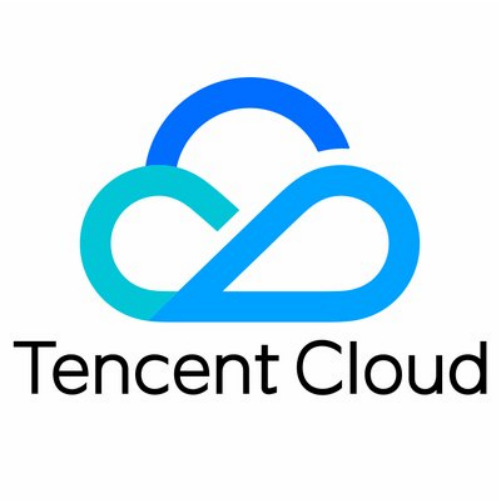
Tencent Infrastructure Automation for Terraform FAQs

Product Documentation



Contents

FAQs

Signature Error

Last updated：2023-03-07 10:35:48

Symptom

The following error message is returned when a provider is downloaded or updated:

```
Initializing the backend...

Initializing provider plugins...
- Checking for available provider plugins on https://releases.hashicorp.com...

Error installing provider "tencentcloud": openpgp: signature made by unknown enti
ty.

Terraform analyses the configuration and state and automatically downloads
plugins for the providers used. However, when attempting to download this
plugin an unexpected error occured.

This may be caused if for some reason Terraform is unable to reach the
plugin repository. The repository may be unreachable if access is blocked
by a firewall.

If automatic installation is not possible or desirable in your environment,
you may alternatively manually install plugins by downloading a suitable
distribution package and placing the plugin's executable file in the
following directory:
terraform.d/plugins/darwin_amd64
```

Locating the Issue

The update failed due to a lower Terraform version as illustrated at the official website.

Troubleshooting the Issue

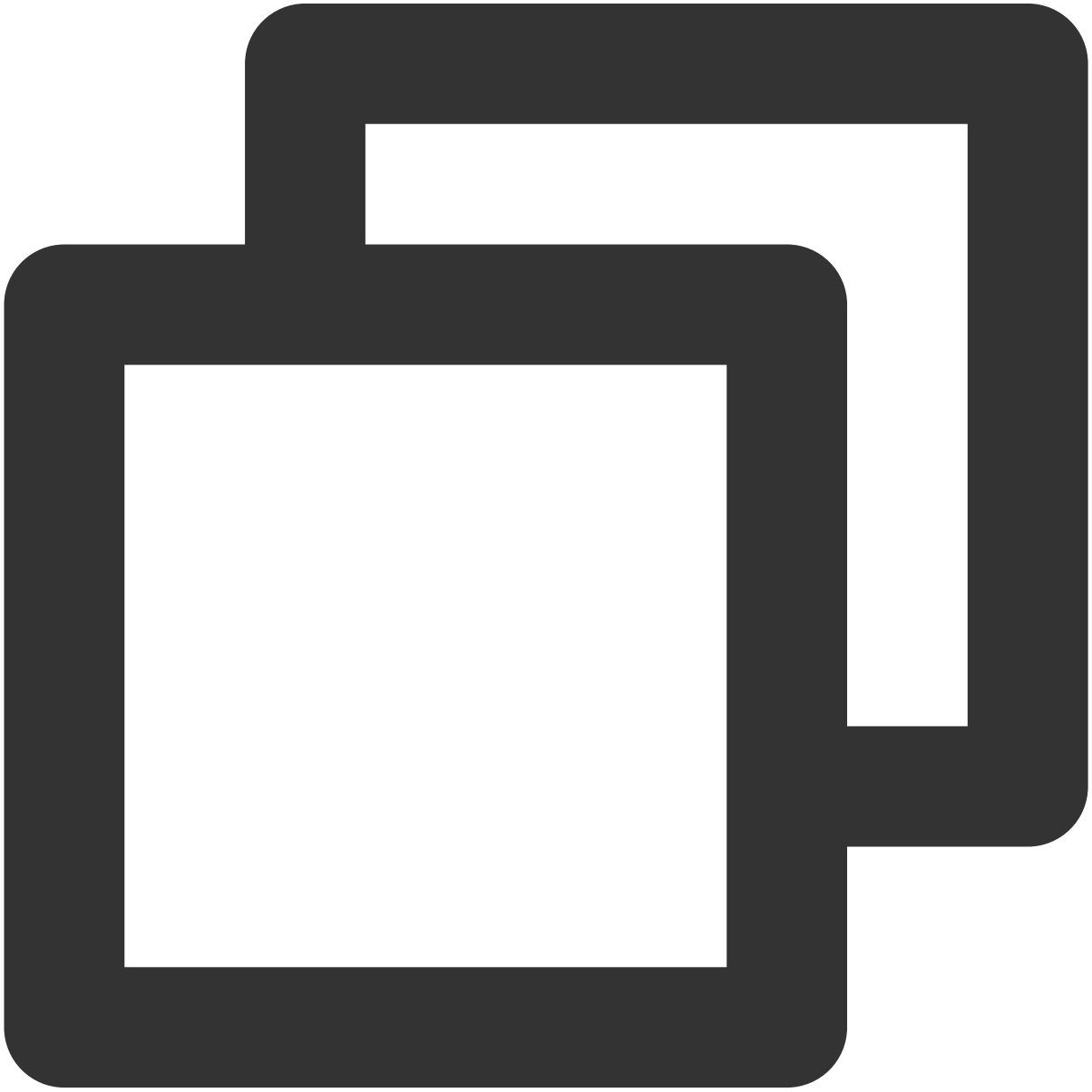
Upgrade Terraform to 0.11.15 or later.

Init Acceleration

Last updated：2023-05-30 10:08:19

Symptom

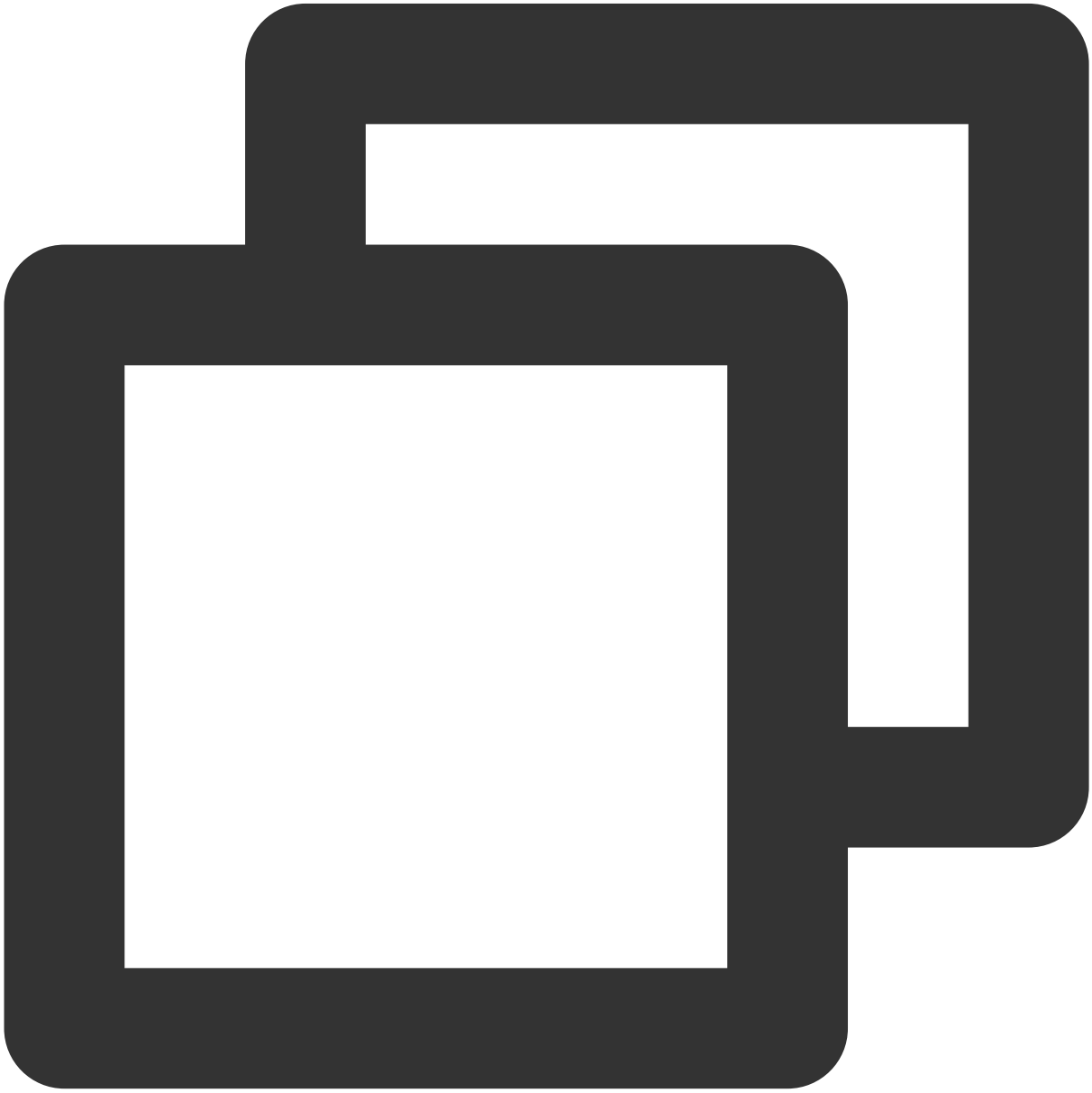
The following error message is returned when a provider is downloaded or updated:



```
Initializing the backend...
```

```
Initializing provider plugins...
- Finding latest version of tencentcloudstack/tencentcloud...

| Error: Failed to install provider
|
| Error while installing tencentcloudstack/tencentcloud v1.78.5: could not query pr
| "https://github.com/tencentcloudstack/terraform-provider-tencentcloud/releases/do
|
```

Or the download is slow:



```
Initializing the backend...

Initializing provider plugins...
- Finding latest version of tencentcloudstack/tencentcloud...
|
| Error: Failed to query available provider packages
|
| Could not retrieve the list of available versions for provider tencentcloudstack/
| (Client.Timeout exceeded while awaiting headers)
|
```

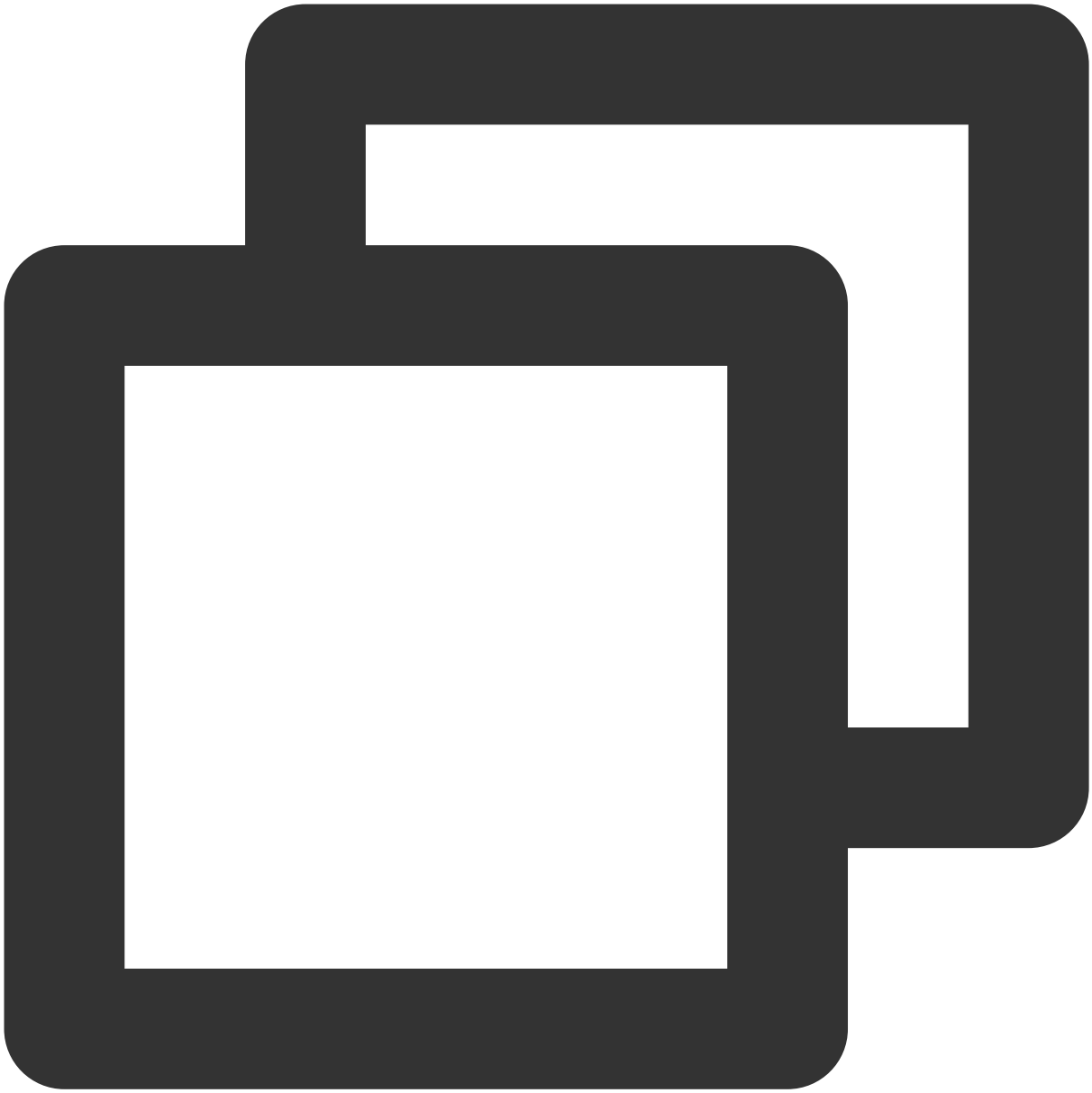
Troubleshooting the Issue

The `registry.terraform.io` default image source of Terraform is deployed outside the Chinese mainland, which means image pull from the Chinese mainland may be slow or fail due to network issues. In this case, you can use the image source provided by Tencent Cloud.

