Deploying a Container to Cloud Run using Terraform on GCP

Overview

In this task, we are deploying an httpd Docker container to **Cloud Run** using **Terraform** on **Google Cloud Platform (GCP)**. Cloud Run is a fully managed compute platform that automatically scales your containerized applications. Terraform helps us automate the provisioning process as Infrastructure as Code (IaC).

Prerequisites

- A GCP project (qwiklabs-gcp-02-908d983288f8)
- Terraform installed locally
- Enabled **Cloud Run Admin**, **IAM**, and **Service Account User** roles
- Docker image: httpd (Apache server)
- GCP credentials configured using gcloud auth application-default login
- Billing enabled for the project

Terraform Project Structure

```
cloud-run-deploy/
|
├── main.tf                 # Main Terraform script
└── terraform.tfstate       # Auto-generated after `apply`
```

Terraform Script (main.tf)

- Vim main.tf

```
terraform {
  required_providers {
    google = {
      source = "hashicorp/google"
    }
  }
}

provider "google" {
  project = "sidhant-19102"
  region  = "us-central1"
  zone    = "us-central1-a"
}

resource "google_cloud_run_v2_service" "default" {
  name     = "cloudrun-service-containers"
  location = "us-central1"

  deletion_protection = false

  template {
    containers {
      image = "docker.io/bharathshetty4/supermario"

      ports {
        container_port = 8080
      }
    }
  }
}

resource "google_cloud_run_service_iam_member" "public_access" {
  service  = google_cloud_run_v2_service.default.name
  location = google_cloud_run_v2_service.default.location
  role     = "roles/run.invoker"
  member   = "allUsers"
}

output "cloud_run_url" {
  value = google_cloud_run_v2_service.default.uri
}
```

Steps to Deploy

1. Initialize Terraform

- Terraform init

```
PS C:\Users\botes\Gcp\Cloud-run> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/google...
- Installing hashicorp/google v6.33.0...
- Installed hashicorp/google v6.33.0 (signed by HashiCorp)

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.

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.
```

2. Review the Plan

- **Terraform plan**

```
PS C:\Users\botes\Gcp\Cloud-run> terraform plan

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:

  # google_cloud_run_v2_service.default will be created
  + resource "google_cloud_run_v2_service" "default" {
      + conditions                = (known after apply)
      + create_time               = (known after apply)
      + creator                   = (known after apply)
      + delete_time               = (known after apply)
      + deletion_protection       = false
      + effective_annotations     = (known after apply)
      + effective_labels          = {
          + "goog-terraform-provisioned" = "true"
        }
```

