INFRA PROJECT : TODO Webapp

Omar Hammad

1. Set up a Compute Engine instance

Here i used this command to make a google compute instance that capable of holding the app and db containers :

```
gcloud compute instances create web-vm \
--zone=europe-west1-b \
--machine-type=e2-micro \
--image-family=ubuntu-2004-lts \
--image-project=ubuntu-os-cloud \
--boot-disk-size=10GB \
--tags http-server,https-server \
--metadata=startup-script='#!/bin/bash
exec > >(tee /var/log/startup-log.txt)
exec 2>&1
sudo apt-get update -y
sudo apt-get install -y ca-certificates curl gnupg
sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
sudo chmod a+r /etc/apt/keyrings/docker.gpg
echo "deb [arch=\"$(dpkg --print-architecture)\"
signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu
$(. /etc/os-release && echo \"$VERSION_CODENAME\") stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update -y
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin
docker-compose-plugin
sudo apt-get update
sudo apt-get install -y software-properties-common
sudo add-apt-repository -y ppa:certbot/certbot
sudo apt-get update
sudo apt-get install -y certbot'
```

Then, use this command : “ `gcloud compute ssh web-vm` ” to be in the vm shell
To monitor the installation process using this command : `tail -f /var/log/startup-log.txt`
Finally, execute this command : “ `sudo usermod -aG docker $USER`”

2. Set up a Cloud SQL database

I used here these commands to create a new sql instance called web-sql then i did made the database and database user also i did patched the vm instance ip to the authorized-networks of the sql instance:

```
gcloud sql instances create web-sql --database-version=POSTGRES_14
--root-password=ohammad1997 --tier=db-g1-small --region=europe-west1
```

```
gcloud sql databases create todo_db --instance=web-sql
```

```
gcloud sql users create web_user --instance=web-sql --password=ohammad1997
```

```
gcloud sql instances patch web-sql \
            --authorized-networks="public_vm_instance_ip"
```

Then I've executed this bash script in order to create the desired tables.
Note: after you run the script it will ask for the user `web_user` password

```
#!/bin/bash

# Variables
INSTANCE_NAME="web-sql"
DATABASE_NAME="todo_db"
USERNAME="web_user"

# Check if the SQL instance exists
if ! gcloud sql instances describe $INSTANCE_NAME > /dev/null 2>&1; then
  echo "Warning: SQL instance '$INSTANCE_NAME' does not exist. Please create it first."
  exit 1
fi

# Connect to Google Cloud SQL
gcloud sql connect $INSTANCE_NAME --user=$USERNAME --database=$DATABASE_NAME --quiet <<EOF
CREATE TABLE IF NOT EXISTS users (
  id SERIAL PRIMARY KEY,
  email VARCHAR(255),
  username VARCHAR(255),
  first_name VARCHAR(255),
  last_name VARCHAR(255),
  hashed_password VARCHAR(255),
  is_active BOOLEAN,
  phone_numbers VARCHAR(255)
);

CREATE TABLE IF NOT EXISTS todos (
  id SERIAL PRIMARY KEY,
  title VARCHAR(255),
  description VARCHAR(255),
  priority INTEGER,
  complete BOOLEAN,
  owner_id INTEGER REFERENCES users (id)
);
SELECT * FROM todos;
SELECT * FROM users;
EOF
```

3. Configure Cloud Storage for static files

In this task i created a storage bucket that holds all static files to copy them later in the deployment script:

```
gsutil mb -c Standard -l europe-west1  gs://infra3_todoapp_bucket
gsutil cp -r  ~/todoapp/static/  gs://infra3_todoapp_bucket
gsutil acl ch -u AllUsers:R gs://infra3_todoapp_bucket
```

Here i give all users the permission to access the buckets and copy from it :

```
gsutil iam ch allUsers:objectViewer gs://infra3_todoapp_bucket
```

4. Configure VPC Networks

Here i created a VPC network with a subnet also created fire-rules to be added to the vm instance and the sql instance to make sure both on the same network :

```
gcloud compute networks create todoapp-network --subnet-mode=custom
```

```
gcloud compute networks subnets create app-subnet --network=todoapp-network
--range=192.168.1.0/24
```

add the vpc network to the sql and vm instance using the console

These are the rules needed to be created :

```
gcloud compute firewall-rules create http-allow \
   --network todoapp-network  \
   --allow tcp:80
```

```
gcloud compute firewall-rules create ssh-allow \
   --network todoapp-network  \
   --allow tcp:22
```

```
gcloud compute firewall-rules create https-allow \
   --network todoapp-network  \
   --allow tcp:443
```

```
gcloud compute firewall-rules create sql-allow \
   --network todoapp-network  \
   --allow tcp:5432
```

5. DNS and DHCP

In this task i've registered a domain name for the web application to make it easy accessible using this DN : **simpletodo.be**

In Addition, here i registered it in google Cloud DNS using these commands :
`gcloud dns managed-zones create --dns-name="simpletodo.be." --description="My DNS Zone" "simpletodobe-zone"`

`gcloud dns record-sets transaction start --zone="simpletodobe-zone"`

`gcloud dns record-sets transaction add --zone="simpletodobe-zone" --name="simpletodo.be." --type=A --ttl=300 "public_vm_instance_ip"`

`gcloud dns record-sets transaction execute --zone="simpletodobe-zone"`

This command shows the nameServers to be used in the domain name provider:
`gcloud dns managed-zones describe simpletodobe-zone`
e.g :
- ns-cloud-e1.googledomains.com.
- ns-cloud-e2.googledomains.com.
- ns-cloud-e3.googledomains.com.
- ns-cloud-e4.googledomains.com.

