terrafo... / terraform-google-i...

Type / to search

Code Issues 8 Pull requests 6 Actions Security Insights

terraform-google-iam

Public

29 Branches 27 Tags Go to file t Go to file About Add file Code

release-please[bot] and [redacted] chore(main): relea...

171c15b · 3 months ago 314 Commits

.github	chore: update .gith...	4 months ago
build	chore(deps): Upda...	8 months ago
docs	Merge branch 'ma...	6 years ago
examples	fix: Removed Privil...	8 months ago
modules	chore(main): relea...	3 months ago
test	chore(deps): Upda...	7 months ago
.gitignore	chore: update test...	4 years ago
.kitchen.yml	fix: fixes lint issues...	3 years ago
CHANGELO...	chore(main): relea...	3 months ago
CODEOWN...	chore: update CO...	4 months ago
CONTRIBUT...	Update testing to ...	6 years ago
LICENSE	Update testing to ...	6 years ago
Makefile	chore(deps): Upda...	8 months ago
README.md	chore(main): relea...	3 months ago
metadata.ya...	feat: add dns zone...	2 years ago

README Apache-2.0 license Security

Google IAM Terraform Module

This is a collection of submodules that make it easier to non-destructively manage multiple IAM roles for resources on Google Cloud Platform:

Manages multiple IAM roles for resources on Google Cloud

registry.terraform.io/modules/terrafor...

#cft-terraform #security-identity

Readme
Apache-2.0 license
Security policy
Activity
Custom properties
208 stars
43 watching
179 forks
Report repository

Releases 26

v8.1.0 Latest
on Feb 10

+ 25 releases

Packages

No packages published

Contributors 49

+ 35 contributors

Languages

HCL 78.2% Ruby 20.1%
Makefile 1.7%

- Artifact Registry IAM
- Audit Config
- BigQuery IAM
- Billing Accounts IAM
- Cloud Run Service IAM
- Custom Role IAM
- DNS Zone IAM
- Folders IAM
- KMS Crypto Keys IAM
- KMS_Key Rings IAM
- Organizations IAM
- Projects IAM
- Pubsub Subscriptions IAM
- Pubsub Topics IAM
- Secret Manager IAM
- Service Accounts IAM
- Storage Buckets IAM
- Subnets IAM
- Tag Keys IAM
- Tag Values IAM
- Secure Source Manager

Compatibility

This module is meant for use with Terraform 1.3+ and tested using Terraform 1.3+. If you find incompatibilities using Terraform >=1.3, please open an issue.

Upgrading

The following guides are available to assist with upgrades:

- 4.0 -> 5.0
- 3.0 -> 4.0
- 2.0 -> 3.0

Usage

Full examples are in the examples folder, but basic usage is as follows for managing roles on two projects:

```
module "projects_iam_bindings" {
  source  = "terraform-google-modules/iam/google//modules/projects_iam"
  version = "~> 8.1"

  projects = ["project-123456", "project-9876543"]

  bindings = {
    "roles/storage.admin" = [
```

```
      "group:test_sa_group@lnescidev.com",
      "user:someone@google.com",
    ]

    "roles/compute.networkAdmin" = [
      "group:test_sa_group@lnescidev.com",
      "user:someone@google.com",
    ]

    "roles/compute.imageUser" = [
      "user:someone@google.com",
    ]
  }
}
```

The module also offers an **authoritative** mode which will remove all roles not assigned through Terraform. This is an example of using the authoritative mode to manage access to a storage bucket:

```
module "storage_buckets_iam_bindings" {
  source  = "terraform-google-modules/iam/google//modules/storage_buckets_iam"
  version = "~> 8.0"

  storage_buckets = ["my-storage-bucket"]

  mode = "authoritative"

  bindings = {
    "roles/storage.legacyBucketReader" = [
      "user:josemanuelt@google.com",
      "group:test_sa_group@lnescidev.com",
    ]

    "roles/storage.legacyBucketWriter" = [
      "user:josemanuelt@google.com",
      "group:test_sa_group@lnescidev.com",
    ]
  }
}
```

Additive and Authoritative Modes

The `mode` variable controls a submodule's behavior, by default it's set to "additive", possible options are:

- additive: add members to role, old members are not deleted from this role.
- authoritative: set the role's members (including removing any not listed), unlisted roles are not affected.

In authoritative mode, a submodule takes full control over the IAM bindings listed in the module. This means that any members added to roles outside the module will be removed the next time Terraform runs. However, roles not listed in the module will be unaffected.

In additive mode, a submodule leaves existing bindings unaffected. Instead, any members listed in the module will be added to the existing set of IAM bindings. However, members listed in the module *are* fully controlled by the module. This means that if you add a binding via the module and later remove it, the module will correctly handle removing the role binding.