Directions

Creating the Terraform CLI configuration file

Create the `.terraformrc` or `terraform.rc` configuration file as needed and put it together with other configuration files in the same folder. The path varies by server operating system:
In the Windows system, the file must be named `terraform.rc` and put in the `%APPDATA%` directory of the user. The physical path of the directory varies by Windows version and system configuration and can be located through `$env:APPDATA` in PowerShell.
In other operating systems, the file must be named `.terraformrc` and put in the root directory of the user. You can also use the `TF_CLI_CONFIG_FILE` environment variable to specify the path of the Terraform CLI configuration file. Any file of this type should follow the `*.tfrc` naming convention. For more information, see CLI Configuration File.
Taking macOS as an example, create the `.terraformrc` file in the root directory of the user as follows:

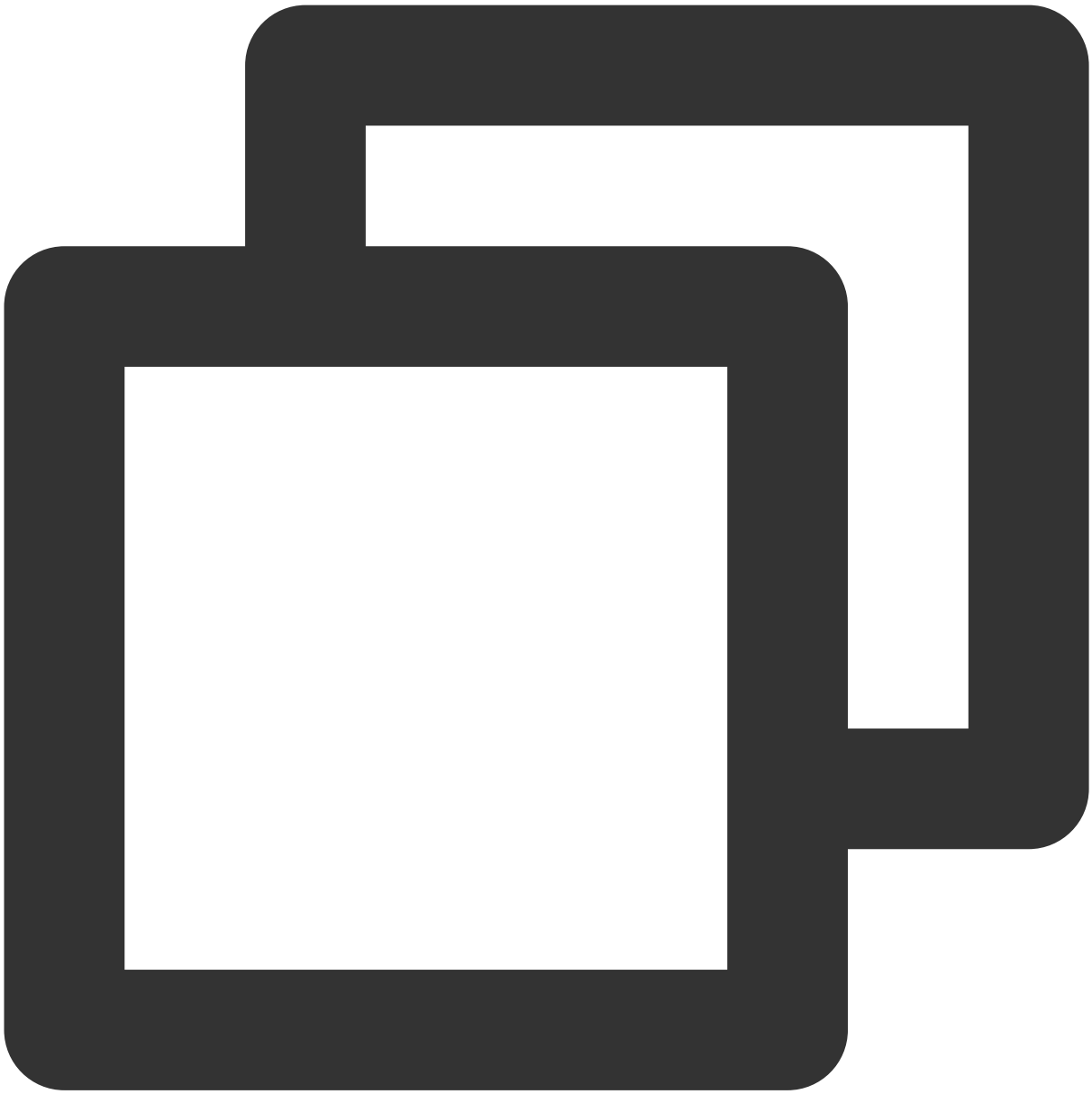


```
provider_installation {
    network_mirror {
        url = "https://mirrors.tencent.com/terraform/"
    }
}
```

Running `terraform init`

In the directory of the Terraform configuration file, run the `terraform init` command to initialize the configuration.

In this step, Terraform will automatically check the `provider` field in the configuration file and download the latest module and plugin. If the following message is printed, the initialization is successful.



```
Initializing the backend...

Initializing provider plugins...
- Finding latest version of tencentcloudstack/tencentcloud...
- Installing tencentcloudstack/tencentcloud v1.78.5...
- Installed tencentcloudstack/tencentcloud v1.78.5 (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 cal
│   - tencentcloudstack/tencentcloud
│
│ The current .terraform.lock.hcl file only includes checksums for darwin_amd64, so
│
│ 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.
```

Enabling Log Tracking

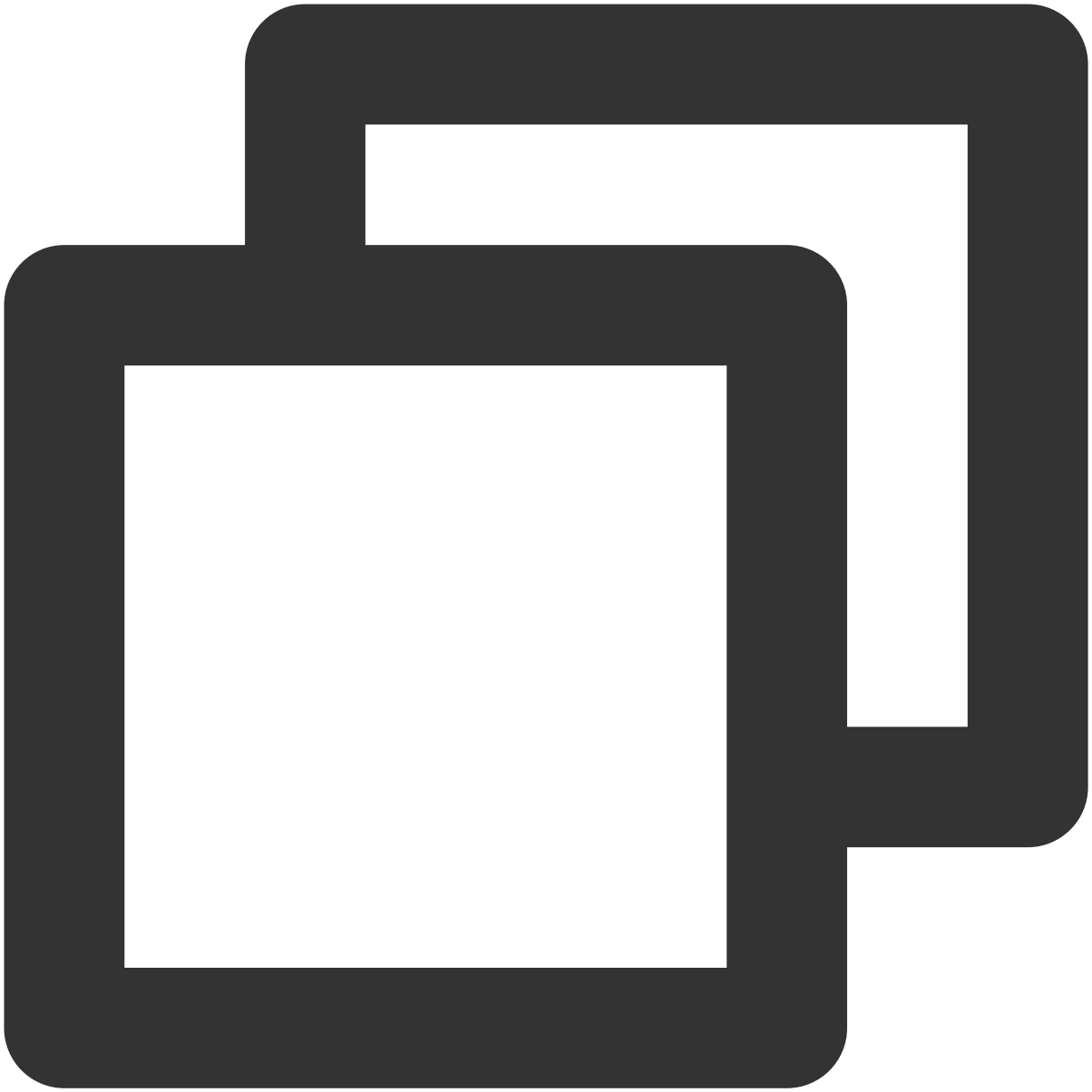
Last updated：2023-05-29 15:22:17

Scenario

This document describes how to enable local log tracking to get more detailed logs for self-check and assistance with ticket processing.

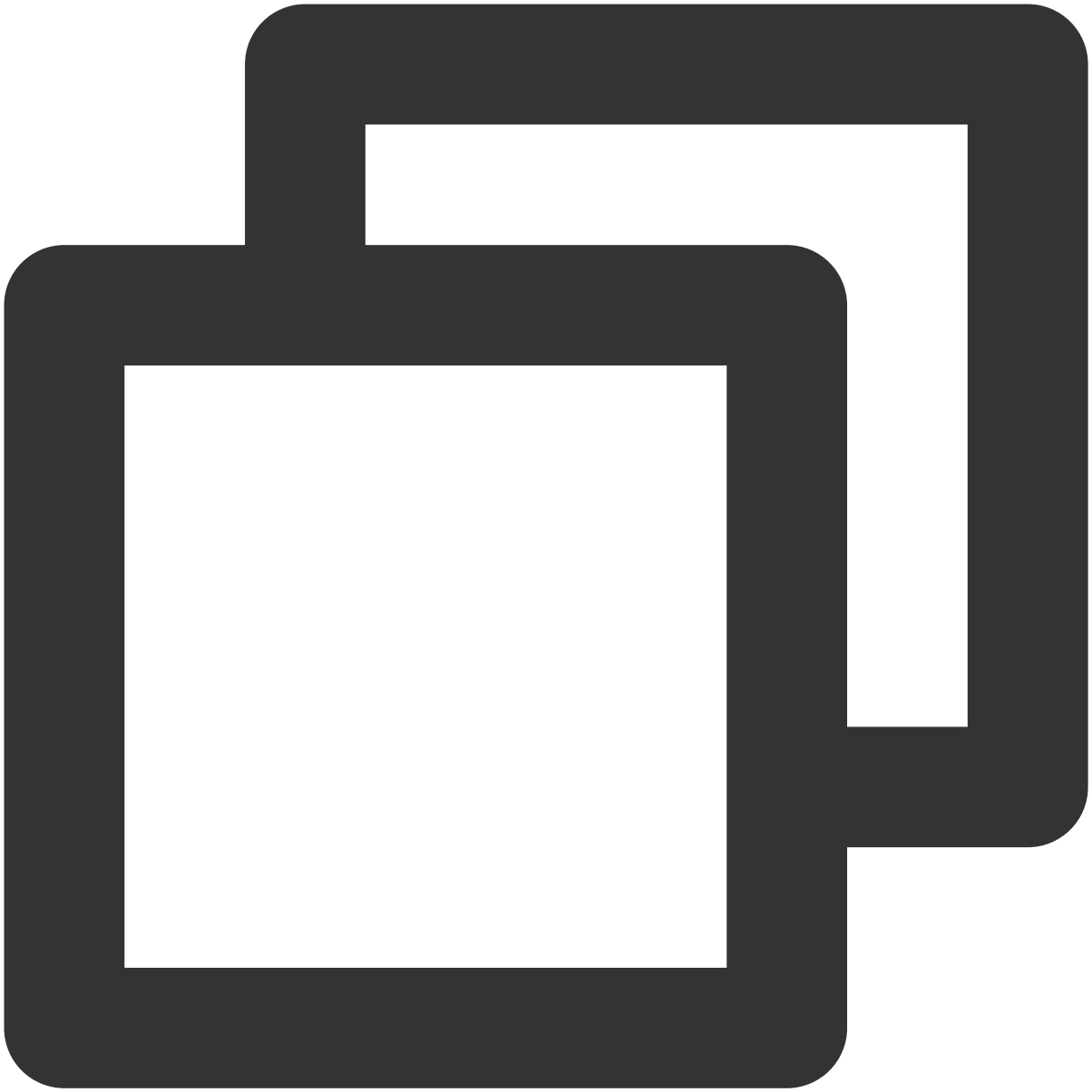
Directions

1. Before running `terraform apply` on the CLI, you can enable local log tracking with the following command:



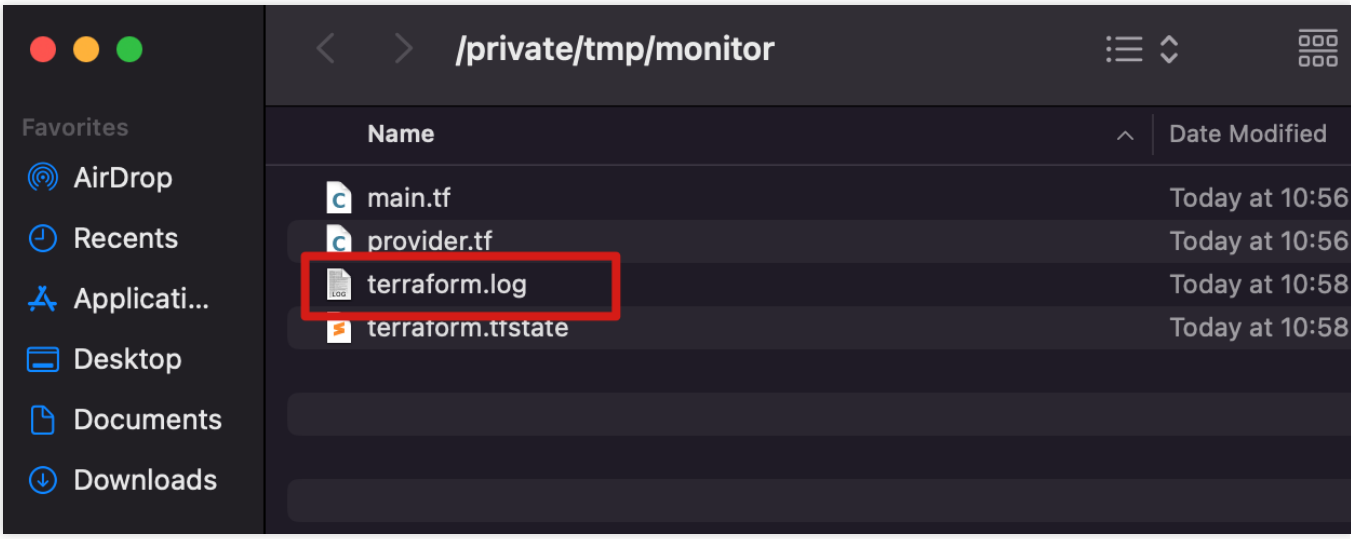
```
export TF_LOG=DEBUG
export TF_LOG_PATH=./terraform.log
```

2. Run the following command:



```
terraform apply/destroy
```

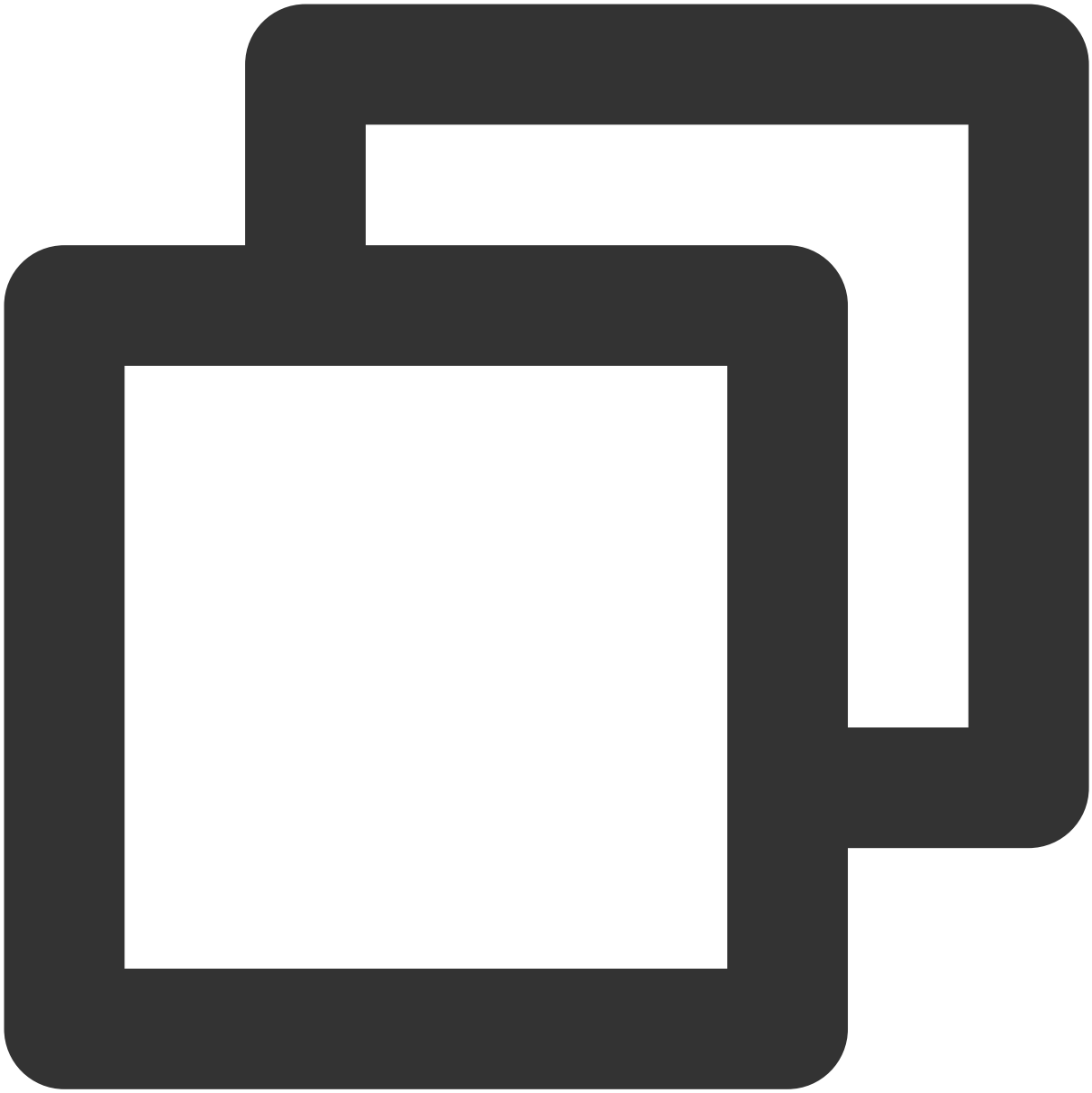
After execution, you can see that the Terraform local folder generates a `terraform.log` file, which records the log output as defined by TencentCloud Provider.



