云资源自动化 for Terraform

快速开始

腾讯云

文档目录

快速开始
本地使用
CLI 方式
VS Code 插件方式
云端使用

快速开始
本地使用
CLI 方式

最近更新时间：2024-11-27 14:15:02

安装 Terraform

下载安装包

前往 Terraform 官网，使用命令行直接安装 Terraform 或下载二进制安装文件。

解压并配置全局路径

- Linux/MAC：`export PATH=$PATH:${可执行文件所在目录}`
- Windows：可执行文件所在目录添加到系统环境变量 PATH 中

验证是否安装成功

执行以下命令，查看是否安装成功。

```
terraform  -version
```

返回信息如下所示（版本号可能存在差异），则表示安装成功。

```
Terraform v1.3.0
on darwin_amd64

Your version of Terraform is out of date! The latest version
is 1.3.2. You can update by downloading from https://www.terraform.io/downloads.html
```

认证和鉴权

凭证获取

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。

静态凭证鉴权

在用户目录下创建 `provider.tf` 文件，输入如下内容：
`my-secret-id` 及 `my-secret-key` 请替换为 获取凭证 中的 SecretId 和 SecretKey。

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

环境变量鉴权

请将如下信息添加至环境变量配置：
`YOUR_SECRET_ID` 及 `YOUR_SECRET_KEY` 请替换为 获取凭证 中的 SecretId 和 SecretKey。

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

使用 Terraform 创建腾讯云 VPC

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"
}
```

2. 创建 main.tf 文件，配置腾讯云 Provider 并创建私有网络 VPC。文件内容如下：

```
resource "tencentcloud_vpc" "foo" {
    name         = "ci-temp-test-updated"
    cidr_block   = "10.0.0.0/16"
    dns_servers  = ["119.29.29.29", "8.8.8.8"]
    is_multicast = false

    tags = {
        "test" = "test"
    }
}
```

3. 执行以下命令，初始化工作目录并下载插件。

```
terraform init
```

返回信息如下所示：

> **说明：**
> 遇到下载失败问题，请参考 Init 加速 。

```
➜  terraform_workspace terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of tencentcloudstack/tencentcloud...
- Installing tencentcloudstack/tencentcloud v1.60.18...
- Installed tencentcloudstack/tencentcloud v1.60.18 (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，升级 Provider 版本。**

```
terraform init -upgrade
```

返回信息如下所示：

```
➜  terraform_workspace terraform init -upgrade

Initializing the backend...

Initializing provider plugins...
- Finding tencentcloudstack/tencentcloud versions matching ">= 1.60.18"...
- Installing tencentcloudstack/tencentcloud v1.60.19...
- Installed tencentcloudstack/tencentcloud v1.60.19 (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 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.
```

5. **执行以下命令，查看执行计划，显示将要创建的资源详情。**

```
terraform plan
```

返回信息如下所示：

```
➜  terraform_workspace terraform plan

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_vpc.foo will be created
  + resource "tencentcloud_vpc" "foo" {
      + cidr_block             = "10.0.0.0/16"
      + create_time            = (known after apply)
      + default_route_table_id = (known after apply)
      + dns_servers            = [
          + "119.29.29.29",
          + "8.8.8.8",
        ]
      + id                     = (known after apply)
      + is_default             = (known after apply)
      + is_multicast           = false
      + name                   = "ci-temp-test-updated"
      + tags                   = {
          + "test" = "test"
        }
    }

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 guarantee to take exactly
these
actions if you run "terraform apply" now.
```

6. 执行以下命令，创建资源。

```
terraform apply
```

根据提示输入 yes 创建资源，返回信息如下所示：

```
➜  terraform_workspace terraform apply

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_vpc.foo will be created
  + resource "tencentcloud_vpc" "foo" {
      + cidr_block             = "10.0.0.0/16"
      + create_time            = (known after apply)
      + default_route_table_id = (known after apply)
      + dns_servers            = [
          + "119.29.29.29",
          + "8.8.8.8",
        ]
      + id                     = (known after apply)
      + is_default             = (known after apply)
      + is_multicast           = false
      + name                   = "ci-temp-test-updated"
      + tags                   = {
          + "test" = "test"
```

```
        }
    }

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_vpc.foo: Creating...
tencentcloud_vpc.foo: Still creating... [10s elapsed]
tencentcloud_vpc.foo: Creation complete after 13s [id=vpc-07mx4yfd]

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

执行完毕后，您可以在腾讯云控制台查看创建的资源。

7.（可选）更新资源。

若您将资源配置信息修改为如下所示内容：

```
resource "tencentcloud_vpc" "foo" {
    name         = "ci-temp-test-updated2"
    cidr_block   = "10.0.0.0/16"
    dns_servers  = ["119.29.29.29", "8.8.8.8"]
    is_multicast = false

    tags = {
        "test" = "test"
    }
}
```

请执行 `terraform plan` **命令更新计划。返回信息如下所示：**

```
➜  terraform_workspace terraform plan
tencentcloud_vpc.foo: Refreshing state... [id=vpc-jhmdf9q9]

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_vpc.foo will be updated in-place
  ~ resource "tencentcloud_vpc" "foo" {
        id                    = "vpc-jhmdf9q9"
      ~ name                  = "ci-temp-test-updated" -> "ci-temp-test-updated2"
        tags                  = {
            "test" = "test"
        }
        # (6 unchanged attributes hidden)
    }

Plan: 0 to add, 1 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.
```

执行 `terraform apply` 命令，应用更新的数据创建资源。返回信息如下：

