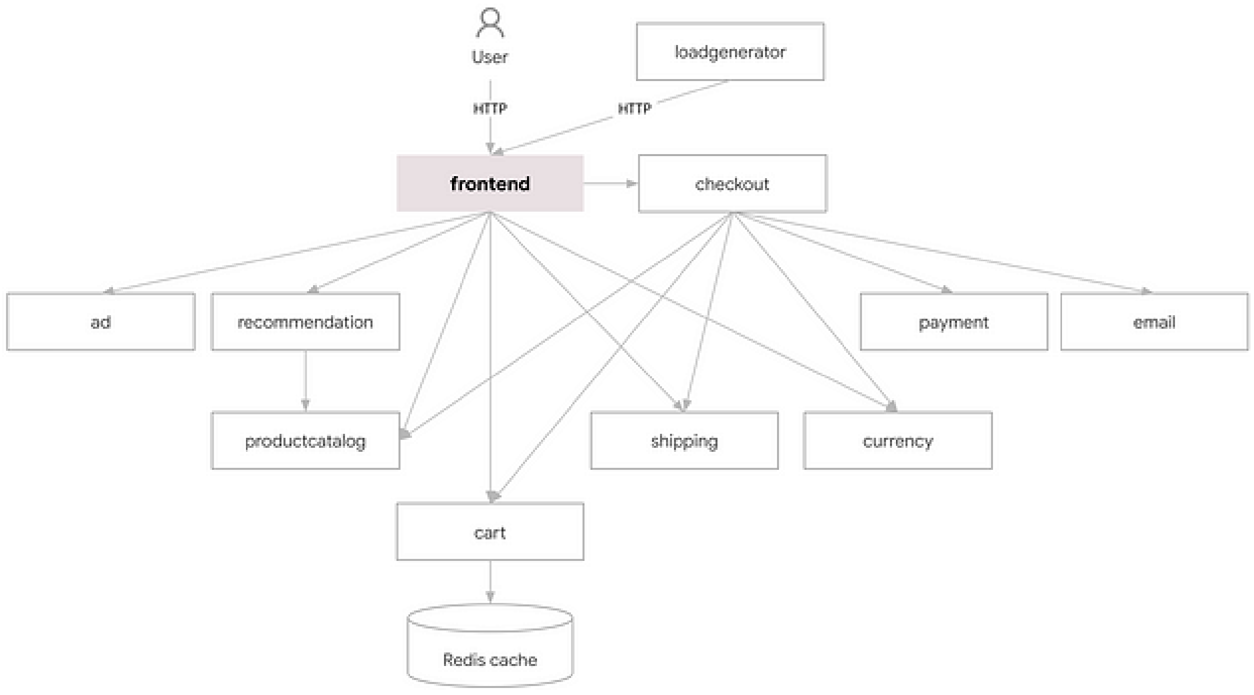
Microservice E-commerce Application

Online Boutique is a cloud-first microservices demo application. The application is a web-based e-commerce app where users can browse items, add them to the cart, and purchase them.

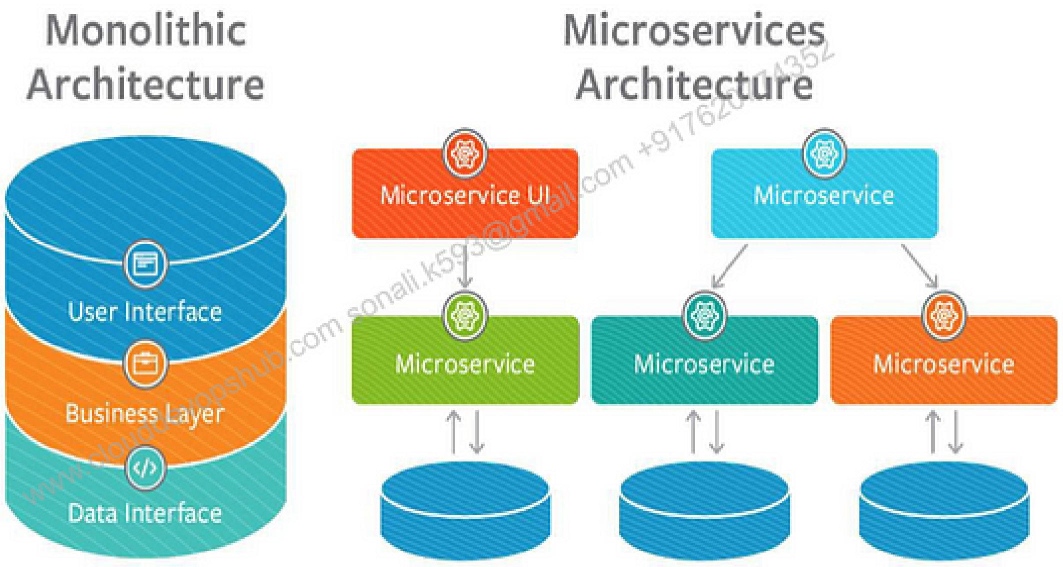
Google uses this application to demonstrate how developers can modernize enterprise applications using Google Cloud products. This application works on any Kubernetes cluster.

