**Kubernetes**

I have documented my learning and implementation of kubernetes by containerizing Microservices using Docker and have deployed & managed those containers using kubernetes. I have used (GKE)Google Kontainer Engine as the cloud provider with my personal account. I did kubernetes container managerment in a declarative way by using yaml file’s

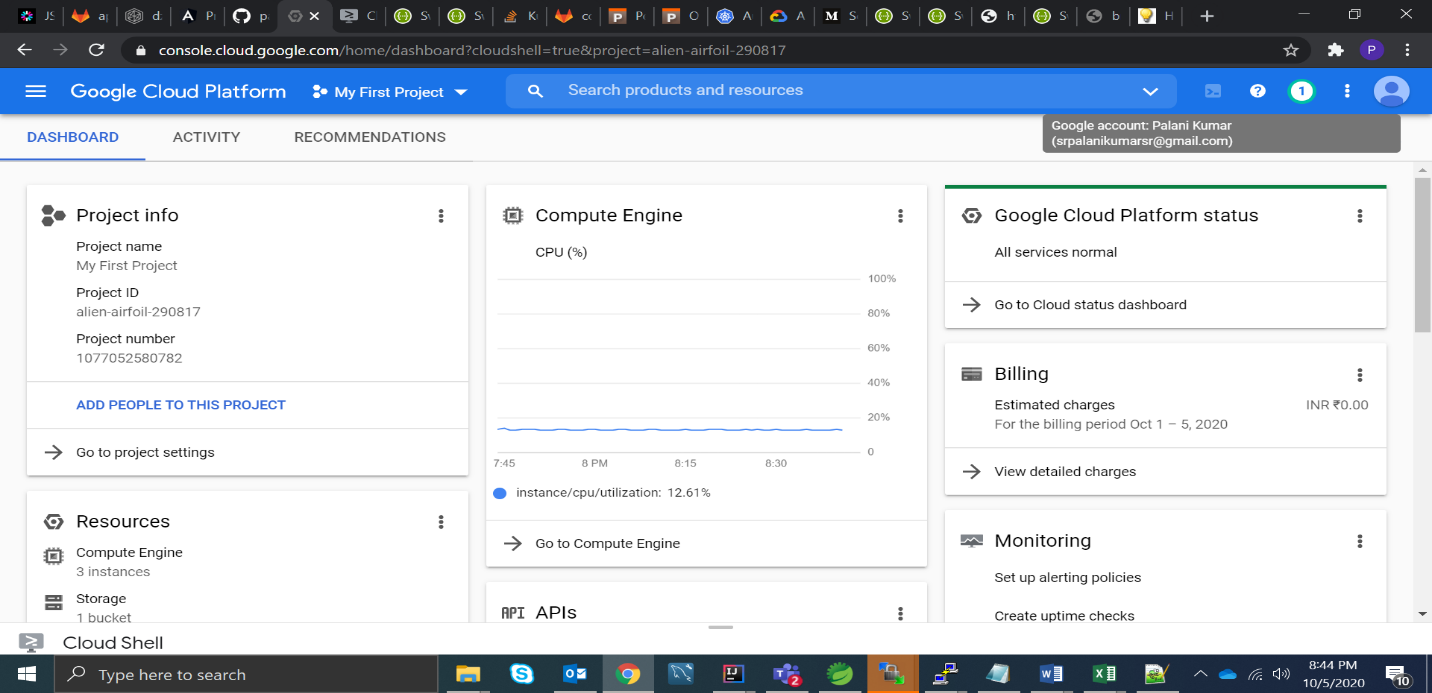
* Containerization
* Deployment
* Data Persistance
* Scalability
* Demo

**Google Kontainer Engine(GKE)**

Have used GKE as the cloud provider for implementing kubernetes.

I Have attached edited screenshots in this document.

Created a Kubernetes cluster with auto-scaling enabled with command

gcloud container node-pools create cluster --cluster=cluster-1 --enable-autoscaling  
--min-nodes=2 --max-nodes=5  --zone=us-central-c 

**Containerization**

I have used Java based MicroServices for this kubernetes course.