```
➜  terraform_workspace terraform apply
tencentcloud_vpc.foo: Refreshing state... [id=vpc-jhmdf9q9]

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_vpc.foo will be updated in-place
  ~ resource "tencentcloud_vpc" "foo" {
        id                    = "vpc-jhmdf9q9"
      ~ name                  = "ci-temp-test-updated" -> "ci-temp-test-updated2"
        tags                  = {
            "test" = "test"
        }
        # (6 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_vpc.foo: Modifying... [id=vpc-jhmdf9q9]
tencentcloud_vpc.foo: Modifications complete after 1s [id=vpc-jhmdf9q9]

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

8. 您可根据实际需求，执行以下命令销毁资源。

```
terraform destroy
```

返回信息如下所示：

```
➜  terraform_workspace terraform destroy
tencentcloud_vpc.foo: Refreshing state... [id=vpc-07mx4yfd]

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_vpc.foo will be destroyed
  - resource "tencentcloud_vpc" "foo" {
      - cidr_block             = "10.0.0.0/16" -> null
      - create_time            = "2021-12-15 16:20:32" -> null
      - default_route_table_id = "rtb-4m1nmo0e" -> null
      - dns_servers            = [
```

```
            - "119.29.29.29",
            - "8.8.8.8",
          ] -> null
        - id                      = "vpc-07mx4yfd" -> null
        - is_default              = false -> null
        - is_multicast            = false -> null
        - name                    = "ci-temp-test-updated" -> null
        - tags                    = {
            - "test" = "test"
          } -> 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_vpc.foo: Destroying... [id=vpc-07mx4yfd]
tencentcloud_vpc.foo: Destruction complete after 7s

Destroy complete! Resources: 1 destroyed.
```

VS Code 插件方式

最近更新时间：2023-12-25 16:02:01

Tencent Cloud Terraform VS Code 插件可有效改善您在使用 Terraform 创建和管理腾讯云资源时的效率。同时降低初次使用 Terraform 的难度和管理腾讯云 Terraform 资源的复杂性。

1. 预配置环境

安装 Terraform

操作步骤详情请参见 Terraform 安装。

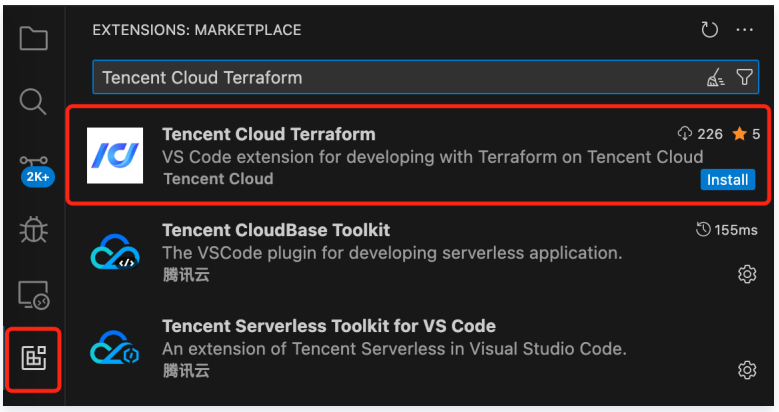
获取 AKSK

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。

2. 安装 Tencent Cloud Terraform VS Code 插件

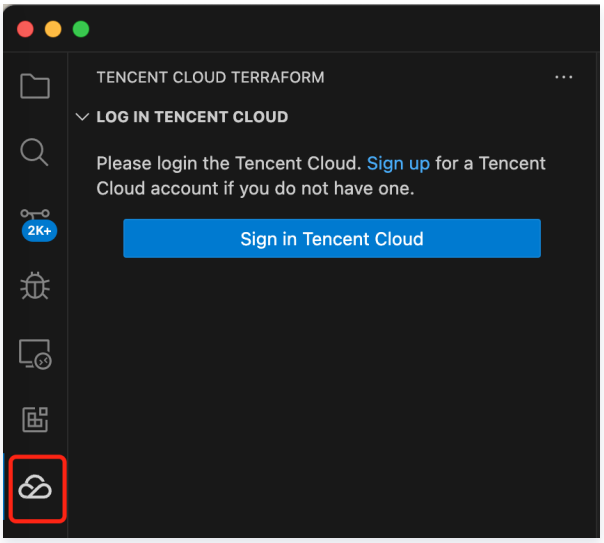
1. 运行 VS Code。
2. 在左侧**插件 (extensions)** 菜单，输入 `Tencent Cloud Terraform` 搜索插件。如下图所示：



