

# **Tencent Infrastructure Automation for Terraform Get Started Product Documentation**



## Copyright Notice

©2013-2023 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

## Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

## Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

# Contents

## Get Started

- Use Terraform in Cloud Shell

- Use Terraform in Local PC

# Get Started

## Use Terraform in Cloud Shell

Last updated : 2023-03-07 10:35:48

Cloud Shell is a free Ops product pre-installed with Terraform components and configured with Tencent Cloud temporary credentials to directly run Terraform commands. You can also quickly create [VPC](#) resources in Cloud Shell as instructed.

### Accessing Cloud Shell

Log in to the [Tencent Cloud console](#) and select **Tools > Cloud Shell** at the top of the page to start Cloud Shell.

### Writing the `main.tf` file

You can run the `vim` command to write the `main.tf` file.

```
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",
  }
}
```

## Initializing the provider

Run the `terraform init --upgrade` command to download the latest Tencent Cloud provider plugin.

```
[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 c
alculate lock file checksums locally for the following providers:
| - tencentcloudstack/tencentcloud
|
| The current .terraform.lock.hcl file only includes checksums for linux_amd64, s
o 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.

## Running `terraform plan`

Run the `terraform plan` command to generate the execution plan.

```
[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.

## Running `terraform apply`

Run the `terraform apply --auto-approve` command to create the resource.

```
[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-5jrlhibn]

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

## Viewing in the console

Log in to the [VPC console](#) to view the created instance.

ID/Name	IPv4 CIDR Block	Multicast	Subnet	Route table	NAT gateway	VPN gateway	CVM	Direct connect g...	Default VPC	Creation time	Tag	Operation
		<input type="checkbox"/>	3	2	2	1	0	1	No	2021-04-26 19:40:30		Delete More

## Running `terraform destroy`

Run the `terraform destroy --auto-approve` command to terminate the resource.

```
[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.



# Use Terraform in Local PC

Last updated : 2023-05-29 16:41:10

## Installing Terraform

### Downloading the installation package

Go to [Terraform official website](#) and use the command line to install Terraform directly or download the binary installation package file.

### Decompressing the package file and configuring the global path

Linux: See [How to permanently set \\$PATH on Linux/Unix \[closed\]](#).

Windows: See [Where can I set path to make.exe on Windows?](#).

macOS: See [How to permanently set \\$PATH on Linux/Unix \[closed\]](#).

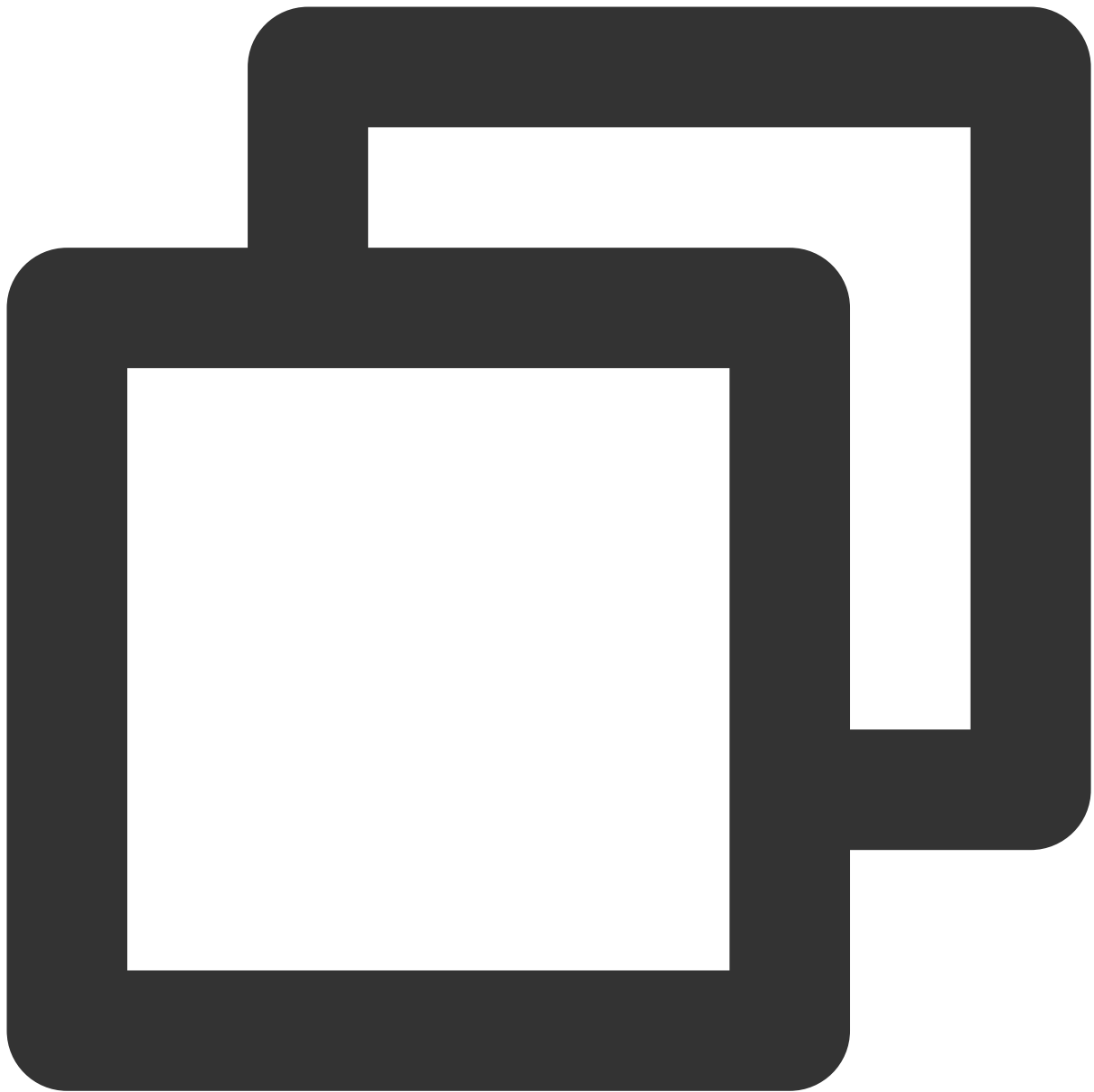
### Verify the installation

Run the following command to check whether the installation worked.



```
terraform -version
```

If the following information is returned (the version number may be different), the installation is successful:



```
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.htm
```