Github link of MicroServices: <https://github.com/palani16/spring-boot-301>

Dockerfile:

FROM openjdk:8

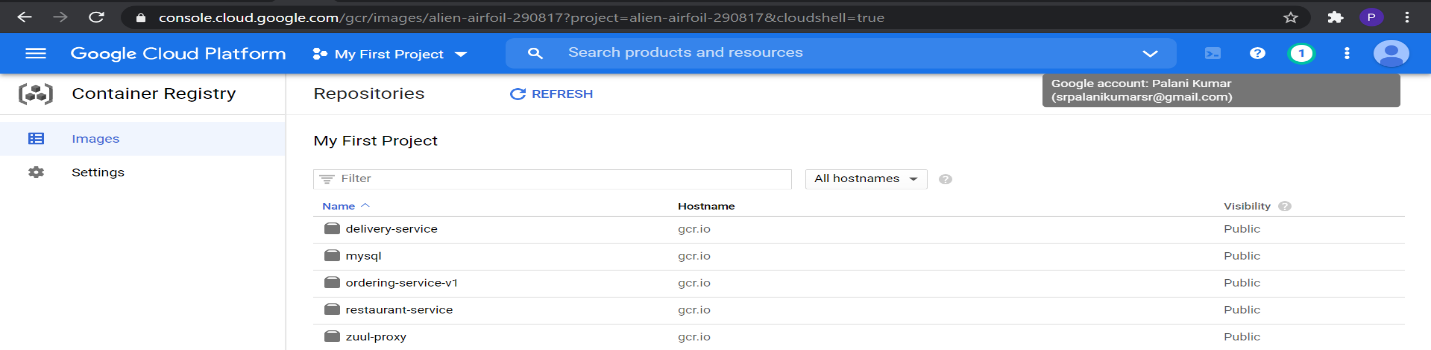
COPY ./target/orderingservice-\*.jar orderingservice.jar

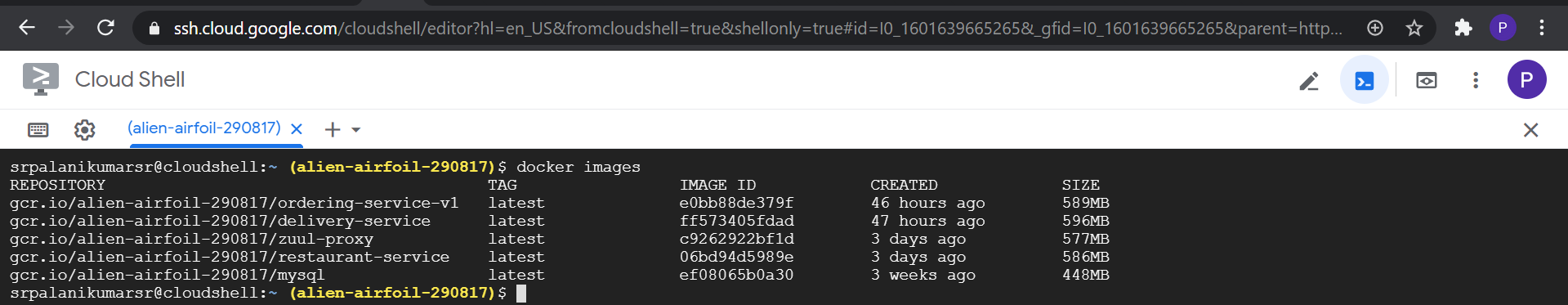
EXPOSE 9112

CMD ["java","-jar","-Dspring.profile.active=dev","orderingservice.jar"]

Containerized the services by creating Docker file and building the Docker image using docker command docker build –t <image-name> .

Tagged and Pushed Docker images of individual services into google container registry





