Introduction

This is a simple gallery web application that is using the following technologies:

- Python Flask App
- GCP hosting
- Terraform

Demo Video: (https://youtu.be/GnCnnmCHBEw)

This application utilizes terraform for the GCP initilization and tear down. A Cloud SQL database instance is used to store user and photo information. A bucket is used to store the pictures for the web application allowing them to be retrieved, downloaded, or deleted.

All terraform files can be found here on the github. These files include:

- **app-deploy.tf** : configures the vm instance on gcp for the application along with application source files (flask app)
- **bucket.tf** : confirgures and creates the bucket used to store and retrieve images for the web app
- **database.tf** : configures the Cloud SQL instance that the flask app utilizes and connects the db instance to the vm instance via a vpc
- **network.tf** : configures the vpc_network, subnet, and private_vpc_connection for the database instance. also configures the firewall rules to allow traffic on ports 22, 80, and 443
- **output.tf** : displays useful information for production and development including the databaes connection url, bucket url, and VM public IP
- **provider.tf** : configures the provider along with the associated GCP project and service account (utilizes terraform json key)
- **requirments.tf** : specifies the versions for terraform and other associated systems
- **variables.tf** : specifies env variables used by the terraform files such as projectid, region, and zone

Set-up

NOTE: if you simply pull this repo and try to run the application it will not work, env variables are hidden to limit access to the assoicated GCP project

```
cd gallery
./creation_script.sh
```

This shell script creates a tar.gz file for the source code of the project which terraform then utilizes on the VM instance it creates

```
cd ..
terraform init
terraform apply
```

This will create the vm instance as well as configure run a script on the vm to host the flask app, terraform apply will output the public ip of the vm instance which you can then navigate to. Note: the flask app takes a minute or so to launch on the created vm and wont be available instantly after running terraform apply.

The output from running terraform apply is as follows:

```
Apply complete! Resources: 0 added, 0 changed, 8 destroyed.

Outputs:

bucket_name = "dc635ace-05de-65a5-38b2-fcd15cc876fb"
bucket_url = "gs://dc635ace-05de-65a5-38b2-fcd15cc876fb"
db_connection_name = "proj-459616:us-central1:flask-mysql-db"
vm_ip = "34.41.180.136"
```

Note that the public IP is suseptible to change.