Example

The following describes an execution error, along with the problem analysis and locating process.

In this example, a K8s cluster is created and an existing CVM instance is mounted to it as a node.



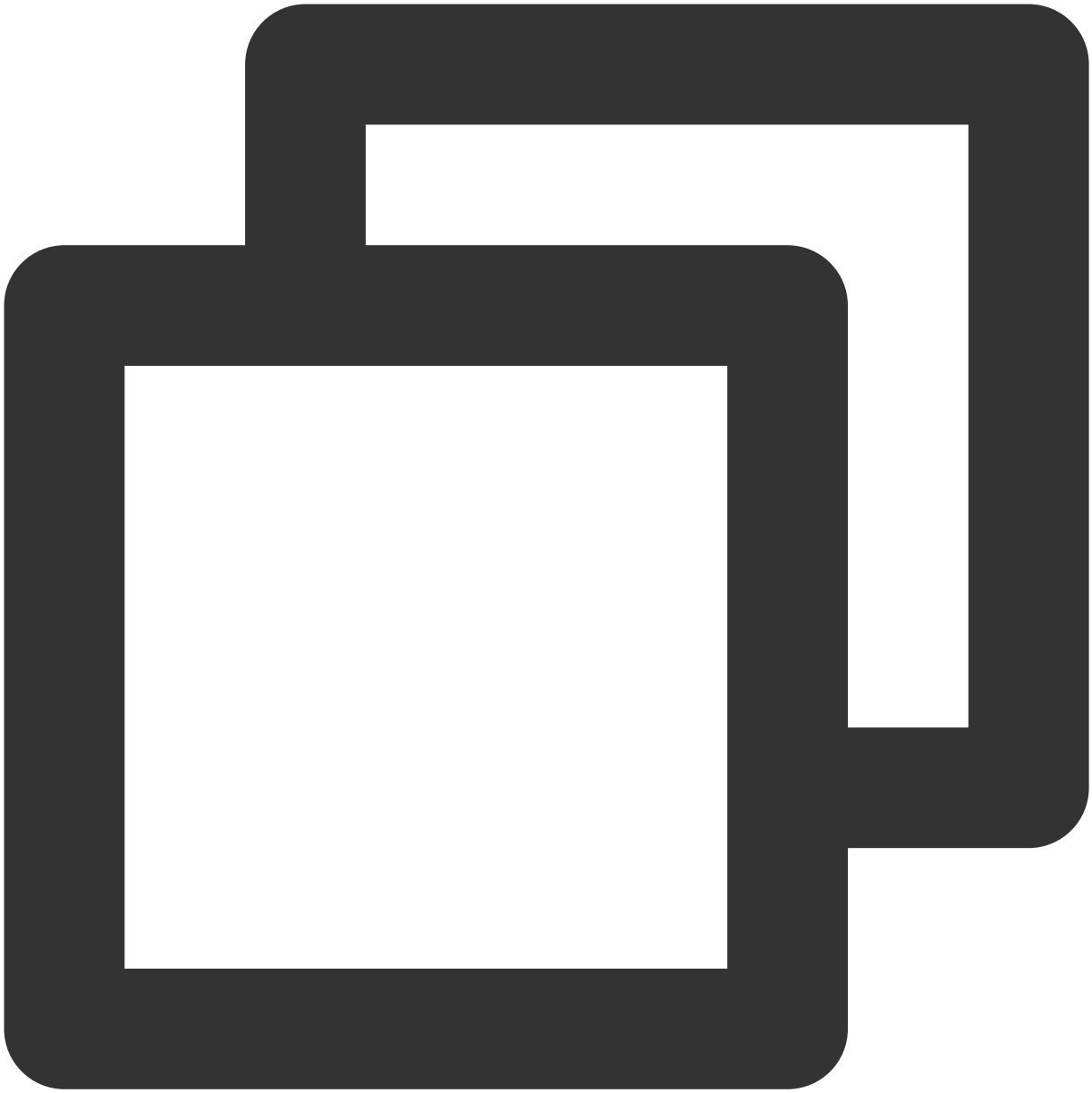
```
➜ terraform apply

2021/12/09 17:53:02 [WARN] Log levels other than TRACE are currently unreliable, an
  Use TF_LOG=TRACE to see Terraform's internal logs.
  ----
data.tencentcloud_instance_types.default: Refreshing state...
data.tencentcloud_cbs_storages.storages: Refreshing state...
data.tencentcloud_vpc_subnets.vpc2: Refreshing state...
data.tencentcloud_images.default: Refreshing state...
data.tencentcloud_vpc_subnets.vpc: Refreshing state...
An execution plan has been generated and is shown below.
```

```
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # tencentcloud_kubernetes_cluster.managed_cluster will be created
  + resource "tencentcloud_kubernetes_cluster" "managed_cluster" {
      + certification_authority      = (known after apply)
      + claim_expired_seconds        = 300
      + cluster_as_enabled           = false
      + cluster_cidr                 = "10.1.0.0/16"
      + cluster_deploy_type          = "MANAGED_CLUSTER"
      + cluster_desc                 = "test cluster desc"
      + cluster_external_endpoint    = (known after apply)
      + cluster_internet             = false
      + cluster_intranet             = false
      + cluster_ipvs                 = true
      + cluster_max_pod_num          = 32
      + cluster_max_service_num      = 32
      + cluster_name                 = "keep"
      + cluster_node_num             = (known after apply)
      + cluster_os                   = "ubuntu16.04.1 LTSx86_64"
      + cluster_os_type              = "GENERAL"
      + cluster_version              = "1.10.5"
      + container_runtime            = "docker"
      + deletion_protection          = false
      + domain                       = (known after apply)
      + id                           = (known after apply)
      + ignore_cluster_cidr_conflict = false
      + is_non_static_ip_mode        = false
      + kube_config                  = (known after apply)
      + network_type                 = "GR"
      + node_name_type               = "lan-ip"
      + password                     = (known after apply)
      + pgw_endpoint                 = (known after apply)
      + security_policy              = (known after apply)
      + user_name                    = (known after apply)
      + vpc_id                       = "vpc-h70b6b49"
      + worker_instances_list        = (known after apply)

      + worker_config {
          + availability_zone                      = "ap-guangzhou-3"
          + count                                  = 1
          + enhanced_monitor_service               = false
          + enhanced_security_service              = false
          + instance_charge_type                   = "POSTPAID_BY_HOUR"
          + instance_charge_type_prepaid_period    = 1
```

```
          + instance_charge_type_prepaid_renew_flag = "NOTIFY_AND_MANUAL_RENEW"
          + instance_name                           = "sub machine of tke"
          + instance_type                           = "S1.SMALL1"
          + internet_charge_type                    = "TRAFFIC_POSTPAID_BY_HOUR"
          + internet_max_bandwidth_out              = 100
          + password                                = (sensitive value)
          + public_ip_assigned                      = true
          + subnet_id                               = "subnet-1uwh63so"
          + system_disk_size                        = 60
          + system_disk_type                        = "CLOUD_SSD"
          + user_data                               = "dGVzdA=="

          + data_disk {
              + disk_size = 50
              + disk_type             = "CLOUD_PREMIUM"
            }
        }
    }

  # tencentcloud_kubernetes_cluster_attachment.test_attach will be created
  + resource "tencentcloud_kubernetes_cluster_attachment" "test_attach" {
      + cluster_id      = (known after apply)
      + hostname        = "user"
      + id              = (known after apply)
      + instance_id     = "ins-lmnl6t1g"
      + labels          = {
          + "test1" = "test1"
          + "test2" = "test2"
        }
      + password        = (sensitive value)
      + security_groups = (known after apply)
      + state           = (known after apply)

      + worker_config {
          + docker_graph_path = "/var/lib/docker"
          + is_schedule       = true

          + data_disk {
              + auto_format_and_mount = false
              + disk_size             = 50
              + disk_type             = "CLOUD_PREMIUM"
            }
        }
    }

Plan: 2 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_kubernetes_cluster.managed_cluster: Creating...

Error: [TencentCloudSDKError] Code=InternalError.CidrConflictWithOtherCluster, Mess

  on main.tf line 424, in resource "tencentcloud_kubernetes_cluster" "managed_clust
 424: resource "tencentcloud_kubernetes_cluster" "managed_cluster" {
```

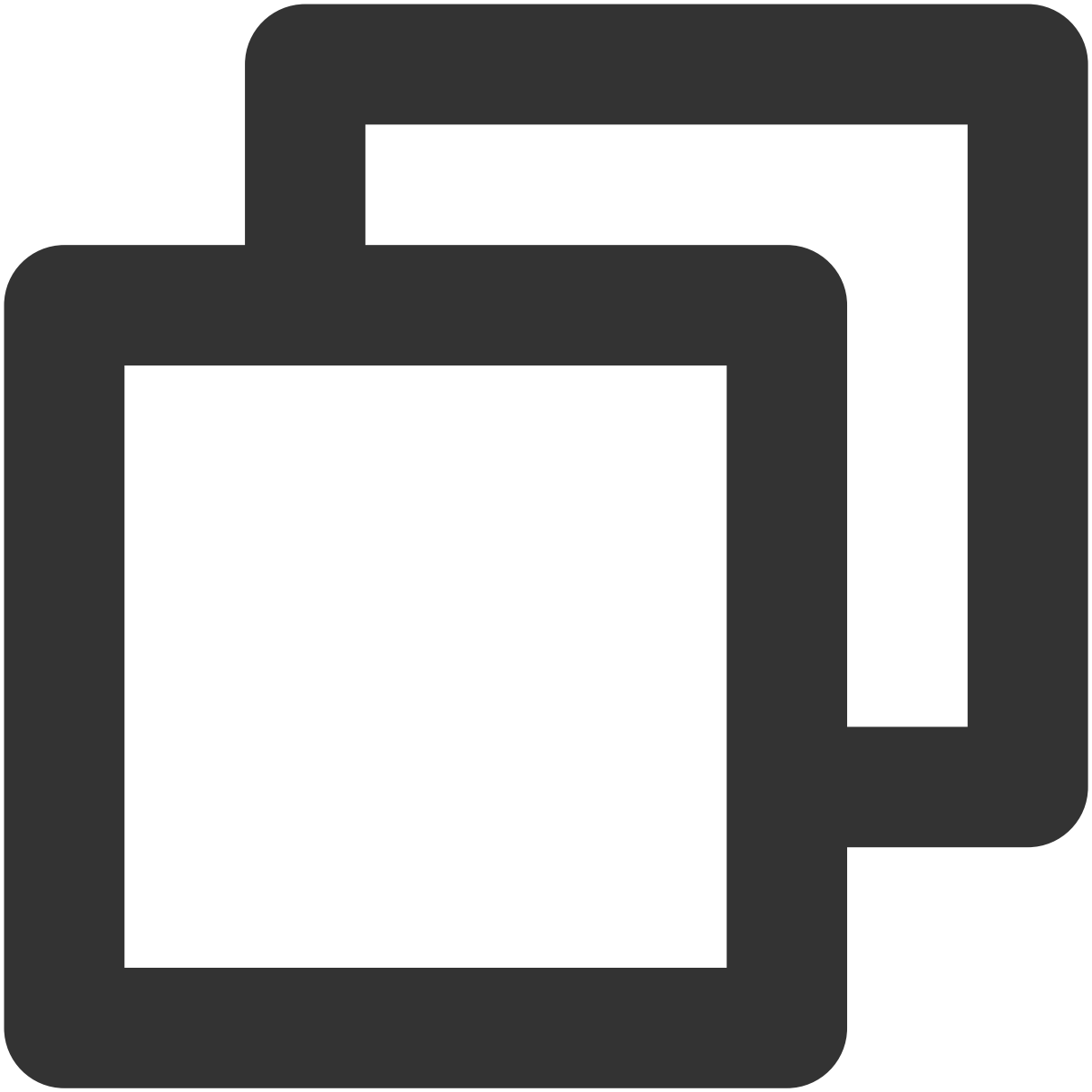
The CLI returns the following error:



```
[TencentCloudSDKError] Code=InternalError.CidrConflictWithOtherCluster, Message=Das
```

Problem analysis and locating:

1. Find `requestId: d7dfb178-f081-480a-9bc3-89efc5fb1db5` .
2. Open `terraform.log` , search for the `RequestId` , and find the following context:



```
2021-12-09T17:53:20.222+0800 [DEBUG] plugin.terraform-provider-tencentcloud.exe: 20

_CONFLICT_WITH_OTHER_CLUSTER[cidr 10.1.0.0/16 is conflict with cluster id: cls-1zc0

6 is conflict with cluster id: cls-1zc0kpyo], err : CheckCIDRWithVPCClusters failed

2021-12-09T17:53:20.593+0800 [DEBUG] plugin.terraform-provider-tencentcloud.exe: 20
```

3. Log analysis shows that the problem occurred during the creation of the K8s cluster. Specifically, a conflict existed between the CIDR and another existing K8s cluster.

Note：

If problem locating is difficult because the CLI prompt isn't clear enough or the error doesn't contain the `requestID`, you can send the TF project file, CLI error message, and its resulting `terraform.log` file by submitting a ticket for assistance.

Managing Existing Resource

Last updated：2023-05-29 16:14:31

Scenario

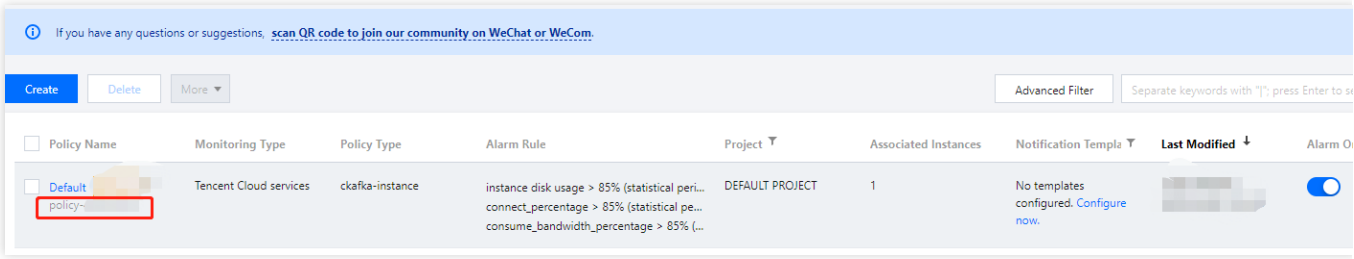
This document describes how to use Terraform to manage resources created in the Tencent Cloud console.

Directions

To take over an existing resource (for example, TencentDB for PostgreSQL alarm policy in this document) on Terraform, you only need to reflect its state in both the source and state files of Terraform.

