Chronicle Zeek Automation

Terraform User Guide

Table of Contents

Pre-requisites

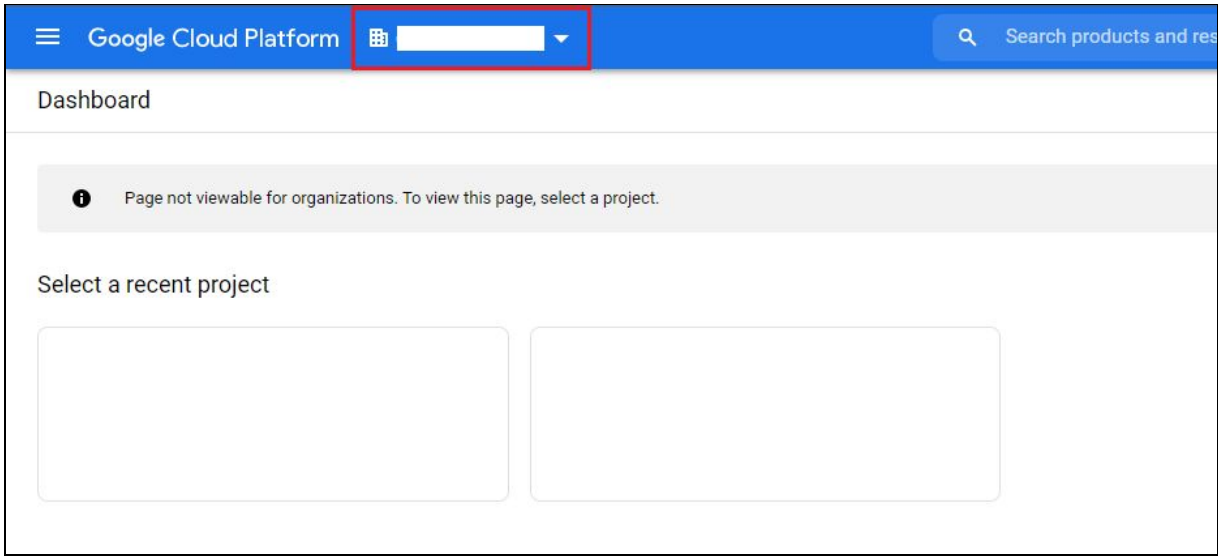
- GCP service account with appropriate roles
- Required Terraform version **0.14.5** or higher
- Generate service account key
- Enable API's inside project
- GCS Bucket should be created and its name to be noted from user side

GCP service account with appropriate roles

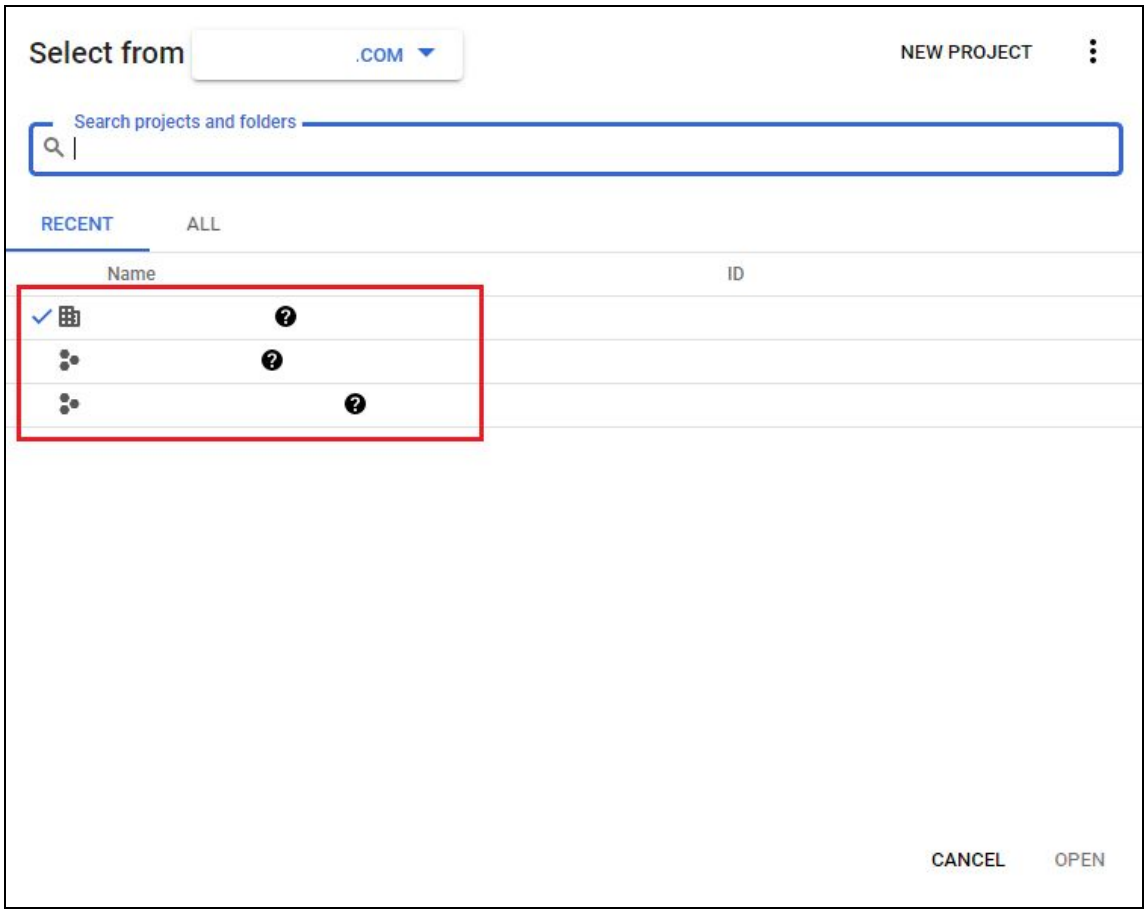
In order to run terraform scripts, user needs to authenticate with google cloud services via JSON Service Account Key