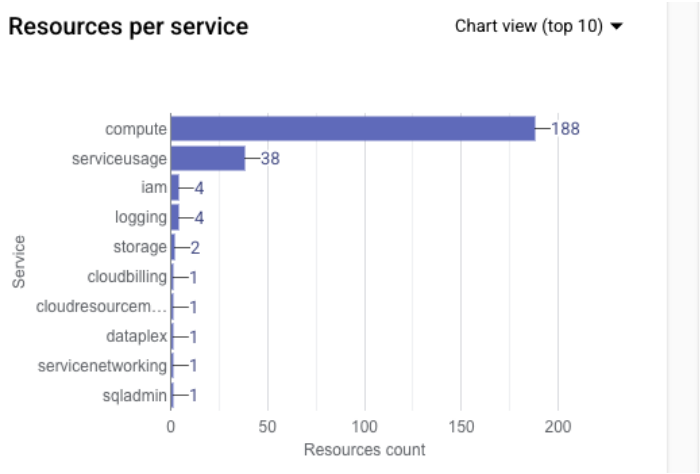
Architecture Diagram

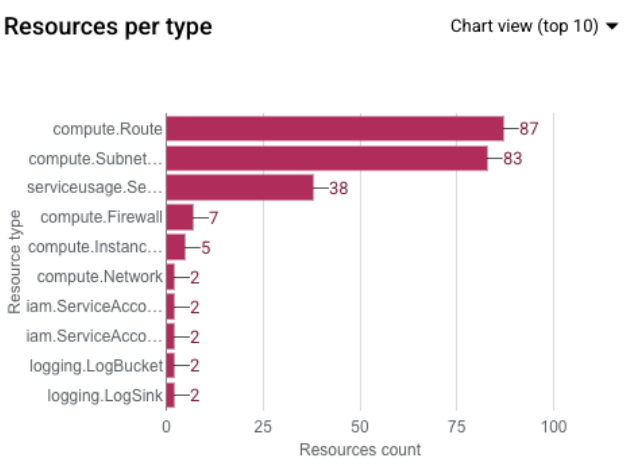
```
    graph TD
    subgraph Architecture Diagram
        firewall_http["google_compute_firewall.allow_http"]
        firewall_port_5000["google_compute_firewall.allow_port_5000"]
        firewall_ssh["google_compute_firewall.allow_ssh"]
        global_addr["google_compute_global_address.private_ip_address"]
        flask_vm["google_compute_instance.flask-vm"]
        vpc_network["google_compute_network.vpc_network"]
        subnetwork["google_compute_subnetwork.default"]
        sqladmin_service["google_project_service.sqladmin"]

vpc_connection["google_service_networking_connection.private_vpc_connectio
n"]
        db["google_sql_database.app_db"]
        db_instance["google_sql_database_instance.default"]
        db_user["google_sql_user.app_user"]
        bucket["google_storage_bucket.flask_app_bucket"]
        bucket_access["google_storage_bucket_iam_member.public_access"]
        google_provider["google provider"]
        random_provider["random provider"]
        uuid["random_uuid.uuid"]
        var_db_name["var.db_name"]
        var_db_password["var.db_password"]
        var_db_username["var.db_username"]
        var_github_token["var.github_token"]
        var_network["var.network"]
        var_project_id["var.project_id"]
        var_region["var.region"]
        var_vm_user["var.vm_user"]
```

```
        var_zone["var.zone"]
        output_bucket_name["output.bucket_name"]
        output_bucket_url["output.bucket_url"]
        output_db_connection["output.db_connection_name"]
        output_vm_ip["output.vm_ip"]
    end

    firewall_http --> vpc_network
    firewall_port_5000 --> vpc_network
    firewall_ssh --> vpc_network
    global_addr --> vpc_network
    flask_vm --> subnetwork
    flask_vm --> db_instance
    flask_vm --> bucket
    flask_vm --> var_db_name
    flask_vm --> var_db_password
    flask_vm --> var_db_username
    vpc_network --> google_provider
    subnetwork --> vpc_network
    sqladmin_service --> google_provider
    vpc_connection --> global_addr
    db --> db_instance
    db --> var_db_name
    db_instance --> vpc_connection
    db_user --> db_instance
    db_user --> var_db_password
    db_user --> var_db_username
    bucket --> google_provider
    bucket --> uuid
    bucket_access --> bucket
    output_bucket_name --> bucket
    output_bucket_url --> bucket
    output_db_connection --> db_instance
    output_vm_ip --> flask_vm
    google_provider --> var_project_id
    google_provider --> var_region
    google_provider --> var_zone
    uuid --> random_provider
```

