

REPORT

Report of the voting-app-ielm project, everything i did, step by step.

Complete Dockerfiles

vote

```
# Choose a base image for Python (typically python:3.9)
FROM python:3.9

# Set the working directory inside the container
WORKDIR /app

# Copy the dependencies file and install dependencies
COPY requirements.txt .
RUN pip3 install -r requirements.txt

# Copy the rest of the application code (from the vote folder to /app,
# that's why . .)
COPY . .

# Expose the internal port on which Flask listens
EXPOSE 80
CMD ["python3", "app.py"]
```

result

```
# Choose a base image for Node.js (typically node:18)
FROM node:18

# Set the working directory inside the container
WORKDIR /app

# Copy package.json / package-lock.json and run npm install (* allows to
# copy both files)
COPY package*.json ./
RUN npm install

# Copy the rest of the application code (from the result folder to /app,
# that's why . .)
COPY . .

# Expose the internal port on which Express listens
EXPOSE 80
CMD ["node", "server.js"]
```

worker

```
# Choose a base image for .NET SDK (typically
[mcr.microsoft.com/dotnet/sdk:8.0]
(https://mcr.microsoft.com/dotnet/sdk:8.0))
FROM [mcr.microsoft.com/dotnet/sdk:8.0]
(https://mcr.microsoft.com/dotnet/sdk:8.0)

# Set the working directory inside the container
WORKDIR /app

# Copy the project files
COPY . .

# Restore packages and compile in Release mode
RUN dotnet restore
RUN dotnet build -c Release

# Define the command to run the container
CMD ["dotnet", "run", "-c", "Release"]
```

Complete docker-compose.yml

We define 5 services:

- vote
- result
- worker
- redis
- db

Build the images:

- Docker build each image from the Dockerfile in the corresponding folder

Port mapping (front accessible ports):

- 5000:80: vote
- 5001:80: result

Networks:

- front-tier: front accessible
- back-tier: back accessible
- vote and result connect to front-tier and back-tier networks

Volumes:

- pg-data: persistent storage for Postgres (if the container is removed, the data is not lost)

Environment variables:

- POSTGRES_PASSWORD

- POSTGRES_DB
- For server.js (Node.js) and Program.cs (.NET)

```
services:
  vote:
    # Create image from ./vote Dockerfile
    build: ./vote
    # Map port of host to port of container
    ports:
      - '5000:80'
    # Connect this service to the networks, (front-tier and back-tier
to be accessible
    # from outside (front) and to back-tier to be accessible from
Redis)
    networks:
      - front-tier
      - back-tier
    # This service depends on Redis (wait until Redis is up and
running)
    depends_on:
      - redis
  result:
    # Create image from ./result Dockerfile
    build: ./result
    # Map port of host to port of container
    ports:
      - '5001:80'
    # Connect this service to the networks, (front-tier and back-tier
to be accessible
    # from outside (front) and to back-tier to be accessible from DB)
    networks:
      - front-tier
      - back-tier
    # This service depends on DB (wait until DB is up and running)
    depends_on:
      - db
  worker:
    # Create image from ./worker Dockerfile
    build: ./worker
    # The worker process data, it connects both networks (front-tier
and back-tier)
    networks:
      - front-tier
      - back-tier
  db:
    # Use the official Postgres image from Docker Hub
    image: postgres:15
    # Set environment variables for Postgres
    environment:
      POSTGRES_PASSWORD: issa1234
      POSTGRES_DB: votes
    # Mount the volume to persist data
```

```
volumes:
  - pg-data:/var/lib/postgresql/data
networks:
  # Connect DB only to back-tier, it is not accessible from
outside
  - back-tier

networks:
  # Define the networks needed front-tier and back-tier
front-tier:
back-tier:

volumes:
  # Define the persistent storage for Postgres
pg-data:
```

docker-compose up

Execute the following command to start the containers:

```
docker compose up -d
```

Result if everything is working:

