息。文件内容如下：

```
terraform {
  required_providers {
    tencentcloud = {
      source = "tencentcloudstack/tencentcloud"
      # 通过version指定版本
      # version = ">=1.60.18"
    }
  }
}

provider "tencentcloud" {
  region = "ap-guangzhou"
  # secret_id = "my-secret-id"
```

```
  # secret_key = "my-secret-key"
}
```

- region：填写地域 ID。
- 获取凭证：在 API 密钥管理 面中创建并复制 SecretId 和 SecretKey。

2. 创建 `main.tf` 文件，配置腾讯云 `Provider` 并创建弹性微服务应用。文件内容如下：

```
resource "tencentcloud_tem_application" "application" {
  application_name        = "demo"
  description             = "demo for terraform"
  coding_language         = "JAVA"
  repo_type               = 2
  repo_name               = "qcloud/nginx"
  repo_server             = "ccr.ccs.tencentyun.com"
}
```

- `repo_type` 选择为公共镜像仓库。
- `repo_name` 和 `repo_server` 填写相关镜像仓库名称和地址。

3. 执行以下命令，初始化工作目录并下载插件。

```
terraform init
```

返回信息如下：

```
Initializing the backend...
Initializing provider plugins...
- Finding latest version of tencentcloudstack/tencentcloud...
- Installing tencentcloudstack/tencentcloud v1.77.4...
- Installed tencentcloudstack/tencentcloud v1.77.4 (signed by a HashiCorp partner,
key ID 84F69E1C1BECF459)

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.
```

4. 执行以下命令，查看更新计划。

```
Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # tencentcloud_tem_application.application will be created
  + resource "tencentcloud_tem_application" "application" {
      + application_name         = "demo_tf"
      + coding_language          = "JAVA"
      + description              = "demo for terraform"
      + id                       = (known after apply)
      + repo_name                = "qcloud/nginx"
      + repo_server              = "ccr.ccs.tencentyun.com"
      + repo_type                = 2
      + use_default_image_service = 0
    }

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

5. 执行以下命令，创建资源。

```
terraform apply
```

返回信息如下：

```
Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:
```

```
  # tencentcloud_tem_application.application will be created
  + resource "tencentcloud_tem_application" "application" {
      + application_name         = "demo"
      + coding_language          = "JAVA"
      + description              = "demo for terraform"
      + id                       = (known after apply)
      + repo_name                = "qcloud/nginx"
      + repo_server              = "ccr.ccs.tencentyun.com"
      + repo_type                = 2
      + use_default_image_service = 0
    }

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_tem_application.application: Creating...
tencentcloud_tem_application.application: Creation complete after 1s [id=app-vpzq42ep]
```

6. 执行完毕后，您可以在 弹性微服务控制台 应用列表中查看创建的应用。

部署应用

此步骤演示以拉取镜像的方式将上面创建的应用部署至 `test-tf` 环境中。

1. 在 `main.tf` 文件，添加弹性微服务工作负载资源。内容如下：

```
resource "tencentcloud_tem_workload" "workload" {
  application_id = "my-app-id"
  environment_id = "my-env-id"
  deploy_version = "1.9"
  deploy_mode    = "IMAGE"
  img_repo       = "qcloud/nginx"
  init_pod_num   = 1
  cpu_spec       = 1
  memory_spec    = 1
}
```

2. 执行 `terraform plan` 命令，查看资源计划更新。

```
Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # tencentcloud_tem_workload.workload will be created
  + resource "tencentcloud_tem_workload" "workload" {
      + application_id = "my-app-id"
      + cpu_spec       = 1
      + deploy_mode    = "IMAGE"
      + deploy_version = "1.9"
      + environment_id = "my-evn-id"
      + id             = (known after apply)
      + img_repo       = "qcloud/nginx"
      + init_pod_num   = 1
      + memory_spec    = 1

      + env_conf {
          + config = (known after apply)
          + key    = (known after apply)
          + secret = (known after apply)
          + type   = (known after apply)
          + value  = (known after apply)
        }
    }

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

3. 执行 `terraform apply` 命令，执行更新计划。

```
Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # tencentcloud_tem_workload.workload will be created
  + resource "tencentcloud_tem_workload" "workload" {
      + application_id = "my-app-id"
      + cpu_spec       = 1
      + deploy_mode    = "IMAGE"
      + deploy_version = "1.9"
      + environment_id = "my-env-id"
```

```
      + id           = (known after apply)
      + img_repo       = "qcloud/nginx"
      + init_pod_num   = 1
      + memory_spec    = 1

      + env_conf {
        + config = (known after apply)
        + key    = (known after apply)
        + secret = (known after apply)
        + type   = (known after apply)
        + value  = (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_tem_workload.workload: Creating...
tencentcloud_tem_workload.workload: Creation complete after 6s [id=en-
ejoqxm65#app-vpzq42ep]

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

4. 执行完毕后，您可以在 弹性微服务控制台 应用列表中查看部署的应用。

销毁应用

您可以根据需求，执行以下命令销毁所创建和部署的资源。

