TencentCloud Managed Service for Prometheus

Terraform

Product Documentation

Contents

Terraform

Terraform Overview

Last updated：2024-10-09 10:11:05

Terraform is an Infrastructure as Code (IaC) tool that allows creation of reusable, maintainable infrastructure code. You can manage and update your infrastructure using codes, thereby simplifying the deployment and management of cloud infrastructure. With abundant resource types, powerful modules, and flexible customization options, Terraform makes it easy to create and manage a variety of resources, such as CVM and CLB.

Terraform Features and Advantages

Terraform is a powerful open-source tool for automatically deploying and orchestrating cloud infrastructure.
Terraform enables you to operate, construct, define, deploy, and preview cloud infrastructure on Tencent Cloud using simple template language commands.
Terraform is remarkably user-friendly. It uses the simple HCL or JSON configuration language to define infrastructure, requiring virtually no code writing.
With variables, modules, and data source features, Terraform's configuration files are easy to use, reusable, and maintainable.
Terraform can provide plan reports at different stages, so users can get to know the actual operations executions.
Terraform tracks the actual configuration and status information of infrastructure through state files and supports rollback operations.
Note: For details about the application scenarios of Terraform, see Use Cases.

Terraform Resources

In Terraform, resources mainly fall into two categories: Resource and Data Source.
Resource
Resource enables the creation and management of new cloud infrastructure components. Through resource definitions, Terraform can be used for creating, modifying, and deleting a diverse range of resources supported by cloud service providers, such as creating a new CVM instance.

```
#Resource of cvm chc_assist_vpc.
# Reference document: https://registry.terraform.io/providers/tencentcloudstack/ten
resource "tencentcloud_cvm_chc_assist_vpc" "chc_assist_vpc" {
  chc_id = "chc-xxxxx"
}
```

Data Source
A data source is used to access and retrieve information of already existing resources and subsequently implementing it as an input or attribute within the configuration. A data source can be used to retrieve the configuration and attribute

information from an existing cloud infrastructure, making it available for reference and usage within a Terraform configuration file. The following example shows that it is used to query an instance of a created cvm chc_denied_actions:

```
#Using this data source to extract detailed information from cvm chc_denied_actions
# Reference document: https://registry.terraform.io/providers/tencentcloudstack/ten
data "tencentcloud_cvm_chc_denied_actions" "chc_denied_actions" {
  chc_ids = ["chc-xxxxx"]
}
```

For information about the Resource of the TCOP Prometheus version, see Usage with Terraform.

Terraform Tools

Terraform Command Line Interface (CLI):

Terraform CLI is a command line interface designed to create, manage, and manage infrastructure. It brings a suite of commands and options, enabling users to perform various operations, including initializing configuration, creating plans, and applying changes. Terraform CLI also incorporates several sub-commands associated with additional Terraform functionality, such as configuration validation and code formatting. Employing Terraform CLI enables users to interact with Terraform configuration files and manage the infrastructure effectively.

Terraform Provider:

Each cloud provider provides its proprietary Terraform Provider, which is a plugin used to incorporate the features of the cloud provider into Terraform. Each Terraform Provider is tasked with interacting with the APIs of the corresponding cloud service provider, defining available resources and data sources, and performing operations of creating, modifying, or deleting these resources. By deploying the appropriate provider, you can directly use the resources and features of the cloud provider within Terraform configurations.

Note:

The Terraform CLI serves as the command line interface to manage Terraform, while Terraform Provider is the plugin incorporating the features of diverse cloud providers into Terraform. These two components work collaboratively, enabling users to use Terraform to create and manage infrastructure. For more information about Terraform, see the usage document of Terraform.

Usage Advantages

Multi-Cloud Solution: Applicable to multi-cloud solutions, Terraform can deploy analogous infrastructure to Tencent Cloud and local data centers. Due to Terraform's portability, developers can manage resources from distinct cloud providers simultaneously using the same tools and configuration files. Such flexibility allows users to switch to other cloud platforms at any time without the need to modify their configuration files.