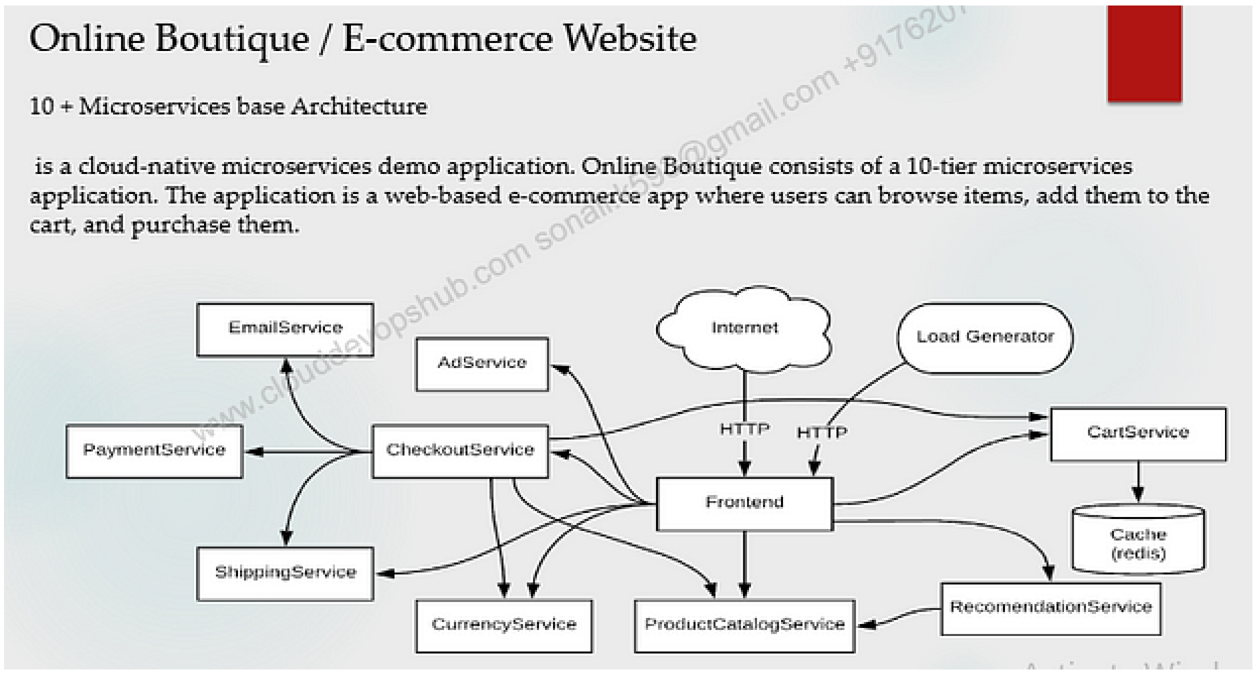
Architecture

Online Boutique is composed of 11 microservices written in different languages that talk to each other



Monolithic Architecture and Microservice Architecture :

