Challenge 1: Creating a 3-tier application.

Environment: GCP (Google Cloud Platform)

Overview: We are going to deploy a Flask application on GCP with the help of different Google Cloud Resources/Services.

Resources: Google Compute Engine, Cloud SQL, Load balancing, and Cloud NAT.

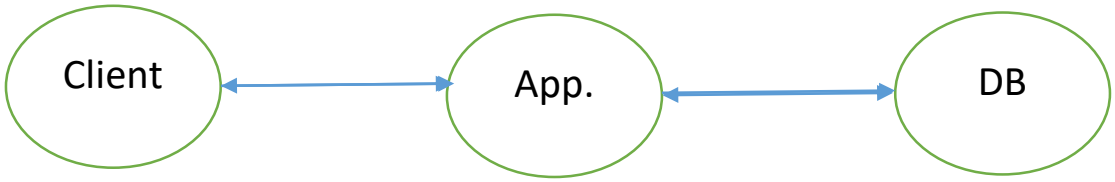
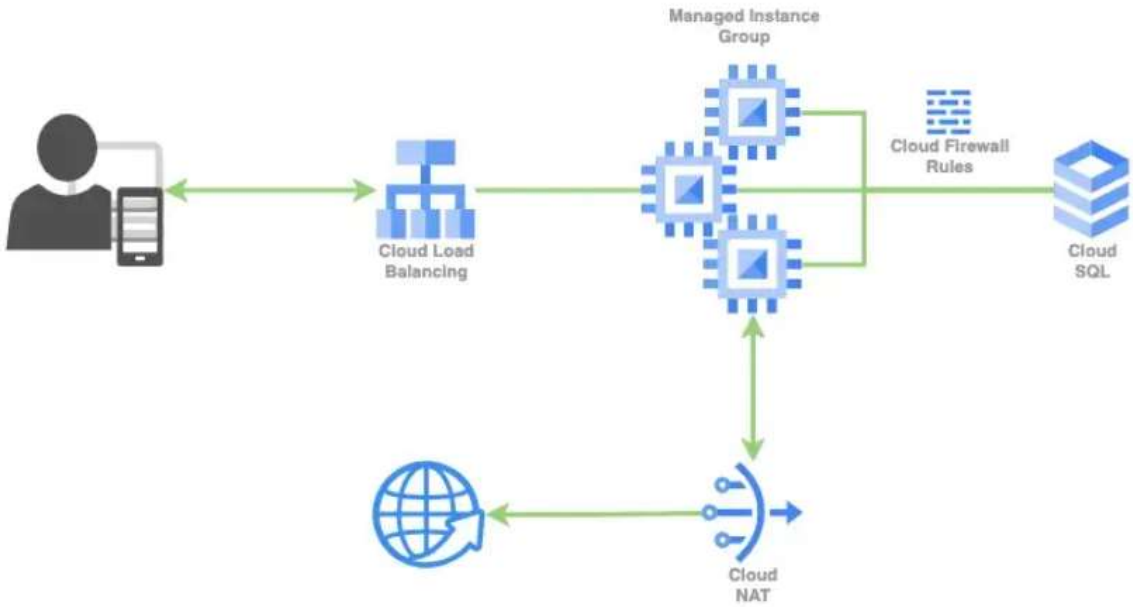


Fig. 3-tier Model

Architecture:



Steps:

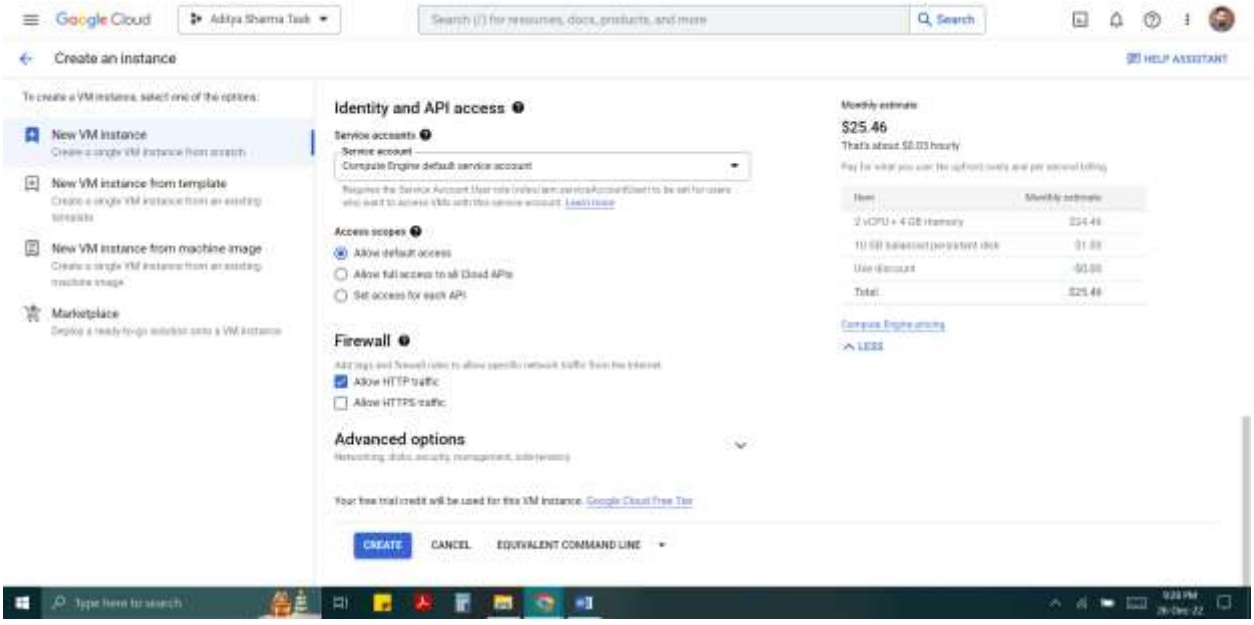
- VM (Virtual Machine) Setup
- NAT (Network Address Translation) Configuration

- Code Integration
- Load-Balancer Setup
- Cloud SQL Integration
- VPC (Virtual Private Cloud) Network Setting
- Testing

Step 1: VM (Virtual Machine) Setup

We are going to host our application in a VM instance, which is similar to running your developed application on your local machine.

Let's start by creating a VM instance using the Compute Engine service in Google Cloud. Go to Google cloud console, search "GCE" then navigate to VM instances and click on Create Instance.



Note:

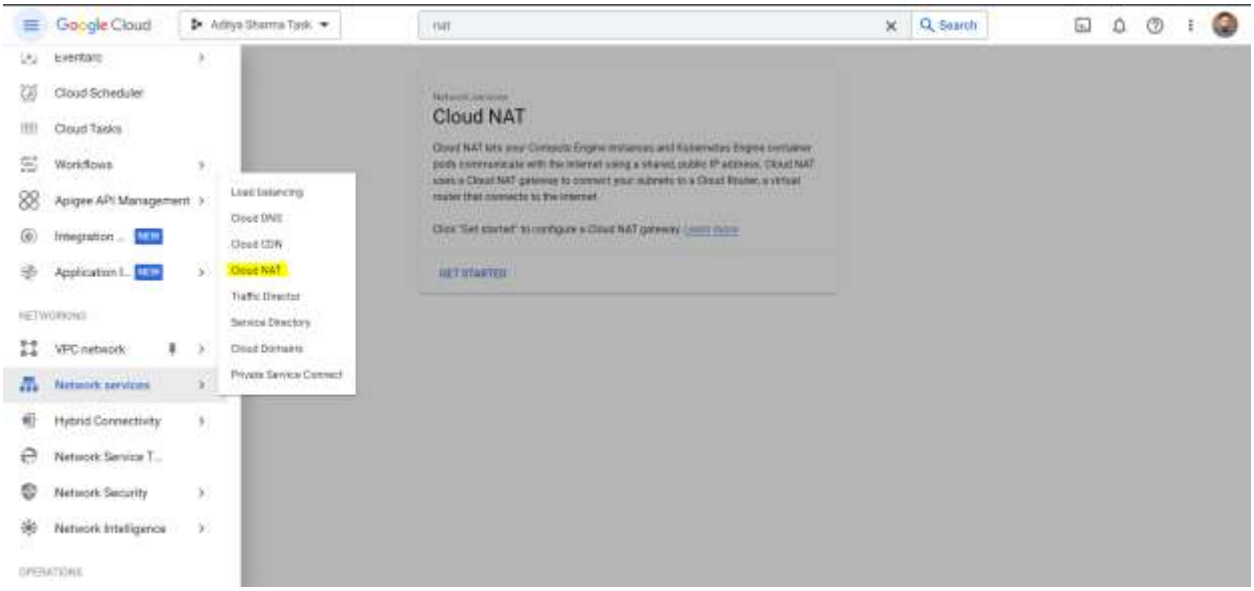
a) Give a proper name to your instance and Allow full access to all cloud APIs and then left everything as default.
b) Under the firewall setting section enable the Allow http checkbox.

c) Under Advanced options select Network Interfaces now disable the External IPv4 address.