## Authentication

## Getting credentials

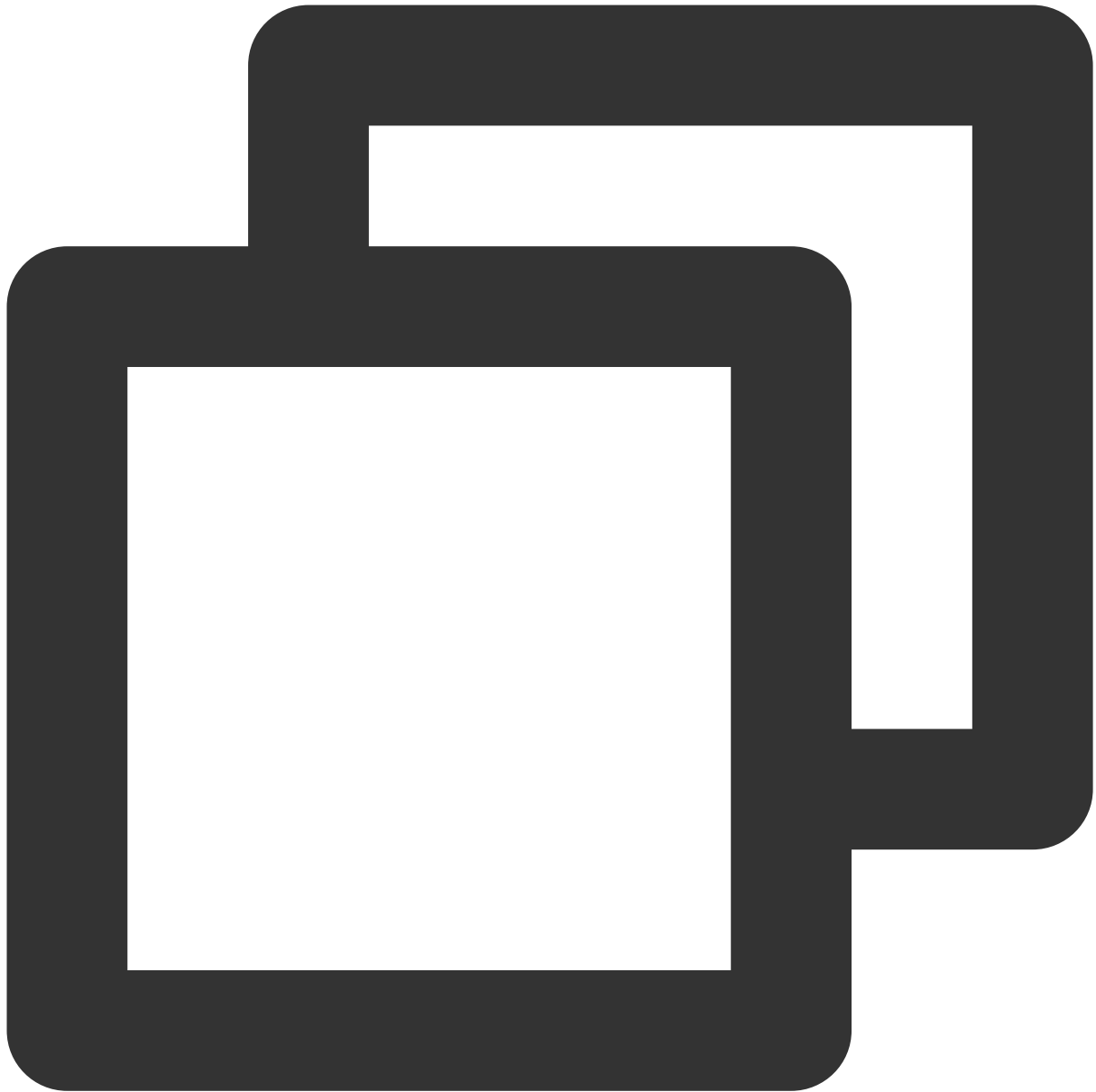
Before using Terraform for the first time, go to the [TencentCloud API Key](#) page to apply for `SecretId` and `SecretKey`. If you already have them, skip this step.

1. Log in to the [CAM console](#) and select **Access Key > Manage API Key** on the left sidebar.
2. On the **Manage API Key** page, click **Create Key** to create a pair of `SecretId/SecretKey`.

## Authentication by static credential

Create a `provider.tf` file in the user directory and enter the following content:

Replace `my-secret-id` and `my-secret-key` with `SecretId` and `SecretKey` obtained in the [Getting credentials](#) step.



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

### Authentication by environment variables

Add the following content to the environment variables:

Replace `YOUR_SECRET_ID` and `YOUR_SECRET_KEY` with `SecretId` and `SecretKey` obtained in the [Getting credentials](#) step.



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

## Creating a VPC via Terraform

1. Create a `provider.tf` file, paste the following content to the file and specify the provider configuration information:

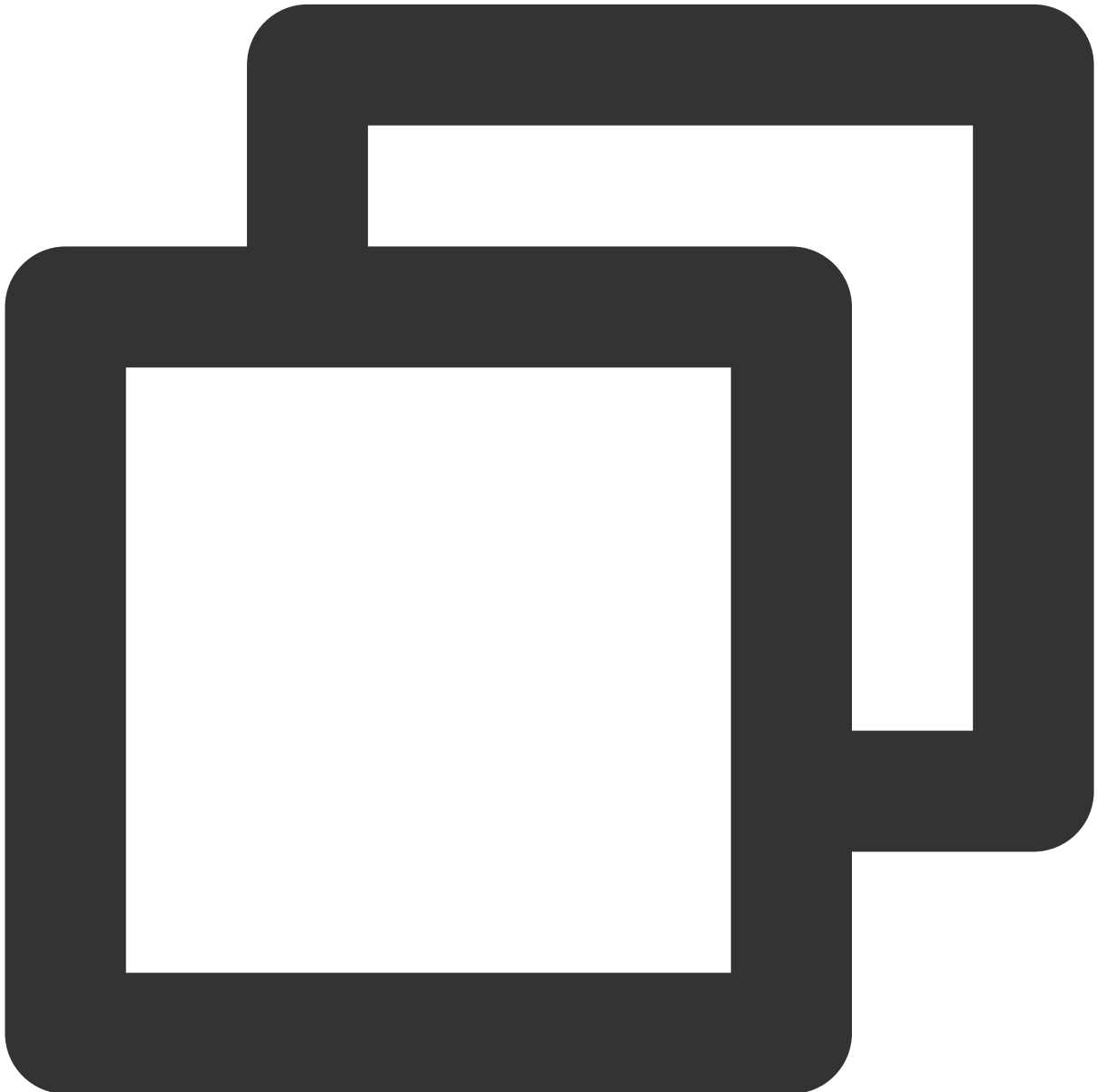


```
terraform {
  required_providers {
    tencentcloud = {
      source = "tencentcloudstack/tencentcloud"
      # Specify the version by `version`
      # version = ">=1.60.18"
    }
  }
}

provider "tencentcloud" {
```

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

2. Create a `main.tf` file and paste the following content to the file to configure TencentCloud Provider and create a VPC.