```
terraform destroy
```

返回信息如下：

```
Terraform will perform the following actions:

  # tencentcloud_tem_application.application will be destroyed
  - resource "tencentcloud_tem_application" "application" {
      - application_name         = "demo" -> null
      - coding_language          = "JAVA" -> null
```

```
    - description             = "demo for terraform" -> null
    - id                      = "my-app-id" -> null
    - repo_name               = "qcloud/nginx" -> null
    - repo_server             = "ccr.ccs.tencentyun.com" -> null
    - repo_type               = 2 -> null
    - use_default_image_service = 0 -> null
  }

# tencentcloud_tem_environment.environment will be destroyed
- resource "tencentcloud_tem_environment" "environment" {
    - description      = "demo for terraform" -> null
    - environment_name = "test_tf" -> null
    - id               = "my-env-id" -> null
    - subnet_ids       = [
        - "subnet-id1",
        - "subnet-id2",
      ] -> null
    - vpc              = "vpc-id" -> null
  }

# tencentcloud_tem_workload.workload will be destroyed
- resource "tencentcloud_tem_workload" "workload" {
    - application_id     = "my-app-id" -> null
    - cpu_spec           = 1 -> null
    - deploy_mode        = "IMAGE" -> null
    - deploy_version     = "1.9" -> null
    - environment_id     = "my-env-id" -> null
    - id                 = "my-env-id#my-app-id" -> null
    - img_repo           = "qcloud/nginx" -> null
    - init_pod_num       = 1 -> null
    - memory_spec        = 1 -> null
    - security_group_ids = [] -> null
  }

Plan: 0 to add, 0 to change, 3 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_tem_application.application: Destroying... [id=my-app-id]
tencentcloud_tem_workload.workload: Destroying... [id=my-env-id#my-app-id]
tencentcloud_tem_environment.environment: Destroying... [id=my-env-id]
```

```
tencentcloud_tem_workload.workload: Destruction complete after 3s
tencentcloud_tem_environment.environment: Destruction complete after 3s
tencentcloud_tem_application.application: Destruction complete after 4s