Caveats

Referencing values/attributes from other resources

Each submodule performs operations over some variables before making any changes on the IAM bindings in GCP. Because of the limitations of `for_each` (more info), which is widely used in the submodules, there are certain limitations to what kind of dynamic values you can provide to a submodule:

1. Dynamic entities (for example `projects`) are only allowed for 1 entity.
2. If you pass 2 or more entities (for example `projects`), the configuration **MUST** be static, meaning that it can't use any of the other resources' fields to get the entity name from (this includes getting the randomly generated hashes through the `random_id` resource).
3. The role names themselves can never be dynamic.
4. Members may only be dynamic in `authoritative` mode.

IAM Bindings

You can choose the following resource types to apply the IAM bindings:

- Projects (`projects` variable)
- Organizations(`organizations` variable)
- Folders (`folders` variable)
- Service Accounts (`service_accounts` variable)
- Subnetworks (`subnets` variable)
- Storage buckets (`storage_buckets` variable)
- Pubsub topics (`pubsub_topics` variable)
- Pubsub subscriptions (`pubsub_subscriptions` variable)
- Kms Key Rings (`kms_key_rings` variable)
- Kms Crypto Keys (`kms_crypto_keys` variable)

- Secret Manager Secrets (`secrets` variable)
- DNS Zones (`managed_zones` variable)
- Secure Source Manager (`entity_ids` and `location` variable)

Set the specified variable on the module call to choose the resources to affect. Remember to set the `mode` variable and give enough permissions to manage the selected resource as well. Note that the `bindings` variable accepts an empty map `{}` passed in as an argument in the case that resources don't have IAM bindings to apply.

Requirements

Terraform plugins

- Terraform >= 0.13.0
- terraform-provider-google 2.5
- terraform-provider-google-beta 2.5

- terraform provider google-beta 2.5

Permissions

In order to execute a submodule you must have a Service Account with an appropriate role to manage IAM for the applicable resource. The appropriate role differs depending on which resource you are targeting, as follows:

- Organization:
  - Organization Administrator: Access to administer all resources belonging to the organization and does not include privileges for billing or organization role administration.
  - Custom: Add resourcemanager.organizations.getIamPolicy and resourcemanager.organizations.setIamPolicy permissions.
- Project:
  - Owner: Full access and all permissions for all resources of the project.
  - Projects IAM Admin: allows users to administer IAM policies on projects.
  - Custom: Add resourcemanager.projects.getIamPolicy and resourcemanager.projects.setIamPolicy permissions.
- Folder:
  - The Folder Admin: All available folder permissions.
  - Folder IAM Admin: Allows users to administer IAM policies on folders.
  - Custom: Add resourcemanager.folders.getIamPolicy and resourcemanager.folders.setIamPolicy permissions (must be added in the organization).
- Service Account:
  - Service Account Admin: Create and manage service accounts.
  - Custom: Add resourcemanager.organizations.getIamPolicy and resourcemanager.organizations.setIamPolicy permissions.
- Subnetwork:
  - Project compute admin: Full control of Compute Engine resources.
  - Project compute network admin: Full control of Compute Engine networking resources.
  - Project custom: Add compute.subnetworks.getIamPolicy and compute.subnetworks.setIamPolicy permissions.
- Storage bucket:
  - Storage Admin: Full control of GCS resources.
  - Storage Legacy Bucket Owner: Read and write access to existing buckets with object listing/creation/deletion.
  - Custom: Add storage.buckets.getIamPolicy and storage.buckets.setIamPolicy permissions.
- Pubsub topic:
  - Pub/Sub Admin: Create and manage service accounts.
  - Custom: Add pubsub.topics.getIamPolicy and pubsub.topics.setIamPolicy permissions.
- Pubsub subscription:
  - Pub/Sub Admin role: Create and manage service accounts.
  - Custom role: Add pubsub.subscriptions.getIamPolicy and pubsub.subscriptions.setIamPolicy permissions.
- Kms Key Ring:
  - Owner: Full access to all resources.
  - Cloud KMS Admin: Enables management of crypto resources.

    - Custom: Add cloudkms.keyRings.getIamPolicy and cloudkms.keyRings.getIamPolicy permissions.
- Kms Crypto Key:
    - Owner: Full access to all resources.
    - Cloud KMS Admin: Enables management of cryptoresources.
    - Custom: Add cloudkms.cryptoKeys.getIamPolicy and cloudkms.cryptoKeys.setIamPolicy permissions.
- Secret Manager:
    - Secret Manager Admin: Full access to administer Secret Manager.
    - Custom: Add secretmanager.secrets.getIamPolicy and secretmanager.secrets.setIamPolicy permissions.
- DNS Zone:
    - DNS Administrator : Full access to administer DNS Zone.
    - Custom: Add dns.managedZones.setIamPolicy, dns.managedZones.list and dns.managedZones.getIamPolicy permissions.

Install

Terraform

Be sure you have the correct Terraform version >= 1.3

Terraform plugins

Be sure you have the compiled plugins on $HOME/.terraform.d/plugins/

- terraform-provider-google >= 5.37
- terraform-provider-google-beta >= 5.37

See each plugin page for more information about how to compile and use them