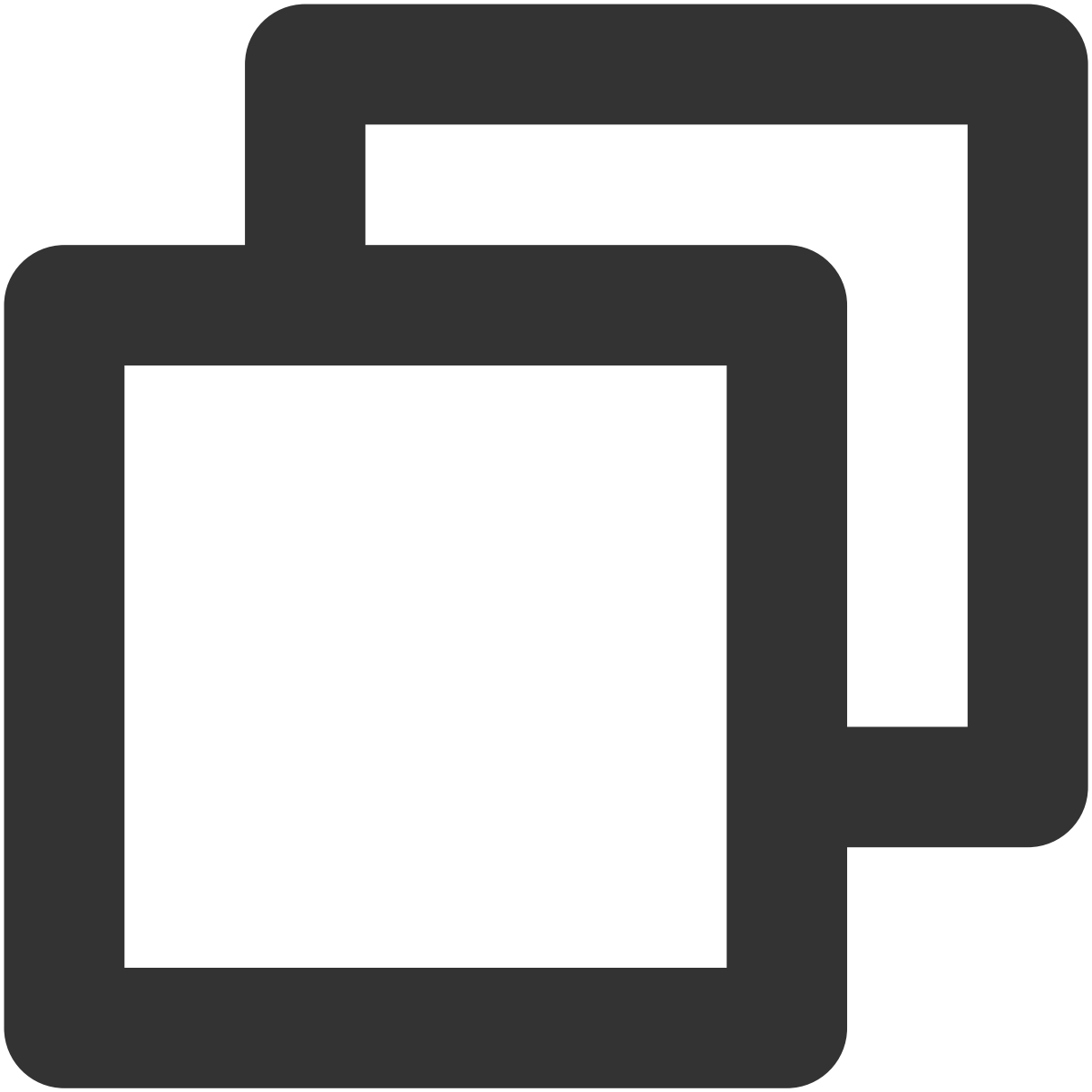
Getting the resource ID

1. Log in to the Cloud Monitor console and select **Alarm Configuration** > **Alarm Policy** on the left sidebar.
2. Find and record the target policy ID as shown below:



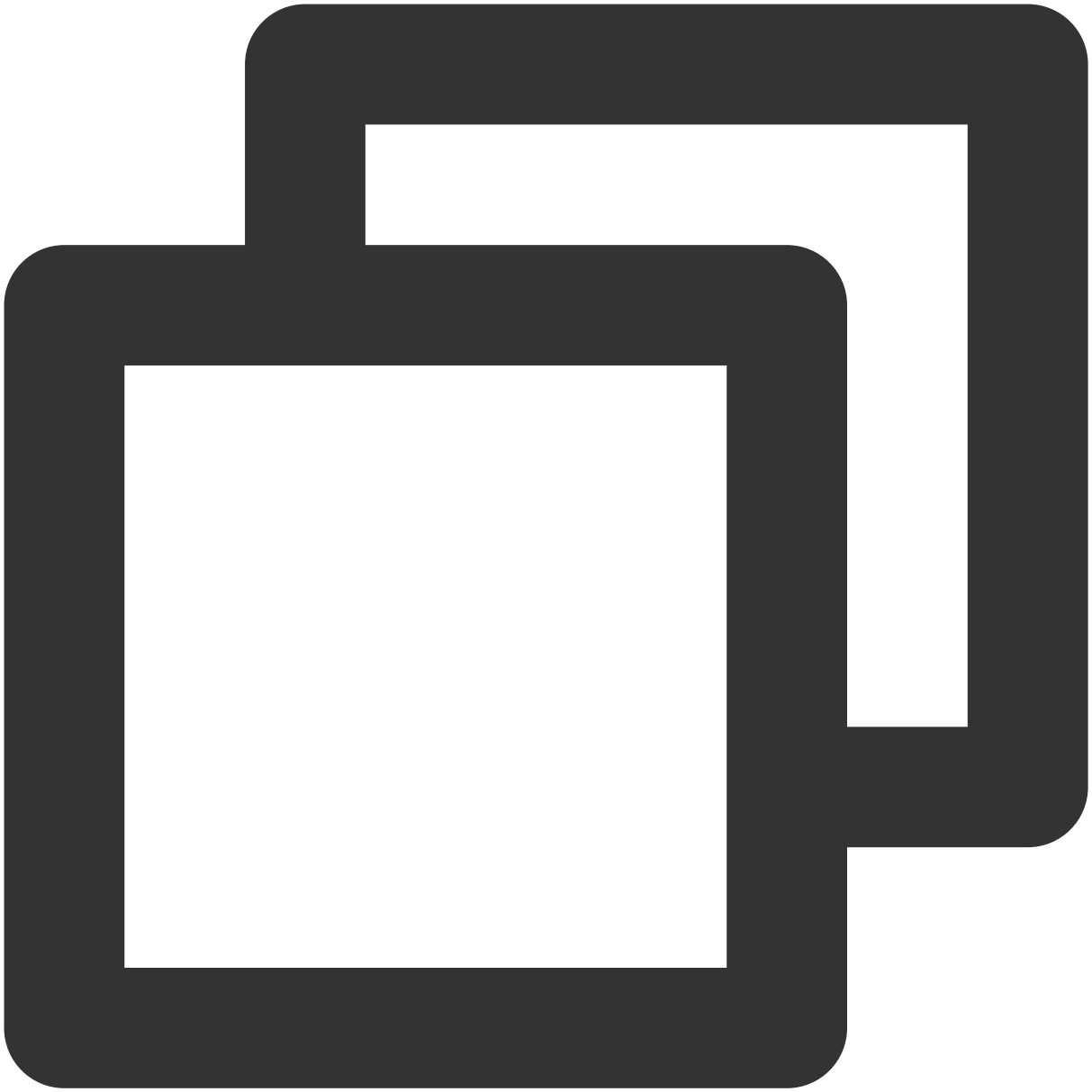
Importing the resource file

1. Go to the Terraform working directory and run the following command to view the `main.tf` content.



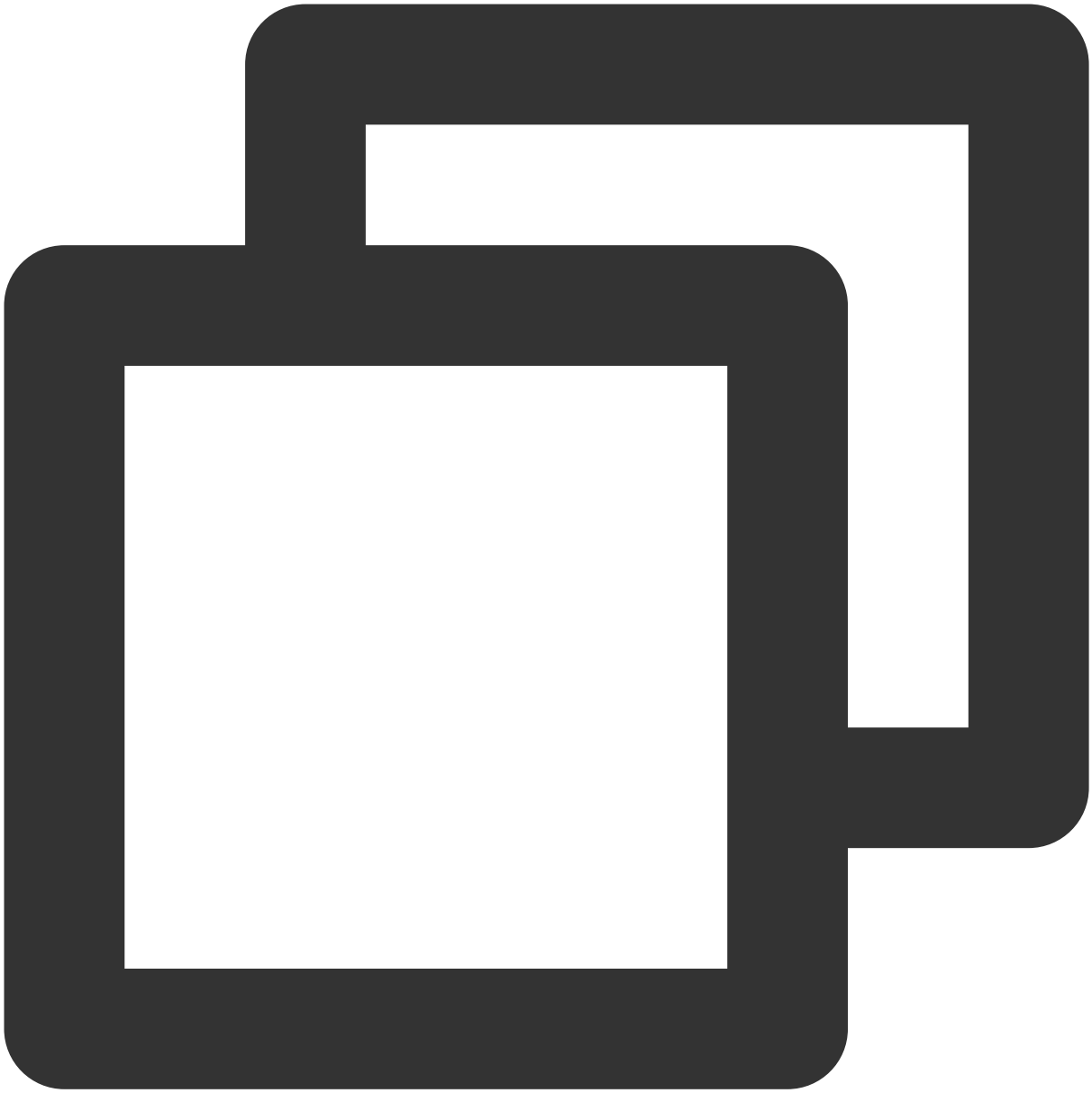
```
tencent-cloud cat main.tf
```

The following information will appear:



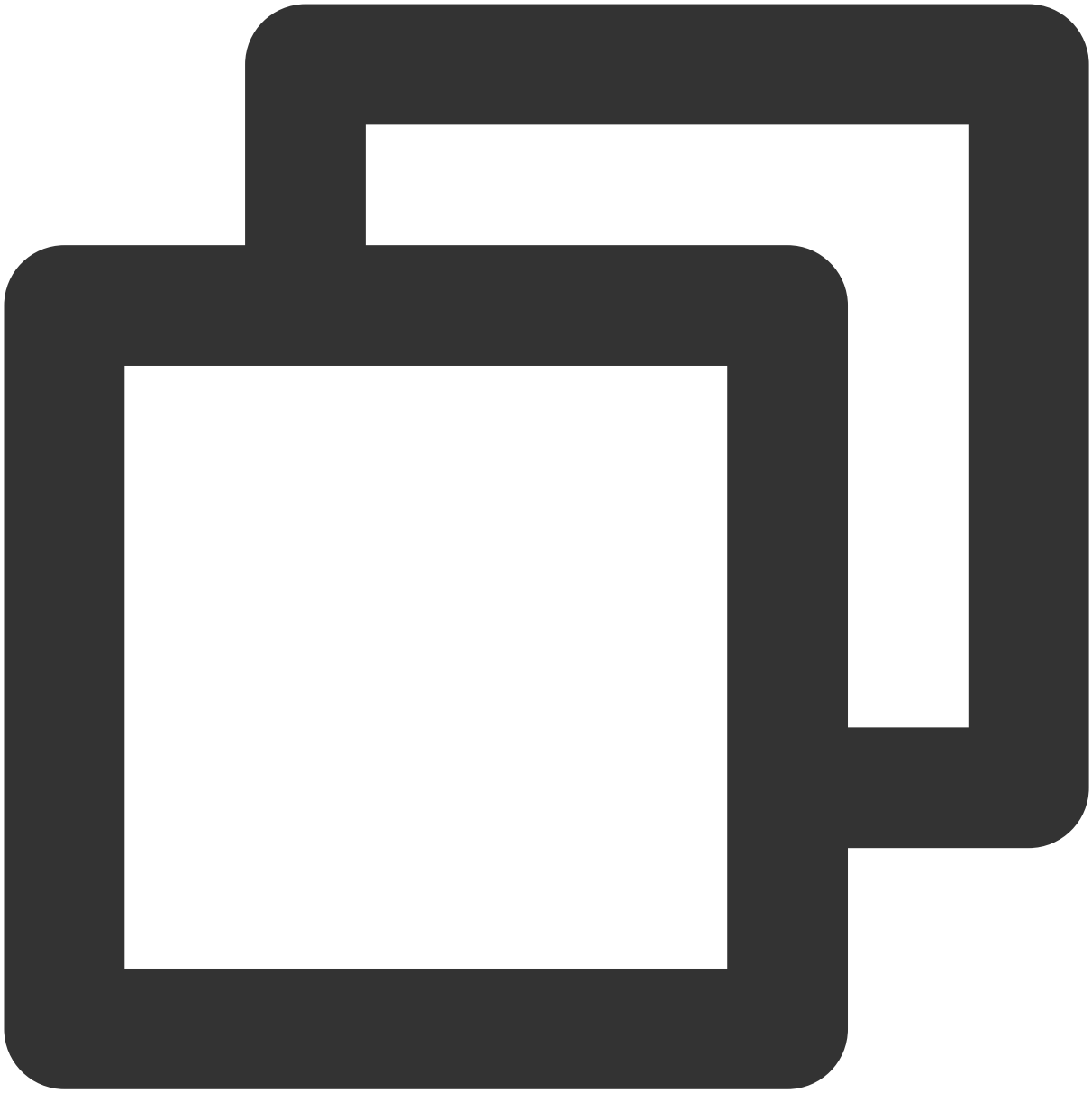
```
resource "tencentcloud_monitor_alarm_policy" "policy" {}
```

2. Run the following command in the directory where the file is located to complete the initialization.



```
terraform init --upgrade
```