Here are the steps to add these nameServers to easyhost.be:

1. Log in to my account on my.easyhost.be. Find my domain in my product menu .
2. Click on my domain to go to its settings. Find the nameservers settings.
3. Replace the existing nameservers with the ones provided by Google Cloud.
4. These are the nameservers that I got from the gcloud dns managed-zones describe simpletodobe-zone command.
5. Save your changes.

Then just reach the website using this url **https://simpletodo.be**

6. Deploy the web application

Before start :
We have to make sure that SSL/TLS certificates are exists on the vm instance, so first ssh the vm instance and list the files of /etc/letsencrypt/live/<domain name> and make sure if it has {cert.pem,privkey.pem} files and also be in sure that these files have been copied to ~/certs because it'll be used as a bind mount in the compose.yml file. If nothing is there as mentioned above then create SSL/TLS certificates using this command :
`sudo certbot certonly --standalone` then use your email address and the domain name, for sure after you finish these steps, you'll find the certs files as mentioned above but we need to copy them to ~/certs using this command :
`sudo cp /etc/letsencrypt/live/<domain-name>/{privkey.pem,cert.pem} ~/certs`

Here we start with the deployment:

First, I did build an image of the application using this Dockerfile:
Command : " docker build -t omarhammad997/todoapp:latest . "

```
FROM tiangolo/uvicorn-gunicorn-fastapi:python3.8

WORKDIR /app

COPY ./requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY ./app .
RUN mkdir /cert

EXPOSE 8000
EXPOSE 5432
EXPOSE 443

# Use uvicorn to serve the application with HTTPS
CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "443", "--ssl-keyfile
", "/certs/privkey.pem", "--ssl-certfile", "/certs/cert.pem"]
```

Then i pushed it using this command :
" `docker push omarhammad997/todoapp:latest` "

Then i've copied the compose file to the vm instance using the scp command also i did upload the service account key that has the CLOUD SQL CLIENT AND CLOUD SQL EDITOR roles so that i use it in the cloud_sql_proxy command:

In order to get this service-account file, I have to go to the sql instance in the console and copy the url in the Service account section e.g : p848444805203-0e5tfi@gcp-sa-cloud-sql.iam.gserviceaccount.com to use it for name of the service account. And here are the steps to create a one :

1. Select or create your project from the top dropdown.
2. Click on the navigation menu > IAM & Admin > Service Accounts. Click on "CREATE SERVICE ACCOUNT", fill out the details and click "CREATE".
3. Select a role for the service account and click "CONTINUE".
4. Back in the Service Account list, click on the newly created service account.
5. In the "KEYS" tab, click "ADD KEY" > "Create new key" > "JSON". The JSON key file will be downloaded.
6. Change the file name to sql_proxy.json to make it compatible with the docker compose file.

Then i made the docker-compose.yml like this :

```
version: '1'
services:
  web:
    image: omarhammad997/todoapp:latest
    volumes:
      - /home/omar-hammad/todo_bucket/static:/app/static
      - /home/omar-hammad/certs:/certs
    ports:
      - "443:443"
    depends_on:
      - db_proxy
    environment:
      - "DATABASE_URL=postgresql://web_user:ohammad1997@db_proxy:5432/todo_db"
  db_proxy:
    image: gcr.io/cloudsql-docker/gce-proxy:1.17
    command: /cloud_sql_proxy -instances=infra3-hammad-omar:europe-west1:web-sql=tcp:0.0.0.0:5432
-credential_file=/config
    volumes:
      - /home/omar-hammad/sql_proxy.json:/config
    environment:
      - DB_USER=web_user
      - DB_PASS=ohammad1997
      - DB_NAME=todo_db
```

Finally, i have to setup everything to make the deployment script works correctly and make sure to use the correct path of the docker compose file and account service key :

```
#!/bin/bash

# Initialize our own variables
instance_name=""
compose_path=""
key_path=""

# Read the options
while getopts ":i:c:k:" opt; do
  case $opt in
    i) instance_name="$OPTARG"
    ;;
    c) compose_path="$OPTARG"
    ;;
    k) key_path="$OPTARG"
    ;;
    \?) echo "Invalid option -$OPTARG" >&2
    ;;
  esac
done

# Check if the user provided all required fields
if [[ -z "$instance_name" || -z "$compose_path" || -z "$key_path" ]]; then
  echo "Please provide the instance name (-i), Docker Compose file path (-c), and service
account key path (-k)."
  exit 1
fi

# Upload the Docker Compose file and the service key
gcloud compute ssh $instance_name --command "mkdir ~/todoapp/"
gcloud compute scp $compose_path $instance_name:~/todoapp/docker-compose.yml
gcloud compute scp $key_path $instance_name:~


# Deploy the website and run it using Docker Compose
gcloud compute ssh $instance_name --command "bash -s" <<EOF
mkdir ~/todo_bucket
sudo gsutil -m cp -r gs://infra3_todoapp_bucket/static/ ~/todo_bucket/
cd ~/todoapp/
docker compose up -d
EOF
```