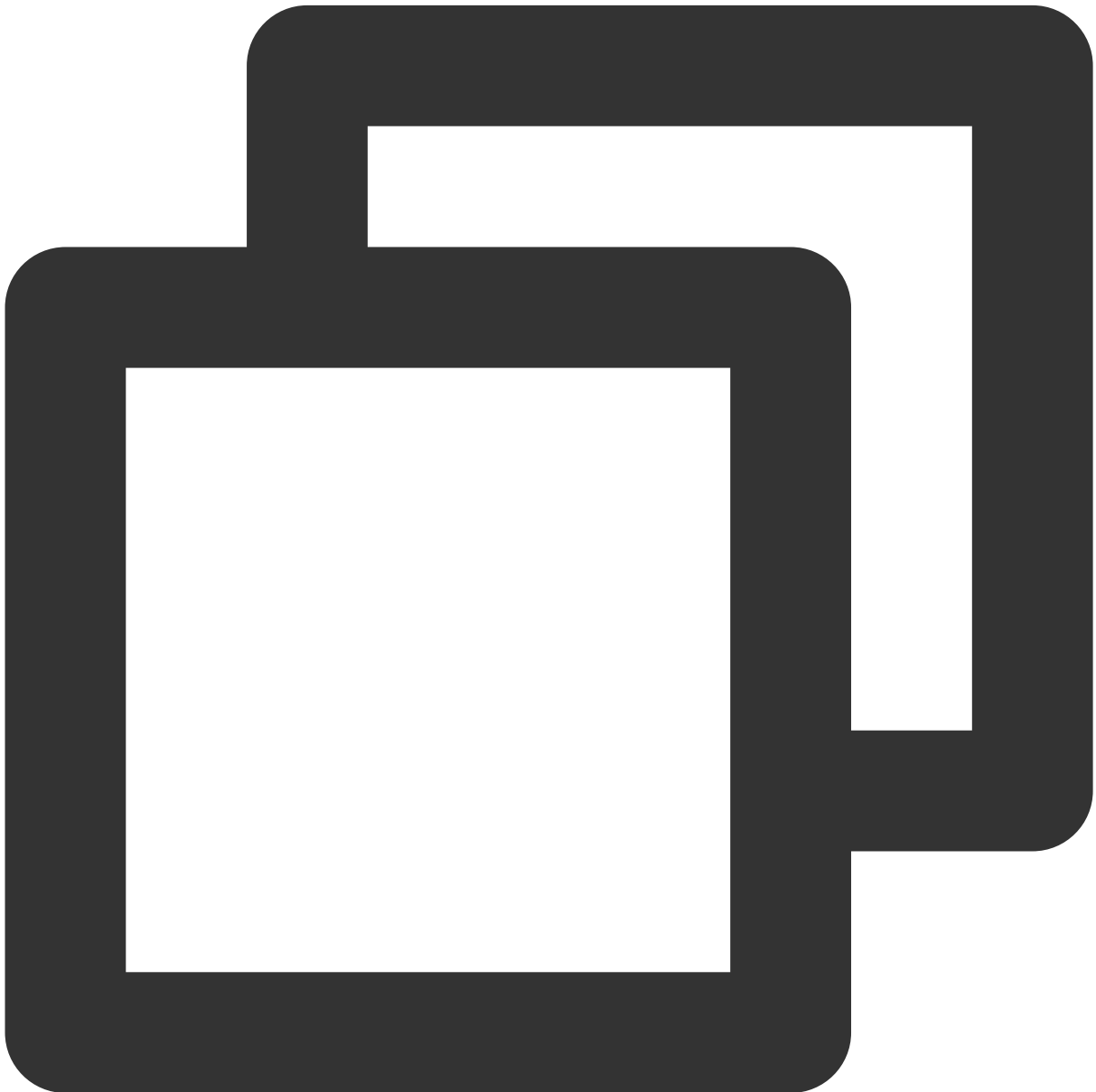
```
resource "tencentcloud_vpc" "foo" {  
  name          = "ci-temp-test-updated"  
  cidr_block    = "10.0.0.0/16"  
}
```



```
dns_servers = ["119.29.29.29", "8.8.8.8"]
is_multicast = false

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

3. Run the following command to initialize the working directory and download the plugin.

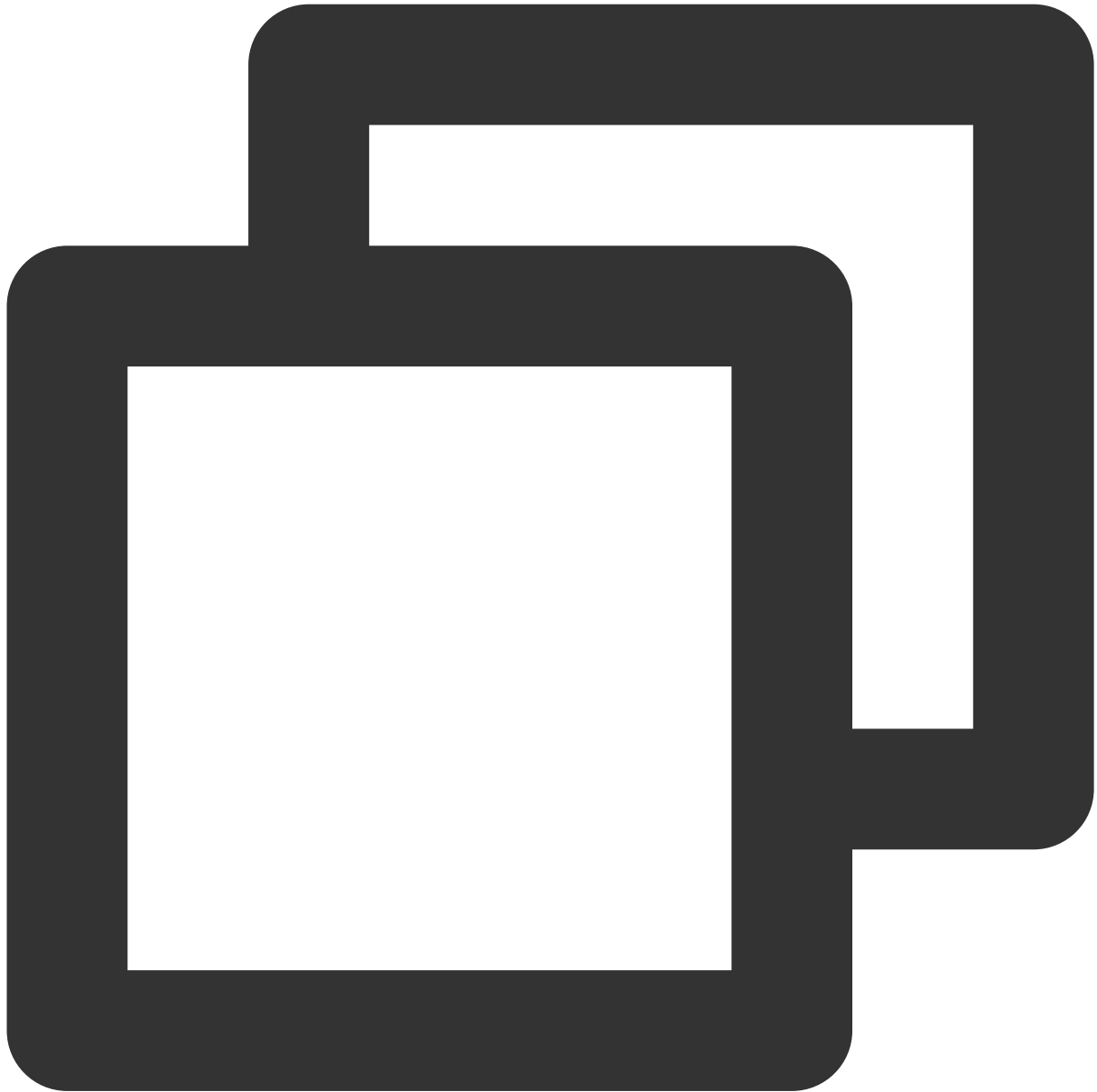


```
terraform init
```

The response is as follows:

**Note :**

In case of a download failure, see [Init Acceleration](#).