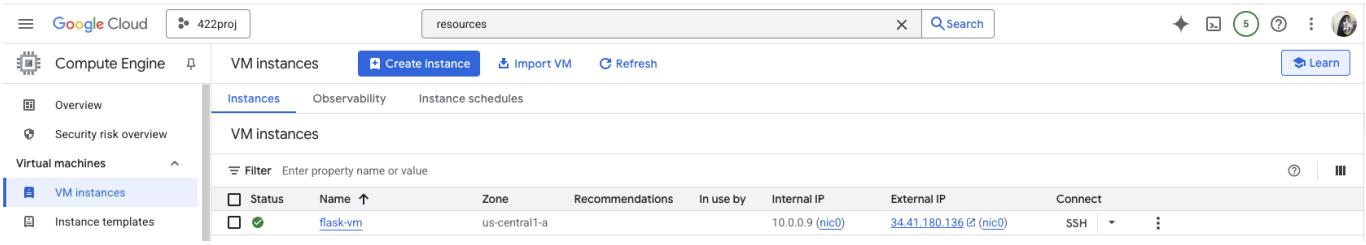
Validation

Google Cloud Console Resources

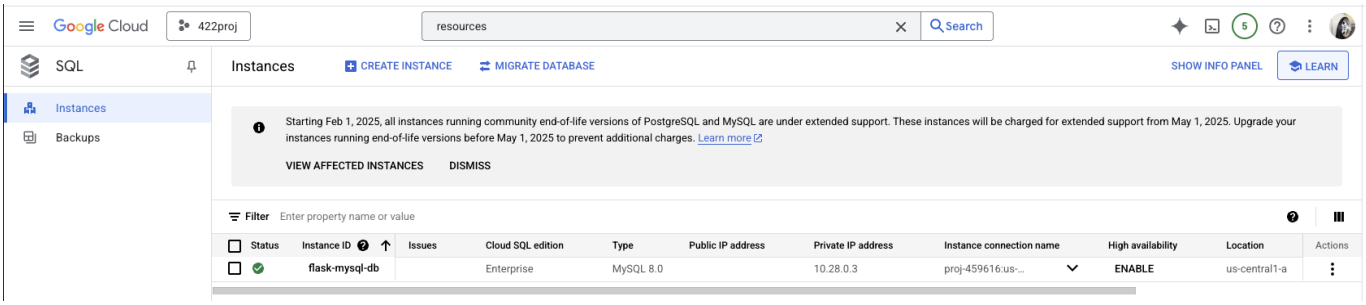




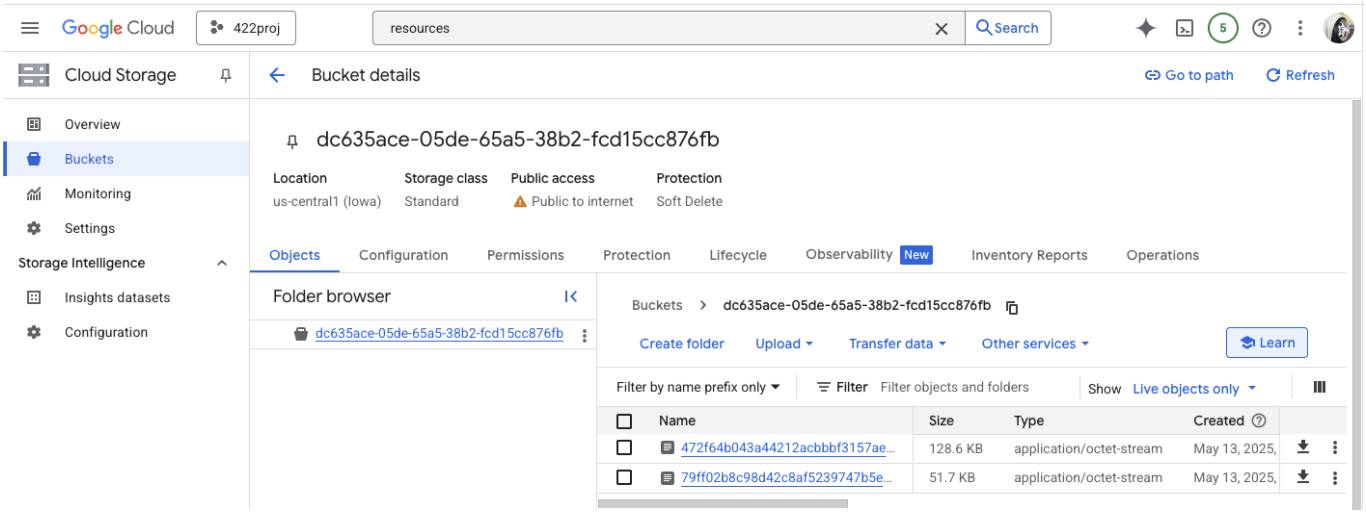
VM Instance



Database Instance