Automated Management: Terraform enables the creation of configuration file templates, defining, provisioning, and configuring resources in a repeatable and predictable manner, reducing deployment and management errors due to human factors. Using Terraform allows users to deploy the same template multiple times, creating identical

development, test, and production environments. Additionally, Terraform automates the process of creating and managing infrastructure, thereby enhancing deployment speed and efficiency.
Infrastructure as Code: The Infrastructure as Code (IaC) method of Terraform allows for the management of resources through code, providing convenience in version control, collaboration, and code reuse. Using Terraform, you can save the state of your infrastructure to track system changes and share these configurations with others. This code-based management approach not only simplifies deployment but also ensures consistency, security, and maintainability.
Visual Interface: Terraform's graphical interface allows for the viewing and management of infrastructure resources. This enables users to view and understand their infrastructures and interdependencies of infrastructures more directly. Through this visual interface, users can locate and rectify potential configuration problems faster and easier.
Community Support: Developed by HashiCorp, Terraform is backed by a large community. This community comprises professionals across various industries and experience levels capable of sharing best practices and solutions, offering supports like forums and email lists.

Using TCOP Prometheus Managed Service via Terraform

Prometheus supports managing the following resources via Terraform:

Name	Description
tencentcloud_monitor_tmp_instance	Prometheus Instances
tencentcloud_monitor_tmp_alert_rule	Prometheus Alarm Rules
tencentcloud_monitor_tmp_recording_rule	Prometheus Recording Rules
tencentcloud_monitor_tmp_cvm_agent	Prometheus Cvm Agent
tencentcloud_monitor_tmp_scrape_job	Prometheus cvm_agent Capture Tasks
tencentcloud_monitor_tmp_exporter_integration	Prometheus Integration Center Resources
tencentcloud_monitor_tmp_manage_grafana_attachment	Prometheus Association with Grafana
tencentcloud_monitor_tmp_tke_cluster_agent	Prometheus Association with Containers
	Prometheus Cluster Collection Configuration

tencentcloud_monitor_tmp_tke_config	

Managing Prometheus Instances Using Terraform

Last updated：2025-03-19 10:30:09

Prerequisites

Installing Terraform

For detailed instructions on installing Terraform, see Installing Terraform.

Note:

The installed version of Terraform must not be below v1.6.3. You can verify the version of Terraform installed by using the command terraform --version.

Configuring Tencent Cloud Account Information

Before using Terraform for the first time, go to API Key to apply for a SecretID and SecretKey of the security credentials. If you already have valid credentials, skip this step.

1. Log in to the CAM Console and choose **Access Key** > **API Keys** from the left sidebar.
2. On the API Keys page, click **Create Key** to create a SecretId/SecretKey pair.

There are two methods for configuring Tencent Cloud account information:

Authentication by Static Credentials

Create a `provider.tf` file in the user directory with the following content. Replace `my-secret-id` and `my-secret-key` with your SecretId and SecretKey, respectively.

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

Authentication by Environment Variables

Configure the computer environment variables or cloud environment variables by running the following command. Replace `YOUR_SECRET_ID` and `YOUR_SECRET_KEY` with your SecretId and SecretKey, respectively.

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

Creating Prometheus Instances

1. Create the configuration file for Terraform. Create a file named main.tf in the project directory and open it with an appropriate editor. Add the Tencent Cloud provider configuration to the main.tf or provider.tf file.

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

2. Defining Tencent Cloud Resources: Use the resources provided by Tencent Cloud in the main.tf file to define the resources you want to create and manage.

```
 #Specifying Provider Configuration Information

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

#Creating a Prometheus Instance

resource "tencentcloud_monitor_tmp_instance" "foo" {
  instance_name       = "tf-tmp-instance-sjtest"
  vpc_id              = "vpc-0n42dxzs"
  subnet_id           = "subnet-es8rv1kx"
  data_retention_time = 30
  zone                = "ap-guangzhou-3"
  tags = {
    "createdBy" = "terraform"
  }
}

#Creating a Grafana Instance(Optional)

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

  tags = {
    "createdBy" = "test"
```

```
  }
}

#Binding Grafana and Prometheus Instances (Optional)

resource "tencentcloud_monitor_tmp_manage_grafana_attachment" "foo" {
  grafana_id  = tencentcloud_monitor_grafana_instance.foo.id
  instance_id = tencentcloud_monitor_tmp_instance.foo.id
}
```