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (2m 42.01s)
docker compose up -d
-> [worker 2/5] WORKDIR /app 0.2s
=> [worker 3/5] COPY . . 0.0s
=> [result 3/5] COPY package*.json ./ 0.0s
=> [worker 4/5] RUN dotnet restore 10.0s
=> [result 4/5] RUN npm install 4.8s
=> [result 5/5] COPY . . 0.3s
=> [result] exporting to image 5.0s
=> => exporting layers 2.9s
=> => exporting manifest sha256:c267b788e0175488002d0dd00e897d416f7cc9304709fce2129b950470319b0f 0.0s
=> => exporting config sha256:fafdc8480f621e64b4c7592da766c8bb9867414ae69bbbd0db12c171f5868ea7 0.1s
=> => exporting attestation manifest sha256:92b2b05197944f9854b5356f3a5da89c88102f210922a2c4d535c319b8ccbe8 0.1s
=> => exporting manifest list sha256:fbe70bb3ac6bb04d4afd9098350c617998e528a3a6e5f40934b61abd5249bab1 0.1s
=> => naming to docker.io/library/app-result:latest 0.0s
=> => unpacking to docker.io/library/app-result:latest 1.7s
=> [vote 5/5] COPY . . 0.4s
=> [worker 5/5] RUN dotnet build -c Release 4.5s
=> [vote] exporting to image 1.8s
=> => exporting layers 1.1s
=> => exporting manifest sha256:9922869f6c41058c288ea04927368f2ae973c268ab646207e3bb1651b9edd9b 0.0s
=> => exporting config sha256:5e3b96617da8acba21b3ef9953607969735434189e3e0cd44e3a67669b1cbde7 0.0s
=> => exporting attestation manifest sha256:ee242ba5cf4f9b3bf37f6deb707c3db1a06d0d4cdc2f2474852032170b722c7 0.1s
=> => exporting manifest list sha256:8902ad4b7323ece8a74d4dda2fde5a2f0468820f50d11108fa83296c6e15ab4f 0.0s
=> => naming to docker.io/library/app-vote:latest 0.0s
=> => unpacking to docker.io/library/app-vote:latest 0.3s
=> [result] resolving provenance for metadata file 0.0s
=> [vote] resolving provenance for metadata file 0.0s
=> [worker] exporting to image 3.8s
=> => exporting layers 3.1s
=> => exporting manifest sha256:05c200ed3defe0f34f5b1a2d3adaf1ac44b83b71d3241daf58664dce1d29712a 0.0s
=> => exporting config sha256:40a6f17bb772c8d714c426fd14b15699ef7222a31e8d28e3cbcd5a97870d430 0.1s
=> => exporting attestation manifest sha256:f7f11020fa04625f95ae6ac1ea3ca44519286b2527be0cf7f4bcdafac8b6f2f 0.1s
=> => exporting manifest list sha256:6706ab2ab74a545f911ec79747fd878353ad2b4cd2e7c09ba274ceeda130de17 0.1s
=> => naming to docker.io/library/app-worker:latest 0.0s
=> => unpacking to docker.io/library/app-worker:latest 0.3s
=> [worker] resolving provenance for metadata file 0.0s
[+] Running 11/11
✔ app-vote Built 0.0s
✔ app-worker Built 0.0s
✔ app-result Built 0.0s
✔ Network app_back-tier Created 0.0s
✔ Network app_front-tier Created 0.0s
✔ Volume app_pg-data Created 0.0s
✔ Container app-redis-1 Started 2.5s
✔ Container app-db-1 Started 2.5s
✔ Container app-vote-1 Started 4.8s
✔ Container app-worker-1 Started 1.9s
✔ Container app-result-1 Started 2.1s
```

Problems

In the previous image, you can see all the services running, but if I execute the following command:

```
docker compose ps
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.105s)
docker compose ps
NAME                IMAGE                COMMAND                SERVICE    CREATED        STATUS        PORTS
app-db-1            postgres:15          "docker-entrypoint.s..." db         2 minutes ago  Up 2 minutes  5432/tcp
app-redis-1         redis:7               "docker-entrypoint.s..." redis      2 minutes ago  Up 2 minutes  6379/tcp
app-result-1        app-result            "docker-entrypoint.s..." result     2 minutes ago  Up 2 minutes  0.0.0.0:5001->80/tcp, [::]:5001->80/tcp
```

We can see that **app-vote-1** and **app-worker-1** are not running, let's check the logs:

1. Vote Service

```
docker logs app-vote-1
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.052s)
docker logs app-vote-1

File "/app/app.py", line 19
    if request.method == "POST":
    ^
IndentationError: expected an indented block
```

- The error is that the script is creating the python index function with incorrect indentation, let's fix it:

```
@app.route("/", methods=["GET", "POST"])
def index():
    if request.method == "POST":
        vote = request.form["vote"]
        redis_client.rpush("votes", vote)
    return render_template_string(TEMPLATE)
```

- Now let's rebuild the vote service:

```
docker compose up -d --build vote
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (1.785s)
docker compose up -d --build vote

[+] Building 1.3s (12/12) FINISHED
=> [internal] load local bake definitions                                0.0s
=> => reading from stdin 569B                                           0.0s
=> [internal] load build definition from Dockerfile                     0.0s
=> => transferring dockerfile: 517B                                      0.0s
=> [internal] load metadata for docker.io/library/python:3.9           1.0s
=> [internal] load .dockerignore                                         0.0s
=> => transferring context: 2B                                            0.0s
=> [1/5] FROM docker.io/library/python:3.9@sha256:da5aee29682d12a6649f51c8d6f15b87deb3e6c524b923c41d0cb3304d07c913 0.0s
=> => resolve docker.io/library/python:3.9@sha256:da5aee29682d12a6649f51c8d6f15b87deb3e6c524b923c41d0cb3304d07c913 0.0s
=> [internal] load build context                                         0.0s
=> => transferring context: 782B                                          0.0s
=> CACHED [2/5] WORKDIR /app                                             0.0s
=> CACHED [3/5] COPY requirements.txt .                                  0.0s
=> CACHED [4/5] RUN pip3 install -r requirements.txt                    0.0s
=> [5/5] COPY . .                                                        0.0s
=> exporting to image                                                    0.1s
=> => exporting layers                                                    0.0s
=> => exporting manifest sha256:91309bc044afed54ddaf5e682948d4e4208ad7d0d253f6ad57c2f6e31f020b3b 0.0s
=> => exporting config sha256:31304288fecadd6088916c3895e97740ed1f90e2001b8988aef91eaeabbd059d 0.0s
=> => exporting attestation manifest sha256:deb6159f11adfbf6a4a3c7f3006eb86a15f2960f27cd953b52d8a13d150f38c 0.0s
=> => exporting manifest list sha256:022d35c59abe34592937e3283b69bd6ea4ac06590de816d08333a8f1602a93e7 0.0s
=> => naming to docker.io/library/app-vote:latest                       0.0s
=> => unpacking to docker.io/library/app-vote:latest                    0.0s
=> resolving provenance for metadata file                                0.0s
[+] Running 3/3
✔ app-vote Built 0.0s
✔ Container app-redis-1 Running 0.0s
✔ Container app-vote-1 Started 0.3s
```

2. Worker Service

```
docker logs app-worker-1
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.049s)
docker logs app-worker-1

/app/Worker.csproj : warning NU1903: Package 'Npgsql' 7.0.6 has a known high severity vulnerability, https://github.com/advisories/GHSA-x9vc-6hfv-hg8c
/app/Worker.csproj : warning NU1903: Package 'Npgsql' 7.0.6 has a known high severity vulnerability, https://github.com/advisories/GHSA-x9vc-6hfv-hg8c
Worker en ejecución. Esperando votos...
Unhandled exception. Npgsql.NpgsqlException (0x80004005): Failed to connect to 172.18.0.3:5432
----> System.Net.Sockets.SocketException (111): Connection refused
   at Npgsql.Internal.NpgsqlConnector.Connect(NpgsqlTimeout timeout)
   at Npgsql.Internal.NpgsqlConnector.Connect(NpgsqlTimeout timeout)
   at Npgsql.Internal.NpgsqlConnector.RawOpen(SslMode sslMode, NpgsqlTimeout timeout, Boolean async, CancellationToken cancellationToken, Boolean isFirstAttempt)
   at Npgsql.Internal.NpgsqlConnector.<Open>g__OpenCore|216_1(NpgsqlConnector conn, SslMode sslMode, NpgsqlTimeout timeout, Boolean async, CancellationToken cancellationTok, Boolean isFirstAttempt)
   at Npgsql.Internal.NpgsqlConnector.Open(NpgsqlTimeout timeout, Boolean async, CancellationToken cancellationToken)
   at Npgsql.PoolingDataSource.OpenNewConnector(NpgsqlConnection conn, NpgsqlTimeout timeout, Boolean async, CancellationToken cancellationToken)
   at Npgsql.PoolingDataSource.<Get>g__RentAsync|28_0(NpgsqlConnection conn, NpgsqlTimeout timeout, Boolean async, CancellationToken cancellationToken)
   at Npgsql.NpgsqlConnection.<Open>g__OpenAsync|45_0(Boolean async, CancellationToken cancellationToken)
   at Npgsql.NpgsqlConnection.Open()
   at Program.<Main>(String[] args) in /app/Program.cs:line 15
```

- The error is that the worker tried to connect to the DB (port 5432), but it was not up and running, this happens because the worker is faster than the DB.
- Now we already have the DB running, let's restart the worker:

```
docker start app-worker-1
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.051s)
docker start app-worker-1
app-worker-1
```

Final Check

Finally, let's check the up containers again:

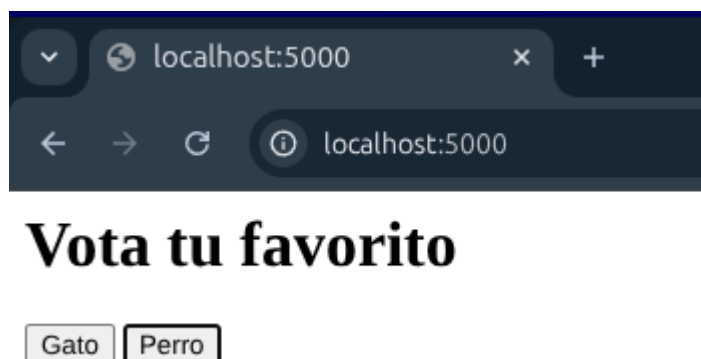
```
docker ps
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.06s)
docker ps
```

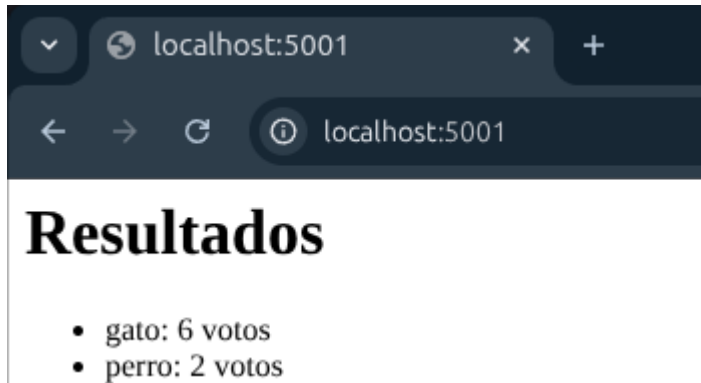
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2b1c175dc99c	app-worker	"dotnet run -c Relea..."	11 seconds ago	Up 10 seconds		app-worker-1
b9be5c03aa58	app-vote	"python3 app.py"	8 minutes ago	Up 8 minutes	0.0.0.0:5000->80/tcp, [::]:5000->80/tcp	app-vote-1
33e2e31e1e6b	app-result	"docker-entrypoint.s..."	32 minutes ago	Up 32 minutes	0.0.0.0:5001->80/tcp, [::]:5001->80/tcp	app-result-1
d7928d02c019	postgres:15	"docker-entrypoint.s..."	32 minutes ago	Up 32 minutes	5432/tcp	app-db-1
fcbb35644f7e	redis:7	"docker-entrypoint.s..."	32 minutes ago	Up 32 minutes	6379/tcp	app-redis-1

Testing

Let's test the **vote service** (<http://localhost:5000>):



Let's test the **result service** (<http://localhost:5001>):



Let's check the **data flow** (Python -> Redis -> Worker -> DB -> Node.js):

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.047s)
docker logs app-worker-1
/app/Worker.csproj : warning NU1903: Package 'Npgsql' 7.0.6 has a known high severity vulnerability, https://github.com/advisories/GHSA-x9vc-6hfv-hg8c
/app/Worker.csproj : warning NU1903: Package 'Npgsql' 7.0.6 has a known high severity vulnerability, https://github.com/advisories/GHSA-x9vc-6hfv-hg8c
Worker en ejecución. Esperando votos...
Procesando voto: perro
Procesando voto: perro
Procesando voto: gato
Procesando voto: gato
Procesando voto: gato
Procesando voto: gato
Procesando voto: gato
Procesando voto: gato
Procesando voto: gato
Procesando voto: gato
```

Data flow explanation:

1. The user sends a vote to the vote service (<http://localhost:5000>).
2. The vote service stores the vote in Redis.
3. The worker processes the vote and stores it in the database.
4. The result service queries the database and displays the results.

Network Design and Volume Persistence

Why is the Worker connected to both networks?

- The Worker acts as a bridge: it needs the **front-tier** to read votes from Redis and the **back-tier** to save them in PostgreSQL.
- The **back-tier** provides security by isolating the database and Redis from public access.