Step 2: NAT CONFIGURATION

Since we have disabled the external IP in our VM, it won't be able to connect to the internet for different purposes like package installation etc. so here we need to configure NAT so that our VM can access internet.

First, Go to search bar at the top and search Network services and click on it. Click on Cloud NAT option on the left of the screen.



Enter a Gateway name and then select the network in which our VM is present and here we can select a region of our choice. Create a router and give it a name as - flask-mig-cr, click create and then
again click create to configure the NAT with the default network.

Step 3: Code Integration

In this step, we will see how we can integrate the application code inside our VM, just like when we develop our application locally. You need to

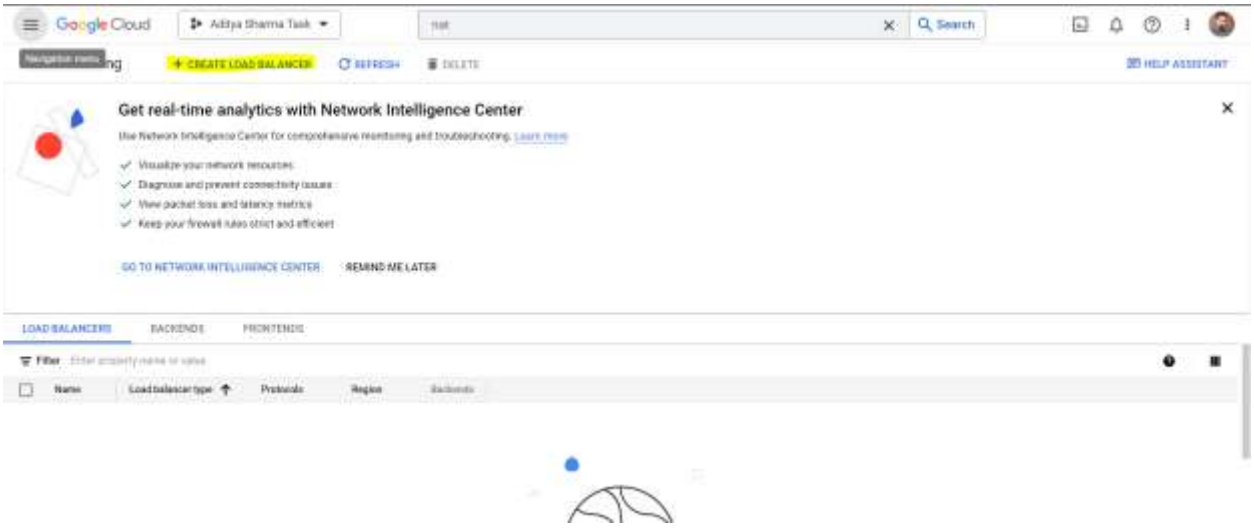
install and configure your environment first, such as programming language dependencies and compatibility, before we can work on it. Transfer all the files from Google cloud shell to the virtual machine using the gcloud compute scp command.

cloud compute scp --recurse /path/of/folder/at/cloud/shell vmuser@hostname:/path/to/folder

Step 4: Load-Balancer Setup

We are going to setup end to end load balancer for VM based workloads. Here we aren't able to directly attach VM to the Load-Balancer so we need to create a UMIG (Unmanaged Instance Group) or MIG (Managed Instance Group) for linking our application to the Load-Balancer. We will be using HTTP(S) Load Balancing to distribute traffic on the VMs of the unmanaged instance group (in our case, it's just a single VM).