To create a custom service account for Terraform and assign it required IAM roles, follow the below instructions:

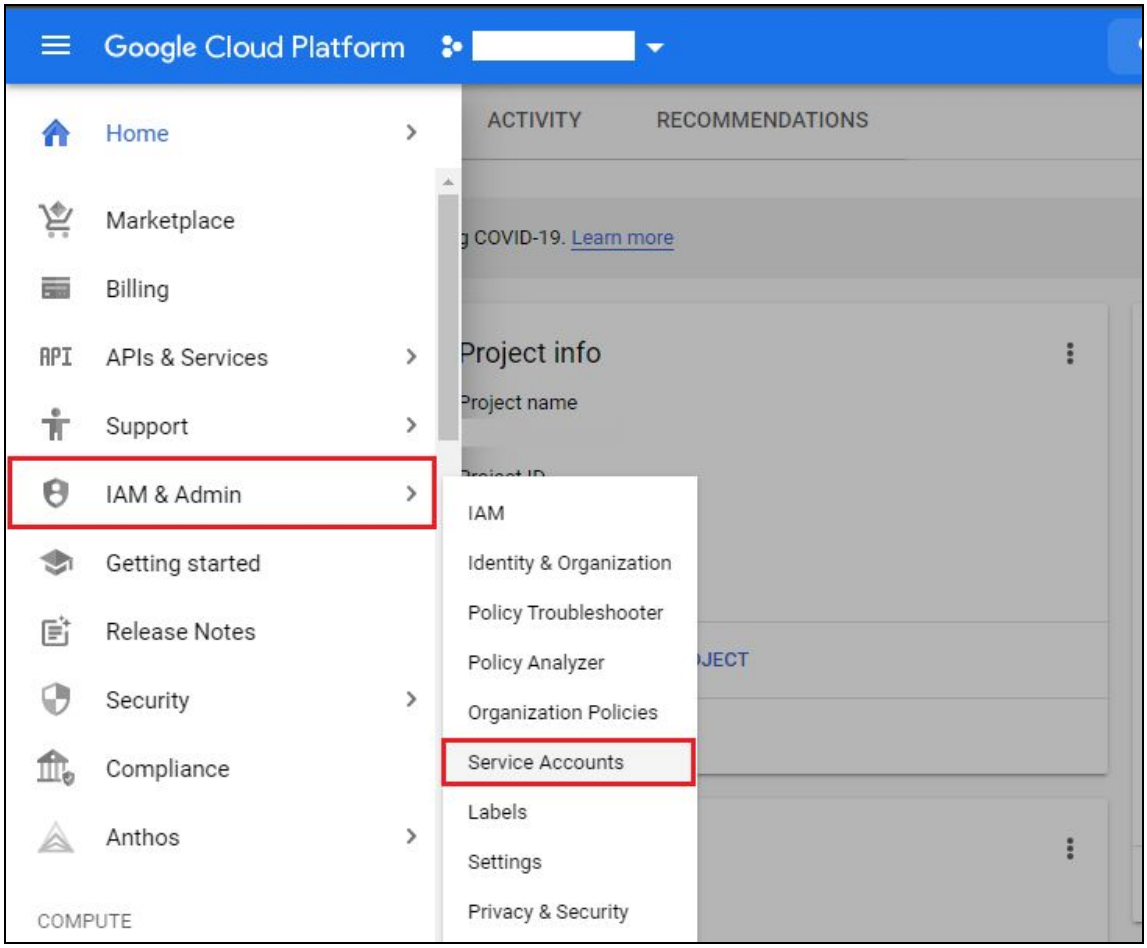
- Log in to the **Google Cloud Console** with this link and select a project.



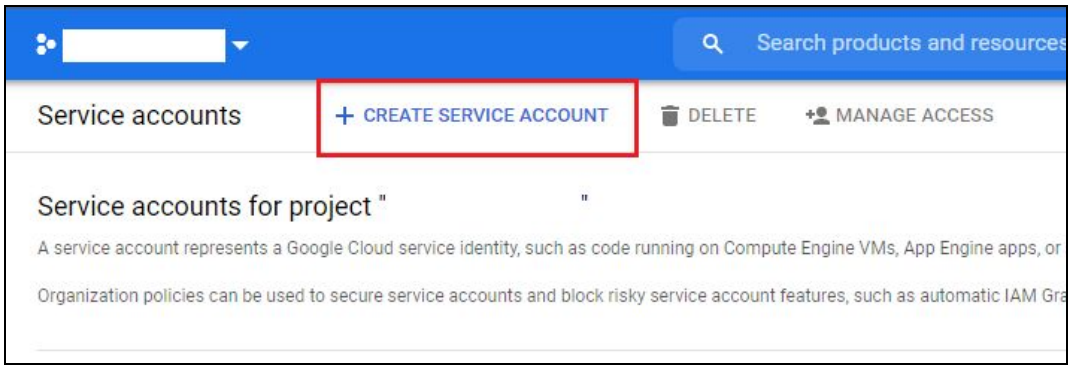
- Click **Select a project**, choose your project, and click **Open**.