```
→ 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,  
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.
```

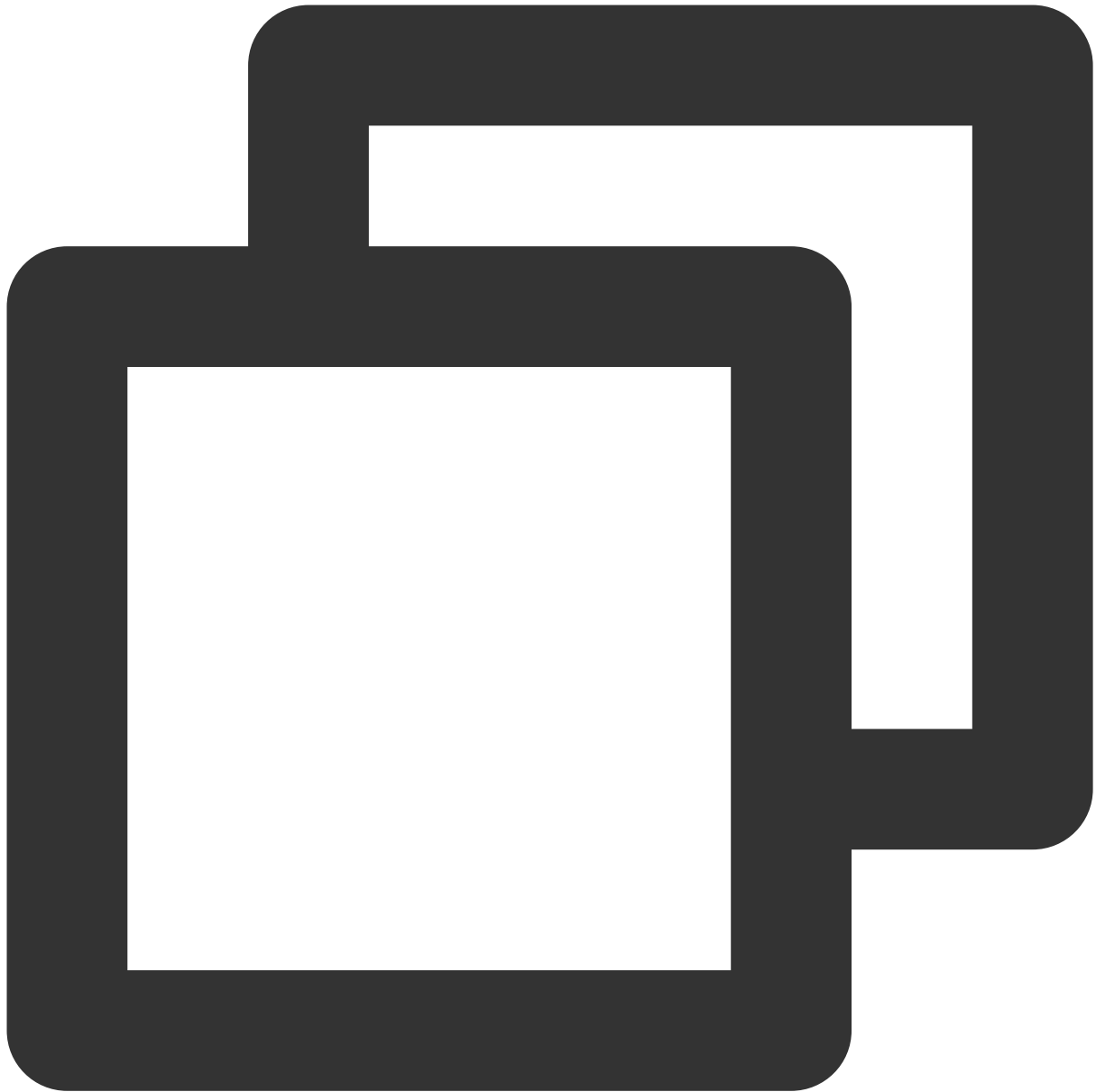
```
If you ever set or change modules or backend configuration for Terraform,  
rerun this command to reinitialize your working directory. If you forget, other  
commands will detect it and remind you to do so if necessary.
```

4. Run the following command to upgrade the provider version.



```
terraform init -upgrade
```

The response is as follows:



```
→ 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,

```
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
```

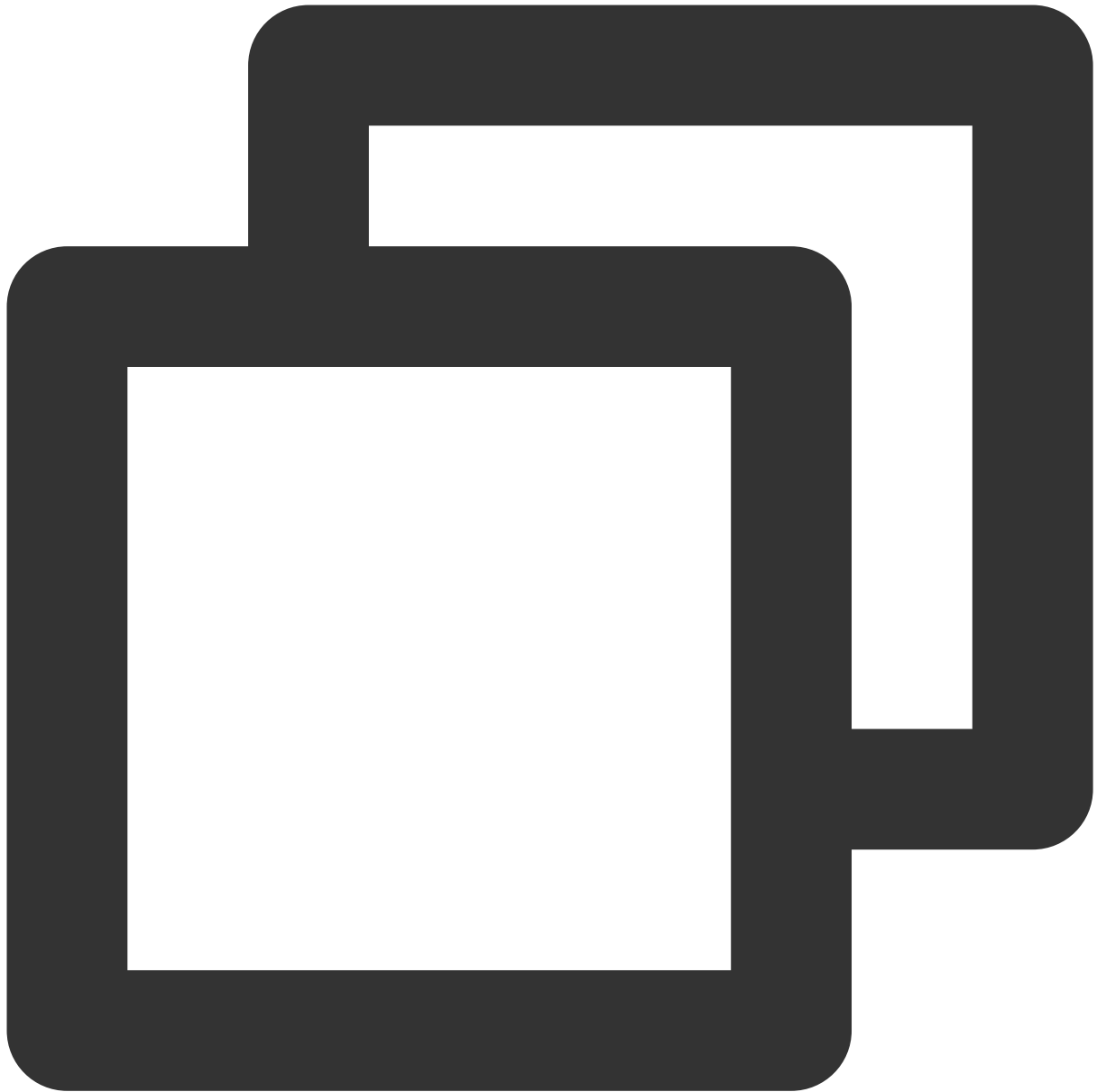
```
Terraform has made some changes to the provider dependency selections recorded in the .terraform.lock.hcl file. Review those changes and commit them to your version control system if they represent changes you intended to make.
```

```
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.
```

