**Create & Setup for Google Cloud Infrastructure using Terraform**

**Project assignments:**

You must create a Terraform deployment script that achieves the following requirements:

* Private CloudSQL Db (Postgres or Mysql)
* Private GCE instance on a dedicated VPC (not default VPC)
* Build and Deploy a sample application on the above GCE instance
* Establish GCE and Private SQL connecting with SQL Auth Proxy using a sidecar container
* Establish CI/CD pipeline (CloudBuild/CloudDeploy, Github Actions or any tools)
* Expose the sample application via Load balancer

Explain why you chose your application and how it interacts with the SQL Database and the networking layers.

Please note that the application deployment must be done within terraform such that applying terraform is sufficient in configuring the application as well.

**Step1: Install Terraform**

In google Cloud Shell, configure your Cloud Shell environment to use Terraform by installing it with the appropriate package.

1.**Download Terraform by running the following command**:

wget https://releases.hashicorp.com/terraform/1.2.7/terraform\_1.2.7\_linux\_amd64.zip

2.**Unzip Terraform by running the following command:**

unzip terraform\_1.2.7\_linux\_amd64.zip

3. **Set the PATH environmental variable to Terraform binaries**:

export PATH="$PATH:$HOME/terraform"

cd $HOME

source ~/.bashrc

4. **Confirm the Terraform installation by running the following command**:

terraform --version

output:

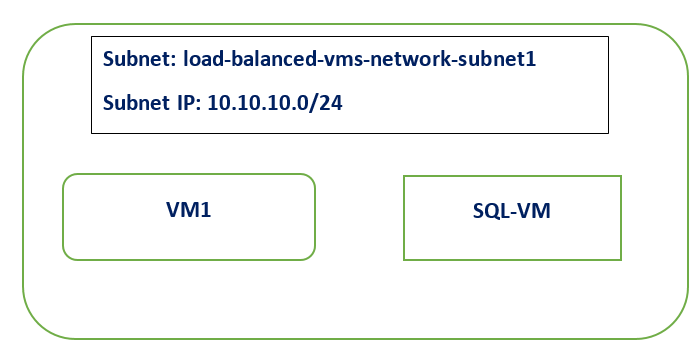
Terraform v1.4.6

on linux\_amd64

+ provider registry.terraform.io/hashicorp/google v4.65.2

**Step2: Create the custom-VPC network along with its firewall rule**

**VPC: load-balanced-vms-network**

****

.

Virtual Private Cloud (VPC) provides networking functionality to Compute Engine virtual machine (VM) instances, Google Kubernetes Engine (GKE) clusters environment. VPC provides networking for your cloud-based resources and services that is global, scalable, and flexible. Virtual Private Cloud (VPC) networks are global resources. Each VPC network consists of one or more IP address ranges called subnets. Subnets are regional resources, and have IP address ranges associated with them.

1. Create a dedicated VPC
2. Create subnetwork
3. Create a firewall rule to allow HTTP, SSH, RDP and ICMP traffic on VPC

**Step 3: Provision a GCE instance and deploy the application**

Google Compute Engine**(GCE) is an infrastructure as a service (IaaS) offering that allows clients to run workloads on Google's physical hardware.**

**Create a GCE instance need as bellow details:**

* **Name of the instance**
* **zone**
* **Region**
* **machine\_type**
* **boot\_disk**
* **network\_interface**

**Step 4: Provision a SQL instance(MYSQL)**

Each Cloud SQL instance is powered by a virtual machine (VM) running on a host Google Cloud server. Each VM operates the database program, such as MySQL Server, PostgreSQL, or SQL Server, and service agents that provide supporting services, such as logging and monitoring.

**Create a SQL instance(MYSQL) need as bellow details:**

* **Name of the instance**
* **zone**
* **Region**
* **machine\_type**
* **boot\_disk**
* **network\_interface**

**Step 5: Configure the SQL Auth Proxy**

1. **Download the Cloud SQL Auth proxy:**

curl -o cloud-sql-proxy <https://storage.googleapis.com/cloud-sql-connectors/cloud-sql->

proxy/v2.2.0/cloud-sql-proxy.linux.amd64

1. **Make the Cloud SQL Auth proxy executable:**

chmod +x cloud-sql-proxy

1. **IAM role for access Cloud SQL instances in a project:**

cloudsql.instances.connect

1. **Start the Cloud SQL Auth proxy**

Some possible Cloud SQL Auth proxy invocation strings:

Using Cloud SDK authentication:

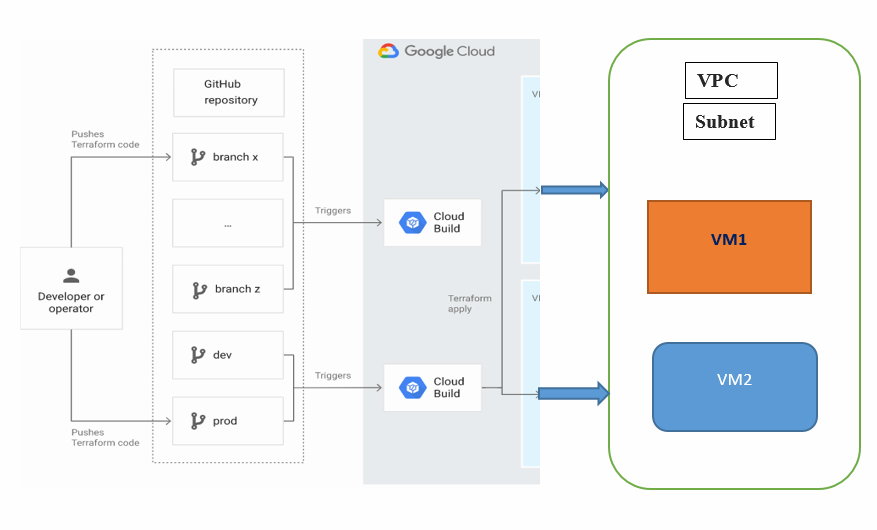
./cloud-sql-proxy --port 3306 *INSTANCE\_CONNECTION\_NAME*

1. **Using a service account and explicitly including the name of the instance connection**

./cloud-sql-proxy \  
--credentials-file *PATH\_TO\_KEY\_FILE* *INSTANCE\_CONNECTION\_NAME* &

**Step 6: Configure CI/CD pipeline**

**Expose the sample application via a Load Balancer**

****

**Create a CI/CD pipeline so we need as bellow details:**

* Set up your GitHub repository.
* Configure Terraform to store state in a Cloud Storage bucket.
* Grant permissions to your Cloud Build service account.
* Connect Cloud Build to your GitHub repository.
* Change your environment configuration in a feature branch.
* Promote changes to the development environment.

**Cloud Build:**

Cloud Build is a service that executes your builds on Google Cloud. Cloud Build can import source code from a variety of repositories or cloud storage spaces, execute a build to your specifications, and produce artefacts such as Docker containers or Java archives.

We need configure for Create a trigger in Cloud Build as below:

* Add a trigger name, such as push-to-branch.
* In the Event section, select Push to a branch.
* In the Source section, select. \* in the Branch field.
* Click Create.

**Terraform Code Overview:**

1. **main.tf**
2. **variable.tf**
3. **provider.tf**