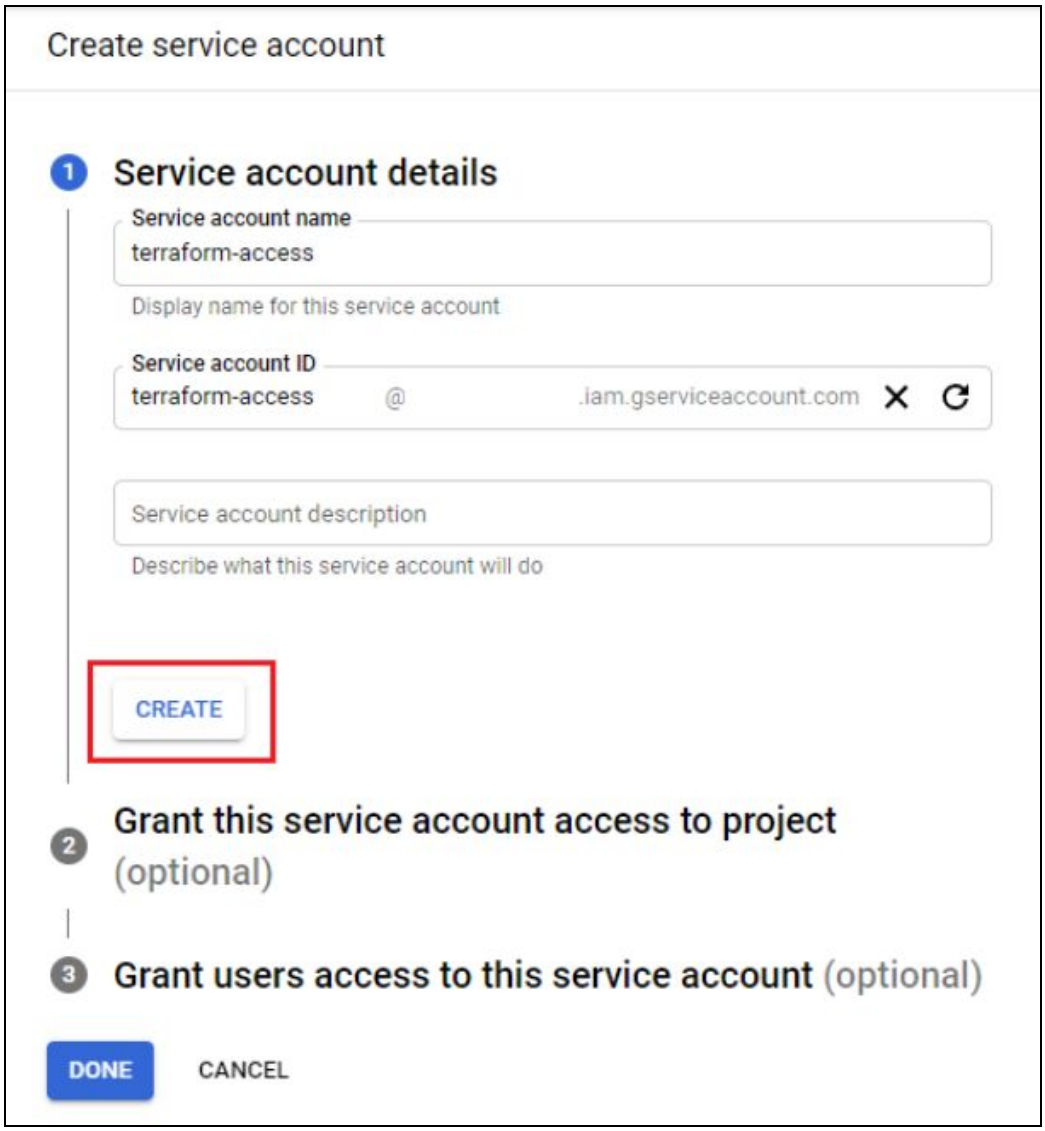
- Now, click on the **Navigation menu** (Hamburger icon) located at top-left corner of console, and then click on **Service Accounts** under **IAM & Admin** menu.



- Click on **Create Service Account** and provide the following information as below:

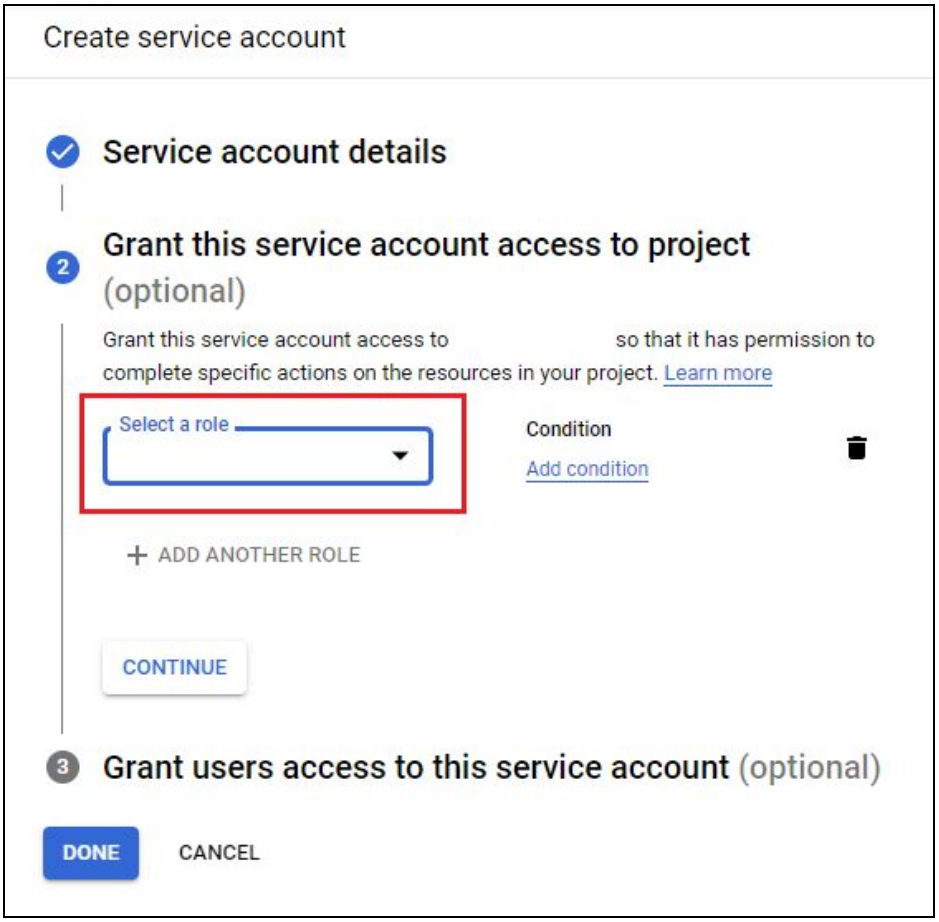


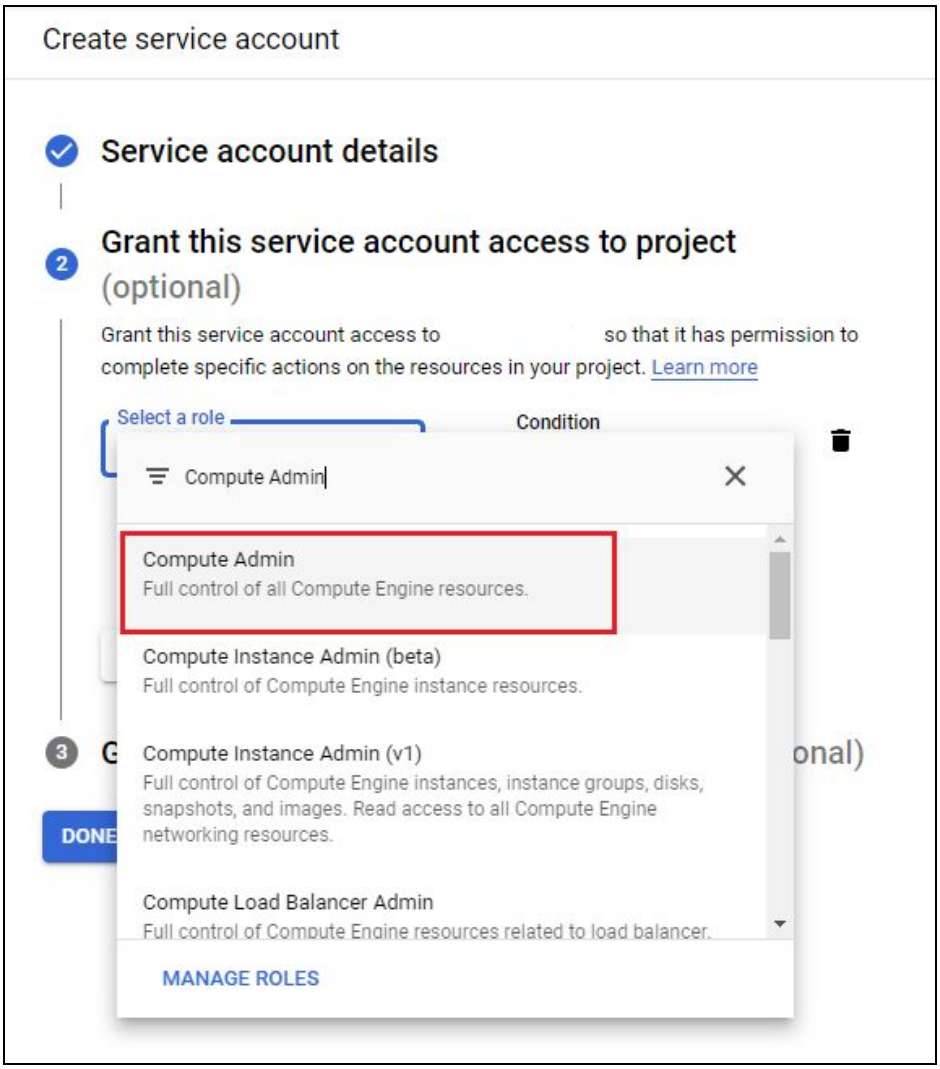
- Enter a **service account name** to display in the Cloud Console. The Cloud Console generates a service account ID based on this name. Edit the ID if necessary. You cannot change the ID later.
- Optional: Enter a description of the service account. Then click on **Create**.



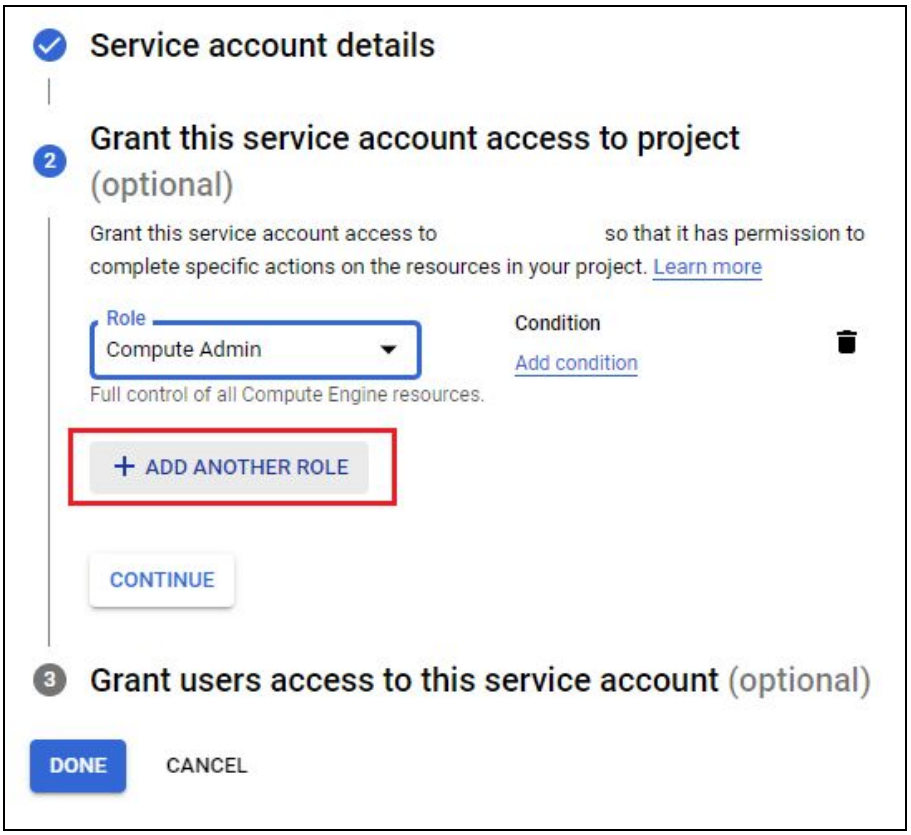
- Under roles, choose one or more IAM roles to grant to the service account on the project. Set below designated roles by clicking on **Select a Role**, then use search bar to find required role and click on **Done**:
    - **Service Account User** - Run operations as the service account.
    - **Service Account Token Creator** - Impersonate service accounts (create OAuth2 access tokens, sign blobs or JWTs, etc).
    - **Compute Admin** - Full control of all Compute Engine resources.
    - **Compute Network Admin** - Permissions to create, modify, and delete networking resources, except for firewall rules and SSL certificates.
    - **Packet Mirroring User** - Use Compute Engine packet mirrorings.
    - **Packet Mirroring Admin** - Specify resources to be mirrored.
    - **Log Writer** - Provides the permissions to write log entries.