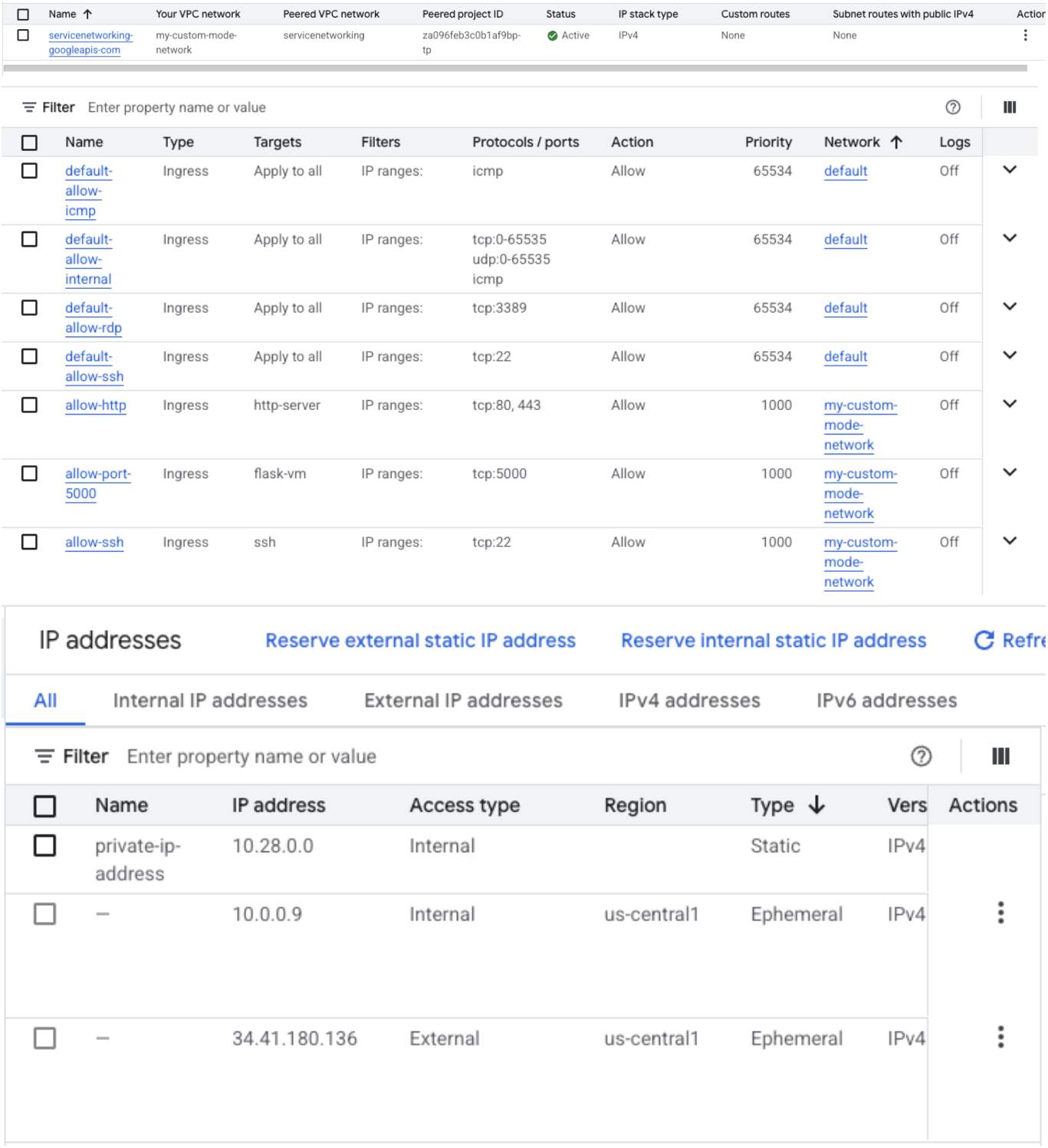


Bucket



Network

Name	Subnets	MTU	Mode	IPv6 ULA range	Gateways	Firewall rules	Global dynamic routing	Network
default	41	1460	Auto			4	Off	
my-custom-mode-network	42	1460	Auto			3	Off	