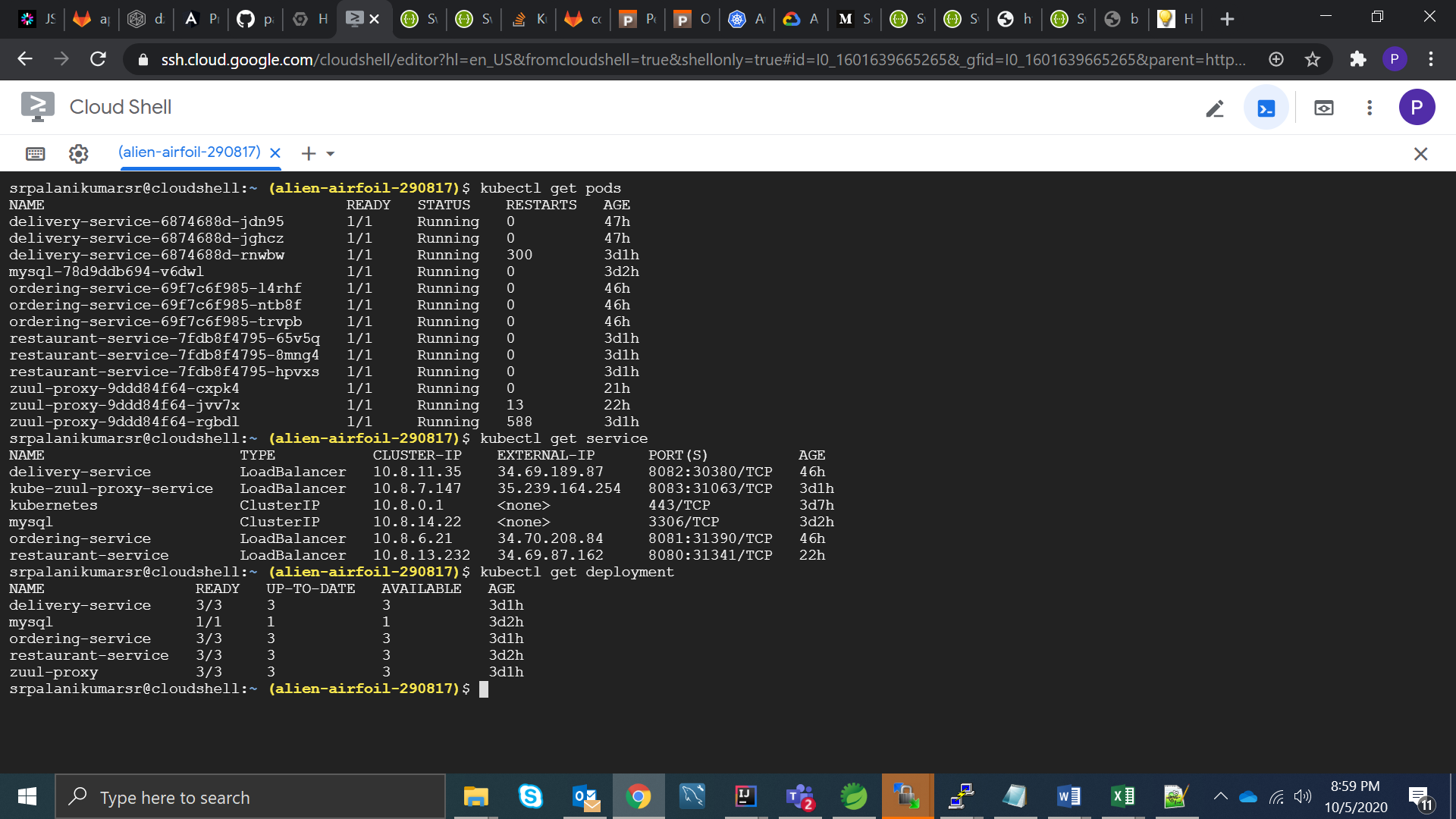
**Deployment**

Have created a individual kubernetes pod for each container

And microservices communicate with each other with kubernetes discovery client by spring cloud.

I have attached all the deployment ymal file along with this document. Deployment yaml also contains kubernetes service configurations in the same yaml files

