Building GCS Buckets and BigQuery Tables with Terraform

Terraform helps data scientists and engineerds to build an infrastructure and to manage its lifecyle.

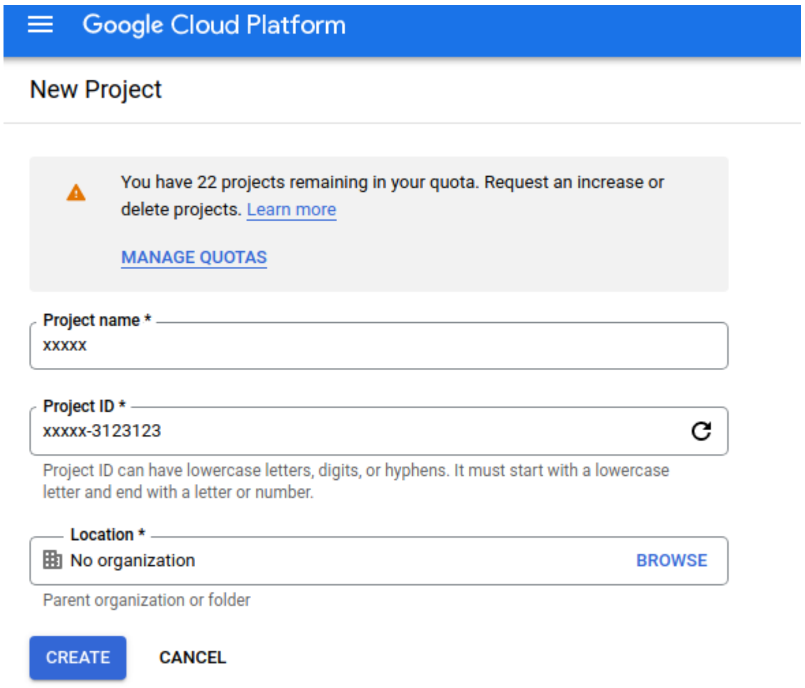
There are two ways to use it: on local & cloud. Below is the description how to install and use it in local to build an infrastructure on Google Cloud Platform (GCP).

Initial Installations: Terraform and Google Cloud SDK

For installing Terraform, pick the proper guide for your operating system provided in their webpage:

- https://www.terraform.io/downloads

Once completing the Terraform installation, you also need to have a GCP account and initiate a project. The ID of the project is importrant to note while proceeding with Terraform.



You have 22 projects remaining in your quota. Request an increase or delete projects. Learn more

Project name *
xxxxx

Project ID *
xxxxx-3123123

Project ID can have lowercase letters, digits, or hyphens. It must start with a lowercase letter and end with a letter or number.

Location *
No organization

Parent organization or folder

The following step is to get key to access and control your GCP project. Pick the project you just created from the pull-down menu on the header of GCP and go to the:

Navigation Menu >> IAM & Admin >> Service Accounts >> Create Service Account

and floow the steps:

- Step1: Assign the name of your preference
- Step2: Pick the role “Viewer” for initiation
- Step3: Skip this optional step for your personal projects.

Then you should see a new account on your Service Accounts list. Click on Actions >> Manage Keys >> Add Key >> JSON to download the key on your local machine.

Next Step is installing Google Cloud SDK to your local machine following the straight forward instractions given in:

https://cloud.google.com/sdk/docs/install-sdk

Then open your terminal (below is a GNU/Linux example) to set the environmental variable on your local machine to link with the key you downloaded (Json file) with the following instructions:

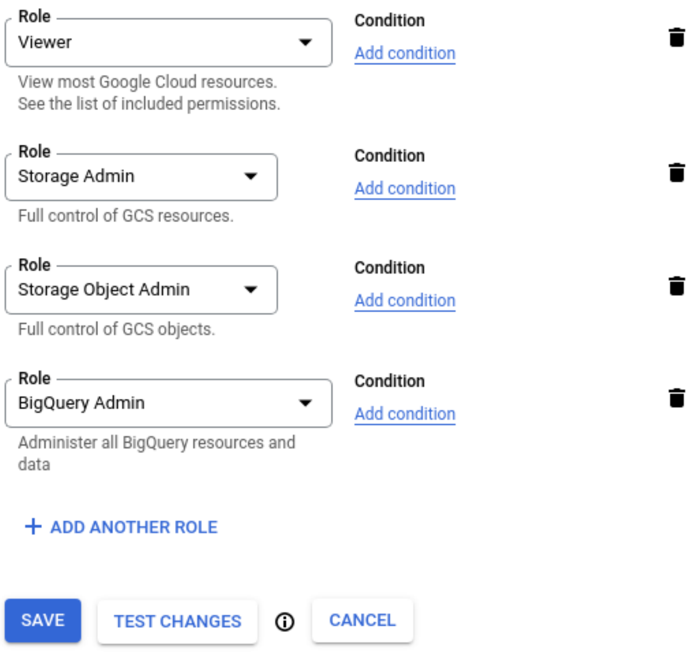
```
export GOOGLE_APPLICATION_CREDENTIALS=/--path to your JSON---/XXXXX-dadas2a4cff8.json

gcloud auth application-default login
```

This redirects you to the browser in order to select your corresponding Google account. Now, your local SDK has credentials to reach and configure your cloud services. However, having these initial authentications, you still need to modify your service account permissions specific for the GCP services you intended to build, namely Google Cloud Storage (GCP) and BigQuery.