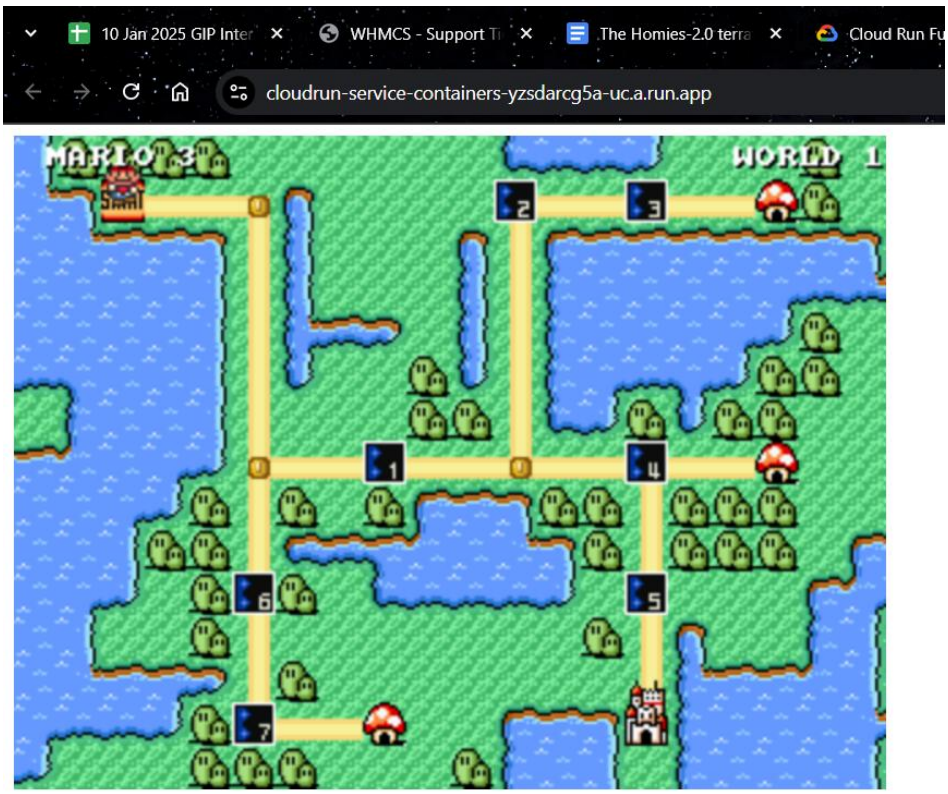
3. Apply the Configuration

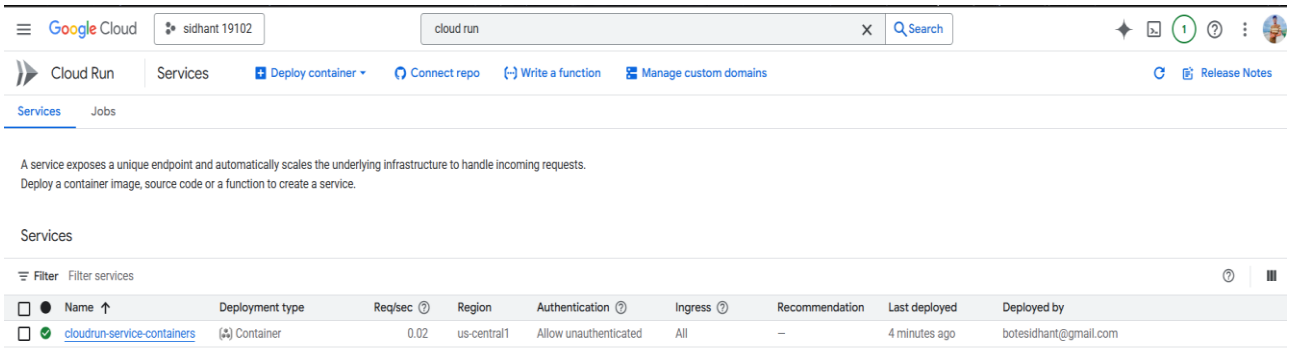
- **Terraform apply**

```
Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + cloud_run_url = (known after apply)

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

google_cloud_run_v2_service.default: Creating...
google_cloud_run_v2_service.default: Still creating... [10s elapsed]
google_cloud_run_v2_service.default: Still creating... [20s elapsed]
google_cloud_run_v2_service.default: Still creating... [30s elapsed]
google_cloud_run_v2_service.default: Still creating... [40s elapsed]
google_cloud_run_v2_service.default: Creation complete after 49s [id=projects/sidhant-19102/locations/us-central1/services/cloudrun-service-cont
ainers]
google_cloud_run_service_iam_member.public_access: Creating...
google_cloud_run_service_iam_member.public_access: Still creating... [10s elapsed]
google_cloud_run_service_iam_member.public_access: Creation complete after 12s [id=v1/projects/sidhant-19102/locations/us-central1/services/clou
drun-service-containers/roles/run.invoker/allUsers]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

cloud_run_url = "https://cloudrun-service-containers-yzsdarcg5a-uc.a.run.app"
```

4. Access the Cloud Run Service

Visit the URL in a browser to see the Apache HTTP server page





Conclusion

With this configuration, we have successfully:

- Automated Cloud Run deployment using Terraform
- Pulled and deployed a Docker container from Docker Hub
- Exposed the container via HTTP on port 8080

This approach is scalable, reusable, and follows best practices for infrastructure automation.