3. Initialize the Terraform runtime environment by running the following command:

```
terraform init
```

Expected output information:

```
  Initializing the backend...

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

Terraform has been successfully initialized!

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

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

4. To generate resource planning, run the following command:

```
terraform plan
```

Expected output information:

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

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

Terraform will perform the following actions:

  # tencentcloud_monitor_tmp_instance.foo will be created
```

```
  + resource "tencentcloud_monitor_tmp_instance" "foo" {
      + api_root_path       = (known after apply)
      + data_retention_time = 30
      + id                  = (known after apply)
      + instance_name       = "tf-tmp-instance-sjtest"
      + ipv4_address        = (known after apply)
      + proxy_address       = (known after apply)
      + remote_write        = (known after apply)
      + subnet_id           = "subnet-es8rv1kx"
      + tags                = {
          + "createdBy" = "terraform"
        }
      + vpc_id              = "vpc-0n42dxzs"
      + zone                = "ap-guangzhou-3"
    }

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

5. To create an instance, run the following command:

```
terraform apply
```

Expected output information:

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

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

Terraform will perform the following actions:

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

```
      + zone                = "ap-guangzhou-3"
    }

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

Enter "yes" to the following information to proceed:

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

  Enter a value:
```

If the following information appear, the creation is successful:

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

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

Viewing the Status of Prometheus Instances

Log in to the TCOP, and choose **Managed Service for Prometheus** from the left-hand navigation bar. Existing instances are displayed in the Prometheus instance list.

Deleting Prometheus Instances

To delete a Prometheus instance, run the following command:

```
terraform destroy
```

If the following information appears, you are prompted to enter "yes" for confirmation:

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

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

Terraform will perform the following actions:

  # tencentcloud_monitor_tmp_instance.foo will be destroyed
```

```
  - resource "tencentcloud_monitor_tmp_instance" "foo" {
      - api_root_path       = "http://10.0.0.34:9090/api/v1" -> null
      - data_retention_time = 30 -> null
      - id                  = "prom-8dyb6iny" -> null
      - instance_name       = "tf-tmp-instance-sjtest" -> null
      - ipv4_address        = "10.0.0.34" -> null
      - proxy_address       = "10.0.0.34:9090" -> null
      - remote_write        = "http://10.0.0.34:9090/api/v1/prom/write" -> null
      - subnet_id           = "subnet-es8rv1kx" -> null
      - tags                = {
          - "createdBy" = "terraform"
        } -> null
      - vpc_id              = "vpc-0n42dxzs" -> null
      - zone                = "ap-guangzhou-3" -> null
    }

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

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

  Enter a value: yes

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

Destroy complete! Resources: 1 destroyed.
```

Note:

When `Destroy complete! Resources: 1 destroyed.` is displayed, the instance has been successfully deleted.

Managing the Integration Center of Prometheus Instances Using Terraform

Last updated：2024-10-09 10:10:30

Prerequisites

Installing Terraform

For detailed instructions on installing Terraform, see Installing Terraform.

Note:

The installed version of Terraform must not be below v1.6.3. You can verify the version of Terraform installed by using the command terraform --version.

Configuring Tencent Cloud Account Information

Before using Terraform for the first time, go to API Key to apply for a SecretID and SecretKey of the security credentials. If you already have valid credentials, skip this step.

1. Log in to the CAM Console and choose **Access Key > API Keys** from the left sidebar.
2. On the API Keys page, click **Create Key** to create a SecretId/SecretKey pair.

There are two methods for configuring Tencent Cloud account information:

Authentication by Static Credentials

Create a `provider.tf` file in the user directory with the following content. Replace `my-secret-id` and `my-secret-key` with your SecretId and SecretKey, respectively.

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

Authentication by Environment Variables

Configure the computer environment variables or cloud environment variables by running the following command. Replace `YOUR_SECRET_ID` and `YOUR_SECRET_KEY` with your SecretId and SecretKey, respectively.

