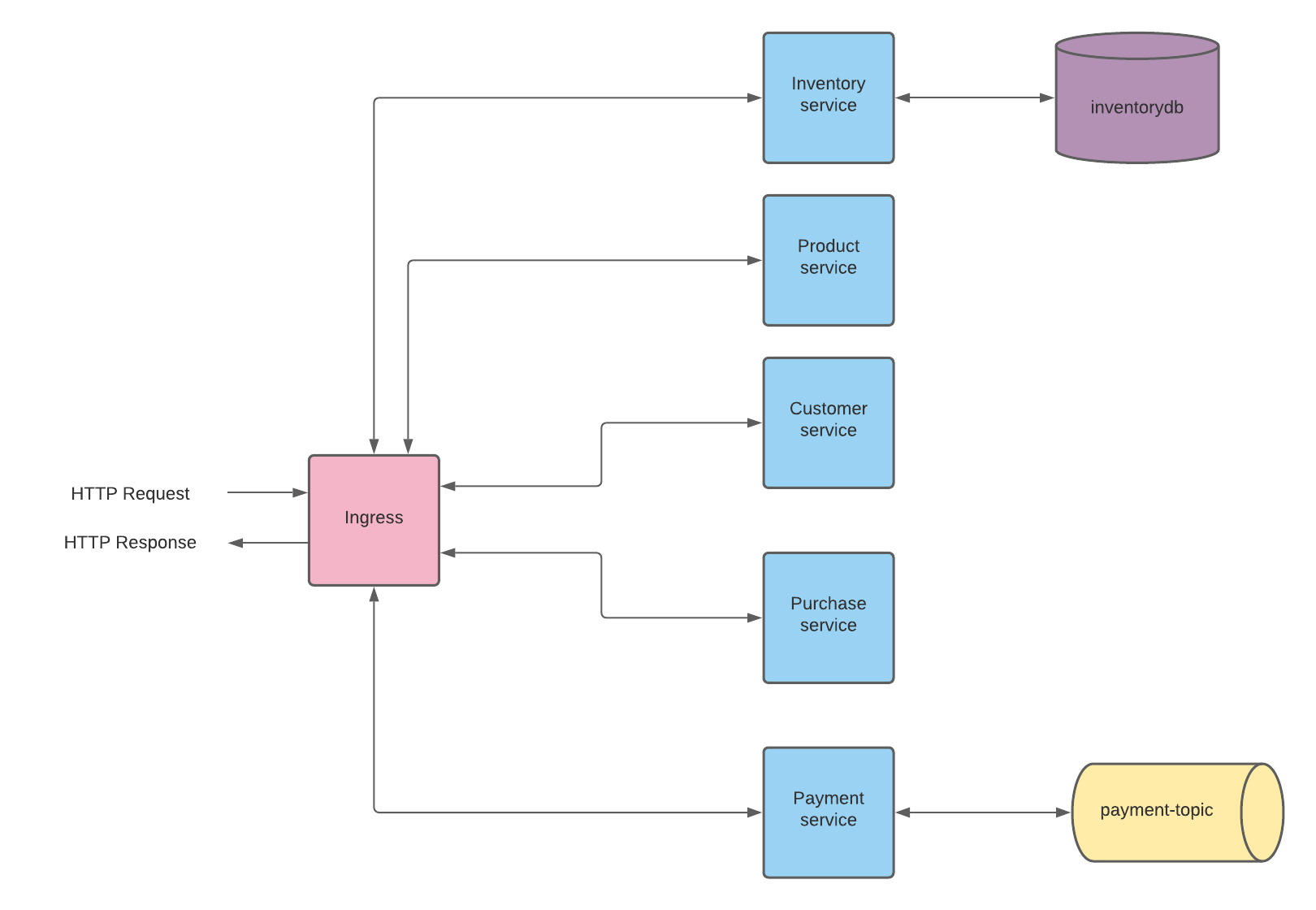
Lab Architecture:

Following diagram depicts the overall lab architecture and inter service communication.