Name	Your VPC network	Peered VPC network	Peered project ID	Status	IP stack type	Custom routes	Subnet routes with public IPv4	Actior
servicenetworking-googleapis-com	my-custom-mode-network	servicenetworking	za096feb3c0b1af9bp-tp	Active	IPv4	None	None	

Filter Enter property name or value

Name	Type	Targets	Filters	Protocols / ports	Action	Priority	Network	Logs
default-allow-icmp	Ingress	Apply to all	IP ranges:	icmp	Allow	65534	default	Off
default-allow-internal	Ingress	Apply to all	IP ranges:	tcp:0-65535 udp:0-65535 icmp	Allow	65534	default	Off
default-allow-rdp	Ingress	Apply to all	IP ranges:	tcp:3389	Allow	65534	default	Off
default-allow-ssh	Ingress	Apply to all	IP ranges:	tcp:22	Allow	65534	default	Off
allow-http	Ingress	http-server	IP ranges:	tcp:80, 443	Allow	1000	my-custom-mode-network	Off
allow-port-5000	Ingress	flask-vm	IP ranges:	tcp:5000	Allow	1000	my-custom-mode-network	Off
allow-ssh	Ingress	ssh	IP ranges:	tcp:22	Allow	1000	my-custom-mode-network	Off

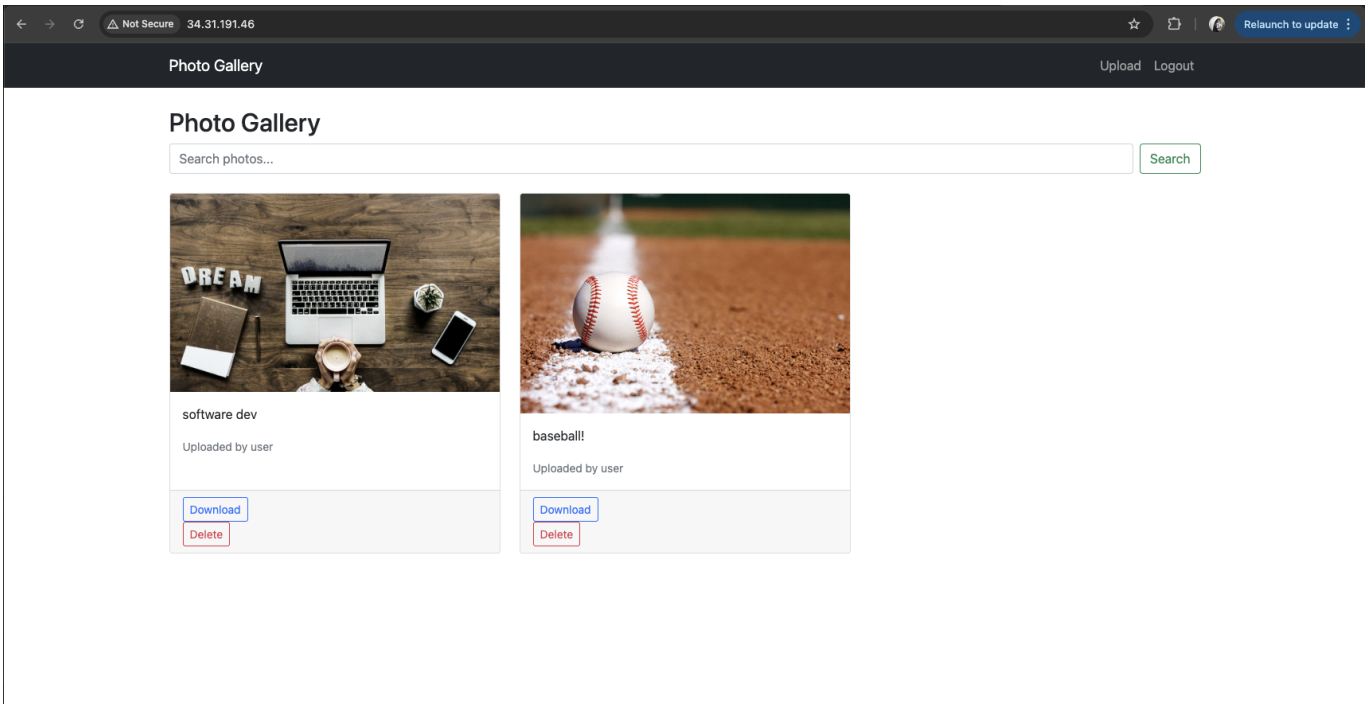
IP addresses Reserve external static IP address Reserve internal static IP address Refre