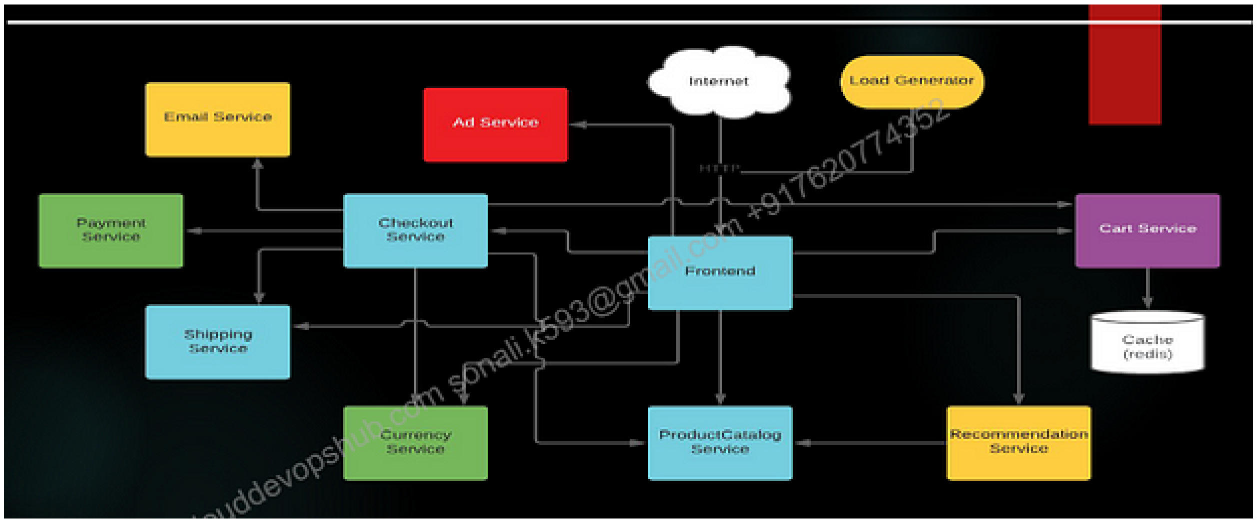




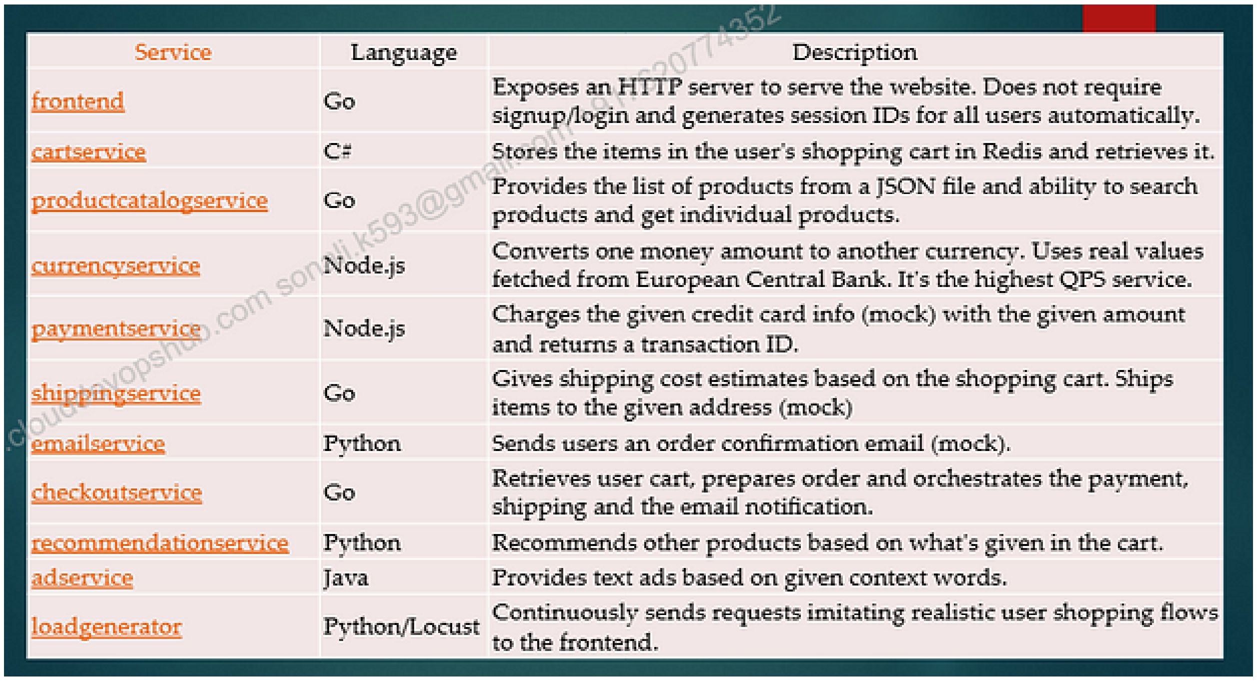
Online Boutique / E-commerce Website

10 + Microservices base Architecture

is a cloud-native microservices demo application. Online Boutique consists of a 10-tier microservices application. The application is a web-based e-commerce app where users can browse items, add them to the cart, and purchase them.



Microservice used in this Project:



Service	Language	Description
frontend	Go	Exposes an HTTP server to serve the website. Does not require signup/login and generates session IDs for all users automatically.
cartservice	C#	Stores the items in the user's shopping cart in Redis and retrieves it.
productcatalogservice	Go	Provides the list of products from a JSON file and ability to search products and get individual products.
currencyservice	Node.js	Converts one money amount to another currency. Uses real values fetched from European Central Bank. It's the highest QPS service.