- **Monitoring Metric Writer** - Provides write-only access to metrics. This provides exactly the permissions needed by the Cloud Monitoring agent and other systems that send metrics.
- **Storage Admin** - Grants full control of objects and buckets.

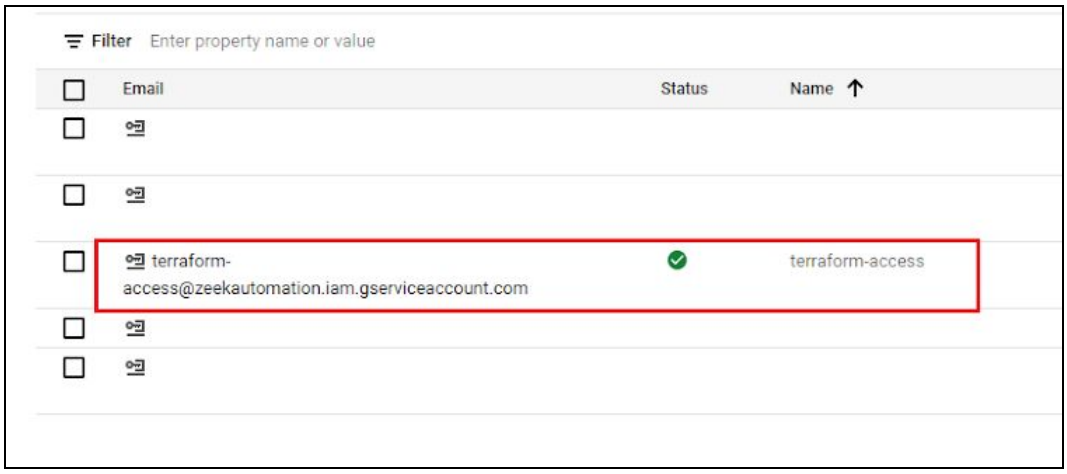



Create service account
Service account details
2
Grant this service account access to project
(optional)
Grant this service account access to so that it has permission to complete specific actions on the resources in your project. Learn more
Select a role
Condition
Compute Admin
Compute Admin
Full control of all Compute Engine resources.
Compute Instance Admin (beta)
Full control of Compute Engine instance resources.
Compute Instance Admin (v1)
Full control of Compute Engine instances, instance groups, disks, snapshots, and images. Read access to all Compute Engine networking resources.
Compute Load Balancer Admin
Full control of Compute Engine resources related to load balancer.
MANAGE ROLES
3
onal)
DONE

- To add other roles, Click on **Add Another Role**, then add all above required roles.



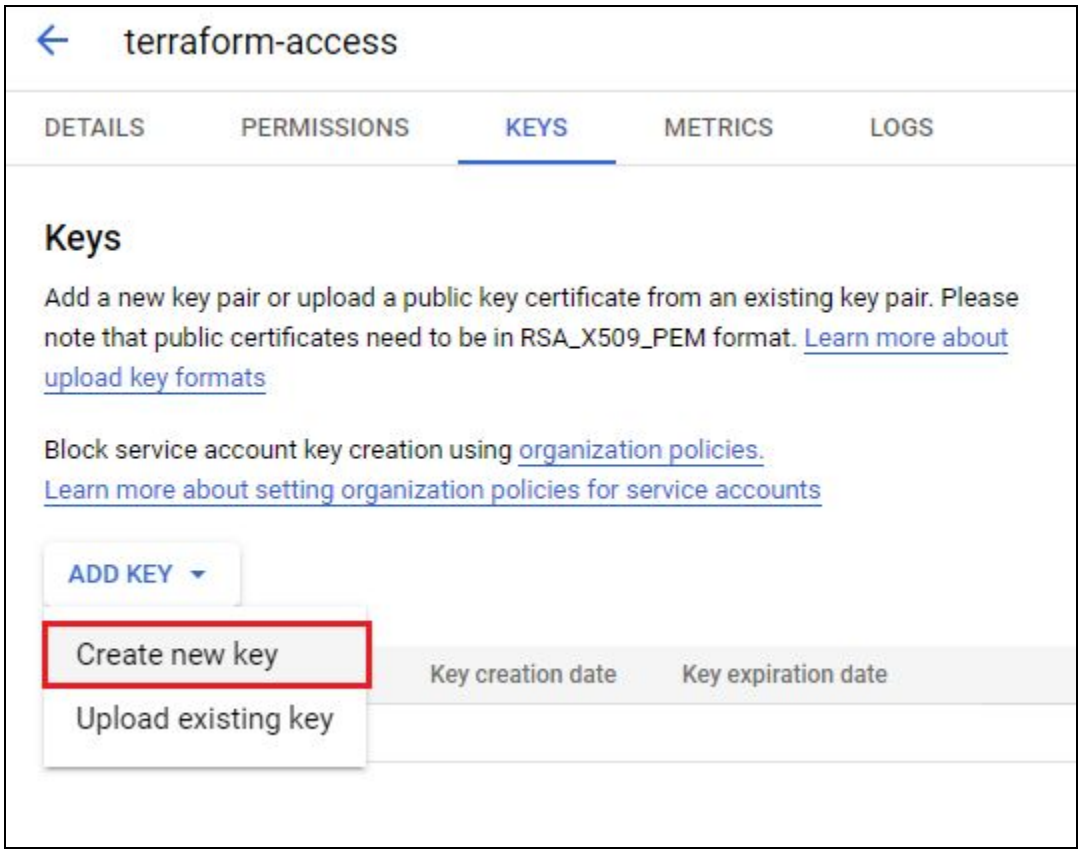
- When you are done adding roles, click **Done.** (After that, you will be redirected back to service accounts console, where your newly created service account is placed in the list of service accounts)



