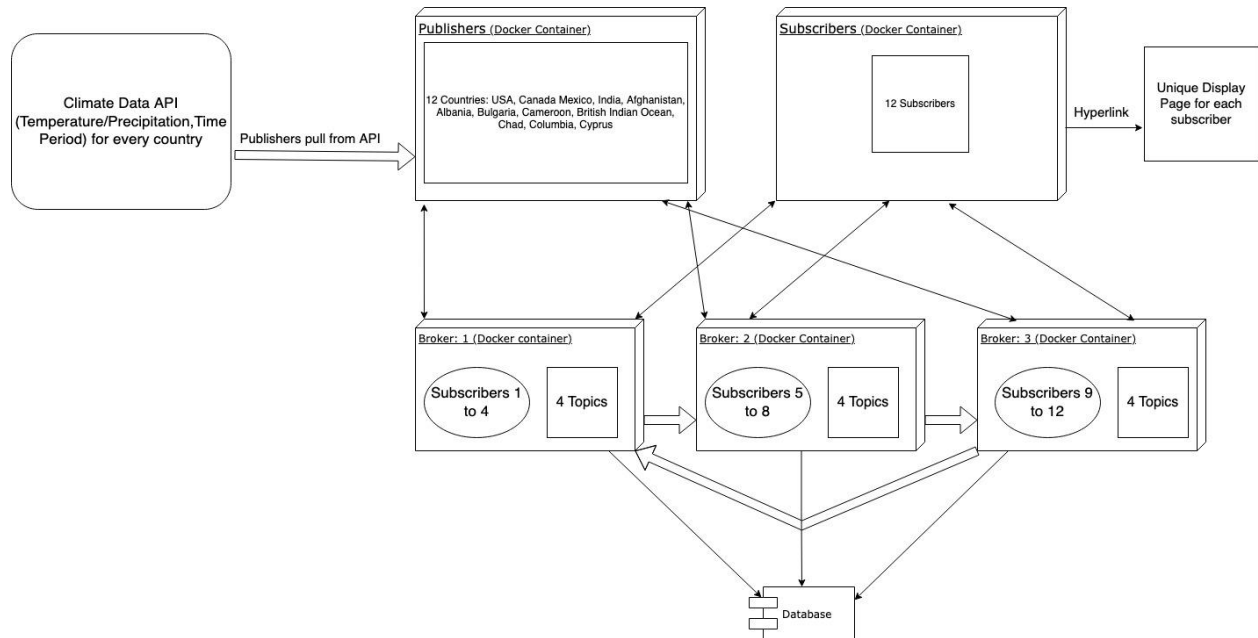# Project report Pub/Sub (Climatic Trends)

A bigger version of the diagram is attached in the zip folder.

- The Pub/Sub system is made up of 12 Publishers, 12 Subscribers, 12 Topics and 3 Brokers.
- The 12 publishers publish the climatic data from the following countries using the world bank open source data and API:
  a. USA
  b. Canada
  c. India
  d. Mexico
  e. Afghanistan
  f. Columbia
  g. Cyprus
  h. Albania
  i. British Indian Ocean
  j. Cameroon
  k. Chad
  l. Bulgarian
- There is a provision to fetch climatic data that includes either the temperature or the precipitation during the following years:
  a. 1920 to 1939
  b. 1940 to 1959

        c. 1960 to 1979
        d. 1980-1999

- The above two aspects are limited for the ease of understanding. The scope of this project extends to far more broader areas and much more data can be obtained using the Climatic data API from the world bank.

Implementation Details:

- The above architectural diagram gives an overview of how the system was defined on a high level.
- The three brokers are separate Docker containers and they communicate with each other using the rendezvous method.
- Canadian, USA, Indian and Mexican temperatures are the four topics mapped to the first Broker. Subscribers 1,2,3 and 4 are also mapped to the first Broker.
- Afghanistan, Albanian, Bulgarian and Cameroonian temperatures as well as Subscribers 5,6,7,8 are mapped to the second Broker.
- British Indian Ocean, Chad, Columbian and Cyprus temperatures as well as Subscribers 9,10,11,12 are mapped to the third Broker.
- The above diagram has the details in visual form.
- There are total of 5 Docker containers: Publisher, Subscriber and 3 Brokers
- There is no direct communication between Publishers and Subscribers, it is all via one of the Brokers.
- The database is common and mounted to each Broker.

