App Tosca

Contents

[App Tosca 1](#_Toc156294034)

[Architecture 2](#_Toc156294035)

[Environment 3](#_Toc156294036)

[Pre action 4](#_Toc156294037)

[This command will download the necessary providers and plugins for Terraform. Base action 5](#_Toc156294038)

[Script 5](#_Toc156294039)

[Developer Environment 5](#_Toc156294040)

Architecture

Immagine che contiene testo, schermata, software, Icona del computer

Descrizione generata automaticamente

Environment

Development

* 2 Compute Engine VM for developer support
* 1 TB Cloud Storage / 1 TB Big Query
* 3 Cloud Run services for input, inference and output orchestrated by Cloud Workflows
* 160 hour/month of Vertex AI Training (considering worst case 8 hours x 20 days)
* Serving 1 model 24x7 using Vertex AI Endpoint (version latest)
* This is considered as a pass-through system where data has a short retention. Data with long retention continues to reside on your Elastic on prem.

Production

* 1 Compute Engine VM for developer support
* 2 TB Cloud Storage / 1 TB Big Query
* 3 Cloud Run services for input, inference and output orchestrated by Cloud Workflows
* 1 Vertex AI Training at quarter
* Serving 2 models 24x7 using Vertex AI Endpoint (version latest and latest-1)
* This is considered as a pass-through system where data has a short retention. Data with long retention continues to reside on your Elastic on prem.

Pre action

1. Create a Google Cloud account

To get started, you need to create a Google Cloud account. You can do this by visiting the Google Cloud Web site and clicking the "Get Started" button.

2. Enable the necessary services

To use Terraform, you need to enable the necessary Google Cloud services. You can do this by visiting the Google Cloud Administration Console and clicking on "Services." Search for the following services and enable them:

Compute Engine

Cloud Storage

BigQuery

Vertex AI

Cloud Run

Cloud Workflows

3. Install Terraform

Terraform is available for Windows, macOS and Linux. To install it, follow the instructions provided on the Terraform Web site.

4. Create a directory for the project

Create a directory in which to store the Terraform configuration for your project. For example, you can create a directory named "my-project."

5. Initialize Terraform

Run the following command to initialize Terraform in your project directory:

terraform init

This command will download the necessary providers and plugins for Terraform. Base action

Here are the steps to describe the architecture in Google Cloud using Terraform:

1. **Install Terraform**: Ensure that Terraform is installed. If not, you can download it from the official website and follow the installation instructions.
2. **Set up Google Cloud SDK**: Make sure you have Google Cloud SDK installed and configured with the necessary permissions.
3. **Create a Terraform file**: Create a new file with a .tf extension to define your infrastructure. Here’s an example of how you can define the resources you mentioned.
4. **Initialize Terraform**: Run terraform init to initialize the working directory containing your Terraform configuration files.
5. **Plan and Apply:** Runn terraform plan to see the changes that will be made to your infrastructure.

Script

Developer Environment

resource "google\_compute\_instance" "vm\_instance" {

count = 2

name = "dev-vm"

machine\_type = "n1-standard-1"

zone = "us-central1-a"

boot\_disk {

initialize\_params {

image = "debian-cloud/debian-9"

}

}

network\_interface {

network = "default"

access\_config {

// Ephemeral IP

}

}

}

resource "google\_storage\_bucket" "bucket" {

name = "<BUCKET\_NAME>"

location = "<LOCATION>"

storage\_class = "<STORAGE\_CLASS>"

}

resource "google\_bigquery\_dataset" "dataset" {

dataset\_id = "<DATASET\_ID>"

friendly\_name = "<FRIENDLY\_NAME>"

description = "<DESCRIPTION>"

location = "<LOCATION>"

default\_table\_expiration\_ms = <EXPIRATION\_TIME>

delete\_contents\_on\_destroy = true

}

resource "google\_cloud\_run\_service" "service" {

count = 3

name = "cloud-run-service-${count.index}"

location = "<LOCATION>"

template {

spec {

containers {

image = "<IMAGE\_NAME>"

}

}

}

}

resource "google\_workflows\_workflow" "workflow" {

name = "<WORKFLOW\_NAME>"

source\_contents = <<EOF

# Cloud Workflows

# Define the workflow

main:

steps:

- initialize:

assign:

- project: "<PROJECT\_ID>"

- location: "<LOCATION>"

- create\_service:

call: googleapis.run.v1.services.create

args:

parent: ${"projects/" + project + "/locations/" + location}

requestBody:

apiVersion: serving.knative.dev/v1

kind: Service

metadata:

name: <SERVICE\_NAME>

spec:

template:

spec:

containers:

- image: <IMAGE\_NAME>

- create\_endpoint:

call: googleapis.ml.v1.projects.locations.endpoints.create

args:

parent: ${"projects/" + project + "/locations/" + location}

requestBody:

displayName: "<ENDPOINT\_NAME>"

serviceSpec:

machineType: "<MACHINE\_TYPE>"

acceleratorConfig:

count: <ACCELERATOR\_COUNT>

type: "<ACCELERATOR\_TYPE>"

deploymentUri: <DEPLOYMENT\_URI>

containerSpec:

imageUri: <IMAGE\_URI>

EOF

}

resource "google\_ai\_platform\_training\_job" "training\_job" {

name = "<TRAINING\_JOB\_NAME>"

project = "<PROJECT\_ID>"

region = "<REGION>"

scale\_tier = "<SCALE\_TIER>"

master\_type = "<MASTER\_TYPE>"

worker\_type = "<WORKER\_TYPE>"

worker\_count = <WORKER\_COUNT>

training\_input {

scaleTier = "<SCALE\_TIER>"

runtimeVersion = "<RUNTIME\_VERSION>"

pythonVersion = "<PYTHON\_VERSION>"

jobDir = "<JOB\_DIR>"

packageUris = ["<PACKAGE\_URIS>"]

pythonModule = "<PYTHON\_MODULE>"

args = ["<ARGS>"]

}

}

resource "google\_ai\_platform\_model" "model" {

name = "<MODEL\_NAME>"

project = "<PROJECT\_ID>"

region = "<REGION>"

description = "<DESCRIPTION>"

online\_prediction\_logging = true

labels = {

key1 = "value1"

key2 = "value2"

}

}

resource "google\_ai\_platform\_endpoint" "endpoint" {

name = "<ENDPOINT\_NAME>"

project = "<PROJECT\_ID>"

region = "<REGION>"

display\_name = "<DISPLAY\_NAME>"

traffic\_split {

percent = 100

latest\_revision = true

}

deployed\_models {

model = "<MODEL\_NAME>"

deployment\_uri = "<DEPLOYMENT\_URI>"

machine\_type = "<MACHINE\_TYPE>"

}

}

Product Environment