paymentservice	Node.js	Charges the given credit card info (mock) with the given amount and returns a transaction ID.
shippingservice	Go	Gives shipping cost estimates based on the shopping cart. Ships items to the given address (mock)
emailservice	Python	Sends users an order confirmation email (mock).
checkoutservice	Go	Retrieves user cart, prepares order and orchestrates the payment, shipping and the email notification.
recommendationservice	Python	Recommends other products based on what's given in the cart.
adservice	Java	Provides text ads based on given context words.
loadgenerator	Python/Locust	Continuously sends requests imitating realistic user shopping flows to the frontend.

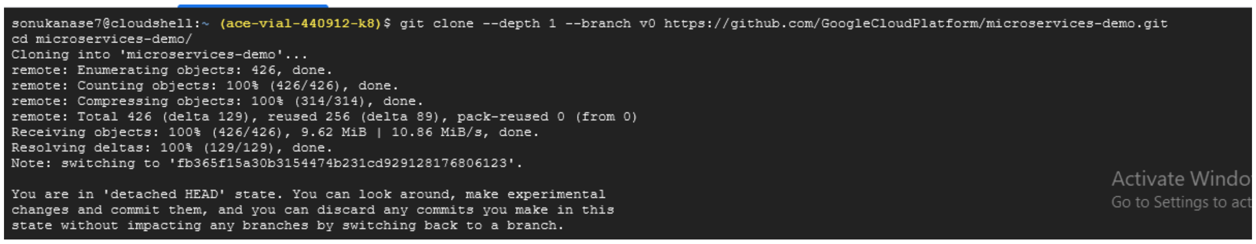
Quickstart (GKE)

1. Ensure you have the following requirements:

- Google Cloud project.
- Shell environment with `gcloud`, `git`, and `kubectl`.