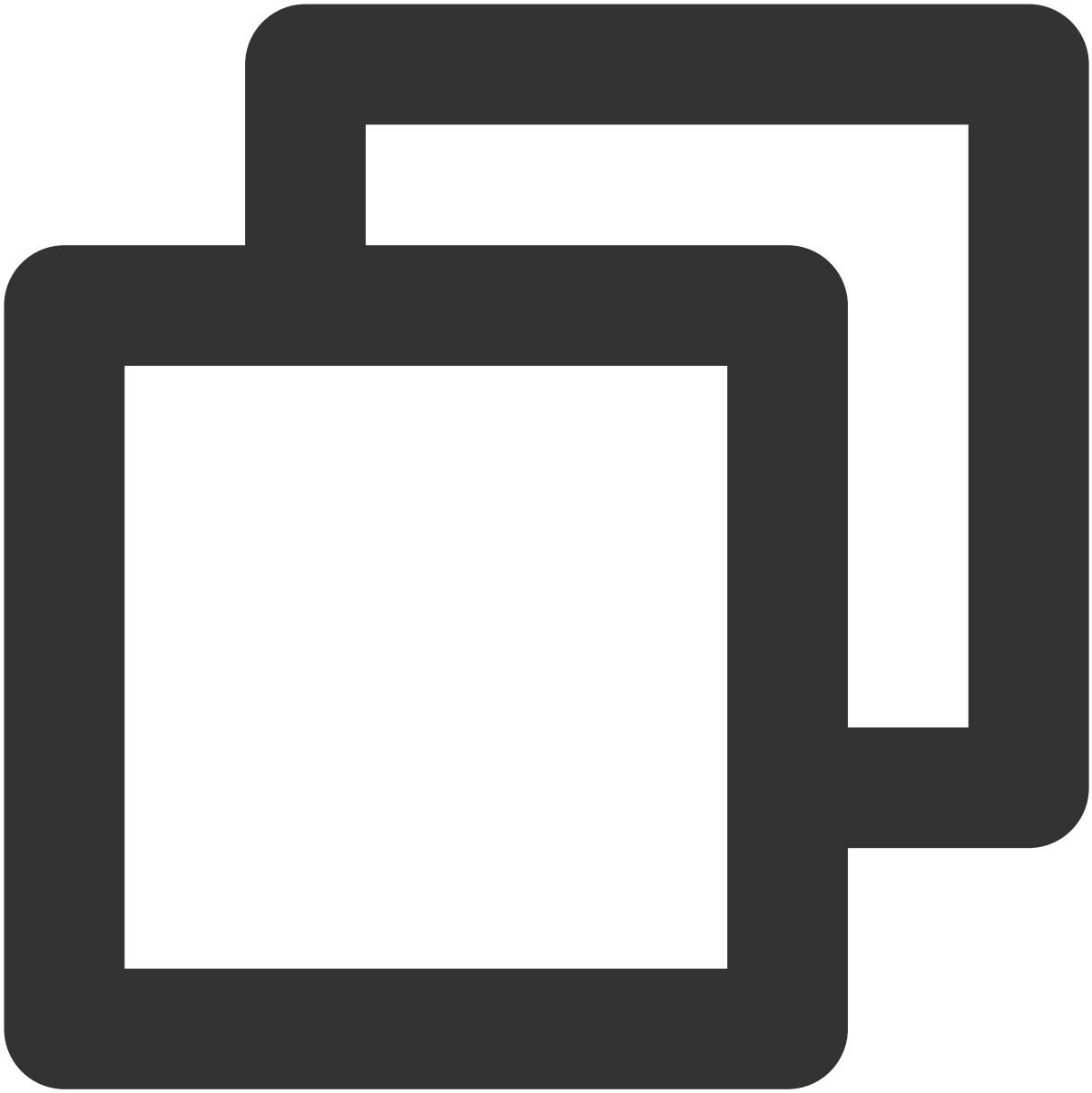
The following information will appear:



```
Initializing the backend...

Initializing provider plugins...
- Finding latest version of tencentcloudstack/tencentcloud...
- Installing tencentcloudstack/tencentcloud v1.60.22...
- Installed tencentcloudstack/tencentcloud v1.60.22 (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
```

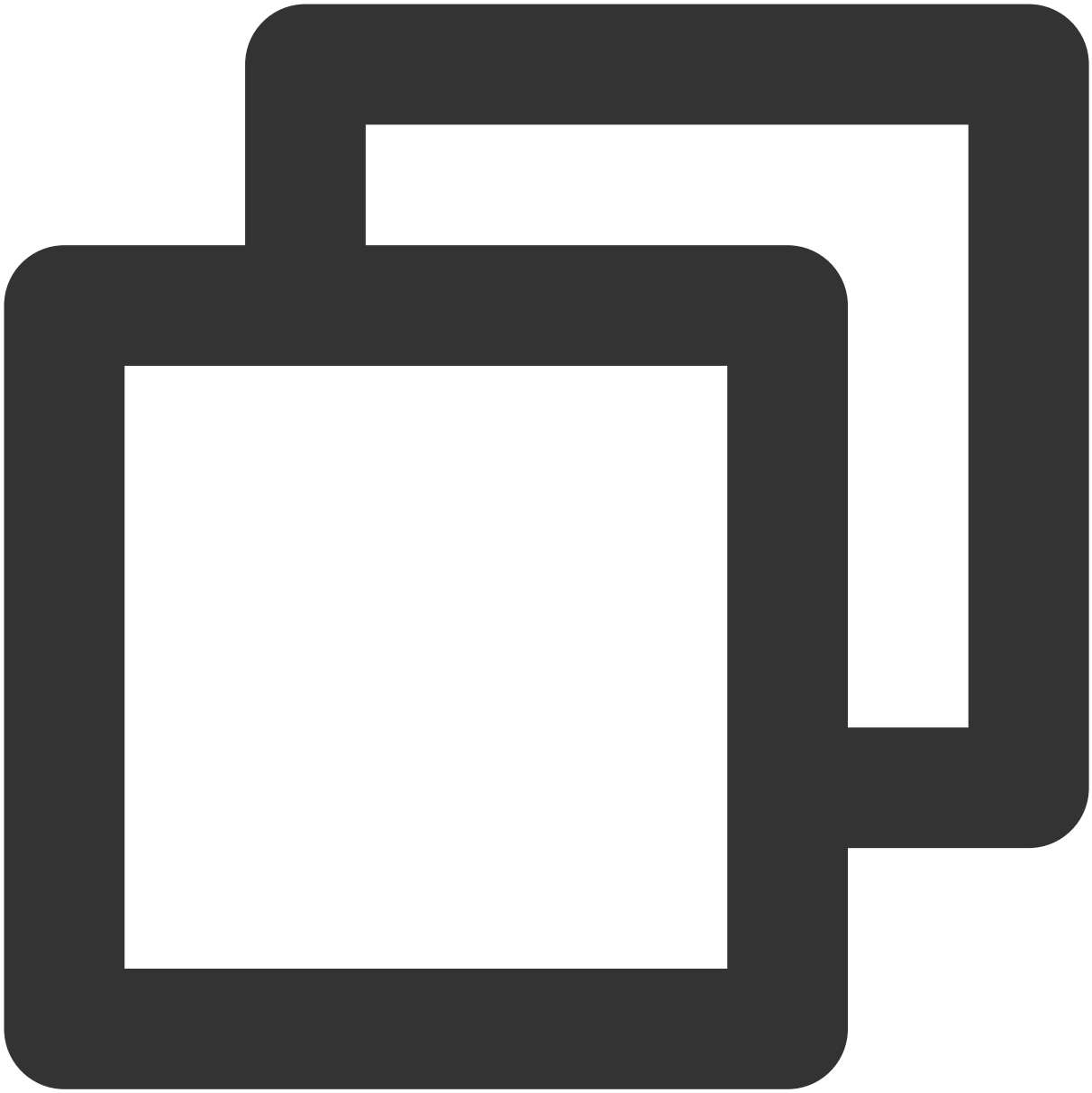
```
Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

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.
```

3. After the initialization is completed, run the following command to import resources into the state file.



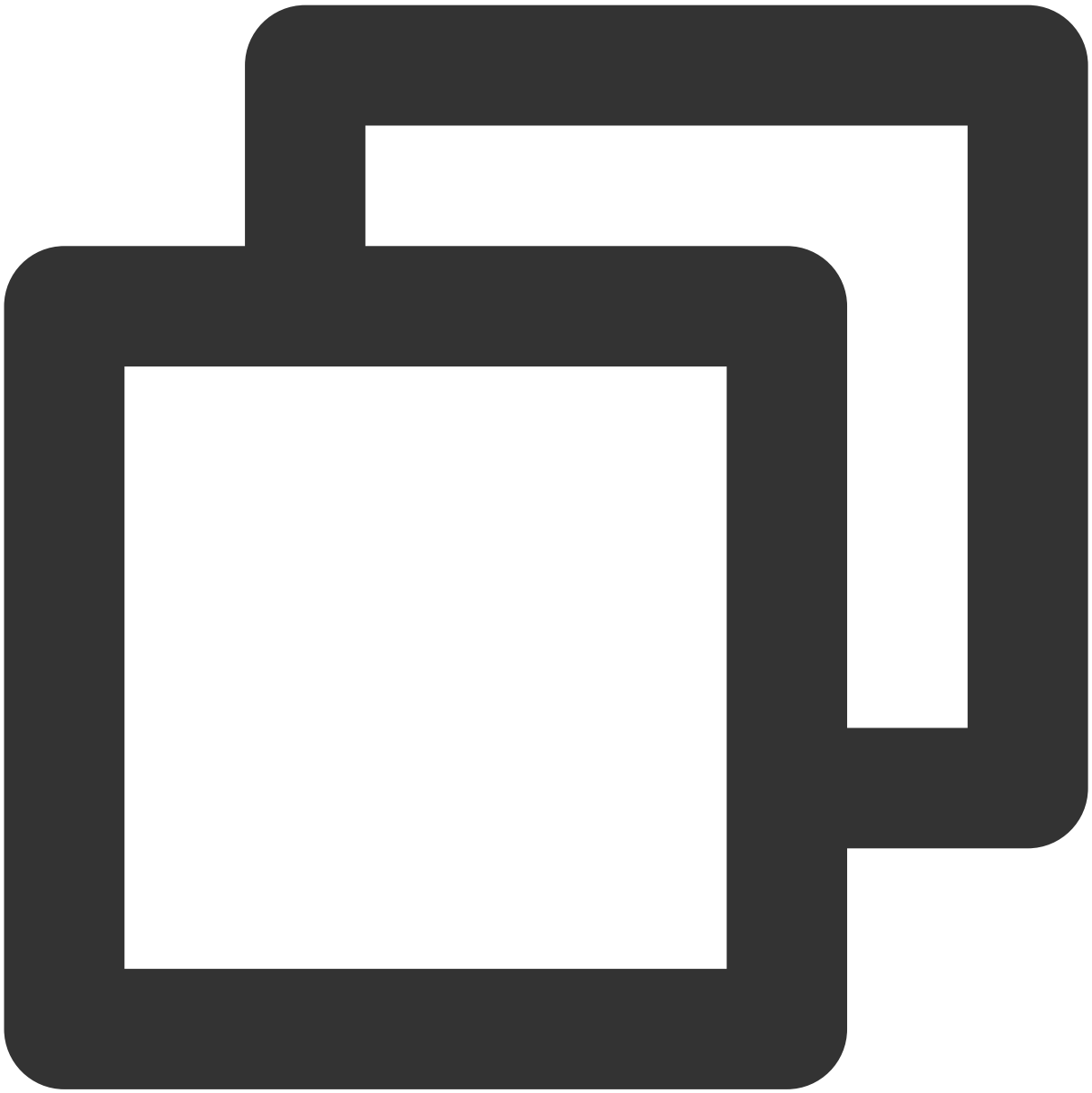
```
terraform import tencentcloud_monitor_alarm_policy.policy policy-vor9w72r
```

The response is as follows:



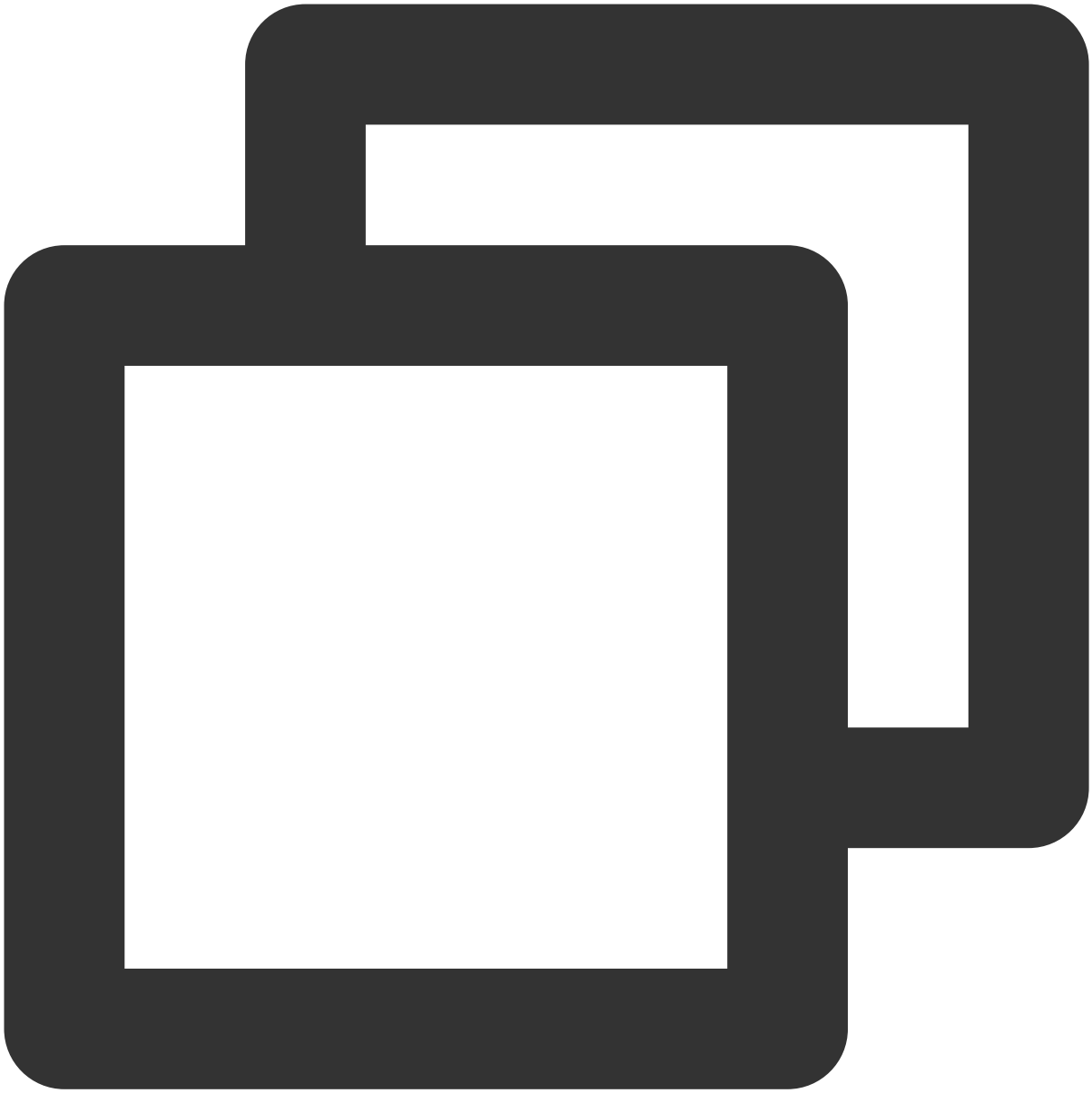
```
➜  terraform import tencentcloud_monitor_alarm_policy.policy policy-vor9w72r

tencentcloud_monitor_alarm_policy.policy: Importing from ID "policy-vor9w72r"...
tencentcloud_monitor_alarm_policy.policy: Import prepared!
  Prepared tencentcloud_monitor_alarm_policy for import
tencentcloud_monitor_alarm_policy.policy: Refreshing state... [id=policy-vor9w72r]

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. Run the following command to view the state file.



