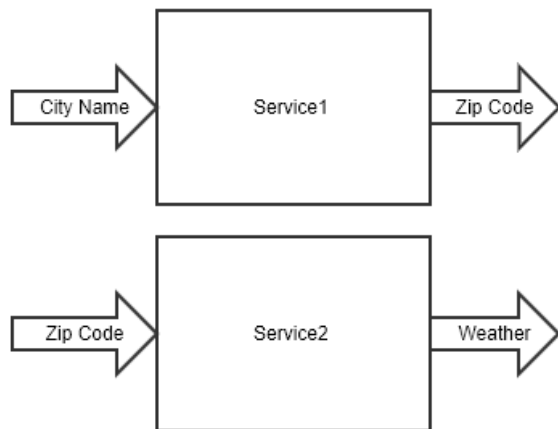


Problem:

Write a web application to find the *weather* of a given city.



1. Design the service using two microservices:



2. Implementation: Write two server programs in python and test them using either a browser client or a curl client.
3. Containerization: Create Dockerfiles and run two containers independently.

Solution:

- First I created python programs and Dockerfile(s) for two services
- Used the command in the terminal to create an image that runs with help of the docker desktop. That is `docker build -t weather .` Here the weather is the name of the image I gave
- Then to run the image and containerize it, we use the command `docker run -p 5000:5000 weather` where 5000 is the port
- Now I tested these services as shown below in screenshots.

1st service which gave Zip code from City Name

The screenshot displays a development environment with three main components:

- Visual Studio Code (Left):** The Explorer sidebar shows a project structure with folders `app`, `assign`, `hw`, `city`, `weather`, and `temp`. The `city` folder contains `city.py` and `Dockerfile`. The `weather` folder contains `Dockerfile` and `weather.py`. The `temp` folder contains `Dockerfile` and `temp.py`. The `city` folder is selected, showing the `Dockerfile` in the editor. The Dockerfile content is:

```
1 FROM python:3.8-alpine
2
3 WORKDIR /app
4
5 COPY . .
6
7 RUN pip install --no-cache-dir flask
8
9 ENV FLASK_APP=city.py
10
11 CMD ["flask", "run", "--host=0.0.0.0"]
```
- Terminal (Bottom Left):** The terminal shows the command `docker run -p 5000:5000 city` being executed. The output indicates that the Flask app is running on `http://127.0.0.1:5000` and `http://172.17.0.2:5000`. A warning message states: "WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead." The terminal also shows a GET request to `/zip_from_city?city_name=Acampo` returning a 200 status.
- Web Browser (Right):** The browser shows the URL `http://localhost:5000/zip_from_city?city_name=Acampo...`. The response is a JSON object: `{"zip code for Acampo":95220}`.

2nd service takes the zip code and gives the weather details:

The screenshot displays a development environment with Visual Studio Code on the left and a web browser on the right.

Visual Studio Code:

- EXPLORER:** Shows a project structure with folders `app`, `assign`, `hw`, `city`, `weather`, and `temp`. The `weather` folder is selected, showing `Dockerfile` and `weather.py`.
- Editor:** Displays the `weather.py` file. The code defines a Flask application with a dictionary of weather conditions for various zip codes and a route `/weather_from_zip` that returns the weather for a given zip code.
- TERMINAL:** Shows the command `docker run -p 5000 weather` being executed. The output indicates that the Flask app is serving on `http://127.0.0.1:5000` and `http://172.17.0.2:5000`.

Web Browser:

- The address bar shows `http://localhost:5000/weather_from_zip?zip_code=9...`.
- The response body shows a JSON object: `{"weather condition for zip(95220)": "Rainy"}`.

4. Networking: Make two containers talk to each other

Solution:

At starting I deleted all the previous images and containers.

Created image and container for the second service on a network named “mynet”:

```
PS C:\Users\Chinni\Documents\2 sem\cloud\assignment-1\city> docker build -t first .  
[+] Building 4.9s (11/11) FINISHED
```

```
PS C:\Users\Chinni\Documents\2 sem\cloud\assignment-1\weather> docker network create mynet  
b75caf96f35e44449152ff3aed23faaee5c2384f1f7b2bd8254eb54da247f578  
PS C:\Users\Chinni\Documents\2 sem\cloud\assignment-1\weather> docker run --network mynet -p 5001:5001 --name second_serv second  
* Serving Flask app 'weather.py'  
* Debug mode: off  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on all addresses (0.0.0.0)  
* Running on http://127.0.0.1:5001  
* Running on http://172.28.0.2:5001  
Press CTRL+C to quit  
172.28.0.3 - - [09/Feb/2023 21:04:57] "GET /weather/95220 HTTP/1.1" 200 -
```

Created image and container for the first service on a network named “mynet”:

```
PS C:\Users\Chinni\Documents\2 sem\cloud\assignment-1\city> docker build -t first .  
[+] Building 4.9s (11/11) FINISHED
```

```
PS C:\Users\Chinni\Documents\2 sem\cloud\assignment-1\city> docker run --network mynet -p 5000:5000 --name first_serv first  
* Serving Flask app 'city.py'  
* Debug mode: off  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
* Running on all addresses (0.0.0.0)  
* Running on http://127.0.0.1:5000  
* Running on http://172.28.0.3:5000  
Press CTRL+C to quit  
172.28.0.1 - - [09/Feb/2023 21:04:57] "GET /cityname/Acampo HTTP/1.1" 200 -  
□
```

Images

[Give feedback](#)

An image is a read-only template with instructions for creating a Docker container. [Learn more](#)

Local

Hub

72.14 MB / 72.14 MB in use 2 images

Last refresh: about 2 hours ago



Search



<input type="checkbox"/>	Name	Tag	Status	Created	Size	Actions
<input type="checkbox"/>	first d4809931f963	latest	In use	2 minutes ago	61.45 MB	▶ ⋮
<input type="checkbox"/>	second 6f0ad444f123	latest	In use	19 minutes ago	58.16 MB	▶ ⋮

Containers

[Give feedback](#)

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)

☒ Only running

Search

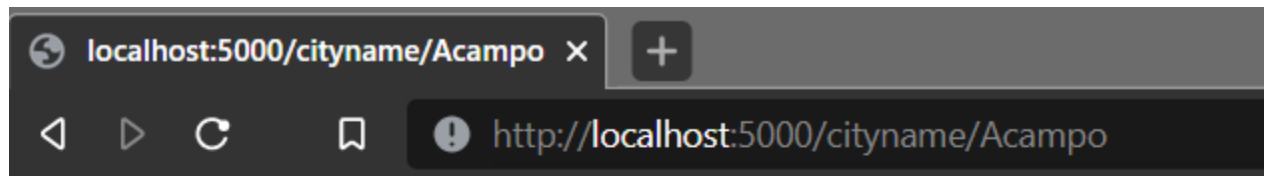


<input type="checkbox"/>		Name	Image	Status	Port(s)	Started	Act
<input type="checkbox"/>		second_serv 292b9a6077a7	second	Running	5001:5001	11 minutes ag	■
<input type="checkbox"/>		first_serv a55d4b0621a5	first	Running	5000:5000	11 minutes ag	■

This is my localhost to test two services:

It takes city name converts it to zip code and send to second service and receive the weather and prints it.

<http://localhost:5000/cityname/Acampo>



The weather in Acampo with zip: 95220 is Rainy