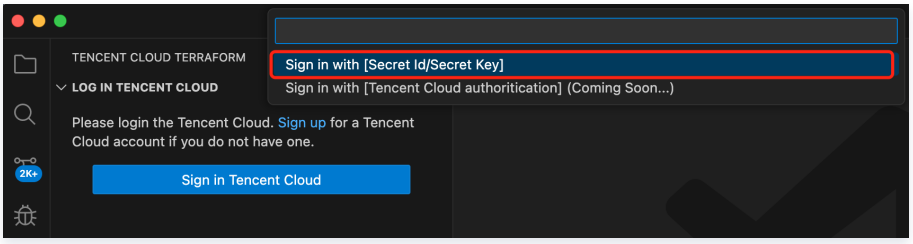
3. 单击**安装 (Install)**。

3. 登录腾讯云

1. 在侧边栏找到 `Tencent Cloud Terraform` 插件，单击**登录 (Sign in)**。如下图所示：

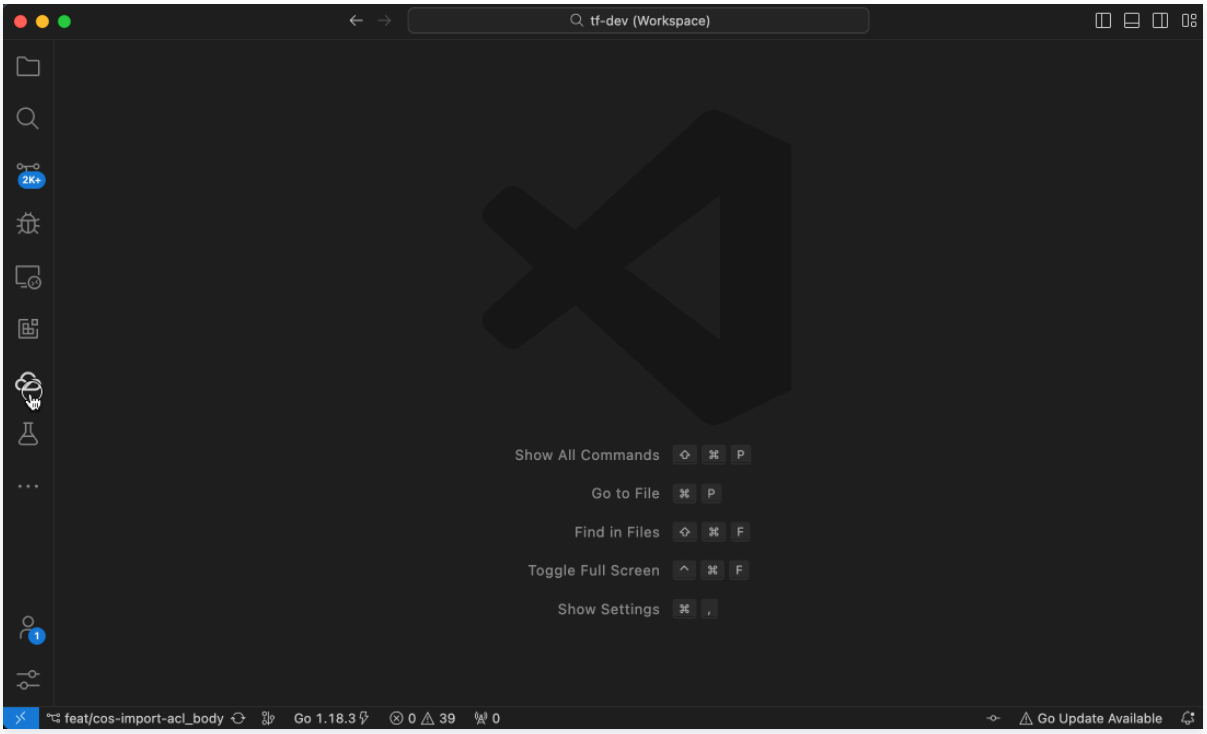


2. 选择 **Login with Secret Id/Secret Key** 方式，使用 AKSK 登录腾讯云。如下图所示：



根据输入框，依次输入账号的 Secret Id, Secret Key 和地域 (Region) 等信息。

3. 当显示账户信息时，表明登录成功。如下图所示：

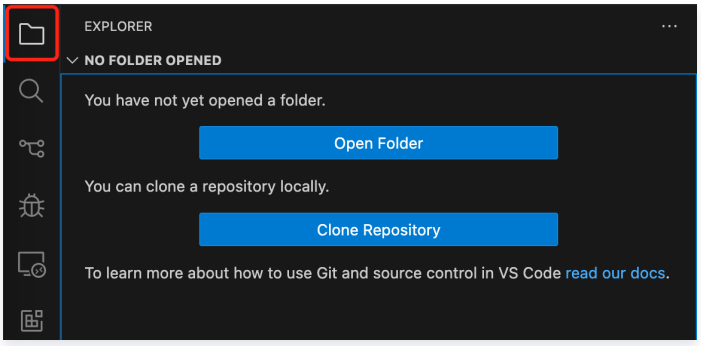


4. 使用插件创建腾讯云 VPC

现在，以创建一个 VPC 资源为例，编写 Terraform 代码。

配置 Provider

1. 在 Explorer 中创建或选择一个文件夹，该文件夹将作为 Terraform 项目的工作空间。如下图所示：