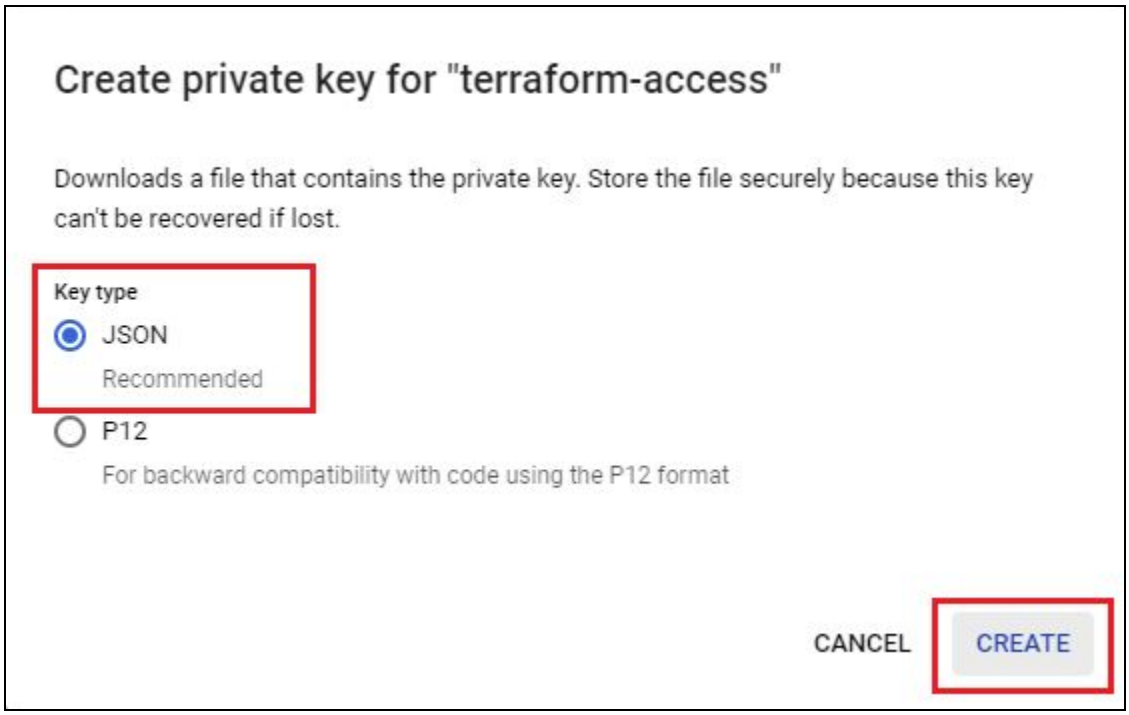
Generate Service account keys

- Locate your new service account from the list of accounts in the console and click on three vertical dots at the rightmost side of your service account row.

- From the drop-down menu in the above step, select **Manage keys** option.
- Then, click on **Add Key > Create new key**.



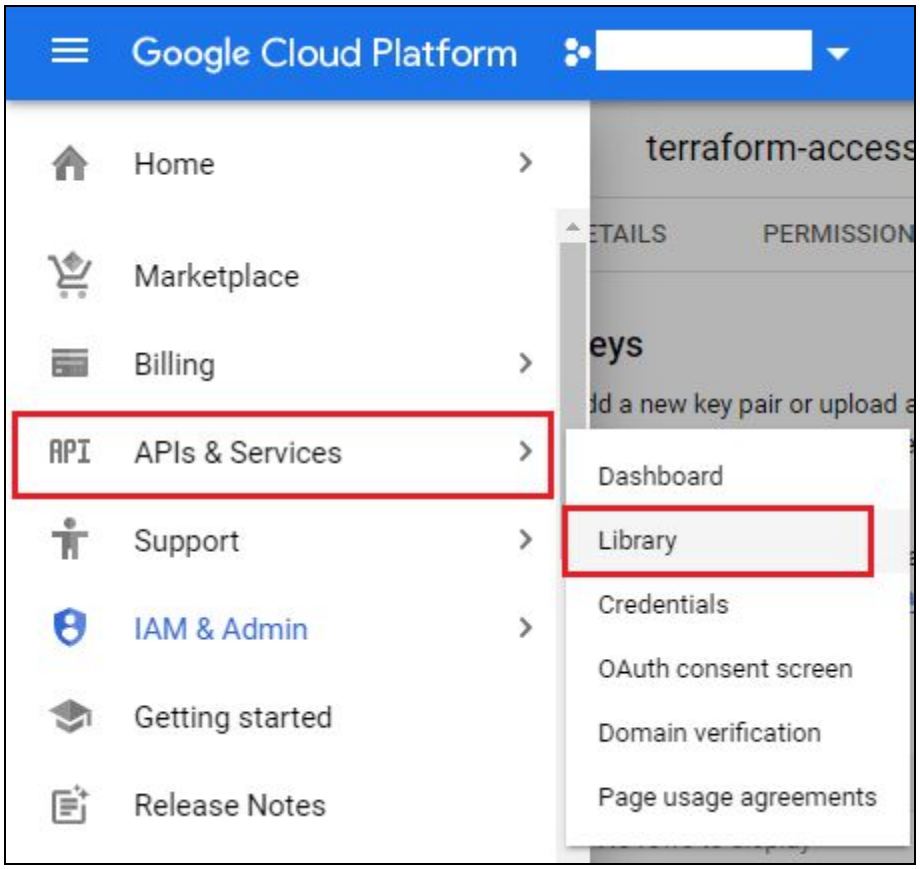
- Select **JSON** type and Click **Create**.