Each pod has been created with 3 replica sets mentioned in yaml file



**Data Persistance**

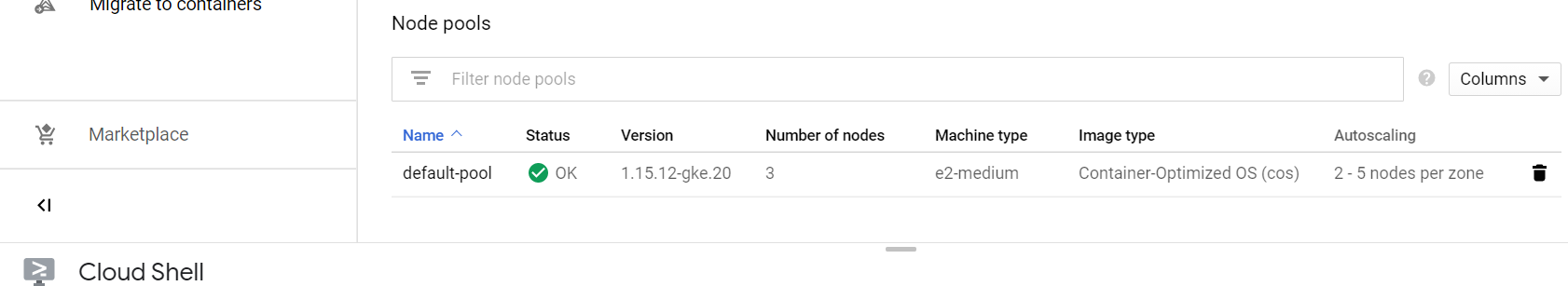
I have used mysql database for my backend services, used docker image of mysql containerized it, packed it into a pod and deployed with kubernetes.

Since mysql is running as a container inside pod if there is restart or failure in pod or container data inside the container would be clear, so I have created a persistant volume in kubernetes where if data is mounted outside the pod or container. So irrespective of pod lifecycle data will be persisted.  
apiVersion: v1  
kind: PersistentVolume  
metadata:  
  name: mysql-pv-volume  
  labels:  
    type: local  
spec:  
  capacity:  
    storage: 2Gi  
  accessModes:  
    - ReadWriteOnce  
  hostPath:  
    path: "/data"

**Scalability**

I have ensured scability by enabling autoscaling and by providing the minimum and maximum number of nodes while creating the cluster with command in GKE which manages scaling up and scaling down automatically

gcloud container node-pools create cluster --cluster=cluster-1 --enable-autoscaling  
--min-nodes=2 --max-nodes=5  --zone=us-central-c



**Auto-Healing of Pods:**

Auto-healing of containers happens in kubernetes by default when the deployement is provided in declarative way using yaml file, as it checks always with desired state & tries to match the desired state and restarts the pods as soon as when pods or containers goes down.