```
cat terraform.tfstate
```

You can view the following resource information:



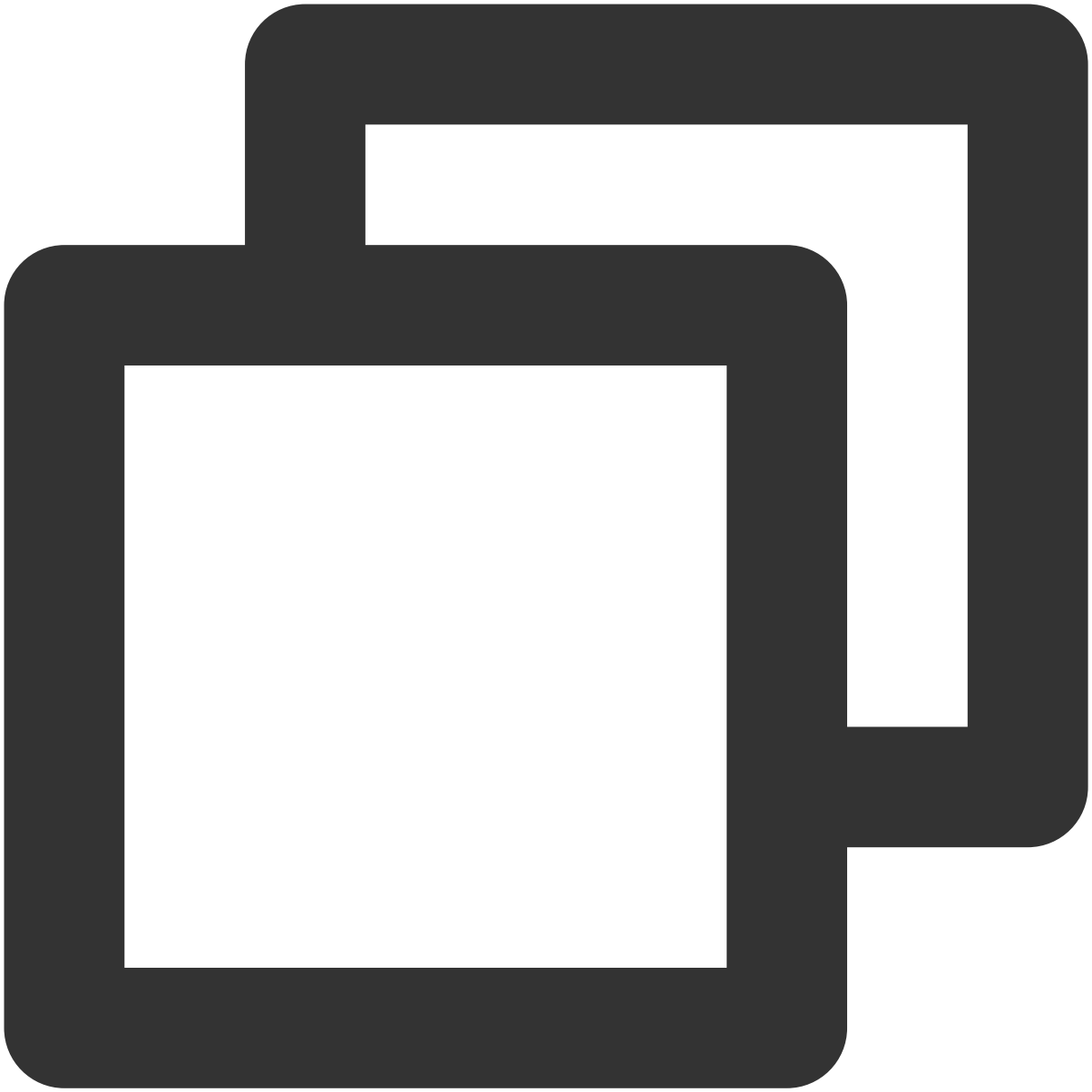

```
{
  "version": 4,
  "terraform_version": "1.1.0",
  "serial": 1,
  "lineage": "35791a73-d371-db51-5871-bfee13426217",
  "outputs": {},
  "resources": [
    {
      "mode": "managed",
      "type": "tencentcloud_monitor_alarm_policy",
      "name": "policy",
```

```
      "provider": "provider[\\"registry.terraform.io/tencentcloudstack/tencentcloud
      "instances": [
        {
          "schema_version": 0,
          "attributes": {
            "conditions": [
              {
                "is_union_rule": 0,
                "rules": [
                  {
                    "continue_period": 5,
                    "description": "cpu",
                    "filter": [],
                    "is_power_notice": 0,
                    "metric_name": "Cpu",
                    "notice_frequency": 86400,
                    "operator": "gt",
                    "period": 60,
                    "rule_type": "STATIC",
                    "unit": "%",
                    "value": "90"
                  }
                ]
              }
            ],
            "conditon_template_id": null,
            "create_time": null,
            "enable": 1,
            "event_conditions": [
              {
                "continue_period": 0,
                "description": "HASwitch",
                "filter": [],
                "is_power_notice": 0,
                "metric_name": "ha_switch",
                "notice_frequency": 0,
                "operator": "",
                "period": 0,
                "rule_type": "",
                "unit": "",
                "value": ""
              }
            ],
            "id": "policy-vor9w72r",
            "monitor_type": "MT_QCE",
            "namespace": "POSTGRESQL",
            "notice_ids": [
```

```
              "notice-l9ziyxw6"
            ],
            "policy_name": "PgSql",
            "project_id": 0,
            "remark": "",
            "trigger_tasks": [],
            "update_time": null
          },
          "sensitive_attributes": [],
          "private": "eyJzY2hlbWFfdmVyc2lvbiI6IjAifQ=="
        }
      ]
    }
  ]
}
```

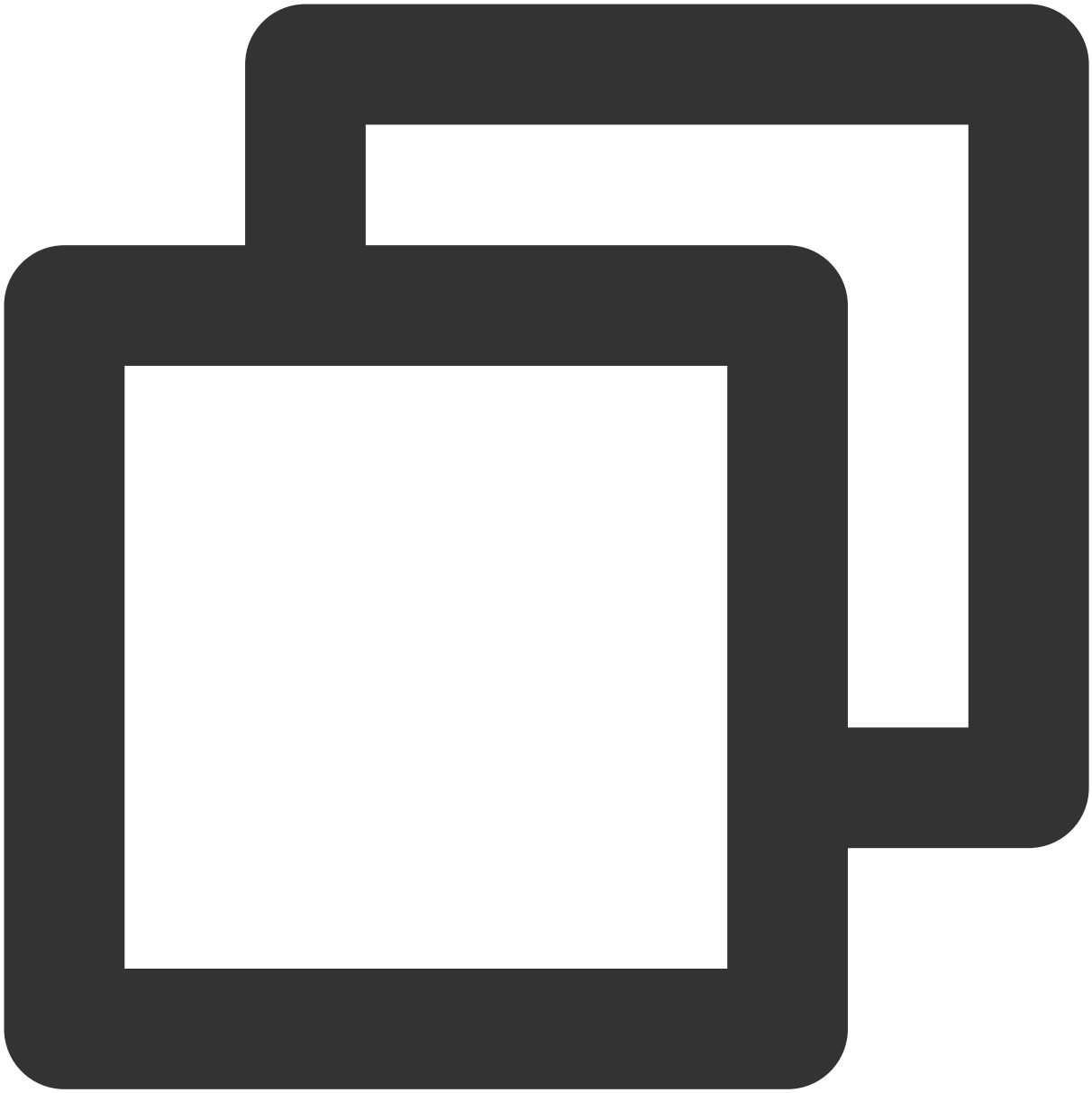
Updating the source file

1. Run the following command to print the resource information.



```
terraform show
```

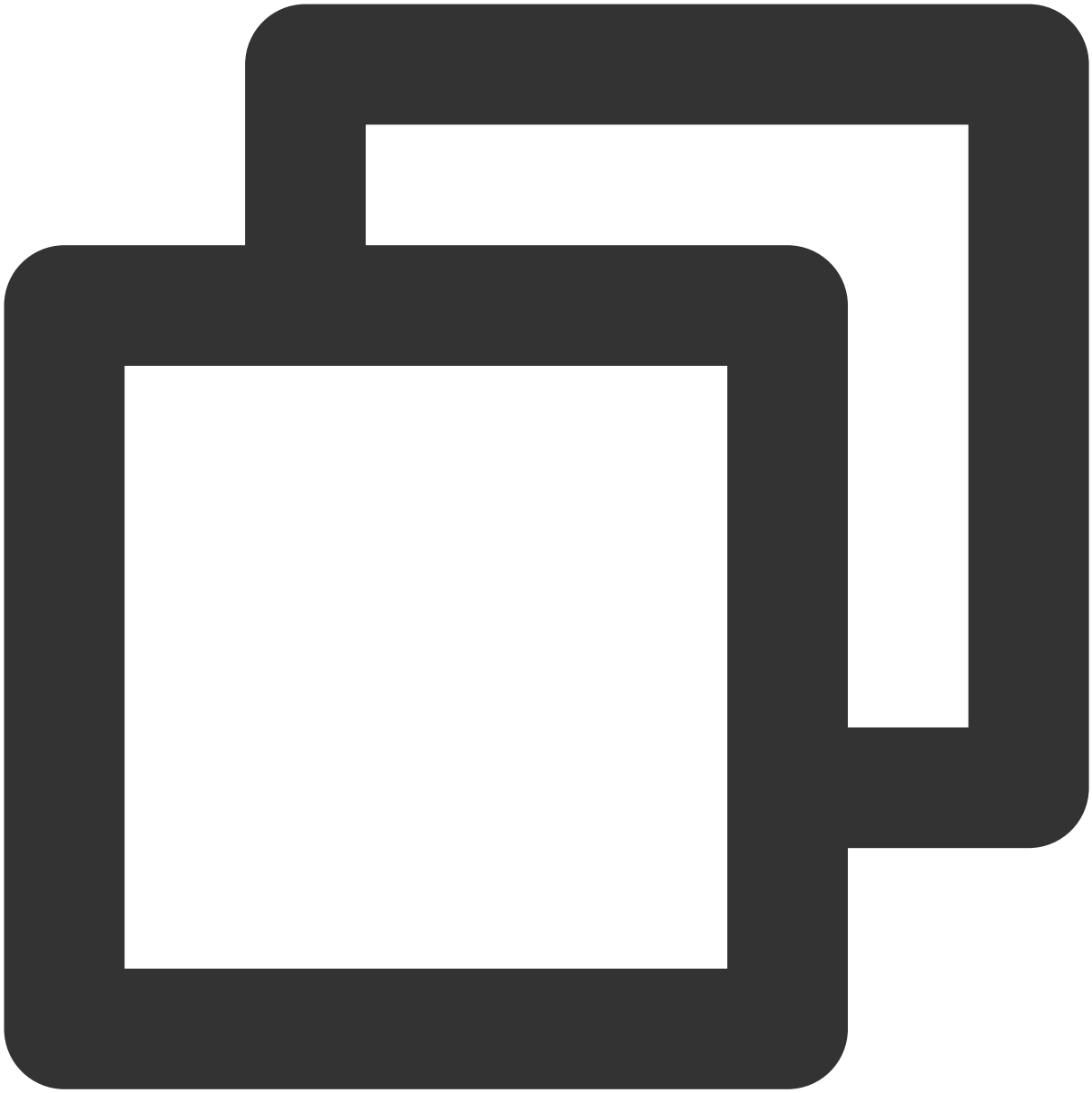
The response is as follows:



```
# tencentcloud_monitor_alarm_policy.policy:
resource "tencentcloud_monitor_alarm_policy" "policy" {
    enable       = 1
    id           = "policy-vor9w72r"
    monitor_type = "MT_QCE"
    namespace    = "POSTGRESQL"
    notice_ids   = [
        "notice-l9ziyxw6",
    ]
    policy_name  = "PgSql"
    project_id   = 0
```

```
    conditions {
        is_union_rule = 0

        rules {
            continue_period  = 5
            description      = "cpu"
            is_power_notice  = 0
            metric_name      = "Cpu"
            notice_frequency = 86400
            operator         = "gt"
            period           = 60
            rule_type        = "STATIC"
            unit             = "%"
            value            = "90"
        }
    }

    event_conditions {
        continue_period  = 0
        description      = "HASwitch"
        is_power_notice  = 0
        metric_name      = "ha_switch"
        notice_frequency = 0
        period           = 0
    }
}
```

2. Copy the resource code to the Terraform source file `tencentcloud.tf` . You need to delete any options that cannot be set, such as ID. Below is the content of the edited `tencentcloud.tf` file:

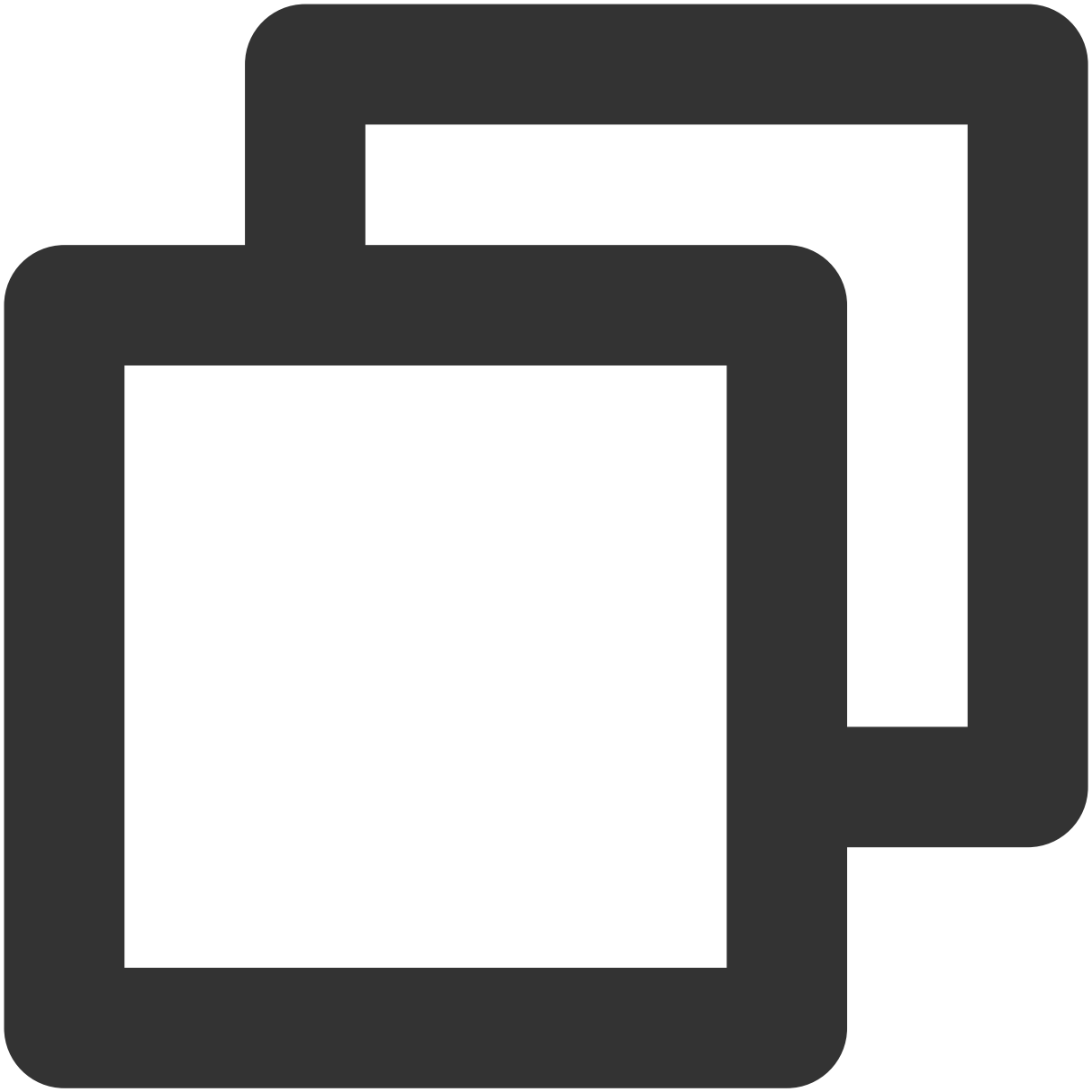


