Document: Must include a description, all the dependencies and requirements, your architecture diagram, steps to instantiate the system, how to interact with it. You must also list down the contribution by each team member

The application is contained in the docker file, and we just have to run the following commands to start the application:

1. docker build -t waldowallpaper:latest --no-cache .
2. docker run -d -p 8000:8000 waldowallpaper:latest
3. After running these commands, waldowallpaper container will start up and we can see the web app in the URL http://127.0.0.1:8000/waldo

Following are the frameworks we have used in this application

* Backend (Python / Django / Flask)
* Frontend (Html / CSS)
* Docker
* Database (SQL)
* Kafka

There is a requirements.txt file alongside the code which contains the complete details of all the libraries and dependencies used by the application.

The ideology behind the application:

* Waldo is a simple pub/sub app which allows users to sign up with their data and we notify them every ‘n’ hour with their choices of wallpaper. The notification will be made through email.
* User preference is saved in the database, Once the user hits the notify method, backend fetches new images and validates against the user selected genres. We construct the email using fetched images & user preference and the users will be notified with it.
* To send out the email, we will use the inbuilt python module.
* Client-server interaction is simple here. The frontend doesn’t have a special framework here. It's a simple html/CSS page. On completion of signup, the action is to record the data in the database through python.
* There are at-present **9 docker containers**. (One for database, one for producer, four for Kafka (Kafka contains three brokers, one zookeeper) and three consumers (b1, b2, b3)).
* Every consumer upon receiving the data validates against the user list and identifies the list of users to be notified. It works on rendezvous algorithm. (Identification of nodes and Identification of users). After the above is completed. Users get notified.

**Dockerised the Database:**

We are using Postgres database inside a container and the command to start the database srever is as given below. We have created a separate docker volume to make postgres persistent. The command to create the docker volume is also given below:

Create volume: *docker create -v /var/lib/postgresql/data --name PostgresData alpine*

Create Network: *docker network create --driver bridge waldo-network*

Run postgres container: *docker run -tid -p 5432:5432 --name postgres --network waldo-network -e POSTGRES\_PASSWORD=admin -d --volumes-from PostgresData postgres*

Run waldowallpaper container: *docker run -tid -p 8000:8000 --network waldo-network waldowallpaper:latest*

Run Broker1 container: *docker run -tid -p 9000:9000 --name b1-broker --network waldo-network b1-broker*

Run Broker2 container: *docker run -tid -p 9001:9001 --name b2-broker --network waldo-network b2-broker*

Run Broker3 container: *docker run -tid -p 9002:9002 --name b3-broker --network waldo-network b3-broker*

To create a kafka topic , you must be in the bin directory

To compose:

docker compose -f docker-compose.yml up -d

To navigate to bin:

cd opt/kafka

cd bin

To create topic :

kafka-topics.sh --create --zookeeper zookeeper:2181 --replication-factor 1 --partitions 3 --topic anime

To list down all the present topics in that zookeeper:

bin/kafka-topics.sh --list --zookeeper zookeeper:2181

Consumer Details:

Topics in the following section means (Genre in the application. Our application allows user to select up to 12 genres).

3 Topics are handled by consumer 1

6 Topics are handled by consumer 2

3 Topic are handled by consumer 3.

contribution by each team member :

The entire project has been contributed by both of us in a fair manner. None of the entire module has been built by either of us or we shared each of the work completely.

**Architecture:**

Diagram

Description automatically generated

Project checklist:

* Built on top of phase 2
* Everything is dockerized
* Specs for the system:
  + Topics : 12
  + Subscriber : 12
  + Kafka broker nodes: 3
* Kafka running with prebuilt docker images (kafka python)
* Topics with multiple partitions and replication.
* Data transmission through producer and consumer.
* All functionalities working.