Network Services -> Load Balancing



Note:

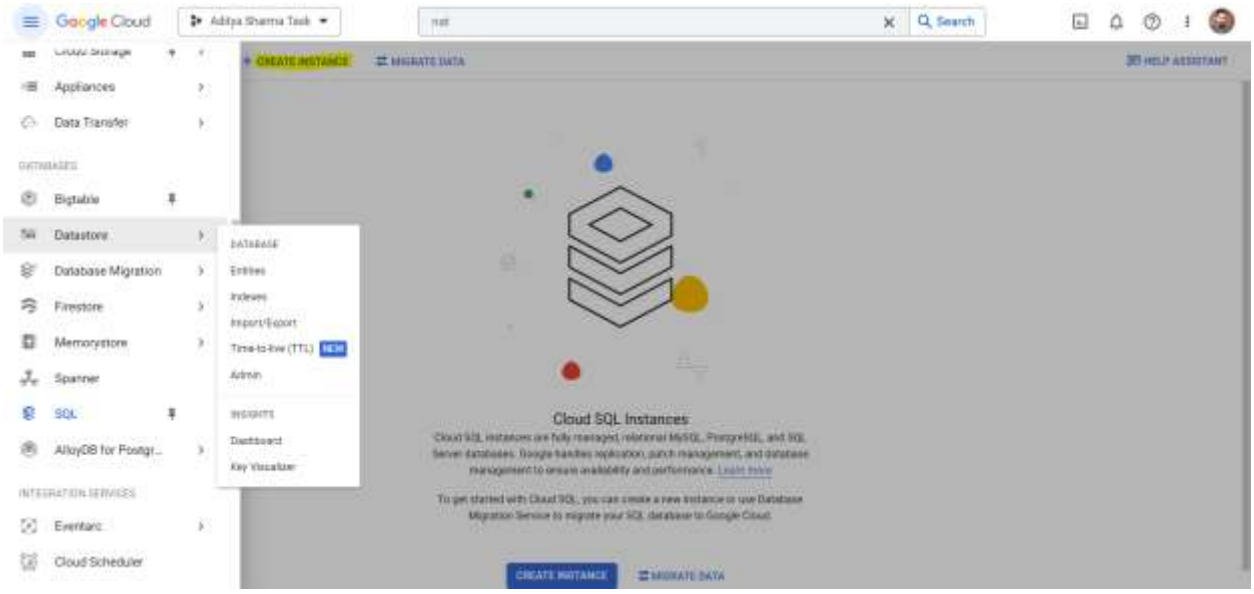
a) Uncheck the enable CDN option as CDN is not needed.

b) Inspect the load balancer and get the IP address which will be used to access the application.

Step 5: Cloud SQL Integration

Now we need some kind of a database to store the data. Here we are storing a structured data thus we need a relational database. In Google Cloud we have a fully-managed database service known as Cloud SQL where we can store our data.

Cloud SQL Services -> Create Instance -> Enter the Instance ID of your cloud SQL instance and generate a password



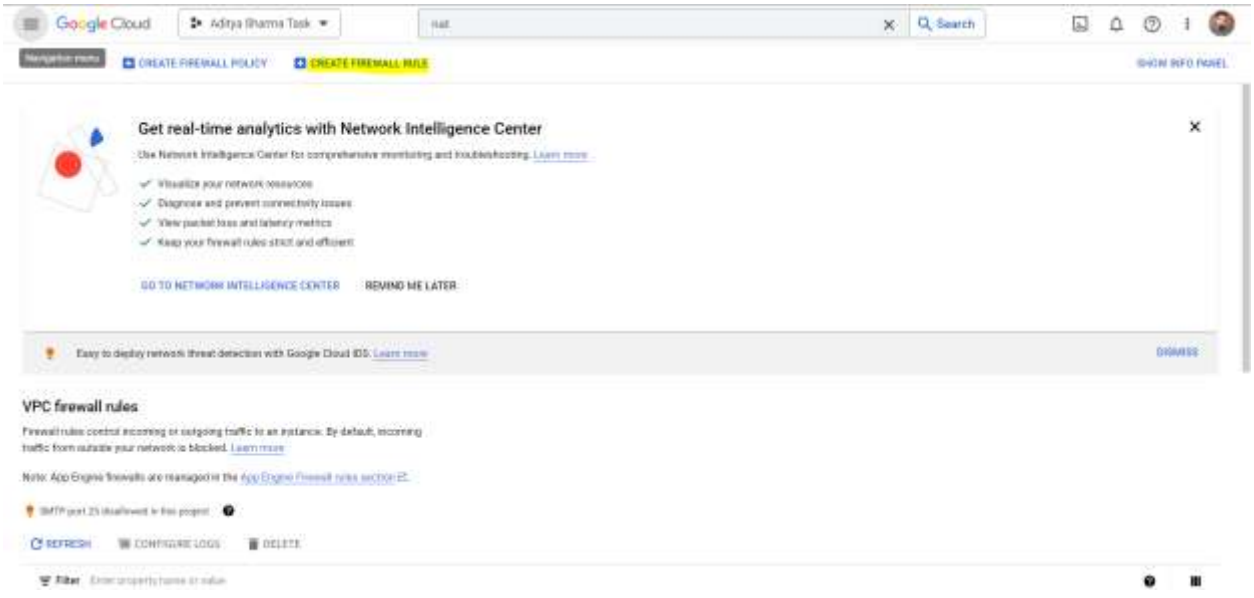
Note:

a) From connection and choose private ip which is a more secure way of accessing the database as it will not have a public ip address and connection from outside are not possible.
b) Enter the frontend-IP (application IP) of load balancer as we will post the data using our application.