Destroy complete! Resources: 3 destroyed.
```

管理现存资源

最近更新时间：2022-11-01 16:39:52

操作场景

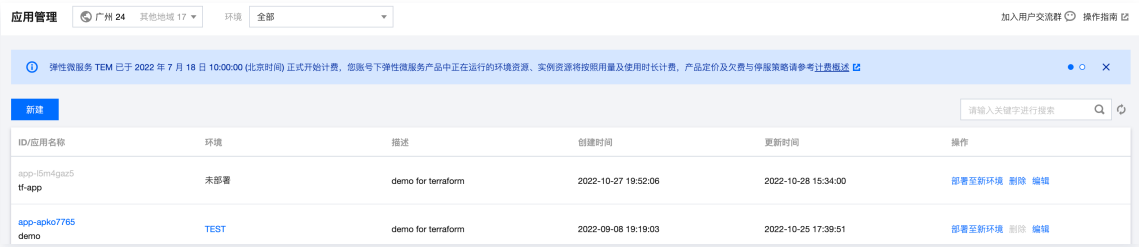
若您在使用 Terraform 管理云资源之前，已经通过腾讯云控制台创建了资源，则可参考本文进行操作，将现存资源使用 Terraform 管理。

操作步骤

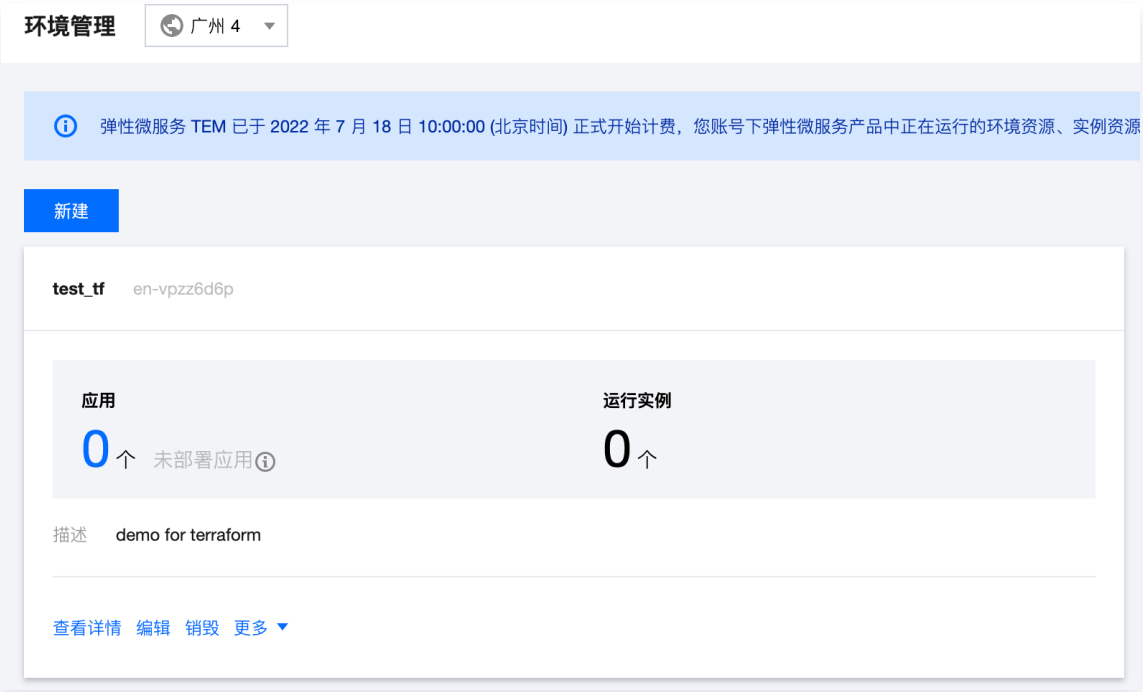
使用 Terraform 接管已经存在的资源，实际上在 Terraform 源文件和状态文件里都反映出该资源的状态即可。本文以使用 Terraform 管理现存的 弹性微服务环境与应用 为例。

步骤1：获取资源 ID

1. 获取应用 ID，您可以在**控制台** > **应用管理**界面中获取现存应用的 ID。



2. 获取环境 ID，您可以在**控制台** > **环境管理**界面中获取现存环境的 ID。



步骤2：导入资源文件

1. 进入 Terraform 工作目录，执行以下命令，查看 main.tf 内容。

```
cat main.tf
```

返回结果如下：

```
resource "tencentcloud_tem_environment" "environment" { }


resource "tencentcloud_tem_application" "application" { }


resource "tencentcloud_tem_workload" "workload" {
  application_id = tencentcloud_tem_application.application.id
  environment_id = tencentcloud_tem_environment.environment.id
  deploy_version = "1.9"
  deploy_mode    = "IMAGE"
  img_repo       = "qcloud/nginx"
  init_pod_num   = 1
  cpu_spec       = 1
  memory_spec    = 1
}
```

在此示例中，main.tf 中已经规划了一个名为 environment 的环境和一个名为 application 的应用，并通过名为 workload 的工作负载部署至以上的环境，但是从以上代码中可以看出，application 和 environment 为空。在此将展示如何将已经创建的存量环境和应用纳入 Terraform 管理。

2. 在文件所在目录下执行以下命令，完成初始化工作。

```
terraform init --upgrade
```

返回结果如下：

```
Initializing the backend...

Initializing provider plugins...
- Finding latest version of tencentcloudstack/tencentcloud...
- Using previously-installed tencentcloudstack/tencentcloud v1.78.6

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. 初始化完成后，执行以下命令，将资源导入状态文件。

```
terraform import tencentcloud_tem_application.application recourceID
terraform import tencentcloud_tem_environment.environment en-vpzz6d6p
```

返回的信息如下所示：

```
tencentcloud_tem_application.application: Importing from ID "app-l5m4gaz5"...
tencentcloud_tem_application.application: Import prepared!
  Prepared tencentcloud_tem_application for import
tencentcloud_tem_application.application: Refreshing state... [id=app-l5m4gaz5]

Import successful!

The resources that were imported are shown above. These resources are now in
your Terraform state and will henceforth be managed by Terraform.
```

```
tencentcloud_tem_environment.environment: Importing from ID "en-vpzz6d6p"...
tencentcloud_tem_environment.environment: Import prepared!
  Prepared tencentcloud_tem_environment for import
tencentcloud_tem_environment.environment: Refreshing state... [id=en-vpzz6d6p]

Import successful!

The resources that were imported are shown above. These resources are now in
your Terraform state and will henceforth be managed by Terraform.
```