2. Create a GKE cluster and get the credentials for it.

3. Clone the latest major version.

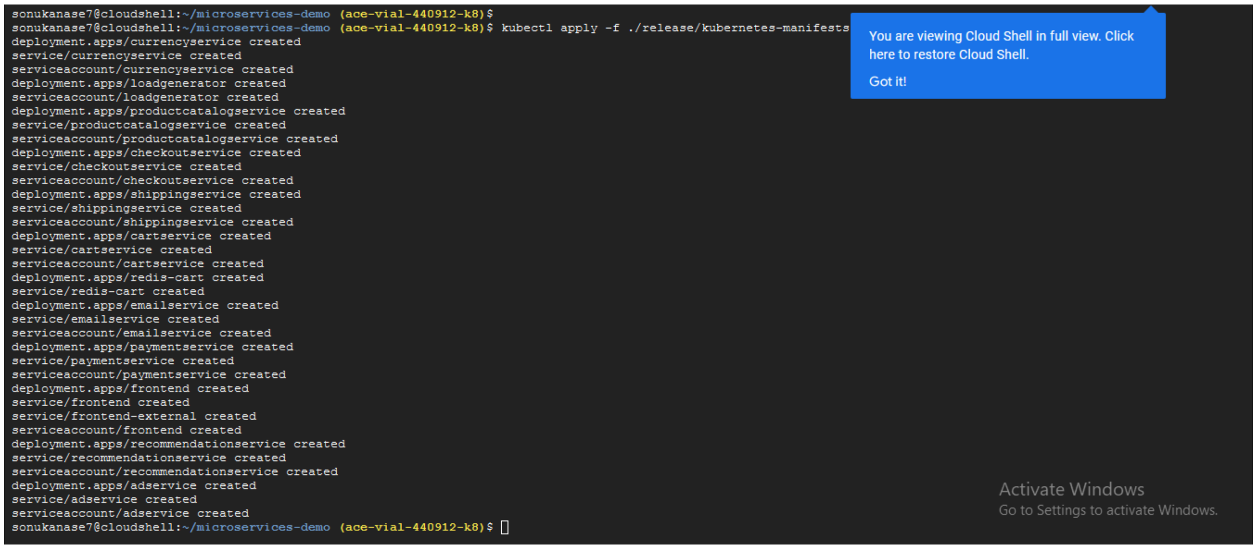


```
sonukanase7@cloudshell:~ (ace-vial-440912-k8)$ git clone --depth 1 --branch v0 https://github.com/GoogleCloudPlatform/microservices-demo.git
cd microservices-demo/
Cloning into 'microservices-demo'...
remote: Enumerating objects: 426, done.
remote: Counting objects: 100% (426/426), done.
remote: Compressing objects: 100% (314/314), done.
remote: Total 426 (delta 129), reused 256 (delta 89), pack-reused 0 (from 0)
Receiving objects: 100% (426/426), 9.62 MiB | 10.86 MiB/s, done.
Resolving deltas: 100% (129/129), done.
Note: switching to 'fb365f15a30b3154474b231cd929128176806123'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.
```