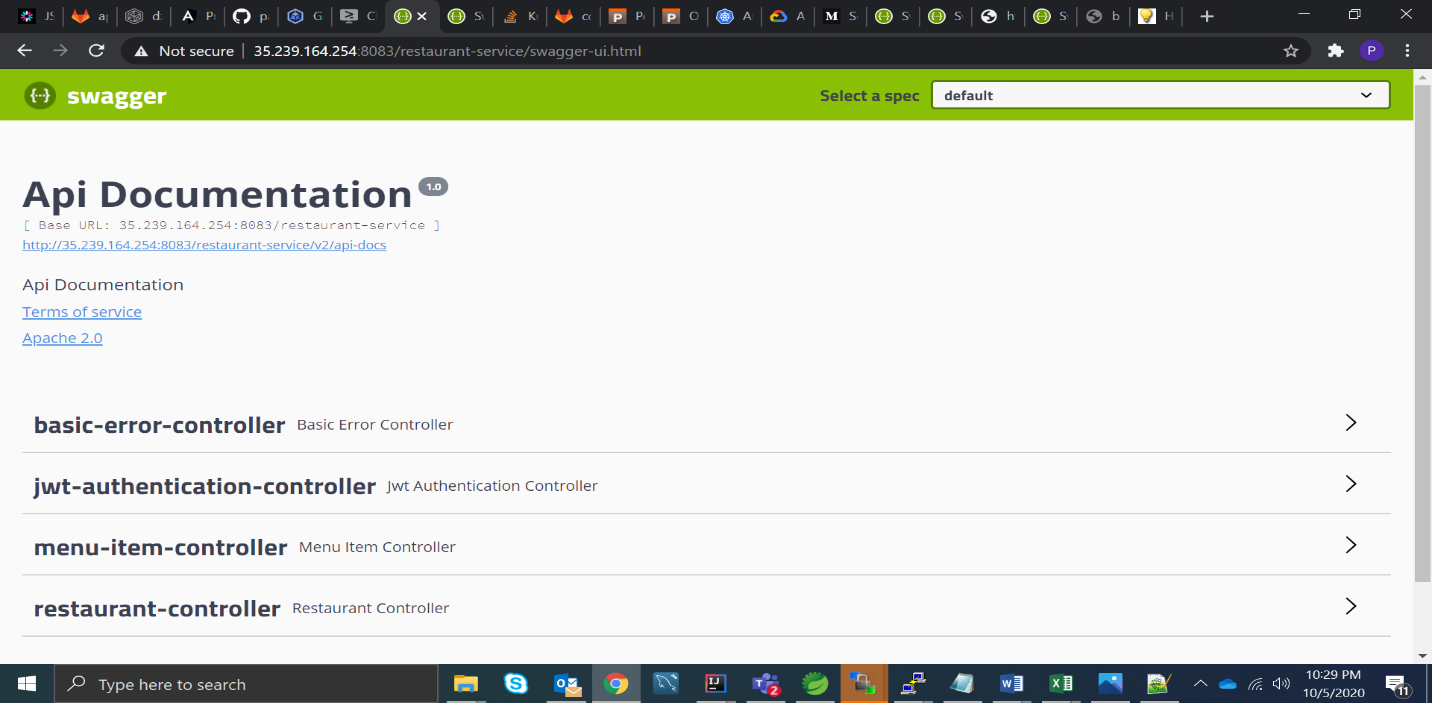
**Demo**

Please find the below links of microservices handled by a zuul-gateway running on port 8083 routes the incoming request to all other services.

<http://35.239.164.254:8083/ordering-service/swagger-ui.html>

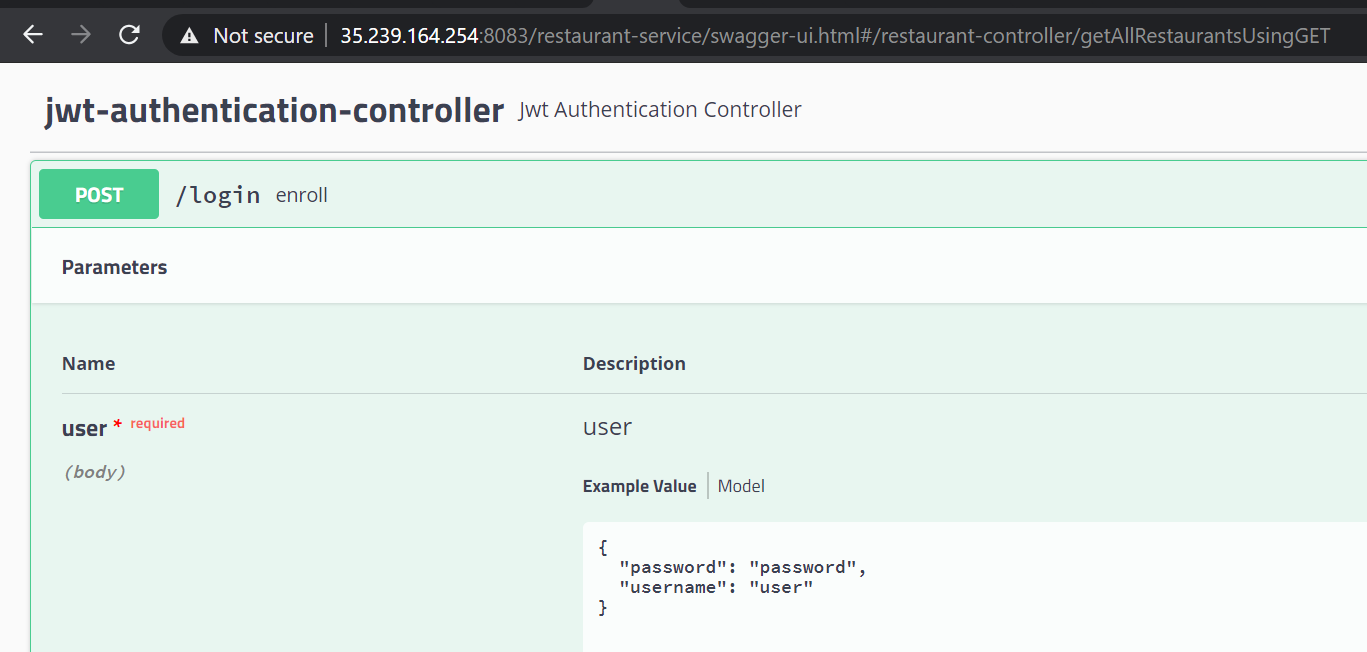
<http://35.239.164.254:8083/restaurant-service/swagger-ui.html>