4. 执行以下命令，查看状态文件

```
cat terraform.tfstate
```

可查看如下关于资源的信息：

```
{
  "version": 4,
```

```
  "terraform_version": "1.2.8",
  "serial": 47,
  "lineage": "c12a26a4-134b-e0e5-9ba1-0b2234xxxxxx",
  "outputs": {},
  "resources": [
  {
    "mode": "managed",
    "type": "tencentcloud_tem_application",
    "name": "application",
    "provider": "provider[\"registry.terraform.io/tencentcloudstack/tencentcloud\"]",
    "instances": [
      {
        "schema_version": 0,
        "attributes": {
          "application_name": "demo",
          "coding_language": "JAVA",
          "description": "demo for terraform",
          "id": "app-apko7765",
          "instance_id": "",
          "repo_name": "qcloud/nginx",
          "repo_server": null,
          "repo_type": 2,
          "use_default_image_service": null
        },
        "sensitive_attributes": [],
        "private": "eyJzY2hlbWFfdmVyc2lvbiI6XXXXXX=="
      }
    ]
  },
  {
    "mode": "managed",
    "type": "tencentcloud_tem_environment",
    "name": "environment",
    "provider": "provider[\"registry.terraform.io/tencentcloudstack/tencentcloud\"]",
    "instances": [
      {
        "schema_version": 0,
        "attributes": {
          "description": "demo for terraform",
          "environment_name": "test_tf",
          "id": "en-vpzz6d6p",
          "subnet_ids": [
            "subnet-5ob92rbm",
            "subnet-dwpro9ni"
          ],
```

```
        "vpc": "vpc-dd5m94px"
      },
      "sensitive_attributes": [],
      "private": "eyJzY2hlbWFfdmVyc2lvbiI6XXXXXX=="
    }
  ]
 }
  ]
}
```

步骤3：更新源文件

1. 执行以下命令，打印资源信息。

```
terraform show
```

返回的信息如下所示：

```
# tencentcloud_tem_application.application:
resource "tencentcloud_tem_application" "application" {
 application_name = "demo"
 coding_language  = "JAVA"
 description      = "demo for terraform"
 id               = "app-apko7765"
 repo_name        = "qcloud/nginx"
 repo_type        = 2
}

# tencentcloud_tem_environment.environment:
resource "tencentcloud_tem_environment" "environment" {
 description      = "demo for terraform"
 environment_name = "test_tf"
 id               = "en-vpzz6d6p"
 subnet_ids       = [
    "subnet-5ob92rbm",
    "subnet-dwpro9ni",
 ]
 vpc              = "vpc-dd5m94px"
}
```

2. 将资源代码拷贝至 Terraform 源文件 main.tf 中。需删除其中不可设置的选项，例如 id。完成编辑后，main.tf 文件内容如下所示：

```
resource "tencentcloud_tem_environment" "environment" {
 description      = "demo for terraform"
 environment_name = "test_tf"
 # id              = "en-vpzz6d6p"
 subnet_ids       = [
    "subnet-5ob92rbm",
    "subnet-dwpro9ni",
 ]
 vpc              = "vpc-dd5m94px"
}

resource "tencentcloud_tem_application" "application" {
 application_name = "demo"
 coding_language  = "JAVA"
 description      = "demo for terraform"
 # id              = "app-apko7765"
 repo_name        = "qcloud/nginx"
 repo_type        = 2

}

resource "tencentcloud_tem_workload" "workload" {
  application_id = "${tencentcloud_tem_application.application.id}"
  environment_id = "${tencentcloud_tem_environment.environment.id}"
  deploy_version = "1.9"
  deploy_mode    = "IMAGE"
  img_repo       = "qcloud/nginx"
  init_pod_num   = 1
  cpu_spec       = 1
  memory_spec    = 1
}
```

步骤4：验证结果

1. 执行以下命令，使用当前工作目录下的代码和状态文件更新 Terraform 规划。

```
terraform plan
```

2. 返回信息如下所示，可查看 Terraform 已成功接管。

```
Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  + create
```

```
Terraform will perform the following actions:

  # tencentcloud_tem_workload.workload will be created
  + resource "tencentcloud_tem_workload" "workload" {
      + application_id = "app-apko7765"
      + cpu_spec       = 1
      + deploy_mode    = "IMAGE"
      + deploy_version = "1.9"
      + environment_id = "en-vpzz6d6p"
      + id             = (known after apply)
      + img_repo       = "qcloud/nginx"
      + init_pod_num   = 1
      + memory_spec    = 1

      + env_conf {
          + config = (known after apply)
          + key    = (known after apply)
          + secret = (known after apply)
          + type   = (known after apply)
          + value  = (known after apply)
        }
    }

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

至此，Terraform 已经接管了现存的 environment 和 application 资源，并将规划创建 workload。之后，您可以通过 Terraform 对资源进行编辑和删除。