**Create and install the database**

Creation and installation of DB is one time activity. To create DB following commands should be executed

Execute following commands to create DB:

gcloud sql instances create inventorydb --database-version=MYSQL\_5\_7 --tier=db-n1-standard-1 --region=us-east1 --root-password=Optimiz@dm1n

gcloud sql databases create DATABASE --instance=inventorydb

Once the database is created, connect to it and create table using following commands on SQL shell:

gcloud sql connect inventorydb --user=root --quiet

show databases;

use inventorydb;

CREATE TABLE `product\_info` (

`product\_id` int(11) NOT NULL AUTO\_INCREMENT,

`product\_name` varchar(200) NOT NULL,

`product\_description` varchar(200) NOT NULL,

`manufacturer\_name` varchar(200) NOT NULL,

`manufacturer\_id` int(11) NOT NULL,

`seller\_name` varchar(200) NOT NULL,

`seller\_id` int(11) NOT NULL,

PRIMARY KEY (`product\_id`)

);

Create user ‘optimiz’ with password ‘Optimiz@Adm1n’ using GCP console

**How to create a pub-sub topic?**

Create topic payment-topic using payment-subscription using GCP console

Note: this is one time activity and don't delete this topic if it exists

**How to start the lab, deploy services and remove all services when work is done?**

1. Start the lab using command - ./startlab.sh lab-application-cluster us-central1-c

Here lab-application-cluster is name of the cluster and ‘us-central1-c’ is region name

1. Deploy lab using following command - ./labdeploy.sh

Once the command finishes see if all services and ingress is deployed in GCP console and using following command

kubectl get services

Kubectl get ing

Check the ingress status in the GCP console, it should be green

1. Once the lab work is done destroy the lab using following command: ./labdeploy.sh

Once the command finishes see if all services and ingress are removed in GCP console and using following command

kubectl get services

Kubectl get ing

Above command should not display inventory lab related services

1. Once the services are deleted stop the lab using following command

./stoplab.sh lab-application-cluster us-central1-c

Ensure that lab is specified by checking the number of allotted for the cluster, they should be 0

1. Stop the Cloud SQL database in the GCP Console using Cloud SQL service. Use search box to locate the service

**How to test services are deployed or not**

If following endpoints are executing successfully then lab is deployed successfully

We can replace the DNS entry with the ingress IP to check ingress deployment

1. HTTP GET : <http://inventoryapp.optimiz.site/api/product/listProductCatalog>

2. HTTP GET : <http://inventoryapp.optimiz.site/api/product/getCustomers>

3. HTTP POST: <http://inventoryapp.optimiz.site/api/product/addProduct>

Body:

{

"productName":"IPhone",

"productDescription":"Phone",

"manufacturerName":"Apple",

"manufacturerId":2,

"sellerName":"DigitalShopee",

"sellerId":2

}

3. HTTP GET : <http://inventoryapp.optimiz.site/productservice/getProducts>

4. HTTP DELETE: <http://inventoryapp.optimiz.site/api/product/deleteProduct>

**How to deploy image of the new service**

Sometimes we might need to modify and change the code or configuration and we need to publish service once it is done. Following sequence of commands will publish the service image to GCP repo

1. Compile service
2. Create docker image of the service
3. Push the image on the GCP image repository

mvn clean install

docker build -t gcr.io/optimiz-lab/<ServiceName>:<ImageTag>.

docker push gcr.io/optimiz-lab/<ServiceName>:<ImageTag>

E.g.

For inventoryservice following are the set of commands. If we have to build it for other service like productservice we will use same steps and replace inventoryservice with productservice

mvn clean install

‘docker build -t gcr.io/optimiz-lab/inventoryservice:v2 .’

‘docker push gcr.io/optimiz-lab/inventoryservice:v2’

Once the lab is published change the image tag in the deployment yaml file for that specific service. E.g we have updated image for inventory from v1 to v2 so we will update deployment file as follows (refer the image section), similar changes should be done with other servies if needed:

apiVersion: apps/v1

kind: Deployment

metadata:

name: inventoryservice

namespace: default

spec:

selector:

matchLabels:

run: inventoryservice

template:

metadata:

labels:

run: inventoryservice

spec:

containers:

- image: gcr.io/optimiz-lab/inventoryservice:v2

imagePullPolicy: IfNotPresent

name: inventoryservice

ports:

- containerPort: 8080

protocol: TCP