All Internal IP addresses External IP addresses IPv4 addresses IPv6 addresses

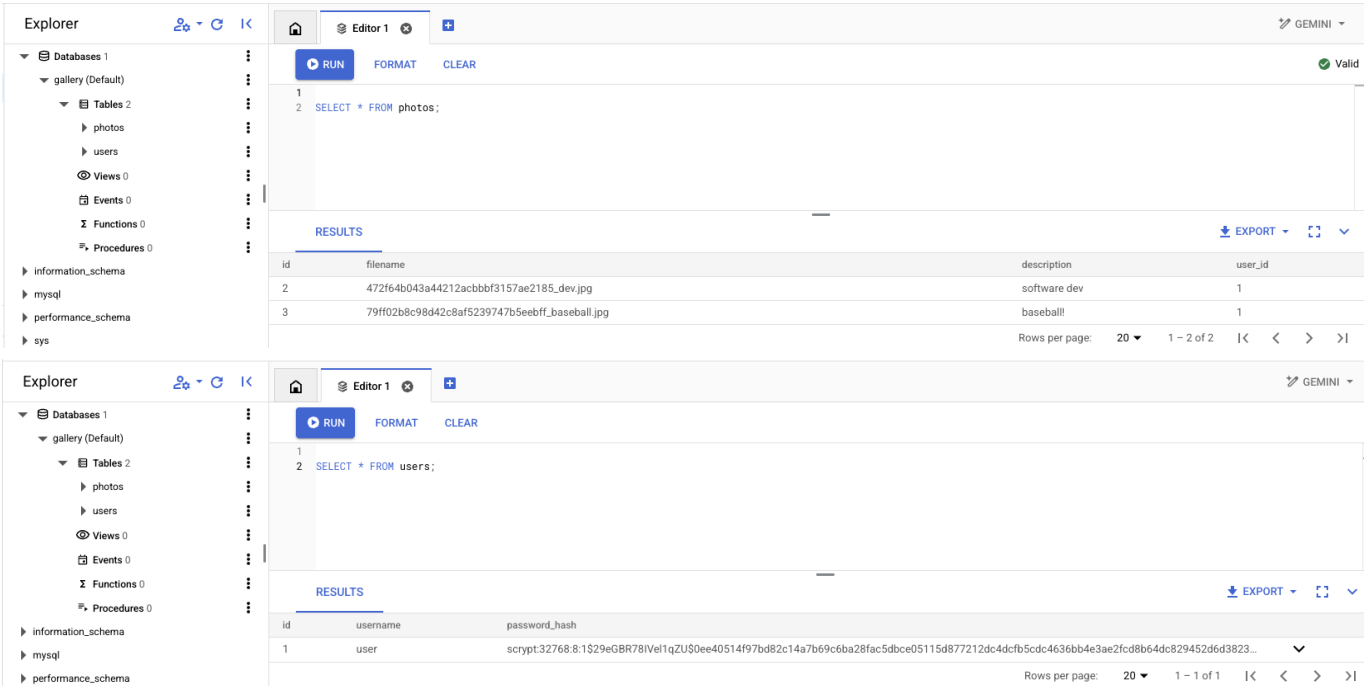
Filter Enter property name or value

Name	IP address	Access type	Region	Type	Vers	Actions
private-ip-address	10.28.0.0	Internal		Static	IPv4	
–	10.0.0.9	Internal	us-central1	Ephemeral	IPv4	
–	34.41.180.136	External	us-central1	Ephemeral	IPv4	

