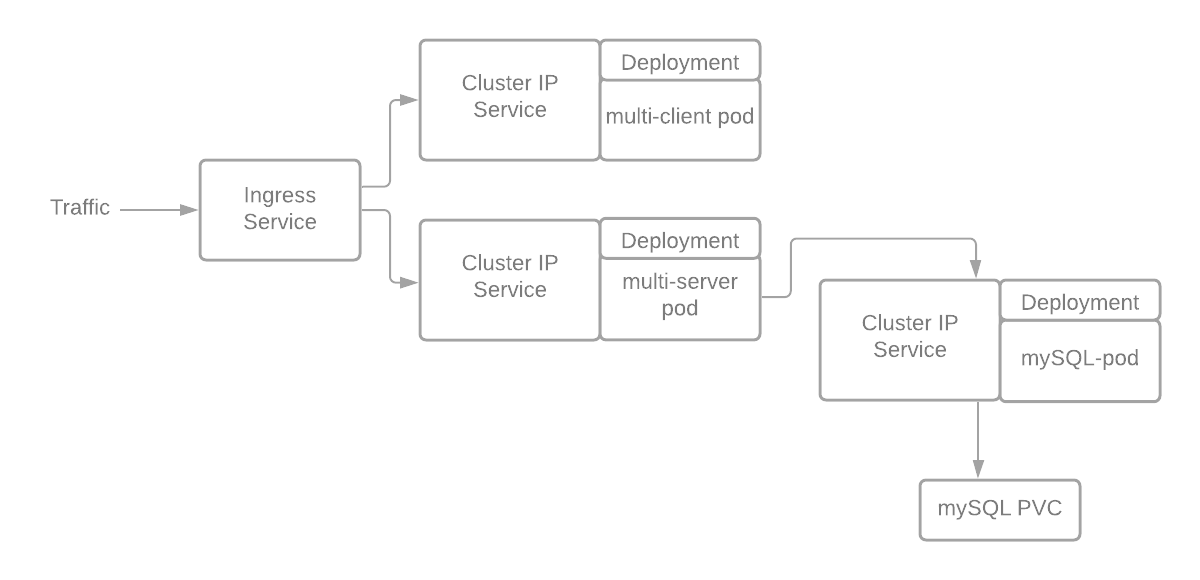
# HOTEL MANAGEMENT SYSTEM

The created website is a hotel management system. This management system will serve as a database of hotels, one that a user can use to book hotels and rooms online and in general, plan their stays at hotels. The database keeps a catalogue of a number of hotels who have registered with the database. It lists its entire selection of hotels, and for all those hotels, its entire selection of rooms. Users will need to sign up with the management system to be able to avail its services, hence the system also has a simple **login** and **signup** functionality. It allows users to **book rooms** that they want to stay in during their trip, and when the time comes, it lets users **check in** to and **check out** of their reservations. Further on, it will also have a **rating and review** system, allowing users to rate and review the hotels and the rooms they stayed at, and their overall experience with the service.

The project is implemented using **NodeJS, MySQL, Kubernetes, Docker, Shell and ReactJS**. The structure of the project is as denoted by the diagram below;



**Usage**

**Using Docker**

1. Once docker and docker-compose are installed run **docker-compose up –build**.
2. This will trigger the pulling and building of the necessary images and will use **docker-compose**to setup the app and the MySQL containers.
3. Once the Postgres MySQL starts up (note that it may take a few minutes until it starts accepting connections the first time) it will automatically run the database migrations using the .sql scripts in **init** directory.
4. After everything is up and running you can start using the app running on port **localhost:3050** in browser of your choice.
5. The MySQL database data directory is setup to be mounted in the **data/db** directory at the root of the project. In case you want to start afresh just issue **rm -rf data**.

**Using Minikube + Kubernetes**

1. **(OPTIONAL STEP; images have already been pushed to the public repository** [**https://hub.docker.com/repository/docker/waleedakramkhann/hotelclient**](https://hub.docker.com/repository/docker/waleedakramkhann/hotelclient)**).**

Create an image for **client** and push it to dockerhub: Move to **client** directory **cd client/**. Build the image with Dockerfile **docker build -t <hub-user>/hotelclient.** Note that we are inside **client** directory. **<hub-user>/hotelclient** would be name of our image. You can check your newly built image using **docker images**. Push the images to Dockerhub **docker push <hub-user>/hotelclient**

1. **(OPTIONAL STEP; images have already been pushed to the public repository** [**https://hub.docker.com/repository/docker/waleedakramkhann/hotelserver**](https://hub.docker.com/repository/docker/waleedakramkhann/hotelserver)**).**

Create an image for **server** and push it to dockerhub: Move to **server** directory **cd server/.** Build the image with Dockerfile **docker build -t <hub-user>/hotelserver.** Note that we are inside **server** directory. **<hub-user>/hotelserver** would be name of our image. You can check your newly built image using **docker images**. Push the images to Dockerhub **docker push <hub-user>/hotelserver**

1. Run **bash deploy.sh**. Once everything is deployed and running the login page will automatically launch in the default browser.