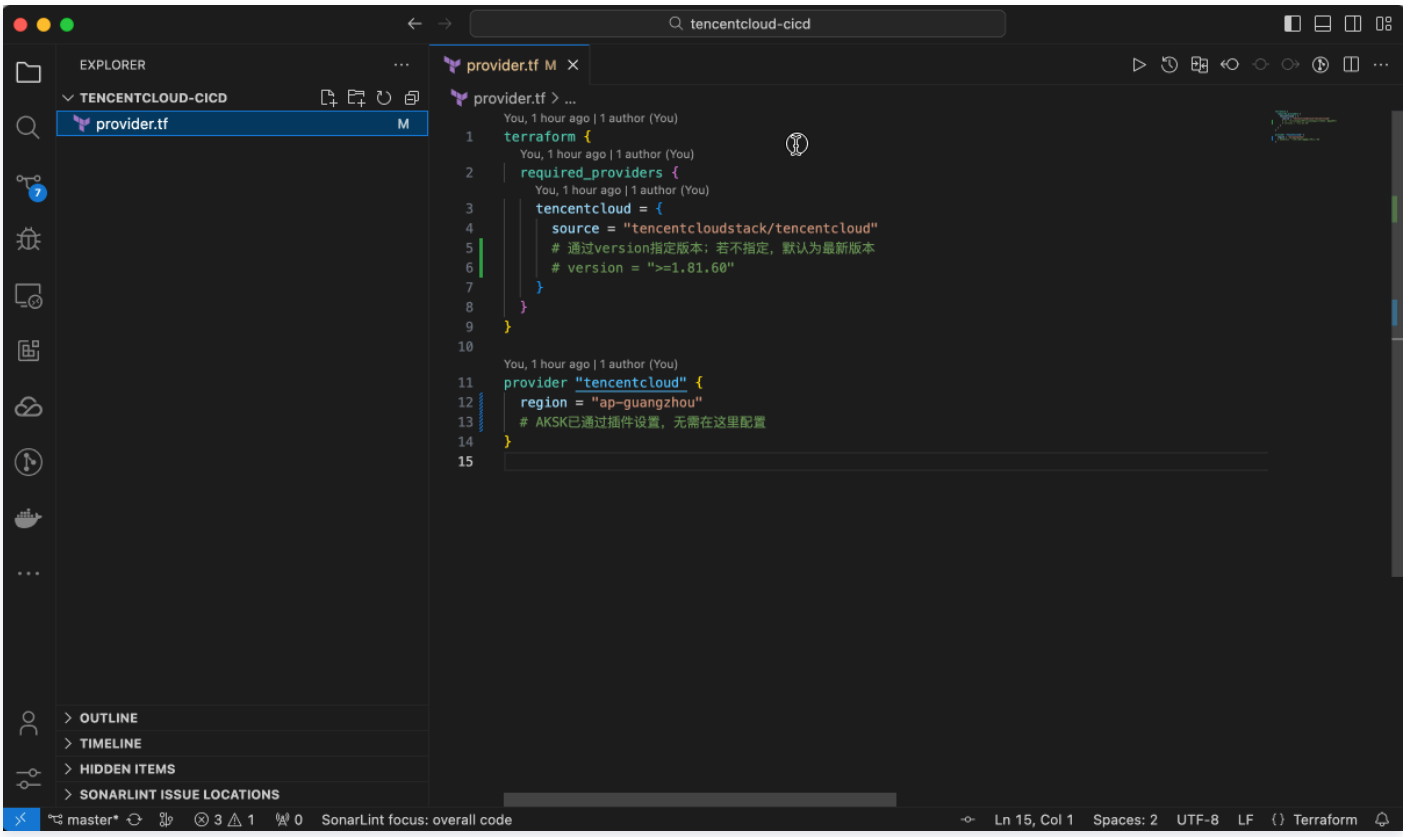
2. 创建一个名为 "provider.tf" 的文件，并在其中指定 Provider 的配置信息。文件内容如下：

```
terraform {
  required_providers {
    tencentcloud = {
      source = "tencentcloudstack/tencentcloud"
      # 通过version指定版本；若不指定，默认为最新版本
```

```
      # version = ">=1.81.60"
    }
  }
}

provider "tencentcloud" {
  region = "ap-guangzhou"
  # AKSK已通过插件设置，无需在这里配置
}
```

3. 初始化 Provider 配置。

   通过调用 VS Code 命令面板（快捷键：Shift + Command + P (Mac) / Ctrl + Shift + P (Windows/Linux)），选择 `TencentCloud Terraform: Init`，执行初始化操作。如下图所示：