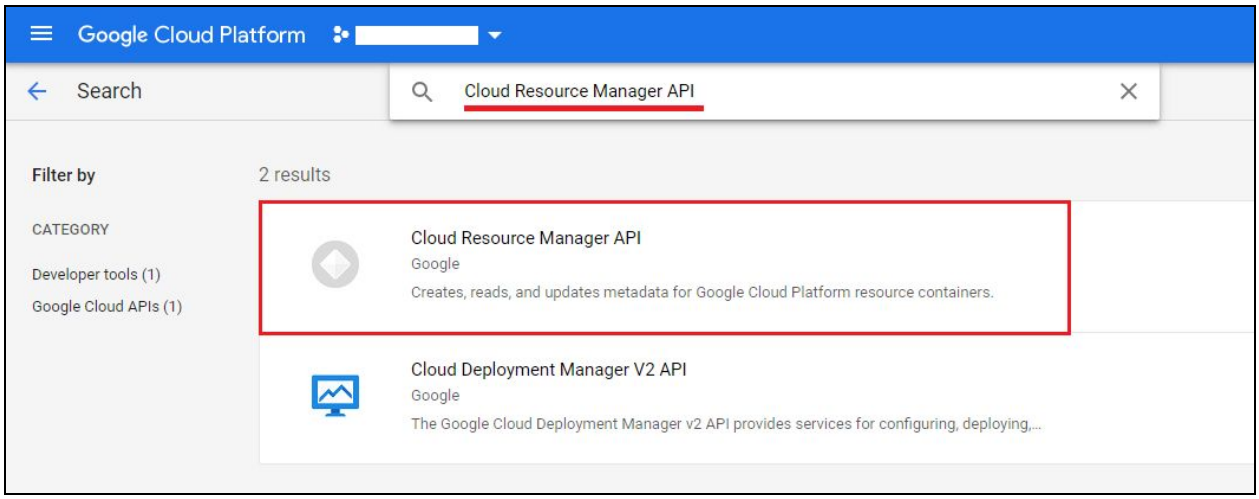
- Save generated Credentials JSON key inside terraform root directory where all scripts and modules are placed.

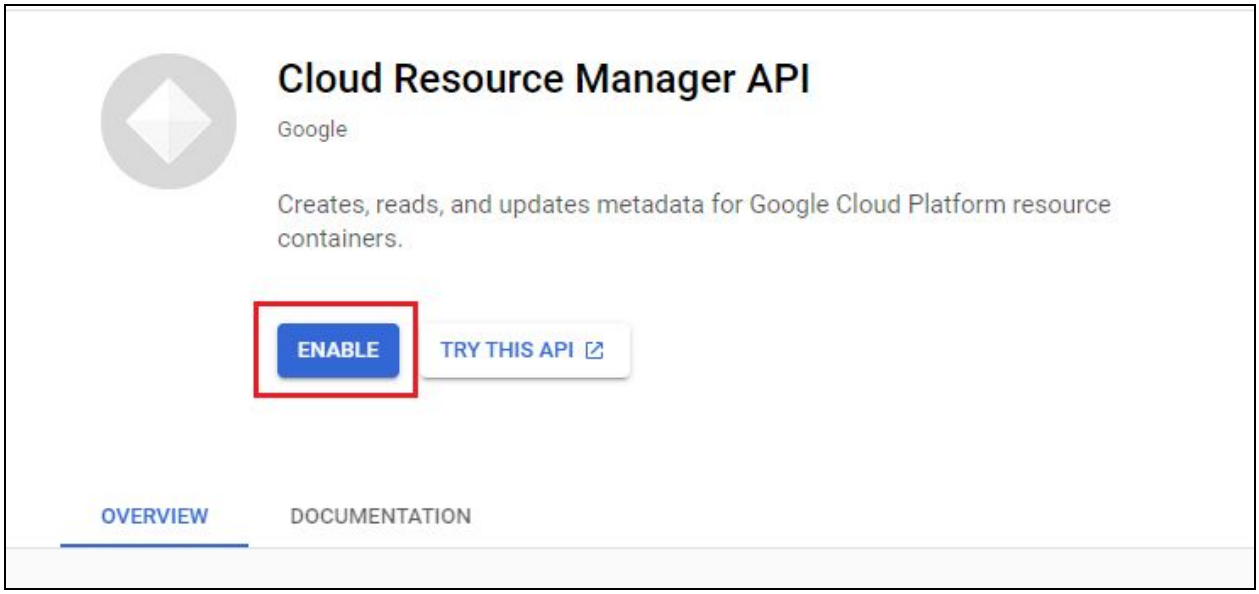
Enable API inside project

- Follow below steps to enable all required API's for terraform to function:
- Click on the Navigation menu (Hamburger icon) located at the top-left corner of the console, and then click on **Library** under **APIs & Services**.