```
provider tencentcloud {}
resource "tencentcloud_monitor_alarm_policy" "policy" {
  enable       = 1
#  id           = "policy-vor9w72r"
  monitor_type = "MT_QCE"
  namespace    = "POSTGRESQL"
  notice_ids   = [
    "notice-l9ziyxw6",
  ]
  policy_name  = "PgSql"
  project_id   = 0
```

```
  conditions {
    is_union_rule = 0

    rules {
      continue_period  = 5
      description      = "cpu"
      is_power_notice  = 0
      metric_name      = "Cpu"
      notice_frequency = 86400
      operator         = "gt"
      period           = 60
      rule_type        = "STATIC"
      unit             = "%"
      value            = "90"
    }
  }

  event_conditions {
    continue_period  = 0
    description      = "HASwitch"
    is_power_notice  = 0
    metric_name      = "ha_switch"
    notice_frequency = 0
    period           = 0
  }
}
```

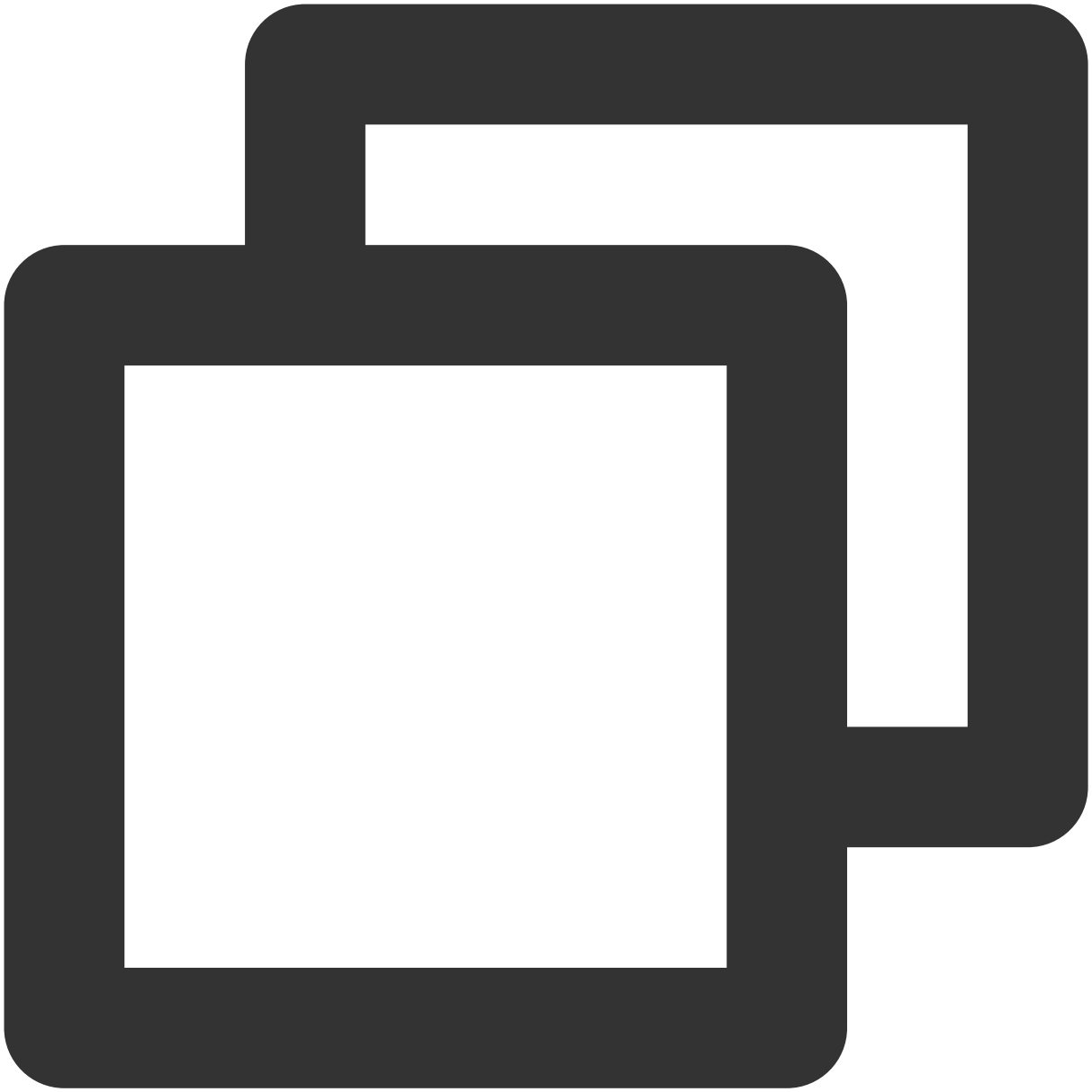
Verification

Run the following command for a refresh using the code and state file in the current working directory.



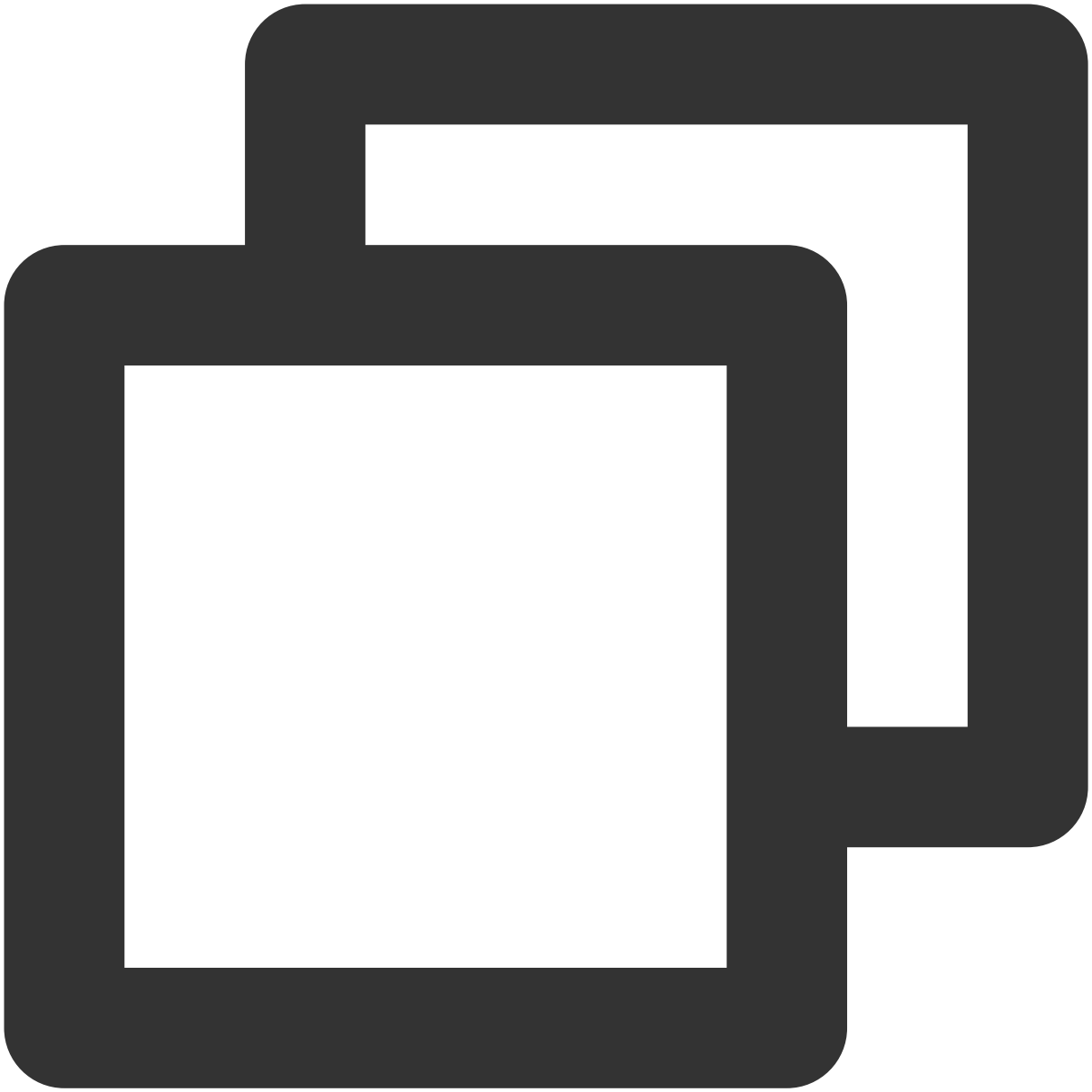
```
terraform plan
```

The following information is returned, showing that the resource has been successfully taken over by Terraform.



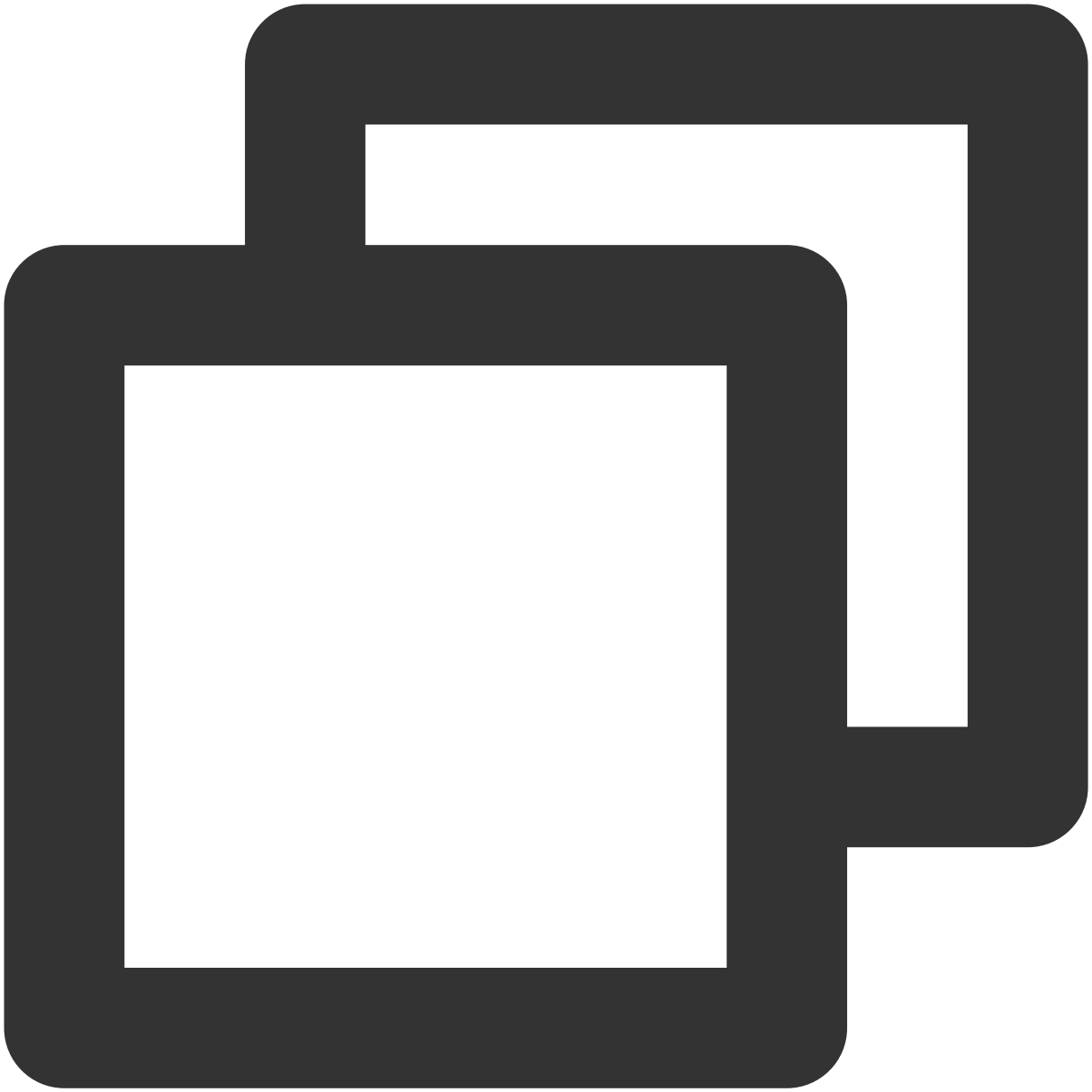
```
tencentcloud_monitor_alarm_policy.policy: Refreshing state... [id=policy-vor9w72r]

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and foun
```

At this point, Terraform has taken over the resource. You can delete it with the `destroy` command or modify it just like the code. After modifying, for example, the alarm threshold, run the following command for an update.



```
terraform plan
```

The following information is returned, showing that Terraform indicates that the alarm policy will be updated after value modification.



```
tencentcloud_monitor_alarm_policy.policy: Refreshing state... [id=policy-vor9w72r]

Terraform used the selected providers to generate the following execution plan. Res
  ~ update in-place

Terraform will perform the following actions:

  # tencentcloud_monitor_alarm_policy.policy will be updated in-place
  ~ resource "tencentcloud_monitor_alarm_policy" "policy" {
        id           = "policy-vor9w72r"
        # (6 unchanged attributes hidden)
```

```
      ~ conditions {
            # (1 unchanged attribute hidden)

          ~ rules {
              ~ value             = "90" -> "99"
                # (9 unchanged attributes hidden)
            }
        }

        # (1 unchanged block hidden)
    }

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

State Lock

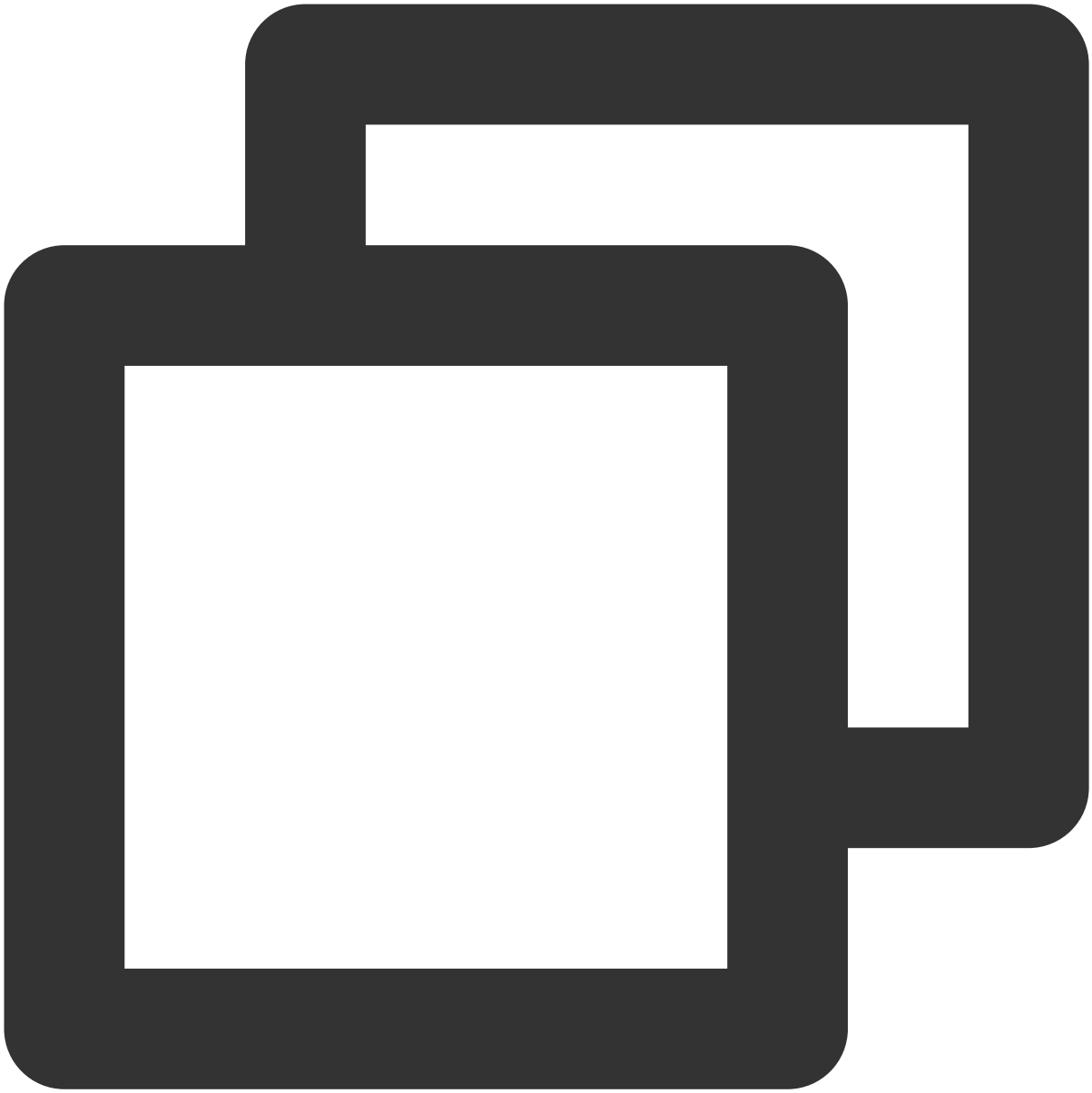
Last updated：2023-05-29 16:32:56

Background

Configuring Terraform via the backend involves the execution environment and remote state sync. Terraform introduces a lock mechanism to prevent the state from being out of sync when multiple execution environments manipulate the same backend. Different vendors can adopt Terraform's lock interface to implement their state lock. This document describes the implementation and troubleshooting of the COS backend lock.