Working:

- The Publisher has the functionality to either Publish the message based on the topics subscribed by the corresponding subscribers or they can Advertise it common to its subscribers as something that is upcoming.
- The Subscriber subscribes or unsubscribes to the topics and all operations are done by their corresponding Brokers, the request is posted from the Subscriber to the Broker, similar is the case for the Publisher who sends the data fetched from the API to the Broker and the Broker writes it to the Database.
- If the Subscribers assigned to their default Brokers does not have the topics that they are looking for, then the request rerouted to the next nearest Broker and so forth until the topic is found.
- The Subscribers can see the data that they are subscribed to, at the bottom part of the landing page that directs them to a display page that contains the information to their subscriptions and the data published by the publishers that they are subscribed to.

Contributions:

- This project was done by Amrit Sreekumar and Nikhil Ramesh.
- Both of us contributed equally to the project.
- Amrit Sreekumar was responsible for writing up the Publisher Functions and Dockerizing the Publishers and for the communication between the Publisher Docker and the Brokers.
- Nikhil Ramesh was responsible for writing up the Subscriber Functions and Dockerizing the
  Subscribers and for the communication between the Subscriber Docker and the Brokers.
- Both our ideas were combined to implement the rendezvous method, Dockerizing the three Brokers and setting up the communication between them.
- This Document was a combined effort and so is the Readme that details how to run the system, attached with the Zip folder.

Containers / Apps
Images
Volumes
Dev Environments PREVIEW

Search...   Sort by

EXITED (0)   PORT: 5000

dazzling_elgamal flaskapp
EXITED (0)   PORT: 5000

recursing_wu flaskapp
EXITED (0)   PORT: 5001

jolly_thompson flaskapp
EXITED (0)   PORT: 5000

wizardly_booth flaskapp
EXITED (0)   PORT: 7000

gifted_shamir flaskapp
EXITED (2)   PORT: 7000

crazy_mendeleev flaskapp
EXITED (0)

peaceful_shaw flaskapp
EXITED (0)

trying_out
EXITED (UNDEFINED)

distributed-system
RUNNING   Open in Visual Studio Code

    sub distributed-syst...
    RUNNING   PORT: 5002

    broker3 distributed-syst...
    RUNNING   PORT: 5004

    pub distributed-syst...
    RUNNING   PORT: 5001

    broker distributed-syst...
    RUNNING   PORT: 5000

    broker2 distributed-syst...
    RUNNING   PORT: 5003

Containers / Apps
Images
Volumes
Dev Environments PREVIEW

distributed-system-main
/Users/amritsreekumar/Desktop/Distributed-System-main

CONTAINERS

sub distributed-syst...
RUNNING   PORT: 5002

broker2 distributed-syst...
RUNNING   PORT: 5003

broker3 distributed-syst...
RUNNING   PORT: 5004

pub distributed-syst...
RUNNING   PORT: 5001

broker distributed-syst...
RUNNING   PORT: 5000

```
Attaching to pub, broker, broker2, broker3, sub
broker2  | * Serving Flask app 'app' (lazy loading)
broker2  | * Environment: production
broker2  |   WARNING: This is a development server. Do not use it in a production deplo
yment.
broker2  |   Use a production WSGI server instead.
broker2  | * Debug mode: off
broker   | * Serving Flask app 'app' (lazy loading)
broker   | * Environment: production
broker   |   WARNING: This is a development server. Do not use it in a production deplo
yment.
broker   |   Use a production WSGI server instead.
broker   | * Debug mode: on
broker3  | * Serving Flask app 'app' (lazy loading)
broker3  | * Environment: production
broker3  |   WARNING: This is a development server. Do not use it in a production deplo
yment.
broker3  |   Use a production WSGI server instead.
broker3  | * Debug mode: on
sub      | * Serving Flask app 'app' (lazy loading)
sub      | * Environment: production
sub      |   WARNING: This is a development server. Do not use it in a production deplo
yment.
sub      |   Use a production WSGI server instead.
sub      | * Debug mode: on
sub      | * Running on all addresses.
sub      |   WARNING: This is a development server. Do not use it in a production deplo
yment.
sub      | * Running on http://172.19.0.2:5000/ (Press CTRL+C to quit)
```

Search...   Stick to bottom