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

Incorporating Component Integration from the Integration Center of a Prometheus Instance

1. Creating a new Terraform configuration file: Create a new directory and a file named main.tf in the directory. The following is the configuration information:

```
#Specifying Provider Configuration Information

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

#Prometheus Managing Cloud Monitoring Integration

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

##Cloud Monitoring Plugin Integration
resource "tencentcloud_monitor_tmp_exporter_integration" "tmpExporterMointor" {
  instance_id = tencentcloud_monitor_tmp_instance.foo.id
  kind        = "qcloud-exporter"
  content     = "{\\"name\\":\\"tf-test-cjtest\\",\\"kind\\":\\"qcloud-exporter\\",
  kube_type   = 1
  cluster_id  = ""
}
```

Note:

The fields to be configured when you create the Integration Center component of the Prometheus instance are as follows:

cluster_id: (Required, string) Cluster ID.

content: (Required, string) Integration configuration (internal parameters within content can be replaced as needed).

instance_id: (Required, string) Instance ID.

kind: (Required, string) Type.

kube_type: (Required, integer) Integration configuration.

For more detailed parameters, see Tencent Cloud Integration with GitHub or Terraform Tencent Cloud Provider.

2. To initialize the Terraform runtime environment, run the following command:

```
terraform init
```

Expected output:

```
Initializing the backend...

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

Terraform has been successfully initialized!

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

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

3. To generate a resource plan, run the following command:

```
  terraform plan
```

Expected output:

```
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_monitor_tmp_exporter_integration.tmpExporterIntegration will
be created
  + resource "tencentcloud_monitor_tmp_exporter_integration"
"tmpExporterIntegration" {
      + content     = jsonencode(
            XXX...
        )
      + id          = (known after apply)
      + instance_id = (known after apply)
```

```
      + kind        = "blackbox-exporter"
      + kube_type   = 1
    }

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

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

Plan: 3 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------------
-----------------------------------------

Note: You didn't use the -out option to save this plan, so Terraform can't
guarantee to take exactly these actions if
you run "terraform apply" now.
```

4. To create component integration of the Integration Center, run the following command:

```
   terraform apply
```

Expected output:

```
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:
XXX...


Plan: 3 to add, 0 to change, 0 to destroy.

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

  Enter a value: yes

tencentcloud_monitor_tmp_instance.foo: Creating...
tencentcloud_monitor_tmp_instance.foo: Still creating... [10s elapsed]
...

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

Note:

When the message "Apply complete! Resources: 3 added, 0 changed, 0 destroyed." is displayed, the creation is successful.

Viewing the Status of Prometheus Instances

Log in to the TCOP, and choose **Managed Service for Prometheus** from the left-hand navigation bar. Existing instances are displayed in the Prometheus instance list.

Deleting the Component Integration in the Prometheus Instance Integration Center

To terminate the resources, run the following command:

```
terraform destroy
```

Expected output:

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

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:
XXX...

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

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

  Enter a value: yes

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

Destroy complete! Resources: 1 destroyed.
```

Note:

When `Destroy complete! Resources: 1 destroyed.` is displayed, the instance has been successfully deleted.

Collecting Container Monitoring Data Using Terraform

Last updated：2024-10-09 10:10:11

Prerequisites

Installing Terraform

For detailed instructions on installing Terraform, see Installing Terraform.

Note:

The installed version of Terraform must not be below v1.6.3. You can verify the version of Terraform installed by using the command terraform --version.

Configuring Tencent Cloud Account Information

Before using Terraform for the first time, go to API Key to apply for a SecretID and SecretKey of the security credentials. If you already have valid credentials, skip this step.

1. Log in to the CAM Console and choose **Access Key > API Keys** from the left sidebar.
2. On the API Keys page, click **Create Key** to create a SecretId/SecretKey pair.

There are two methods for configuring Tencent Cloud account information:

Authentication by Static Credentials

Create a `provider.tf` file in the user directory with the following content. Replace `my-secret-id` and `my-secret-key` with your SecretId and SecretKey, respectively.

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

Authentication by Environment Variables

Set the computer variables or cloud environment variables by running the command below. Replace `YOUR_SECRET_ID` and `YOUR_SECRET_KEY` with your SecretId and SecretKey, respectively.

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

Adding Container Monitoring Data Collection to Prometheus Instances

1. Create a new Terraform configuration file: Create a new directory and a file named main.tf in the directory. The following shows the configuration:

```
#Specifying Provider Configuration Information

terraform {
  required_providers {
    tencentcloud = {
      source = "tencentcloudstack/tencentcloud"
    }
  }
}
#Prometheus Managing Cloud Monitoring Integration

#Prometheus Management of Container Clusters (Collection Containers)

resource "tencentcloud_monitor_tmp_tke_cluster_agent" "foo" {
  instance_id = tencentcloud_monitor_tmp_instance.foo.id
  agents {
    region          = "ap-mumbai"
    cluster_type    = "eks"
    cluster_id      = "cls-1uary7z2" #ID of the cluster that needs to be associated
    enable_external = false
  }
}
```

Note:

To create the Integration Center component of a Prometheus instance, the corresponding fields must be configured as follows:

instance_id: (Required, string, ForceNew) Instance ID.

agents: (Required, list) List of agents.

cluster_id: (Required, string) Identifier for the cluster ID.

region: (Required, string) Region restrictions.

enable_external: (Required, Bool) Whether to enable public network CLB.

cluster_type: (Required, string) Cluster type.

For more details about parameters, see **Detailed Parameters**. You can also see Tencent Cloud Provider on Github and Terraform Tencent Cloud Provider.

2. To initialize the Terraform runtime environment, run the following command:

```
terraform init
```

Expected output:

```
Initializing the backend...

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

Terraform has been successfully initialized!

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

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

3. To generate a resource plan, run the following command:

```
  terraform plan
```

Expected output:

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

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

Terraform will perform the following actions:

  # tencentcloud_monitor_tmp_tke_cluster_agent.foo will be created
  + resource "tencentcloud_monitor_tmp_tke_cluster_agent" "foo" {
      + id          = (known after apply)
      + instance_id = "prom-jh0zntj2"

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

```
    }

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

------------------------------------------------------------------------

Note: You didn't use the -out option to save this plan, so Terraform can't guarante
you run "terraform apply" now.
```

4. To create component integration of the Integration Center, run the following command:

```
   terraform apply
```

Expected output:

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

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

Terraform will perform the following actions:

  # tencentcloud_monitor_tmp_tke_cluster_agent.foo will be created
  + resource "tencentcloud_monitor_tmp_tke_cluster_agent" "foo" {
      + id          = (known after apply)
      + instance_id = "prom-jh0zntj2"

      + agents {
          + cluster_id      = "cls-1uary7z2"
          + cluster_name    = (known after apply)
          + cluster_type    = "eks"
          + enable_external = false
          + region          = "ap-mumbai"
          + status          = (known after apply)
        }
    }

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

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

  Enter a value: yes
```

```
tencentcloud_monitor_tmp_tke_cluster_agent.foo: Creating...
Instance content...

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

Viewing the Status of the Prometheus Instance

Log in to the TCOP, and choose **Managed Service for Prometheus** from the left-hand navigation bar. Existing instances are displayed in the Prometheus instance list.

Deleting the Component Integration in the Prometheus Instance Integration Center

To terminate the resources, run the following command:

```
terraform destroy
```

Expected output:

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

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

Terraform will perform the following actions:
Instance content...