State Lock and File Write Lock

As indicated by the source code, the COS backend involves two locks during execution.
State lock: It is in the form of the `terraform.tfstate.tflock` file that records the information of the current process and is written to the backend bucket.



```
{
    "ID":"1234abcd-1234-cdef-5678-1234567890ab",
    "Operation":"OperationTypePlan",
    "Info":"",
    "Who":"UserName@Host",
    "Version":"1.3.0",
    "Created":"2006-01-02T15:04:05Z07:00",
    "Path":"terraform.tfstate.tflock"
}
```

File write lock: It prevents multiple processes from writing the `terraform.tfstate.tflock` file to the bucket at the same time. The COS backend implements the lock as a Tencent Cloud tag in the format of `tencentcloud-terraform-lock:xxxxxxxxx` (here, `xxxxxxxxx` is the MD5 of the `bucket:lockfile`), which is released automatically after the file is written successfully.

Locking Steps

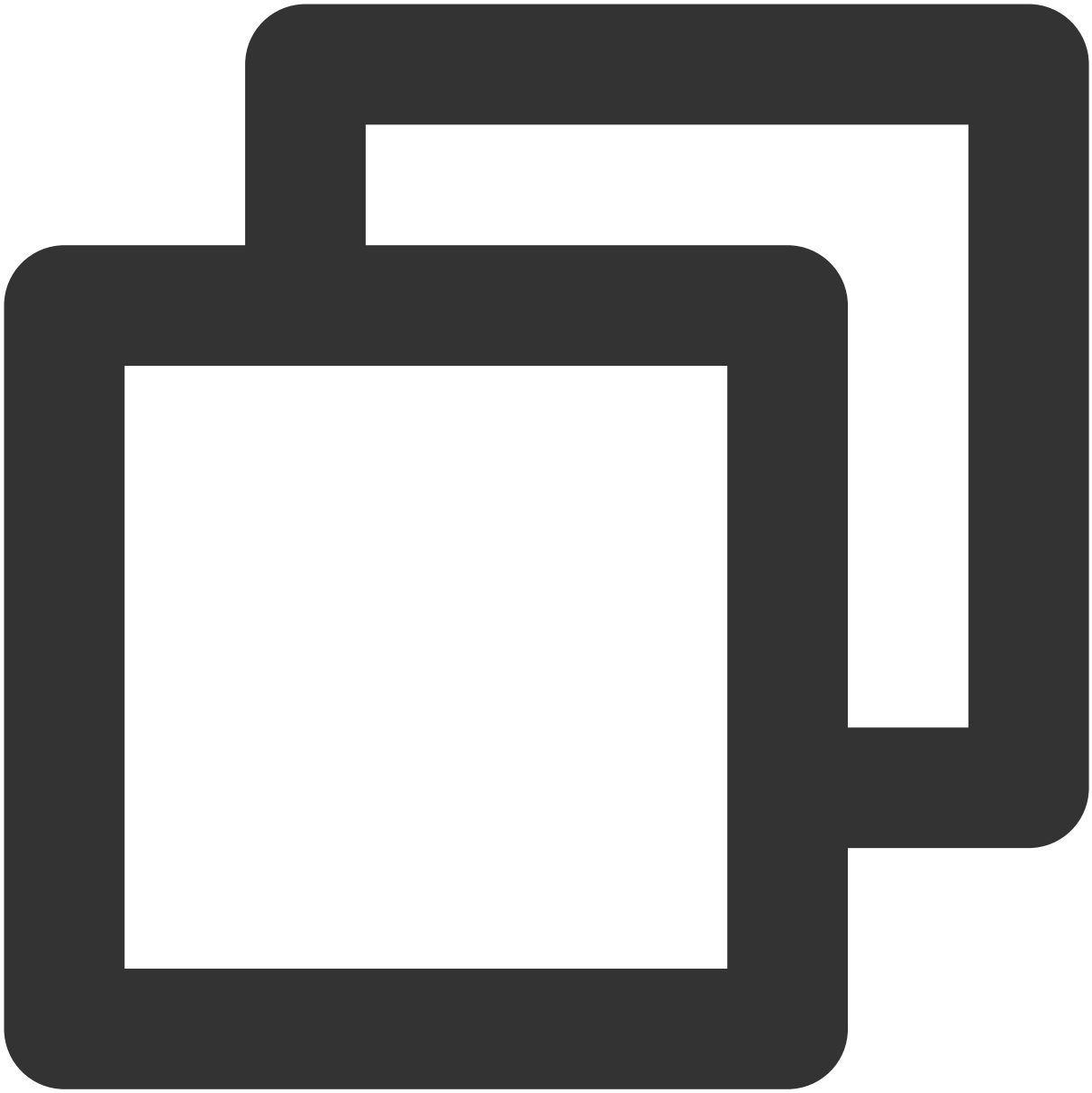
In general, running a Terraform command involves the following steps:

1. Terraform executes the command (such as `plan` or `apply).
2. The backend service adds the Tencent Cloud tag to the current account as the file write lock.
3. The bucket checks and writes `terraform.tfstate.tflock` as the state lock.
4. Delete the tag and release the file write lock.
5. Wait for Terraform to complete the execution.
6. The bucket checks and deletes `terraform.tfstate.tflock` to cancel the state lock.
7. Terraform completes the execution.

Exception Handling

Scenario 1

When a process is performing step 4, 5, 6, or 7 in Locking Steps and another process tries manipulating the same backend, a lock error will be reported as follows:



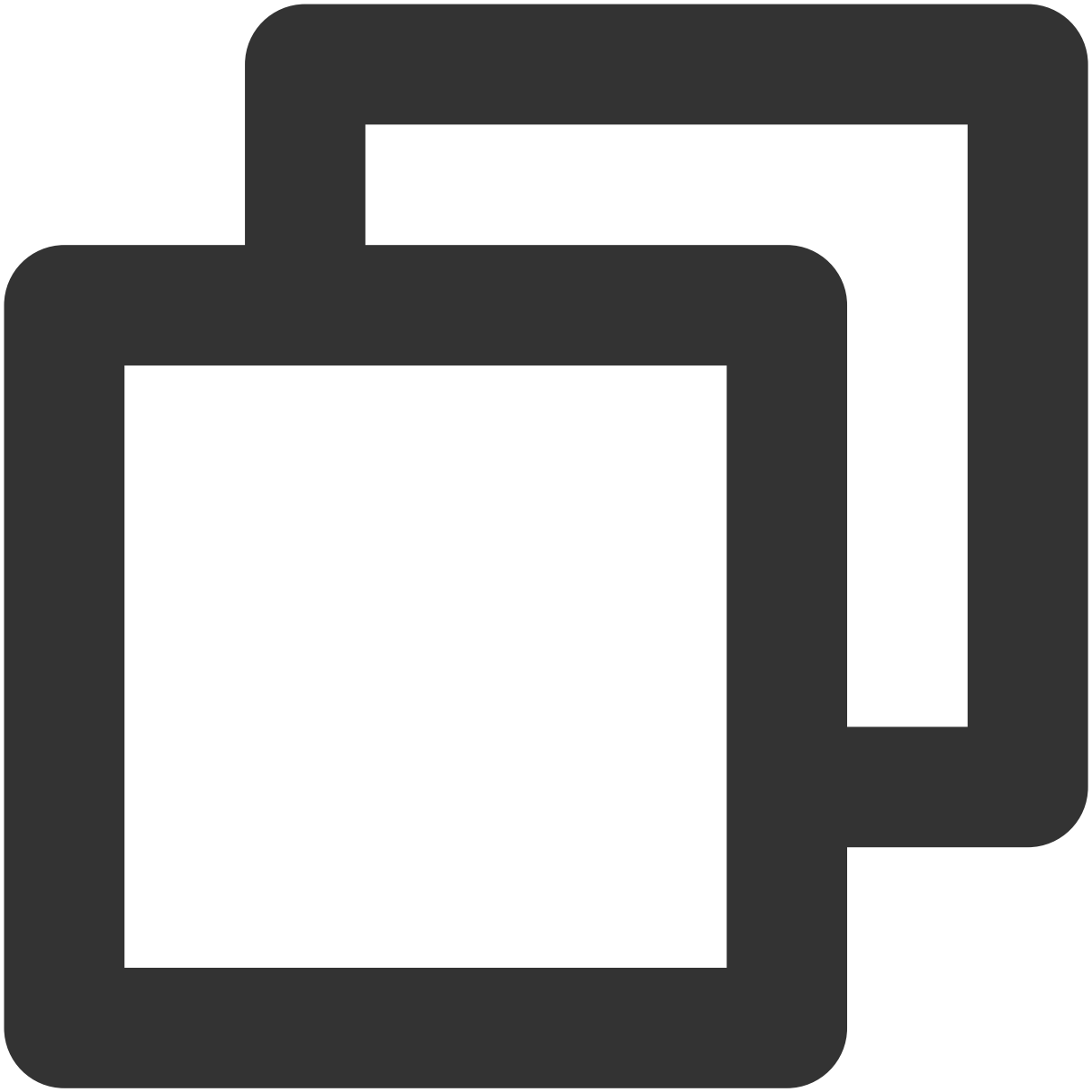
```
|
| Error: Error acquiring the state lock
|
| Error message: lock file terraform.tfstate.tflock exists
| Lock Info:
|   ID:        1234abcd1234cdef56781234567890ab
|   Path:      terraform.tfstate.tflock
|   Operation: OperationTypePlan
|   Who:       UserName@Host
|   Version:   1.1.2
|   Created:   2006-01-02T15:04:05Z07:00
```

```
  Info:


 Terraform acquires a state lock to protect the state from being written
 by multiple users at the same time. Please resolve the issue above and try
 again. For most commands, you can disable locking with the "-lock=false"
 flag, but this is not recommended.
```

At this point, other execution environments need to wait for the current process to complete. Or you can add `-lock=false` as prompted to ignore the state lock for re-execution **(not recommended)**.

At this time, if a Terraform process exits unexpectedly and does not release the state lock, you need to manually release it.

Run the following force-unlock command by lock ID as indicated above.

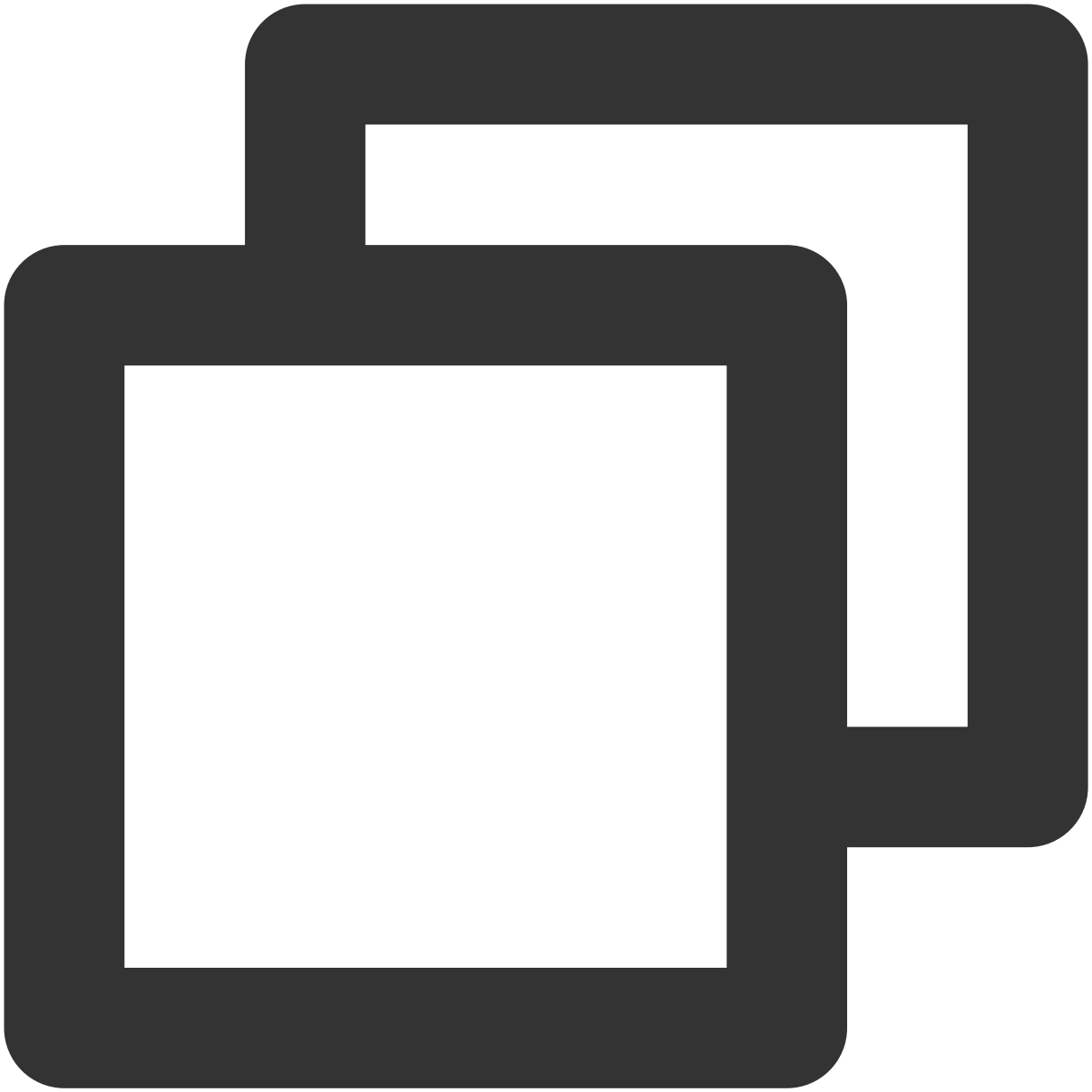


```
terraform force-unlock -force 1234abcd1234cdef56781234567890ab
```

Or you can delete the `terraform.tfstate.tflock` file in the bucket to unlock.

Scenario 2

If a process exits due to interruption when performing step 3 or 4 in Locking Steps and does not release the tag lock, Terraform executions will lead to the exception indicating that the tag cannot be created:



```
| Error: Error acquiring the state lock
|
| Error message: 2 errors occurred:
|       * failed to create tag: tencentcloud-terraform-lock -> xxxxxxxxx: [TencentC
| RequestId=47c57b0b-2491-42cd-9f29-0b14802681e5
|       * lock file terraform/state/terraform.tfstate.tflock not exists
|
|
|
| Terraform acquires a state lock to protect the state from being written
| by multiple users at the same time. Please resolve the issue above and try
```

```
again. For most commands, you can disable locking with the "-lock=false"
flag, but this is not recommended.
```

At this point, Terraform has no command to release the lock, and you need to manually delete the `tencentcloud-terraform-lock:xxxxxxxxx` tag in the console or via the TencentCloud API DeleteTags. Below are directions you can follow in the console:

1. Log in to the tag console and select **Tag List** on the left sidebar.
2. Click **Delete** on the right of the target tag and click **OK** in the pop-up window.