Step 6: VPC (Virtual Private Cloud) Network Setting

Use the following VPC path –

VPC Network -> Firewall -> Create Firewall Rule



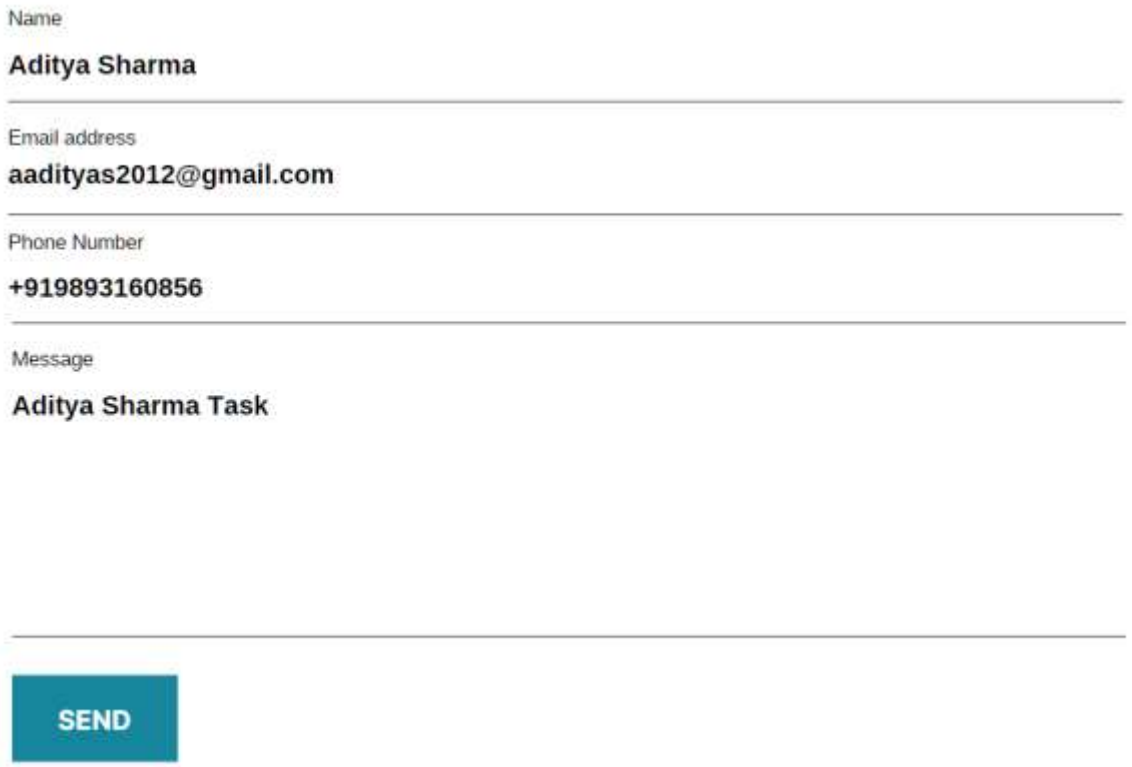
Note:

a) Enter the name default-allow-db as we will be using Cloud SQL later to connect at this port 3306.
b) Targets: All instances in the network only for the simplicity in the real world it is better to use specified target tags for better security.

Step 7: Testing

Now the final phase comes into the picture. Start the Server using command <Python_File_Name>.py

Then Go to the contacts page of your website and fill the form with the details.



After that we can login to our database by following commands and provide the password-

mysql --host=IP_address of CloudSQL instance -u 'username' –p

show databases;
use database_name; #Replace database_name with codingthunder as in my case
show tables;
select * from contacts;

Conclusion: This application has Reusability/Reproducibility feature. We can easily deploy this with the help of following tool-

Deploy Tool: Google Devstack