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

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

  Enter a value: yes

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

Destroy complete! Resources: 1 destroyed.
```

Note:

If `Destroy complete! Resources: (numbers of existing instances) destroyed.` is displayed, the instance has been deleted.

Configuring Alarm Policies Using Terraform

Last updated：2024-10-09 10:09:36

Prerequisites

Installing Terraform

For detailed instructions on installing Terraform, see Installing Terraform.

Note:

The installed version of Terraform must not be below v1.6.3. You can verify the version of Terraform installed by using the command terraform --version.

Configuring Tencent Cloud Account Information

Before using Terraform for the first time, go to API Key to apply for a SecretID and SecretKey of the security credentials. If you already have valid credentials, skip this step.

1. Log in to the CAM Console and choose **Access Key > API Keys** from the left sidebar.
2. On the API Keys page, click **Create Key** to create a SecretId/SecretKey pair.

There are two methods for configuring Tencent Cloud account information:

Authentication by Static Credentials

Create a `provider.tf` file in the user directory with the following content. Replace `my-secret-id` and `my-secret-key` with your SecretId and SecretKey, respectively.

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

Authentication by Environment Variables

Configure the computer environment variables or cloud environment variables by running the following command. Replace `YOUR_SECRET_ID` and `YOUR_SECRET_KEY` with your SecretId and SecretKey, respectively.

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

Adding a Terraform Alarm Policy to a Prometheus Instance

1. Create a new Terraform configuration file. Create a new directory and a file named main.tf in the directory. The following is the configuration information:

```
#Specifying Provider Configuration Information

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

#Configuring Alarm for Prometheus (Cloud Monitoring Side Configuration)

resource "tencentcloud_monitor_tmp_alert_rule" "foo" {
  duration    = "2m"
  expr        = "avg by (instance) (mysql_global_status_threads_connected) / avg by
  instance_id = tencentcloud_monitor_tmp_instance.foo.id
  receivers = ["notice-zmjsavnp"] # Notifications templates can be created here usi
  rule_name = "MySQL Excessive Connections--tf-Cloud Monitor Test"
  rule_state  = 2
  type = "MySQL/Excessive MySQL Connections"

  annotations {
    key   = "description"
    value = "Too many MySQL connections, instance: {{$labels.instance}}, current va
  }
  annotations {
    key   = "summary"
    value = "MySQL has too many connections(>80%)"
  }

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

Note:

The fields of the configuration are as follows:

duration: Rule persistence duration.

expr: Alarm expression.

instance_id: Instance ID.

receivers: List of alarm recipients.

rule_name: Alarm rule name.

rule_state: Alarm rule status.

type: Alarm rule type.

For more detailed parameters, see Tencent Cloud Integration to GitHub or Terraform Tencent Cloud Provider.

2. To initialize the Terraform runtime environment, run the following command:

```
terraform init
```

Expected output:

```
Initializing the backend...

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

Terraform has been successfully initialized!

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

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

3. To generate a resource plan, run the following command:

```
  terraform plan
```

Expected output:

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

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

Terraform will perform the following actions:

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

```
      + duration    = "2m"
      + expr        = "avg by (instance) (mysql_global_status_threads_connected) /
      + id          = (known after apply)
      + instance_id = "prom-jh0zntj2"
      + receivers   = [
          + "notice-zmjsavnp",
        ]
      + rule_name   = "Excessive MySQL connections--tf-Cloud monitoring test"
      + rule_state  = 2
      + type        = "MySQL/Excessive MySQL Connections"

      + annotations {
          + key   = "description"
          + value = "Excessive MySQL connections, instance: {{$labels.instance}}, C
        }
      + annotations {
          + key   = "summary"
          + value = "Excessive MySQL connections (>80%)"
        }

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

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

------------------------------------------------------------------------

Note: You didn't use the -out option to save this plan, so Terraform can't guarante
you run "terraform apply" now.
```

4. To create component integration of the Integration Center, run the following command:

```
   terraform apply
```

Expected output:

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

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

```
Terraform will perform the following actions:
 Instance content...

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

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

  Enter a value: yes

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

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

Viewing the Status of the Prometheus Instance

Log in to the TCOP, and choose **Managed Service for Prometheus** from the left-hand navigation bar. Existing instances are displayed in the Prometheus instance list.

Deleting the Component Integration in the Prometheus Instance Integration Center

To terminate the resources, run the following command:

```
terraform destroy
```

Expected output:

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

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

Terraform will perform the following actions:
Instance content...

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

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

  Enter a value: yes

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

Destroy complete! Resources: 1 destroyed.
```

Note:

If `Destroy complete! Resources: (numbers of existing instances) destroyed.` is displayed, the instance has been deleted.