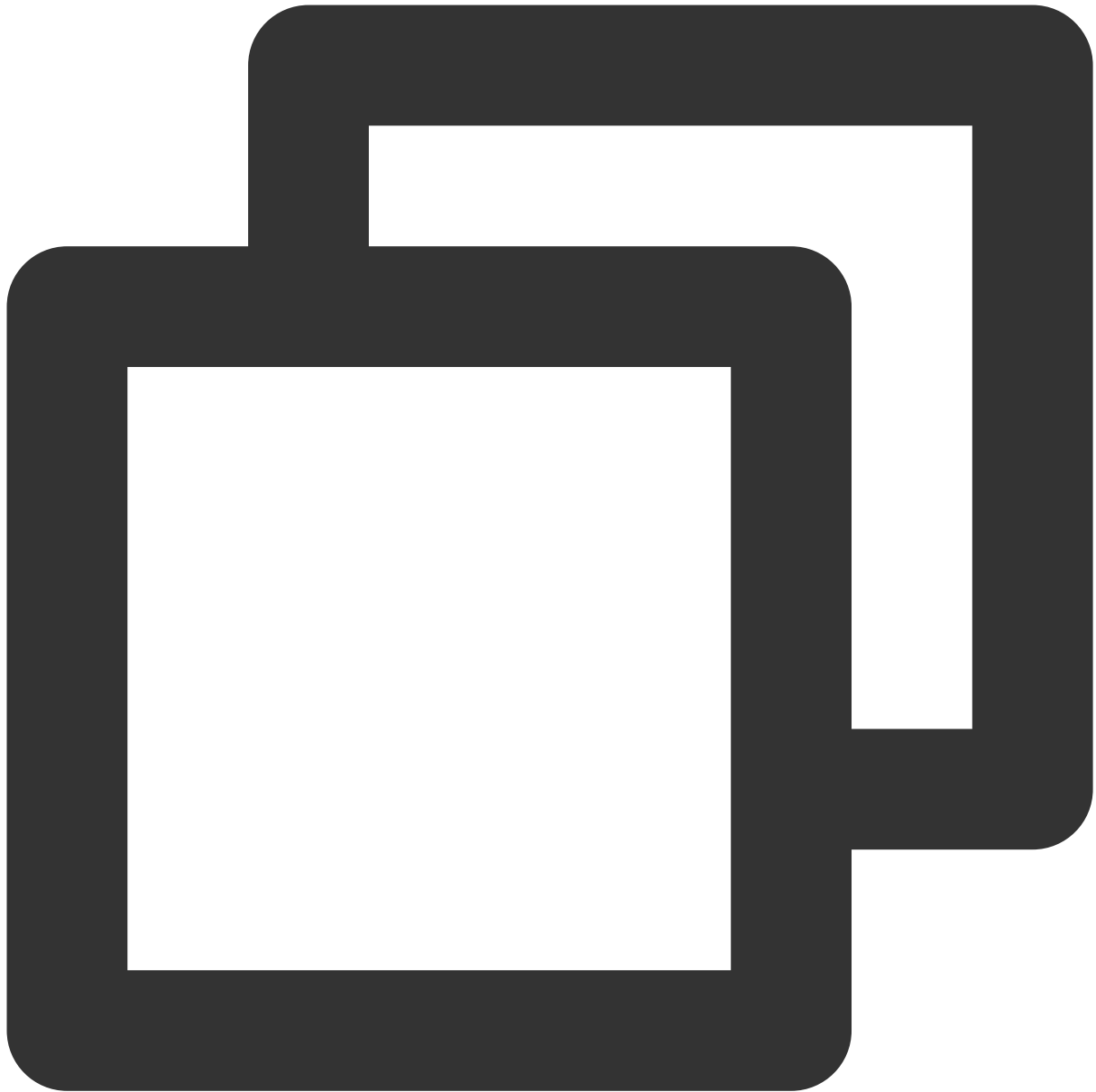
```
If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.
```

5. Run the following command to view the execution plan and display the details of the resource to be created.



```
terraform plan
```

The response is as follows:



```
→ terraform_workspace terraform plan
```

Terraform used the selected providers to generate the following execution plan. Resources 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"
```



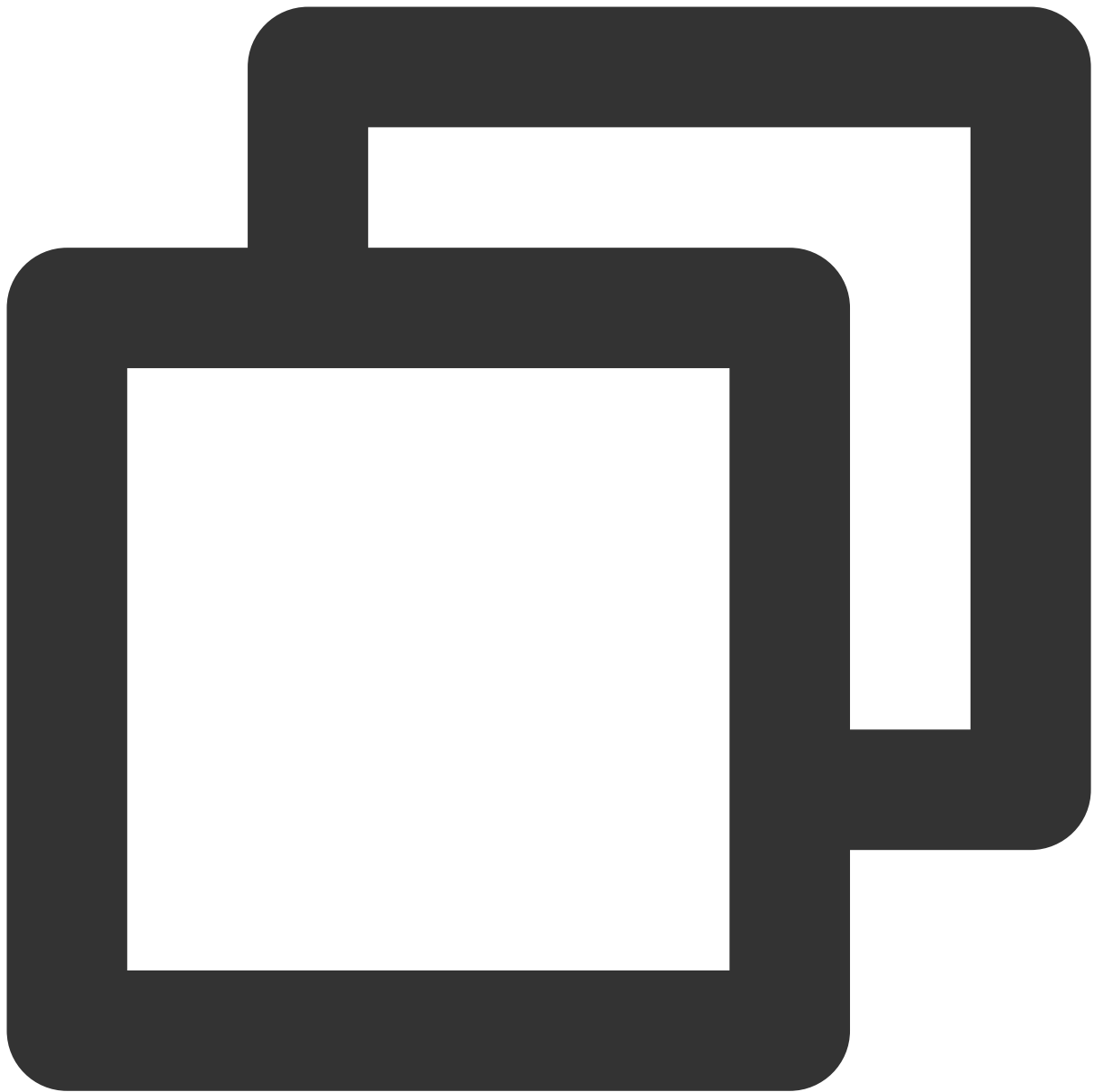
```
+ create_time          = (known after apply)
+ default_route_table_id = (known after apply)
+ dns_servers          = [
  + "119.29.29.29",
  + "8.8.8.8",
]
+ id                   = (known after apply)
+ is_default           = (known after apply)
+ is_multicast         = false
+ name                 = "ci-temp-test-updated"
+ tags                 = {
  + "test" = "test"
}
}
```

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

---

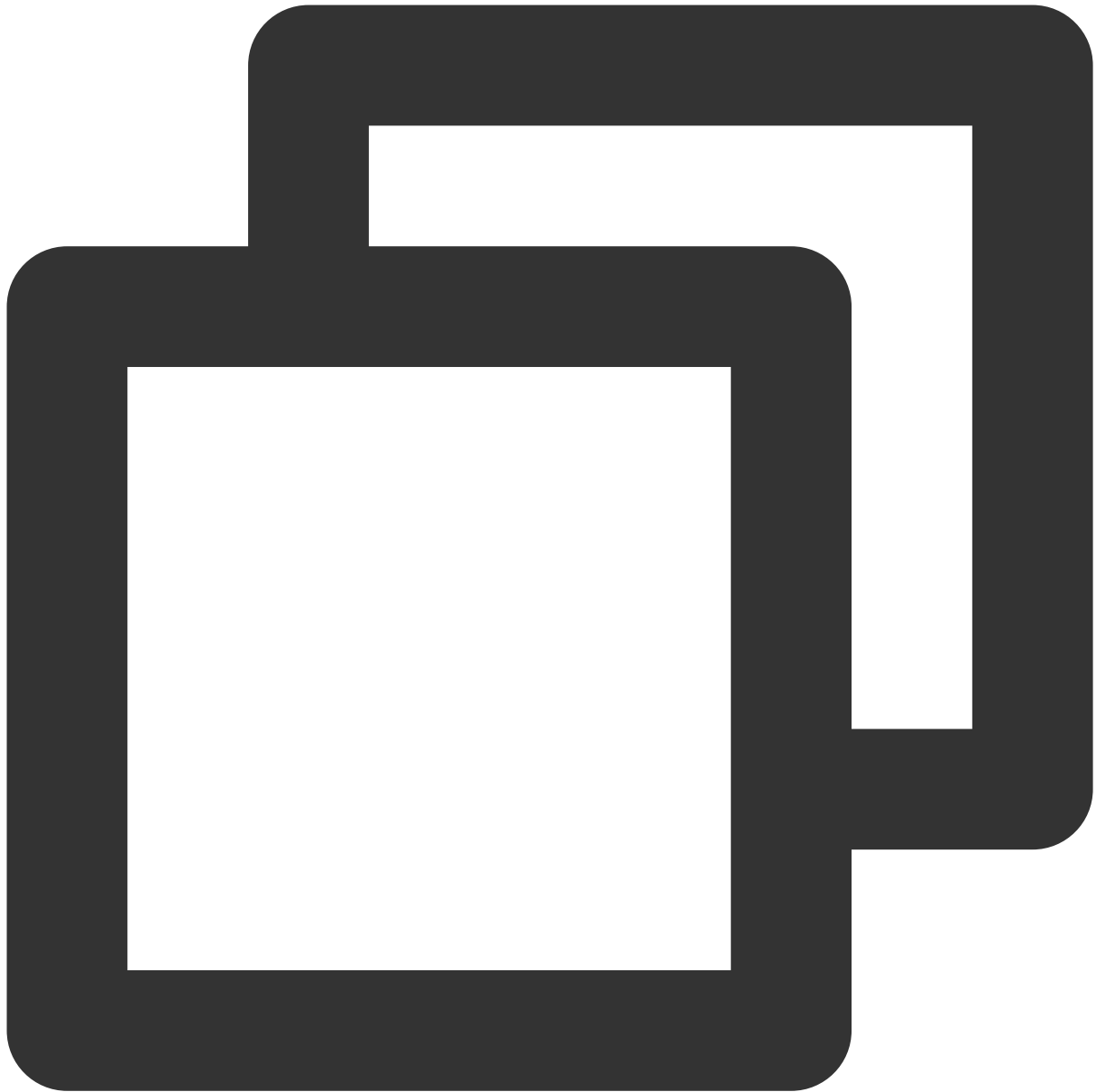
Note: You didn't use the `-out` option to save this plan, so Terraform can't guarantee actions if you run `"terraform apply"` now.

6. Run the following command to create the resource.