Navigation Menu >> IAM & Admin >> IAM

and pick your project to edit its permissions as following:


Role
Viewer
View most Google Cloud resources. See the list of included permissions.
Condition
Add condition

Role
Storage Admin
Full control of GCS resources.
Condition
Add condition

Role
Storage Object Admin
Full control of GCS objects.
Condition
Add condition

Role
BigQuery Admin
Administer all BigQuery resources and data
Condition
Add condition

Next step is enabling the APIs for your project by following the links:

- https://console.cloud.google.com/apis/library/iam.googleapis.com
- https://console.cloud.google.com/apis/library/iamcredentials.googleapis.com

(Take care of the GCP account and the project name while enabling APIs.)

Building GCP Services with Terraform

Completing necessary installations (Teraform and Google Cloud SDK) and authentications, we are ready to build these two GCP services via Terraform from your local machine. Basically, two files are needed to configure the installations: `main.tf` and `variables.tf`. The former one requires the code given below to create GCP services with respect to variables provided in latter (the following code snippet).

```
# The code below is from https://github.com/DataTalksClub/data-engineering-zoomcamp/tree/main/week_1_basics_n_setup/1_terraform_gcp
# --------------------------------------------------

terraform {
  required_version = ">= 1.0"
  backend "local" {}
    google = {
      source  = "hashicorp/google"
    }
  }
}

provider "google" {
  project = var.project
  region = var.region
  // credentials = file(var.credentials)  # Use this if you do not want to set env-var GOOGLE_APPLICATION_CREDENTIALS
}

# Data Lake Bucket
# Ref: https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/storage_bucket
resource "google_storage_bucket" "data-lake-bucket" {
  name          = "${local.data_lake_bucket}_${var.project}" # Concatenating DL bucket & Project name for unique naming
  location      = var.region

  # Optional, but recommended settings:
  storage_class = var.storage_class
  uniform_bucket_level_access = true

  versioning {
    enabled     = true
  }

  lifecycle_rule {
    action {
      type = "Delete"
    }
    condition {
      age = 30  // days
    }
  }

  force_destroy = true
}

# DWH
# Ref: https://registry.terraform.io/providers/hashicorp/google/latest/docs/resources/bigquery_dataset
resource "google_bigquery_dataset" "dataset" {
  dataset_id = var.BQ_DATASET
  project    = var.project
  location   = var.region
}
```

The code for `variables.tf`:

```
# The code below is from https://github.com/DataTalksClub/data-engineering-zoomcamp/tree/main/week_1_basics_n_setup/1_terraform_gcp
# The comments are added by the author

locals {
  data_lake_bucket = "BUCKET_NAME"  # Write a name for the GCS bucket to be created
}

variable "project" {
  description = "Your GCP Project ID"   # Don't write anything here: it will be prompted during installation
}

variable "region" {
  description = "Region for GCP resources. Choose as per your location: https://cloud.google.com/about/locations"
  default = "europe-west6"  # Pick a data center location in which your services will be located
  type = string
}

variable "storage_class" {
  description = "Storage class type for your bucket. Check official docs for more info."
  default = "STANDARD"
}

variable "BQ_DATASET" {
  description = "BigQuery Dataset that raw data (from GCS) will be written to"
  type = string
  default = "Dataset_Name" # Write a name for the BigQuery Dataset to be created
}
```

Once the files given above are located to a folder, it is time to execute them. THere few main commands for Terraform CLI:

Main commands:

- **init:** prepares the directory by adding necessary folders and files for following commands
- **validate: c**hecks the existing configuration if it is valid
- **plan:** shows planned changes for the given configuration
- **apply:** creates the infrastructure for the given configuration
- **destroy:** destroys existing infrastructure

THe init, plan and apply commands will give the following outputs (shortened):

```
x@y:~/-----/terraform$ terraform init

Initializing the backend...

Successfully configured the backend "local"! Terraform will automatically
use this backend unless the backend configuration changes.
.
.
.
```

```
x@y:~/-----/terraform$ terraform plan
var.project
  Your GCP Project ID

  Enter a value: xxx-yyy # write yor GCP project ID here
```

```
x@y:~/-----/terraform$ terraform apply

var.project
  Your GCP Project ID

  Enter a value: xxx-yyy # write yor GCP project ID here


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:
.
.
.
```

After executing the three simple codes above, you will see A new GCS bucket and BigQuery table in your GCP account.