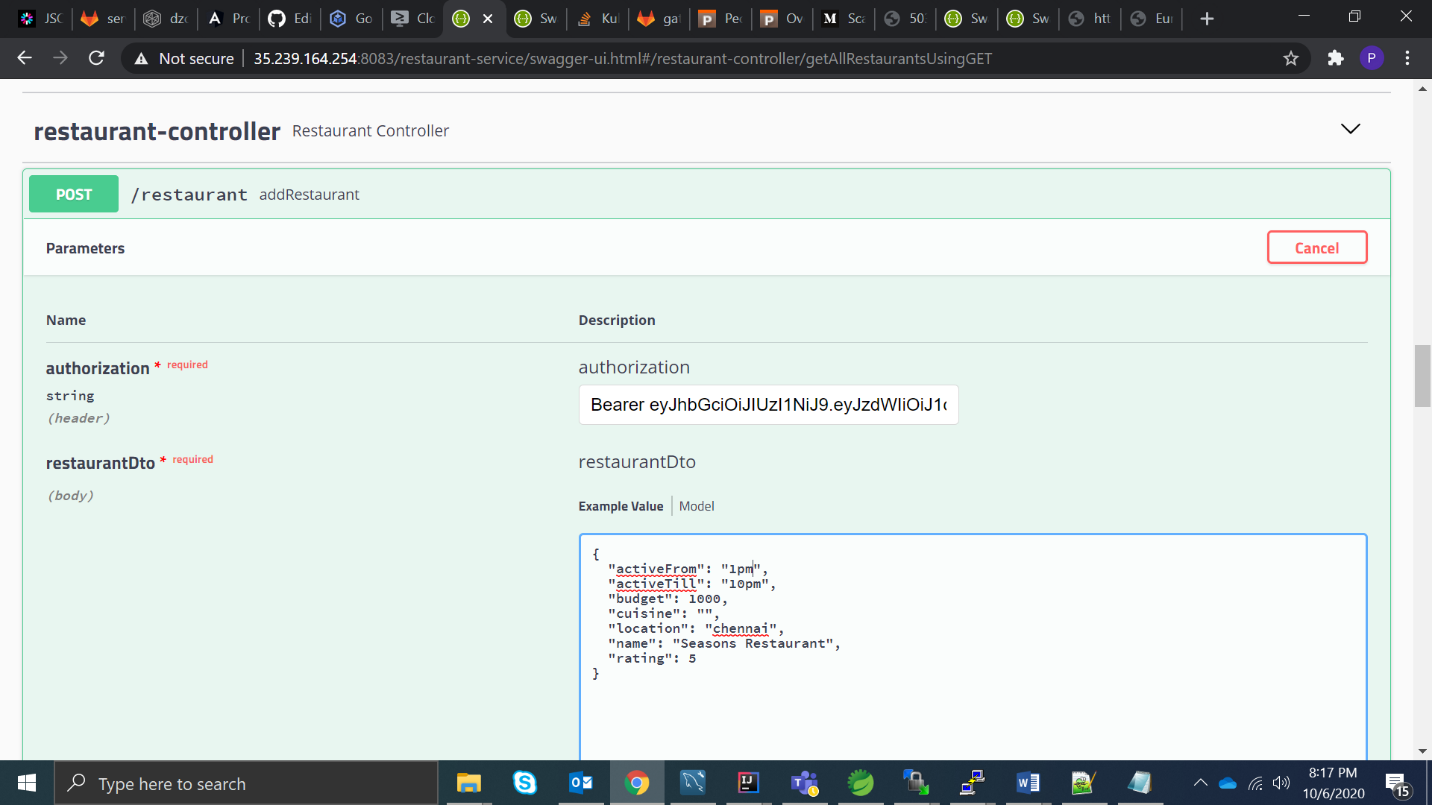
<http://35.239.164.254:8083/delivery-service/swagger-ui.html>