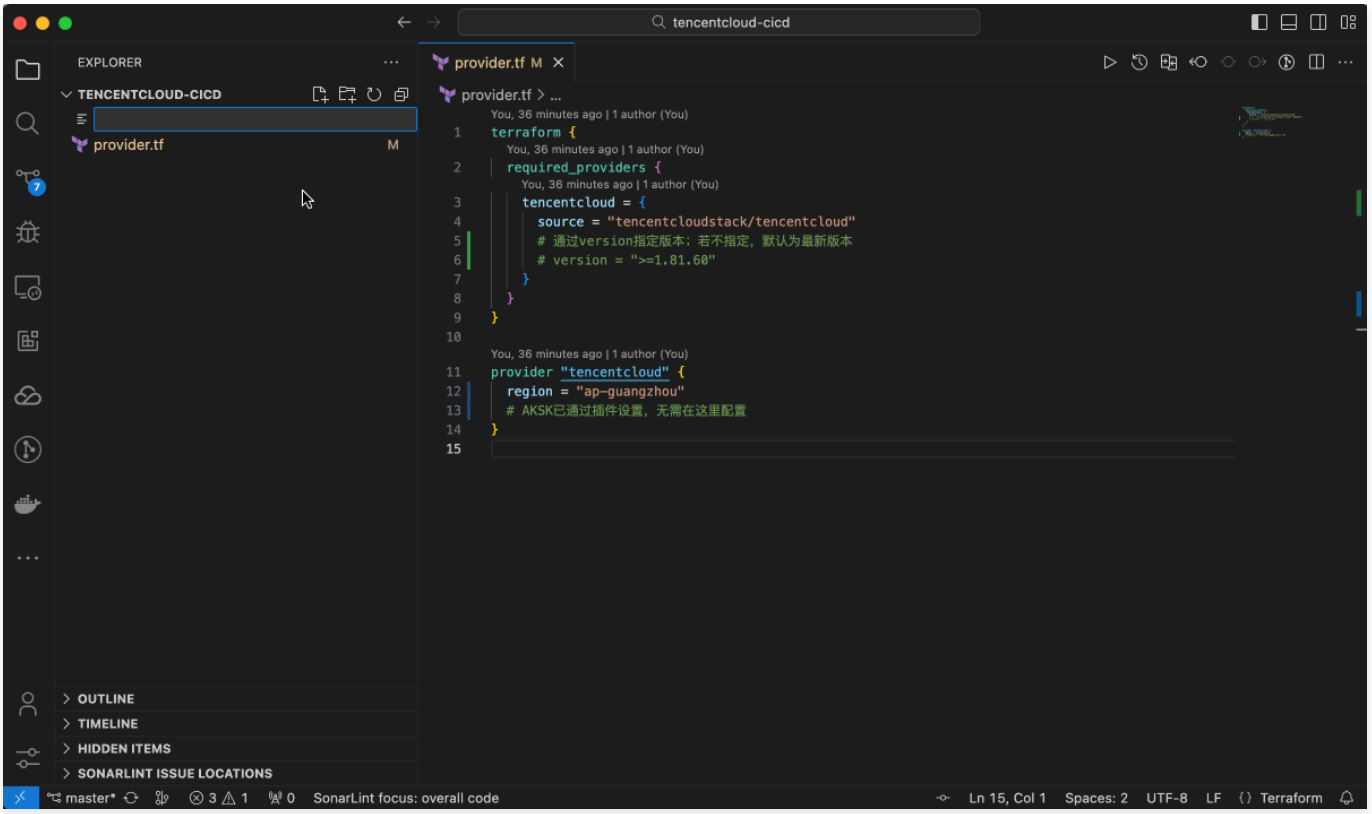


初始化完成后，您将在终端看到 "successfully initialized" 的提示。此时，您的工作空间中会出现两个文件夹：".terraform" 和 ".terraform.lock.hcl"，这表明 Provider 已成功下载到本地。

编写 Terraform 代码

1. 创建一个名为 "main.tf" 的文件。

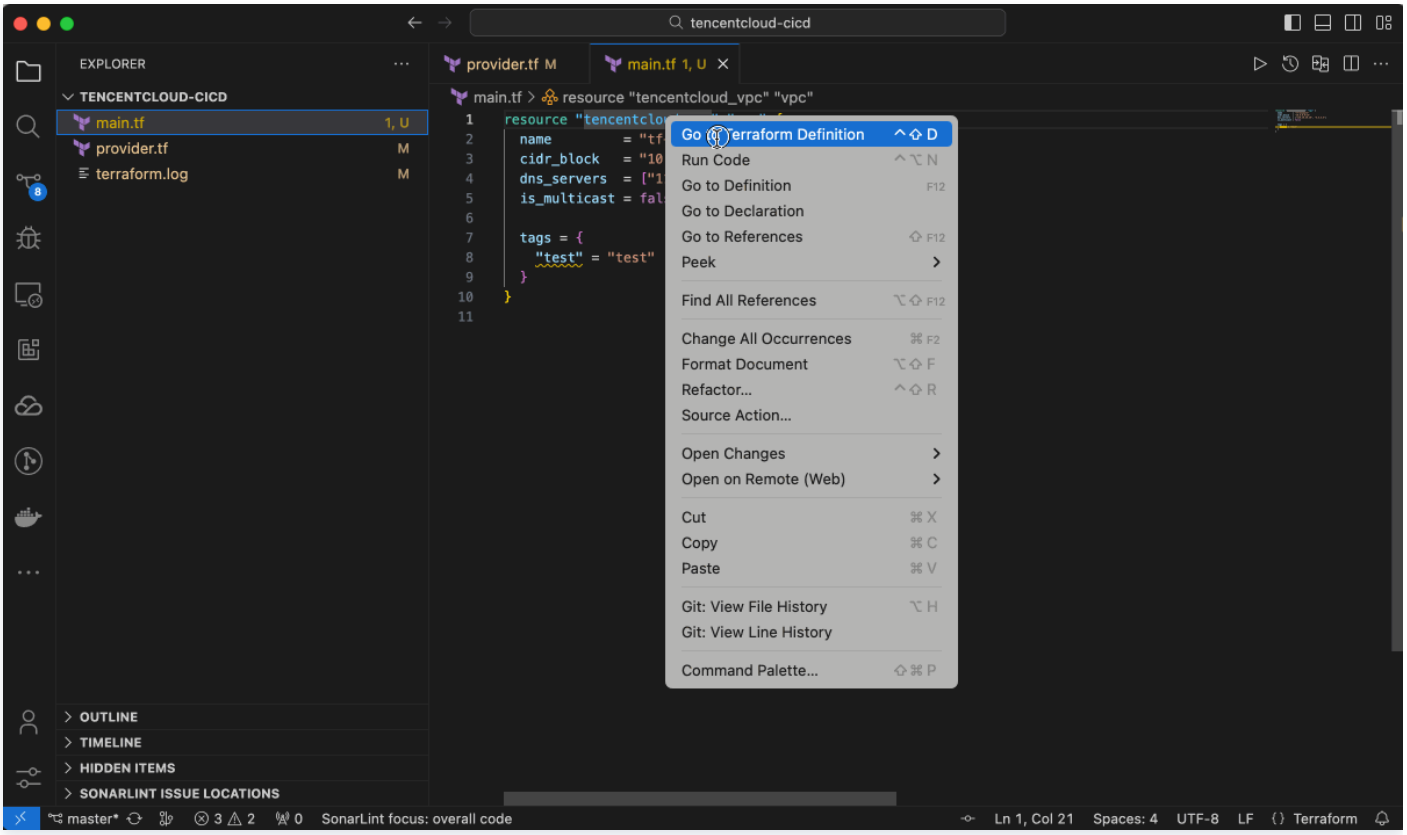
2. 借助插件的**资源提示**和**自动补全**功能，您可以快速创建一个 VPC 资源的代码。在文件中键入 "vpc"，然后根据提示按回车键完成自动补全。如下图所示：