Why is persistence necessary?

- The data on Docker containers disappear when the container is stopped or deleted.
- A **Volume** (pg-data) is necessary to ensure voting records are saved on the host even if the container restarts or is deleted.

Extra report explanation

Networks design evidence:

```
docker network ls
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.048s)
docker network ls

NETWORK ID          NAME                DRIVER              SCOPE
4a5e32a59669        app_back-tier       bridge              local
2fd8e3580dc0        app_front-tier      bridge              local
3d6d4aad6ee7        bridge              bridge              local
860fdb100dbd        host                host                local
929ecdcde156        none                null                local
```

- **Front-tier services:**

```
docker network inspect app_front-tier | grep Name # To get a simple list of
the services
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.049s)
docker network inspect app_front-tier | grep Name

    "Name": "app_front-tier",
      "Name": "app-worker-1",
      "Name": "app-result-1",
      "Name": "app-vote-1",
```

- **Back-tier services:**

```
docker network inspect app_back-tier | grep Name # To get a simple list of
the services
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.054s)
docker network inspect app_back-tier | grep Name

    "Name": "app_back-tier",
      "Name": "app-worker-1",
      "Name": "app-result-1",
      "Name": "app-vote-1",
      "Name": "app-db-1",
      "Name": "app-redis-1",
```

Volumes and persistence:

```
docker volume ls
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.047s)
docker volume ls

DRIVER      VOLUME NAME
local       0714892c24c1c180e2c280427ee52129c05b1bbf81c349e24a30a67896f0d9ba
local       app_pg-data
```

- Once we have registered some votes, let's destroy the db container (without removing the volume):

```
docker rm -f app-db-1 && docker ps
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.195s)
docker rm -f app-db-1
app-db-1

~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.054s)
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2b1c175dc99c	app-worker	"dotnet run -c Relea..."	29 minutes ago	Up 29 minutes		app-worker-1
b9be5c03aa58	app-vote	"python3 app.py"	37 minutes ago	Up 37 minutes	0.0.0.0:5000->80/tcp, [::]:5000->80/tcp	app-vote-1
33e2e31e1e6b	app-result	"docker-entrypoint.s..."	About an hour ago	Up About an hour	0.0.0.0:5001->80/tcp, [::]:5001->80/tcp	app-result-1
fcbb35644f7e	redis:7	"docker-entrypoint.s..."	About an hour ago	Up About an hour	6379/tcp	app-redis-1

- We have checked that the container is down, now let's create it again:

```
docker compose up -d db
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.385s)
docker compose up -d db

[+] Running 1/1
✓ Container app-db-1 Started
```

- Finally, let's check the result service (<http://localhost:5001>):



Script completion and execution

Give execution permissions to the script

```
chmod +x deploy_final.sh
```

Execute the script and follow the instructions

```
./deploy_final.sh
```

```
~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm git:(main) (0.039s)
./deploy_final.sh
==> Generando SOLUCIÓN COMPLETA de voting-app (TODOS completados)...
==> Carpeta raíz: /home/issa/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app
==> Generando vote/app.py (Con indentación arreglada)
==> Generando vote/requirements.txt
==> Generando vote/Dockerfile (Completado)
==> Generando result/server.js
==> Generando result/package.json
==> Generando result/Dockerfile (Completado)
==> Generando worker/Program.cs
==> Generando worker/Worker.csproj
==> Generando worker/Dockerfile (Completado)
==> Generando docker-compose.yml (Completado)

=====
SOLUCIÓN GENERADA EN: /home/issa/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app
=====
Pasos para desplegar:
  1) cd /home/issa/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app
  2) docker compose up -d

Nota: Se ha añadido un pequeño 'sleep' en el worker y corregido
la indentación de Python automáticamente.
=====

~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm git:(main) (0.03s)
cd /home/issa/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app

~/Escritorio/DAW/2DAW/VIRTUALBOS/voting-app-ielm/app git:(main) (0.66s)
docker compose up -d

[+] Running 5/5
 ✓ Container app-redis-1   Started
 ✓ Container app-db-1     Started
 ✓ Container app-vote-1    Started
 ✓ Container app-worker-1  Started
 ✓ Container app-result-1  Started
```

- Note: notice that even we have created a new app, we still have the old data because we have used a volume (pg-data) and the same project name (app), if we had used a different project name, the data would have been lost.