- Search for required API's from below required API list inside search bar and **Enable** it.
    - **Compute Engine API**
    - **Service Usage API**
    - **Cloud Resource Manager API**
    - **Identity and Access Management (IAM) API**
    - **Cloud Logging API**


Google Cloud Platform
Search
Cloud Resource Manager API
Filter by
2 results
CATEGORY
Developer tools (1)
Google Cloud APIs (1)
Cloud Resource Manager API
Google
Creates, reads, and updates metadata for Google Cloud Platform resource containers.
Cloud Deployment Manager V2 API
Google
The Google Cloud Deployment Manager v2 API provides services for configuring, deploying,...


Cloud Resource Manager API
Google
Creates, reads, and updates metadata for Google Cloud Platform resource containers.
ENABLE
TRY THIS API
OVERVIEW
DOCUMENTATION

Getting terraform script ready to execute

Required user input arguments:

1. **credentials:** file name of service account secret key generated during creation of service account located in root directory of terraform scripts.
2. **subnets:** Specify both mirror and collector vpc subnet ip ranges along with their region as key-value pairs inside this list of map variable.
3. **mirror_vpc_network:** Specify mirror vpc id in below format: `'projects/your_project_id/global/networks/your_vpc_name'`
4. **bucket:** Specify unique bucket name where tfstate file will be stored on every execution
5. **prefix:** GCS prefix inside the bucket.State file for your workspace will be stored inside prefix as follows: `<workspace_name>/<folder_name>`

Optional user input arguments:

1. **ip_protocols:** Specify comma-separated protocols that will apply as a filter on mirrored traffic. Possible values: [“tcp”, “udp”, “icmp”]
2. **direction:** Specify comma-separated direction of traffic to mirror. Default value is BOTH, Possible values: “INGRESS”, “EGRESS”, “BOTH”
3. **cidr_ranges:** Specify comma-separated IP CIDR ranges that apply as a filter on the source (ingress) or destination (egress) IP in the IP header. Only IPv4 is supported.

Note: User has to specify atleast one of below optional arguments

4. **mirror_vpc_subnets:** Specify list of regions and their respective subnetwork ids as key-value pairs where key will be region name and value of subnetwork id.
5. **mirror_vpc_instances:** Specify region-wise key-value pairs where key will be region and value will be list of tags set to specific instance in that region.
6. **mirror_vpc_tags:** Specify region-wise key-value pairs where key will be region and value will be list of tags.

Running terraform script

- Verify the installation:

  `$ terraform --version`

- Add modules to backend and show abstract execution plan for review with below command:

  `$ terraform init -backend-config='backend.tfvars'`

  `$ terraform plan -var-file='backend.tfvars'`

- To apply reviewed execution plan from step 2, run below command

  `$ terraform apply -var-file='backend.tfvars' -auto-approve`

- To destroy all the setup, run below command

  `$ terraform destroy -var-file='backend.tfvars' -auto-approve`

Notes:

- Make sure you add required inputs in **terraform.tfvars and backend.tfvars** files with appropriate service account credentials and backend configurations before running terraform scripts.

  Example:

```
credentials = "credentials.json"

subnets = [
  {
    mirror_vpc_subnet_cidr      = "10.128.0.0/20"
    collector_vpc_subnet_cidr   = "10.10.0.0/24"
    collector_vpc_subnet_region = "us-central1"
  },
  {
    mirror_vpc_subnet_cidr      = "10.138.0.0/20"
    collector_vpc_subnet_cidr   = "10.11.0.0/24"
    collector_vpc_subnet_region = "us-west1"
  },
]

mirror_vpc_network = "projects/<project-id>/global/networks/<vpc-name>"

mirror_vpc_subnets = {
  "us-central1" = ["projects/<project-id>/regions/<region>/subnetworks/<subnet-name>"]
  "us-west1"    = ["projects/<project-id>/regions/<region>/subnetworks/<subnet-name>"]
}
```

File: terrafrom.tfvars

```
bucket      = "<your-bucket-name>"
prefix      = "tfstate-files"
credentials = "credentials.json"
```

File: backend.tfvars

- User input sample format is demonstrated in the **examples** folder of the root directory for reference.

  Examples:

```
mirror_vpc_network = "projects/<your-project-id>/global/networks/<vpc-name>"
```

```
mirror_vpc_subnets = {
  "us-central1" = ["projects/<your-project-id>/regions/<region>/subnetworks/<subnet-name>"]
  "us-west1"    = ["projects/<your-project-id>/regions/<region>/subnetworks/<subnet-name>"]
}

mirror_vpc_tags = {
  "us-central1" = ["http-server", "https-server"]
  "us-west1"    = ["mirror-http", "mirror-http"]
}

mirror_vpc_instances = {
  "us-central1" = ["projects/<your-project-id>/zones/<zone>/instances/<instance-name>"]
  "us-west1"    = ["projects/<your-project-id>/zones/<zone>/instances/<instance-name>"]
}
```

```
subnets = [
  {
    mirror_vpc_subnet_cidr      = "192.168.1.0/16"
    collector_vpc_subnet_cidr   = "10.10.10.0/24"
    collector_vpc_subnet_region = "us-west1"
  },
  {
    mirror_vpc_subnet_cidr      = "192.168.2.0/16"
    collector_vpc_subnet_cidr   = "10.10.20.0/24"
    collector_vpc_subnet_region = "us-west1"
  },
]
```

Troubleshooting for API Activation

- If terraform/console restricts you from activating API, confirm that:
    - Current project, which is using this guide, is selected.
    - Billing account is added for current project