Concept Overview

- **Cloud Run**: A fully managed service to run containerized web apps.
- **Docker**: To package your app into a container.
- **Cloud Build**: Builds and stores your image.
- **Container Registry / Artifact Registry**: Stores the Docker image.
- **gcloud CLI**: To manage deployment via terminal.

Tool	**Purpose**
FastAPI	Web framework
SQLAlchemy	ORM for SQLite
SQLite	Local DB
Docker	Containerize app
Cloud Build	Build a container in the cloud
Container Registry	Store container image
Cloud Run	Deploy the app to the cloud
gcloud CLI	Manage deployments via terminal

Overview

FastAPI app that pulls data from an API, stores it in SQLite, and serves endpoints (e.g., /locations, /top-n, etc.).

Steps to run it :

1) Create a Dockerfile

This file packages your Python app, dependencies, and server config (e.g., uses uvicorn to serve FastAPI). Example:

dockerfile

CopyEdit

```
FROM python:3.11-slim

WORKDIR /app

COPY . /app

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

CMD ["uvicorn", "weather_api:app", "--host", "0.0.0.0", "--port", "8000"]
```

2) Build the container image

```
gcloud builds submit --tag gcr.io/weather-466408/weather-api
```

weather-466408 (is the project I created in GCP). You also have to enable some services before in GCP (Cloud Run, Cloud Build, Container Registry)

- Packages your app
- Uploads it to Google Container Registry (gcr.io)
- Tag it as weather-api

3)Deploy the container to Cloud Run

```
gcloud run deploy weather-api-service \
  --image gcr.io/weather-466408/weather-api \
  --platform managed \
  --region europe-west3 \
  --allow-unauthenticated \
  --port 8000
```

Gives you a public URL (like https://weather-api-service-xyz.a.run.app)

4)Access your API online

After deployment, GCP gives you a live URL to your app where you can hit your endpoints just like you would locally:

- GET /locations
- GET /latest
- GET /averages
- GET /top-n?n=2

Known Limitations

- SQLite is not persistent in Cloud Run (not recommended for production).
- API credentials (like `.env`) should be handled using **Secret Manager** or **environment variables** in Cloud Run settings.
- Cold starts can slow initial requests slightly

Future Improvements

- You should move secrets into GCP Secret Manager later
- Consider running the fetcher on a schedule with Cloud Scheduler
- If you want to store more than SQLite, use GCP Cloud SQL

Problems you ran into, and how you solved them

gcloud run deploy weather-api-service --image gcr.io/weather-466408/weather-api --platform managed --region europe-west3 --allow-unauthenticated --port 8000. I was running this command, exposing port 800,0 and the error wasn't clear. After I used port 8080 in my Dockerfile and removed the port from the command, it finally worked.

GET https://weather-api-service-727552370178.europe-west3.run.app/locations

GET https://weather-api-service-727552370178.europe-west3.run.app/forecast/latest

GET https://weather-api-service-727552370178.europe-west3.run.app/forecast/averages

GET
https://weather-api-service-727552370178.europe-west3.run.app/top-locations?metric=the_temp&n=5