3. 创建出的代码如下所示：

```
# 自动补全代码
resource "tencentcloud_vpc" "vpc" {
  name         = "tf-example"
  cidr_block   = "10.0.0.0/16"
  dns_servers  = ["119.29.29.29", "8.8.8.8"]
  is_multicast = false


  tags = {
    "test" = "test"
  }
}
```

4. 如果对资源中的某些**类型**或**参数（Argument）**的定义不清楚，您可以在资源类型上右键选择菜单项 "Go to Terraform Definition"（快捷键：Ctrl+Shift+D），本地加载资源定义。如下图所示：

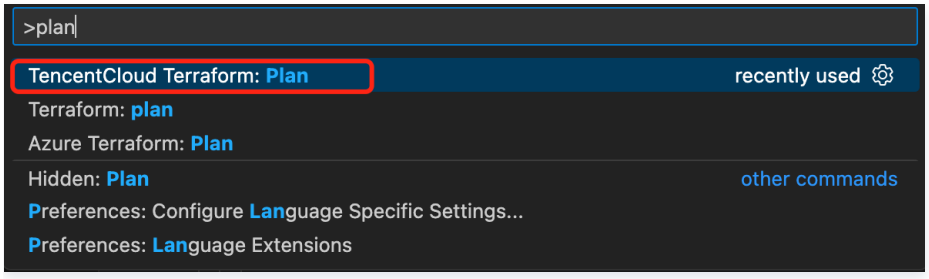


预览待创建 VPC 资源

现在，您可以预览上一节中编写的 main.tf 代码。
如果您对生成的代码不满意，可以进行修改。例如，将代码中 VPC 的名称修改为 "my-first-vpc-demo"。代码示例如下所示：

```
resource "tencentcloud_vpc" "vpc" {
  name         = "my-first-vpc-demo" # 修改为新名称
  cidr_block   = "10.0.0.0/16"
  dns_servers  = ["119.29.29.29", "8.8.8.8"]
  is_multicast = false


  tags = {
    "test" = "test"
  }
}
```

召唤插件的**命令面板**（快捷键：Shift + Command + P (Mac) / Ctrl + Shift + P (Windows/Linux)），执行 `TencentCloud Terraform: Plan`，这将预览即将创建的 VPC 资源。如下图所示：



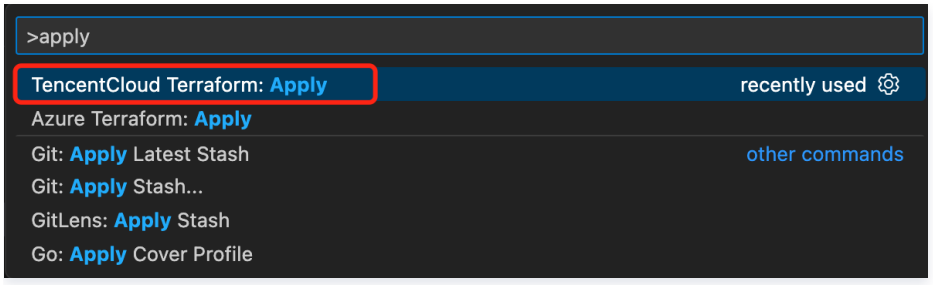
控制台输出如下所示：

```
> terraform plan
```

```
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_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            = [
          + "119.29.29.29",
          + "8.8.8.8",
        ]
      + docker_assistant_cidrs = (known after apply)
      + id                     = (known after apply)
      + is_default             = (known after apply)
      + is_multicast           = false
      + name                   = "my-first-vpc-demo"
      + tags                   = {
          + "test" = "test"
        }
    }

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 guarantee to take exactly these
actions if you run "terraform apply" now.
```