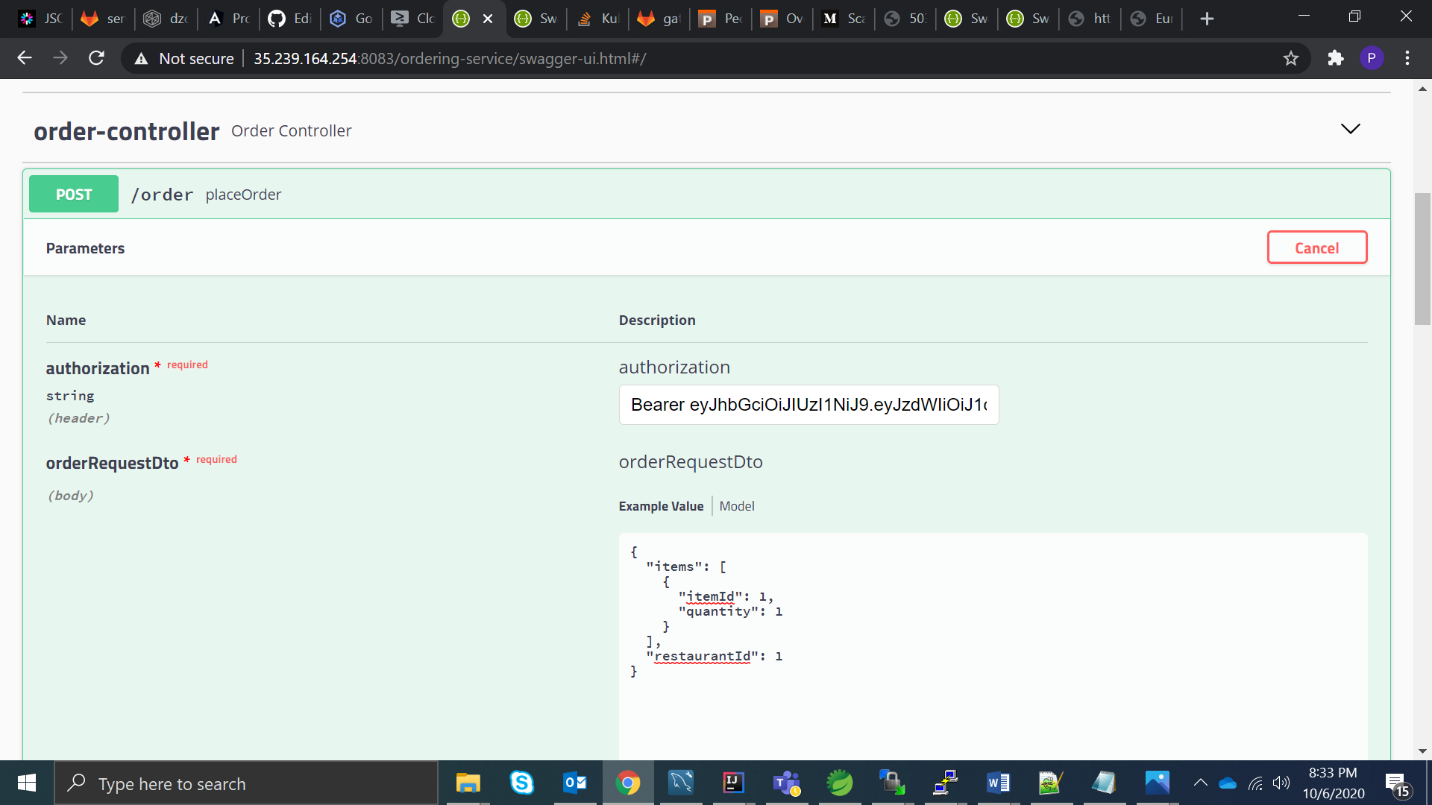


Json’s and Api’s:

All Api’s are secured with a Authorization Header. Use Jwt-Authentication Controller with login Api to get Authorization token with user name as “user” and password as “password”







Place order API has inter-service call with restaurant-service using feign with ability of kubernetes discovery client enabled helping for inter-service communication

Data persistence can be checked by using Get Api’s