4. Deploy Online Boutique to the cluster.

- kubectl apply -f ./release/kubernetes-manifests.yaml

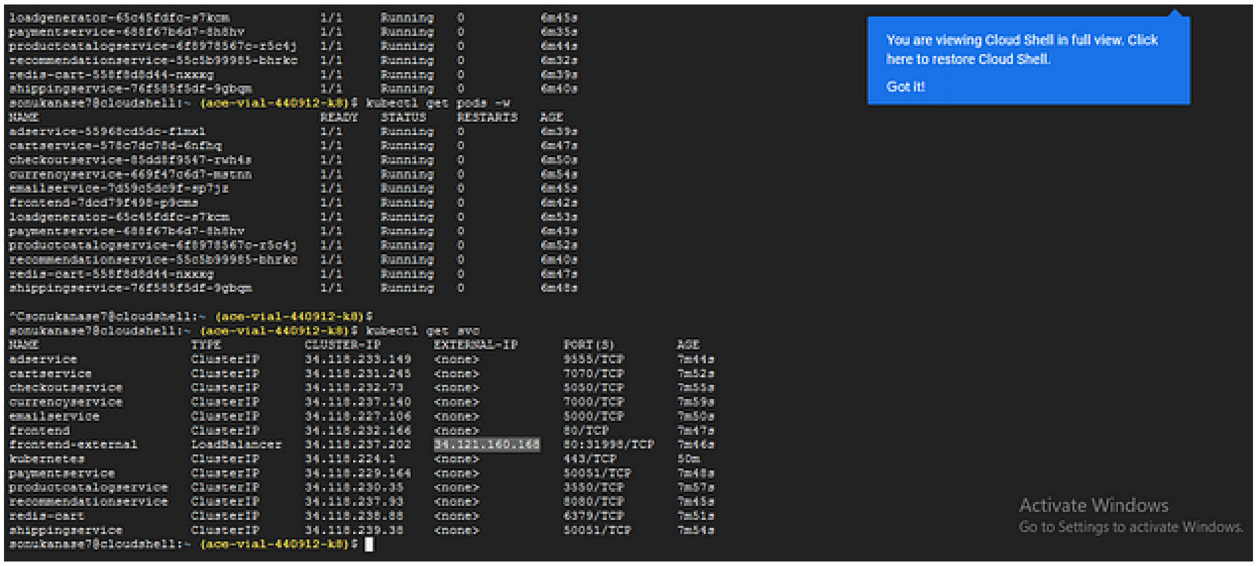


```
sonukanase7@cloudshell:~/microservices-demo (ace-vial-440912-k8)$
sonukanase7@cloudshell:~/microservices-demo (ace-vial-440912-k8)$ kubectl apply -f ./release/kubernetes-manifests
deployment.apps/currencyservice created
service/currencyservice created
serviceaccount/currencyservice created
deployment.apps/loadgenerator created
serviceaccount/loadgenerator created
deployment.apps/productcatalogservice created
service/productcatalogservice created
serviceaccount/productcatalogservice created
deployment.apps/checkoutservice created
service/checkoutservice created
serviceaccount/checkoutservice created
deployment.apps/shippingservice created
service/shippingservice created
serviceaccount/shippingservice created
deployment.apps/cartservice created
service/cartservice created
serviceaccount/cartservice created
deployment.apps/redis-cart created
service/redis-cart created
deployment.apps/emailservice created
service/emailservice created
serviceaccount/emailservice created
deployment.apps/paymentservice created
service/paymentservice created
serviceaccount/paymentservice created
deployment.apps/frontend created
service/frontend created
service/frontend-external created
serviceaccount/frontend created
deployment.apps/recommendationservice created
service/recommendationservice created
serviceaccount/recommendationservice created
deployment.apps/adservice created
service/adservice created
serviceaccount/adservice created
sonukanase7@cloudshell:~/microservices-demo (ace-vial-440912-k8)$
```

You are viewing Cloud Shell in full view. Click here to restore Cloud Shell.

5. Wait for the pods to be ready.

- kubectl get pods



```
loadgenerator-65c45fdfc-s7kcm             1/1     Running   0          6m45s
paymentservice-688f67b6d7-8h8hv           1/1     Running   0          6m35s
productcatalogservice-6f8978567c-r5c4j    1/1     Running   0          6m44s
recommendationservice-55c5b99985-bhrkc    1/1     Running   0          6m32s
redis-cart-558f8d8d44-nxxxg               1/1     Running   0          6m39s
shippingservice-76f585f5df-9gbgm          1/1     Running   0          6m40s
sonukanase7@cloudshell:~ (ace-vial-440912-k8)$ kubectl get pods -w
NAME                                      READY   STATUS    RESTARTS   AGE
adservice-55968cd5dc-f1mxl                1/1     Running   0          6m39s
cartservice-578c7dc78d-6nfbq              1/1     Running   0          6m47s
checkoutservice-85dd8f9547-rwh4s          1/1     Running   0          6m50s
currencyservice-669f47c6d7-mstnn          1/1     Running   0          6m54s
emailservice-7d59c5dc9f-sp7jz             1/1     Running   0          6m45s
frontend-7dcd79f498-p9cms                 1/1     Running   0          6m42s
loadgenerator-65c45fdfc-s7kcm             1/1     Running   0          6m53s
paymentservice-688f67b6d7-8h8hv           1/1     Running   0          6m43s
productcatalogservice-6f8978567c-r5c4j    1/1     Running   0          6m52s
recommendationservice-55c5b99985-bhrkc    1/1     Running   0          6m40s
redis-cart-558f8d8d44-nxxxg               1/1     Running   0          6m47s
shippingservice-76f585f5df-9gbgm          1/1     Running   0          6m48s

^Csonukanase7@cloudshell:~ (ace-vial-440912-k8)$
sonukanase7@cloudshell:~ (ace-vial-440912-k8)$ kubectl get svc
NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)        AGE
adservice               ClusterIP      34.118.233.149   <none>           9555/TCP       7m44s
cartservice             ClusterIP      34.118.231.245   <none>           7070/TCP       7m52s
checkoutservice         ClusterIP      34.118.232.73    <none>           5050/TCP       7m55s
currencyservice         ClusterIP      34.118.237.140   <none>           7000/TCP       7m59s
emailservice            ClusterIP      34.118.227.106   <none>           5000/TCP       7m50s
frontend                ClusterIP      34.118.232.166   <none>           80/TCP         7m47s
frontend-external       LoadBalancer   34.118.237.202   34.121.160.148   80:31998/TCP   7m46s
kubernetes              ClusterIP      34.118.224.1     <none>           443/TCP        50m
paymentservice          ClusterIP      34.118.229.164   <none>           50051/TCP      7m48s
productcatalogservice   ClusterIP      34.118.230.35    <none>           3550/TCP       7m57s
recommendationservice   ClusterIP      34.118.237.93    <none>           8080/TCP       7m45s
redis-cart              ClusterIP      34.118.238.88    <none>           6379/TCP       7m51s
shippingservice         ClusterIP      34.118.239.38    <none>           50051/TCP      7m54s
sonukanase7@cloudshell:~ (ace-vial-440912-k8)$
```