Application Interface

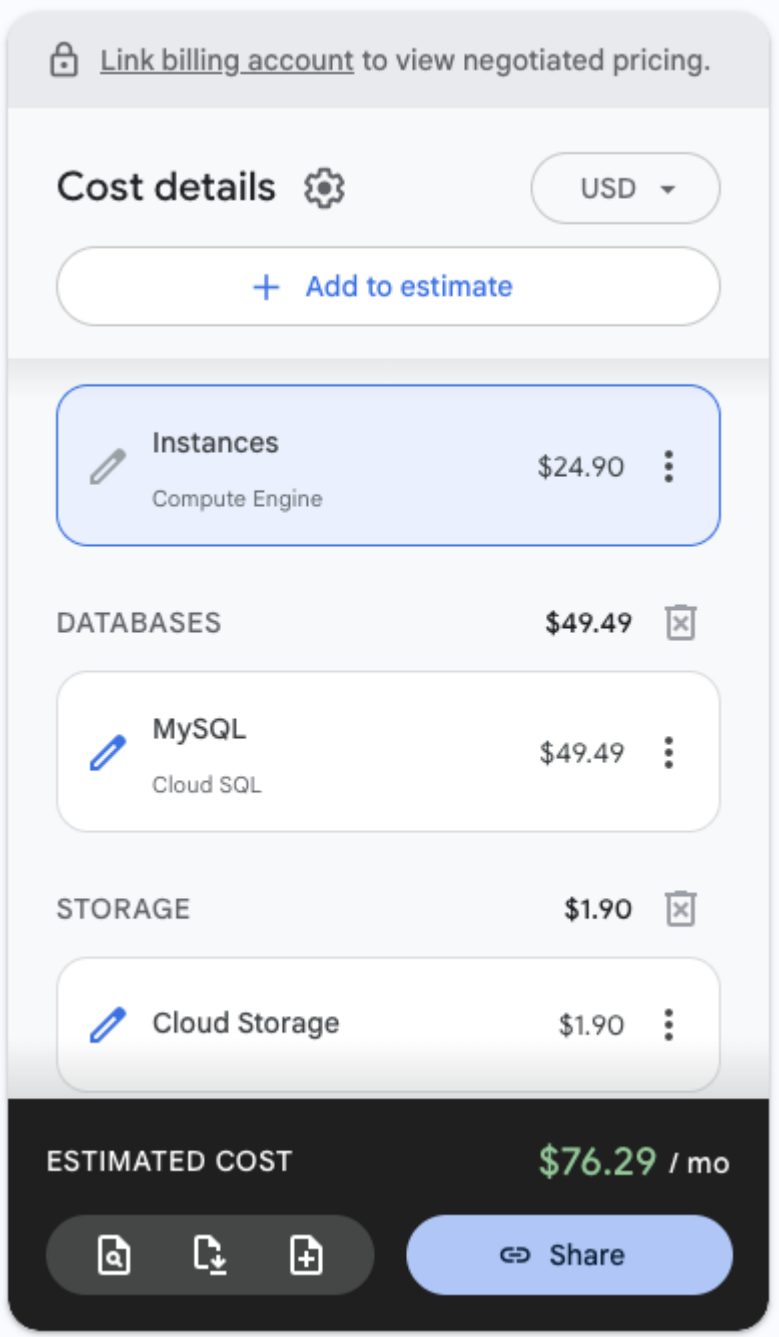


DB Connection



Cost Break Down

Using the specified vm instance and db from the project description results in this monthly price, this could be easily reduced with a less powerful vm and db instance



The runtime for this application depends if your building it from scratch. From scratch the database creation takes a long time, resulting in a run time of around 18 minutes. If your db is already persistent, then the run time is closer to 1 minute.