执行资源创建

召唤插件的命令面板，执行 `TencentCloud Terraform:Apply` ，创建资源。如下图所示：



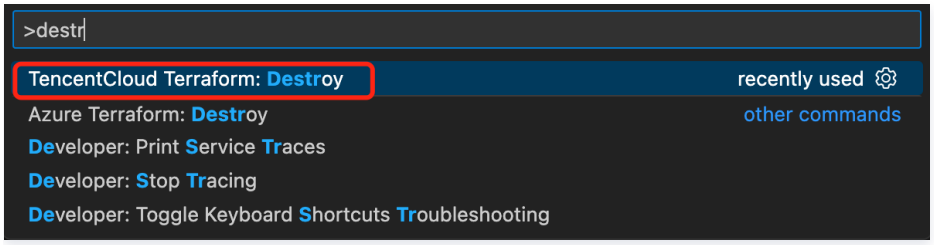
确认无误后，键入"yes"，开始创建资源。

```
> terraform apply

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_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            = [
          + "119.29.29.29",
          + "8.8.8.8",
        ]
      + docker_assistant_cidrs = (known after apply)
      + id                     = (known after apply)
      + is_default             = (known after apply)
      + is_multicast           = false
      + name                   = "my-first-vpc-demo"
      + tags                   = {
          + "test" = "test"
        }
    }

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_vpc.vpc: Creating...
tencentcloud_vpc.vpc: Creation complete after 6s [id=vpc-ou0do5d7]

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

（可选）清理资源

若需要销毁刚才创建的资源。召唤插件的命令面板，执行 `TencentCloud Terraform:Destroy` **，销毁资源。如下图所示：**

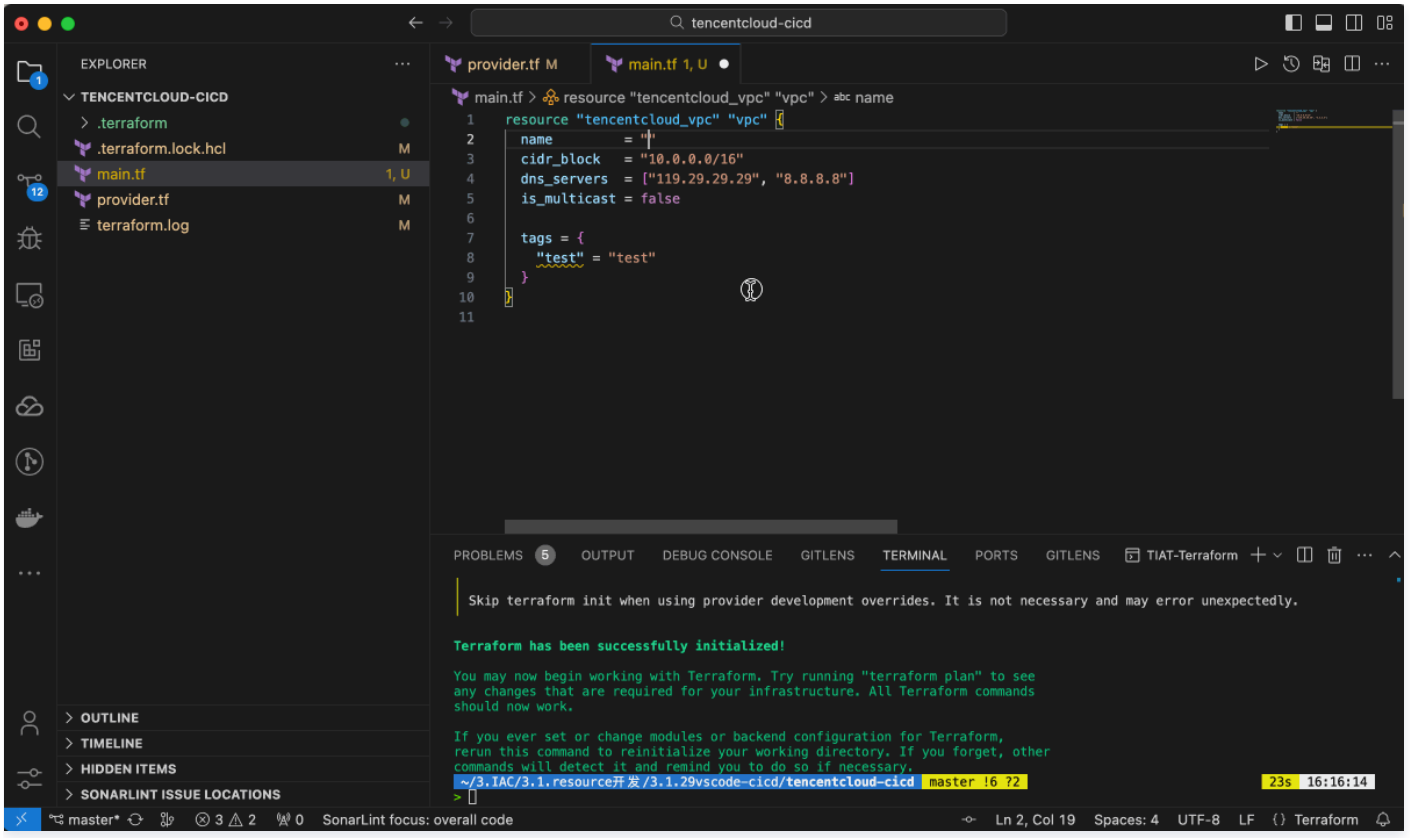


确认无误后，键入"yes"，开始销毁资源。

```
> terraform destroy
tencentcloud_vpc.vpc: Refreshing state... [id=vpc-ou0do5d7]

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_vpc.vpc will be destroyed
  - resource "tencentcloud_vpc" "vpc" {
      - assistant_cidrs        = [] -> null
      - cidr_block             = "10.0.0.0/16" -> null
      - create_time            = "2023-12-25 11:16:14" -> null
      - default_route_table_id = "rtb-jf8ksjl0" -> null
      - dns_servers            = [
```

```
            - "119.29.29.29",
            - "8.8.8.8",
          ] -> null
        - docker_assistant_cidrs = [] -> null
        - id                     = "vpc-ou0do5d7" -> null
        - is_default             = false -> null
        - is_multicast           = false -> null
        - name                   = "my-first-vpc-demo" -> null
        - tags                   = {
            - "test" = "test"
          } -> 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_vpc.vpc: Destroying... [id=vpc-ou0do5d7]
tencentcloud_vpc.vpc: Destruction complete after 3s