```
terraform apply
```

Enter `yes` as prompted to create the resource. The response is as follows:



```
→ terraform_workspace terraform apply
```

Terraform used the selected providers to generate the following execution plan. Resources 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]

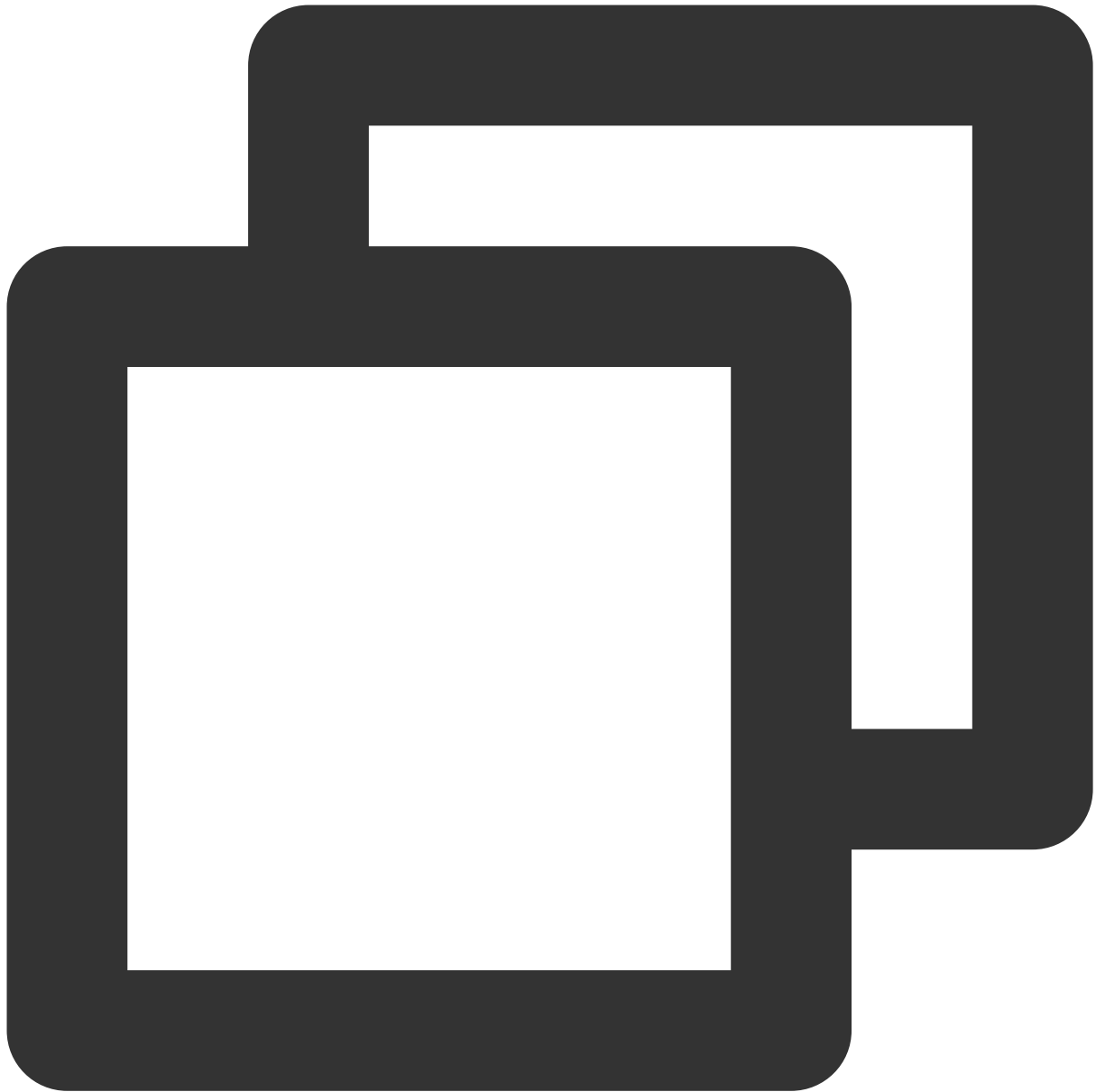
tencentcloud\_vpc.foo: Creation complete after 13s [id=vpc-07mx4yfd]

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

After execution, you can view the created resource in the Tencent Cloud console.

## 7. (Optional) Update the resource.

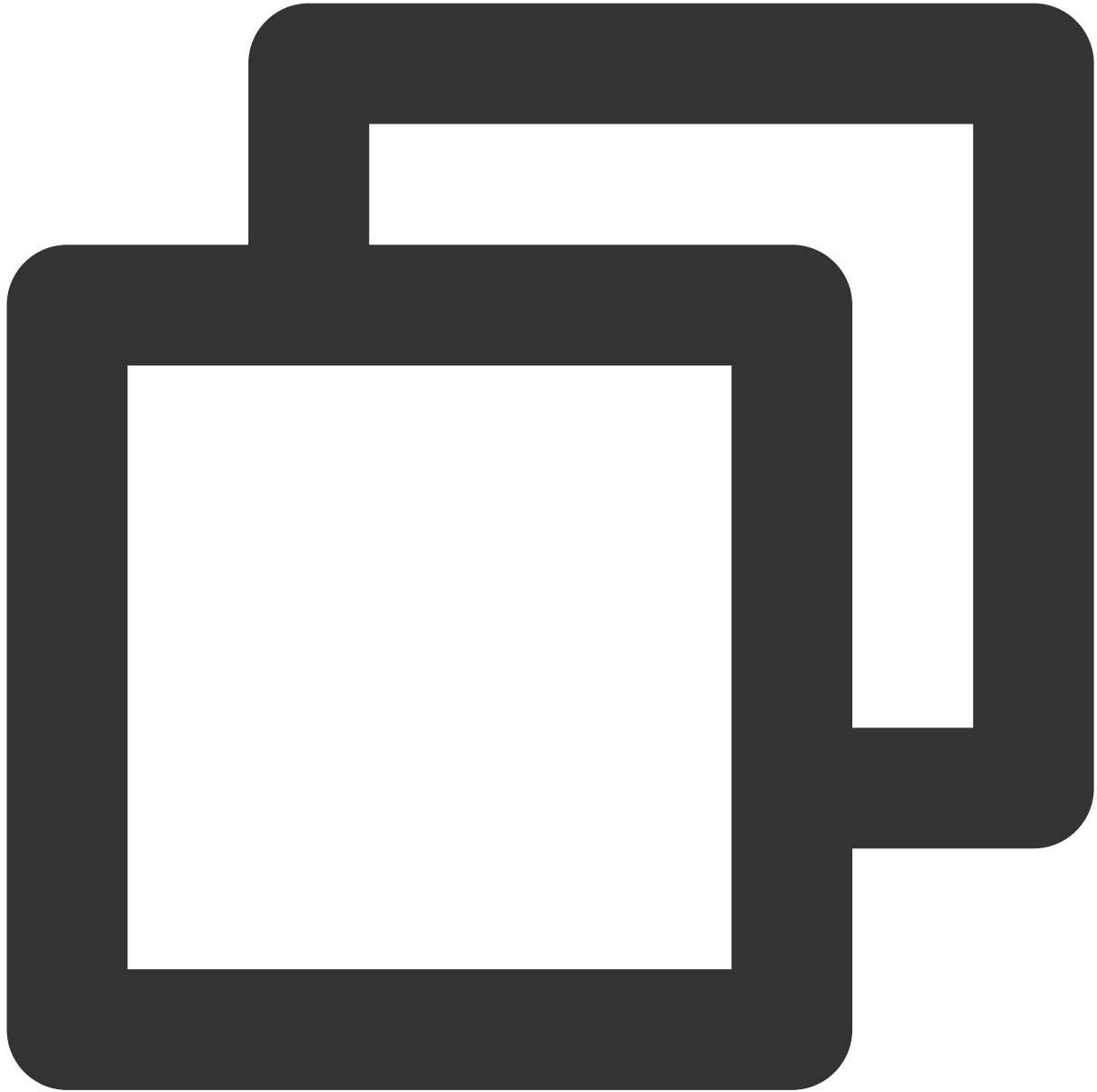
If you change the resource configuration to the following information:



```
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"
  }
}
```

Run the `terraform plan` command to update the plan. The response is as follows:



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

```
Terraform used the selected providers to generate the following execution plan. Resources to be updated are shown below.
~ update in-place
```

```
Terraform will perform the following actions:
```

```
# tencentcloud_vpc.foo will be updated in-place
```

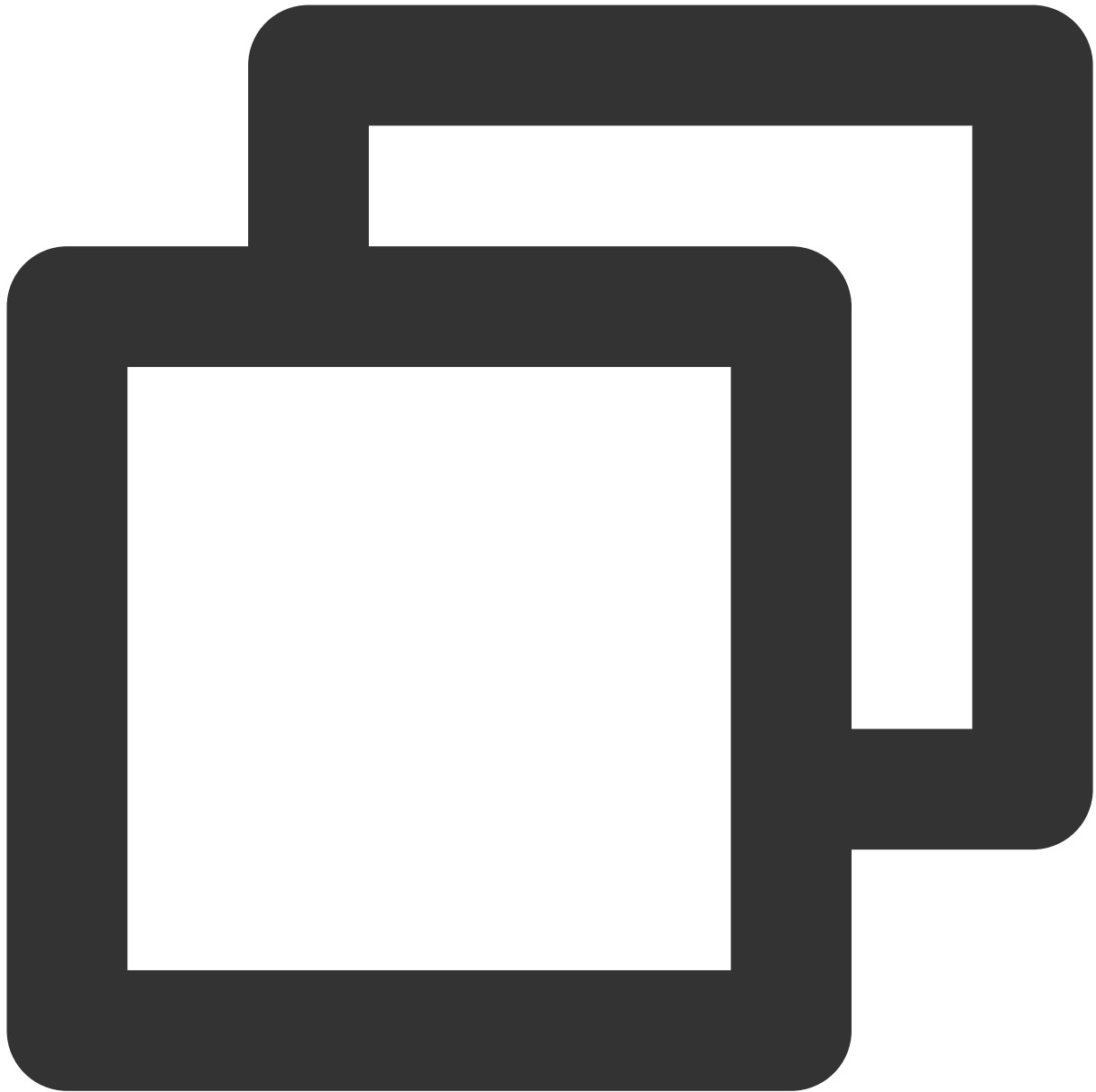
```
~ resource "tencentcloud_vpc" "foo" {
  id          = "vpc-jhmdf9q9"
  ~ name      = "ci-temp-test-updated" -> "ci-temp-test-updated2"
  tags       = {
    "test" = "test"
  }
  # (6 unchanged attributes hidden)
}
```

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 now.

Run the `terraform apply` command to create the resource with the updated data. The following information is returned:



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

Terraform used the selected providers to generate the following execution plan. Resources that will be updated in-place are marked with a tilde (~).

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.

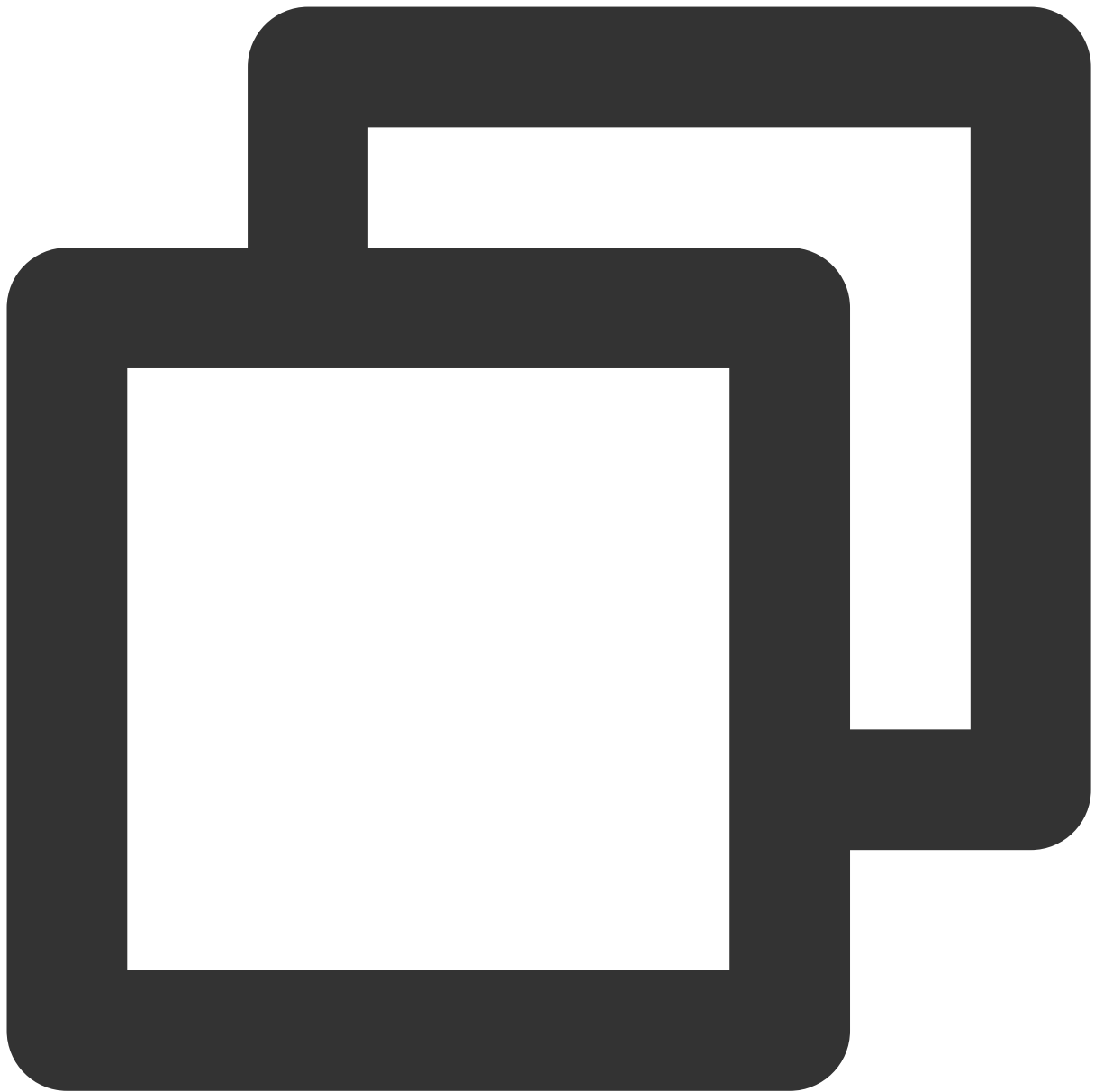
Enter a value: yes

tencentcloud\_vpc.foo: Modifying... [id=vpc-jhmdf9q9]

tencentcloud\_vpc.foo: Modifications complete after 1s [id=vpc-jhmdf9q9]

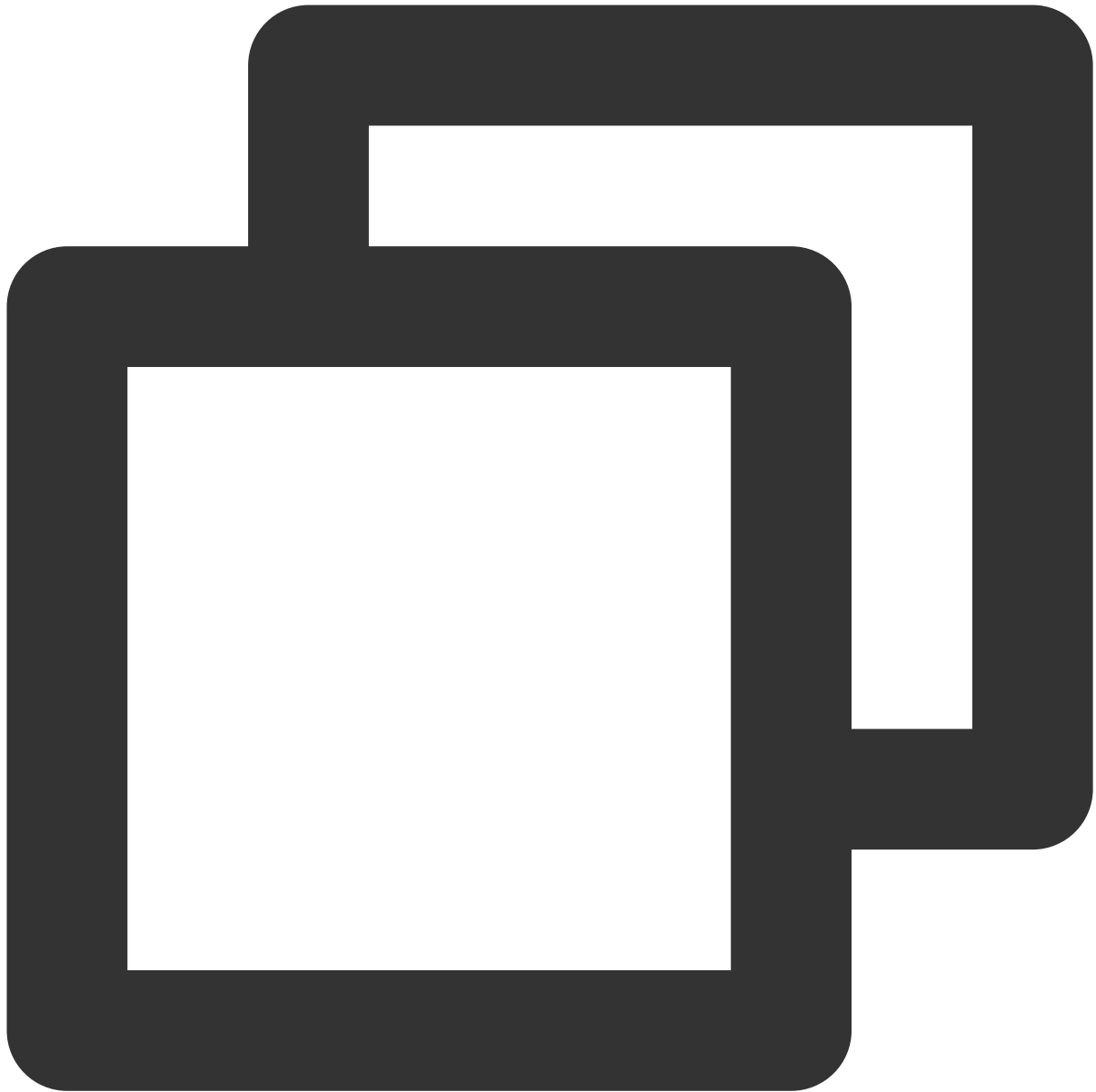
Apply complete! Resources: 0 added, 1 changed, 0 destroyed.

8. You can run the command below to terminate the resource as needed.



```
terraform destroy
```

The returned information is as follows:



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

Terraform used the selected providers to generate the following execution plan. Resources 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.