Destroy complete! Resources: 1 destroyed.
```

创建一个腾讯云 VPC 的完整 Demo



云端使用

最近更新时间：2023-09-21 20:57:31

腾讯云 Cloud Shell 是一款帮助您运维的免费产品，预装了 Terraform 相关组件，并配置好腾讯云临时凭证（credentials）。因此您可直接在 Cloud Shell 中运行 Terraform 命令。参考以下操作，可以在 Cloud Shell 中快速创建 腾讯云 VPC 资源。

访问 Cloud Shell

登录 腾讯云控制台，选择页面上方的工具 > CloudShell，即可启动 CloudShell。

编写 main.tf 文件

您可以使用 vim 命令编写 main.tf 文件。

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

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

resource "tencentcloud_vpc" "test-vpc" {
  name         = "demo-vpc"
  cidr_block   = "10.0.10.0/24"
  dns_servers  = ["119.29.29.29", "8.8.8.8"]
  is_multicast = false
  tags         = {
    "createdBy" = "terraform",
  }
}
```

初始化 Provider

命令行执行 `terraform init --upgrade` 命令，下载最新版本腾讯云 Provider 插件。

```
[cloudshell@TencentCloudshell ~/data/vpc]$ terraform init --upgrade

Initializing the backend...

Initializing provider plugins...
- Finding latest version of tencentcloudstack/tencentcloud...
- Installing tencentcloudstack/tencentcloud v1.78.9...
- Installed tencentcloudstack/tencentcloud v1.78.9 (verified checksum)

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.

╷
| Warning: Incomplete lock file information for providers
|
| Due to your customized provider installation methods, Terraform was forced to calculate lock file
checksums locally for the following providers:
```

```
|   - tencentcloudstack/tencentcloud
|
| The current .terraform.lock.hcl file only includes checksums for linux_amd64, so Terraform running on
another platform will fail to install these providers.
|
| To calculate additional checksums for another platform, run:
|   terraform providers lock -platform=linux_amd64
| (where linux_amd64 is the platform to generate)
'

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.
```

运行 terraform plan

命令行执行 `terraform plan` 命令，生成执行计划。

```
[cloudshell@TencentCloudshell ~/data/vpc]$ terraform plan

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_vpc.test-vpc will be created
  + resource "tencentcloud_vpc" "test-vpc" {
      + assistant_cidrs        = (known after apply)
      + cidr_block             = "10.0.10.0/24"
      + create_time            = (known after apply)
      + default_route_table_id = (known after apply)
      + dns_servers            = [
          + "119.29.29.29",
          + "8.8.8.8",
        ]
      + docker_assistant_cidrs = (known after apply)
      + id                     = (known after apply)
      + is_default             = (known after apply)
      + is_multicast           = false
      + name                   = "tf-ci-test-vpc"
      + tags                   = {
          + "createdBy" = "terraform"
        }
    }

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 guarantee to take exactly these
actions if you run "terraform apply" now.
```

运行 terraform apply

命令行执行 `terraform apply --auto-approve` 命令，创建资源。

```
[cloudshell@TencentCloudshell ~/data/vpc]$ terraform apply --auto-approve

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_vpc.test-vpc will be created
  + resource "tencentcloud_vpc" "test-vpc" {
      + assistant_cidrs        = (known after apply)
      + cidr_block             = "10.0.10.0/24"
      + create_time            = (known after apply)
      + default_route_table_id = (known after apply)
      + dns_servers            = [
          + "119.29.29.29",
          + "8.8.8.8",
        ]
      + docker_assistant_cidrs = (known after apply)
      + id                     = (known after apply)
      + is_default             = (known after apply)
      + is_multicast           = false
      + name                   = "tf-ci-test-vpc"
      + tags                   = {
          + "createdBy" = "terraform"
        }
    }

Plan: 1 to add, 0 to change, 0 to destroy.
tencentcloud_vpc.test-vpc: Creating...
tencentcloud_vpc.test-vpc: Creation complete after 4s [id=vpc-5jr1hibn]

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

控制台查看

登录 私有网络控制台，查看已创建的实例。

ID/名称	IPv4 CIDR	子网	路由表	NAT 网关	VPN 网关	云服务器	专线网关	默认私有网络	创建时间	标签	操作
vpc-[redacted] tf-ci-test-vpc	10.0.10.0/24	0	1	0	0	0	0	否	2022-11-10 16:16:36	1	删除 更多

运行 terraform destroy

命令行执行 `terraform destroy --auto-approve` 命令，销毁资源。

```
[cloudshell@TencentCloudshell ~/data/vpc]$ terraform destroy --auto-approve
tencentcloud_vpc.test-vpc: Refreshing state... [id=vpc-5jr1hibn]

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_vpc.test-vpc will be destroyed
  - resource "tencentcloud_vpc" "test-vpc" {
      - assistant_cidrs        = [] -> null
      - cidr_block             = "10.0.10.0/24" -> null
      - create_time            = "2022-11-10 16:16:36" -> null
      - default_route_table_id = "rtb-0oiqq8mc" -> null
      - dns_servers            = [
          - "119.29.29.29",
          - "8.8.8.8",
        ] -> null
      - docker_assistant_cidrs = [] -> null
      - id                     = "vpc-5jr1hibn" -> null
      - is_default             = false -> null
      - is_multicast           = false -> null
      - name                   = "tf-ci-test-vpc" -> null
      - tags                   = {
          - "createdBy" = "terraform"
        } -> null
    }

Plan: 0 to add, 0 to change, 1 to destroy.
tencentcloud_vpc.test-vpc: Destroying... [id=vpc-5jr1hibn]
tencentcloud_vpc.test-vpc: Destruction complete after 4s

Destroy complete! Resources: 1 destroyed.
```