You are viewing Cloud Shell in full view. Click here to restore Cloud Shell.
Got it!

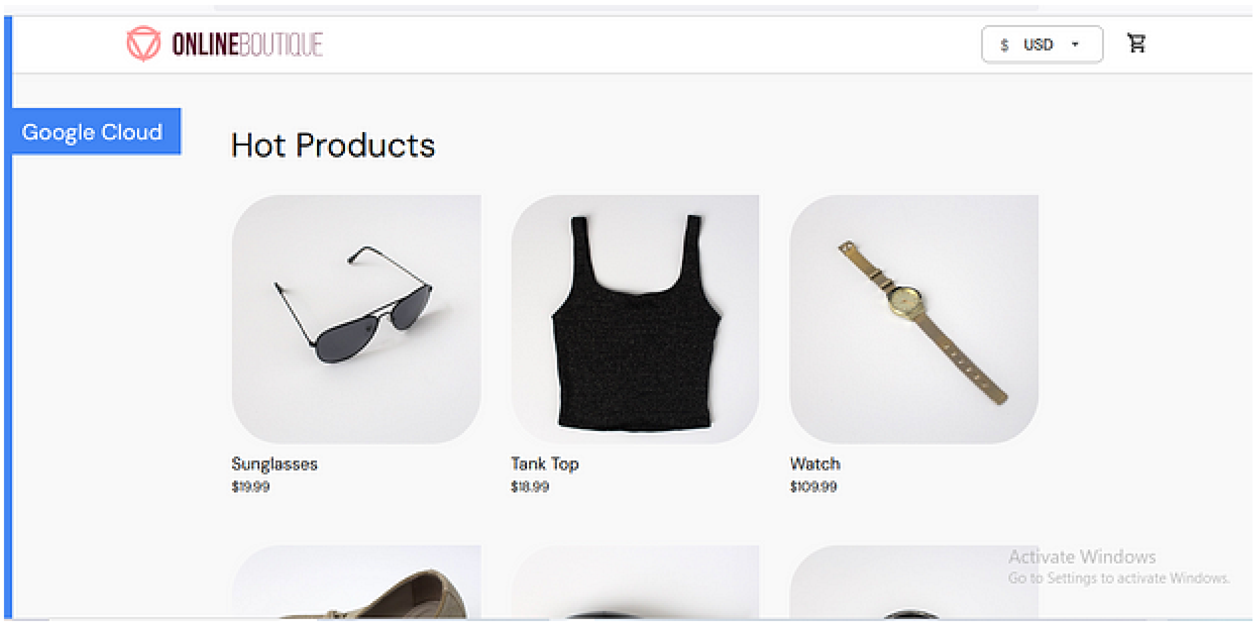
6. After a few minutes, you should see the Pods in a `Running` state:

7. Access the web frontend in a browser using the frontend's external IP.

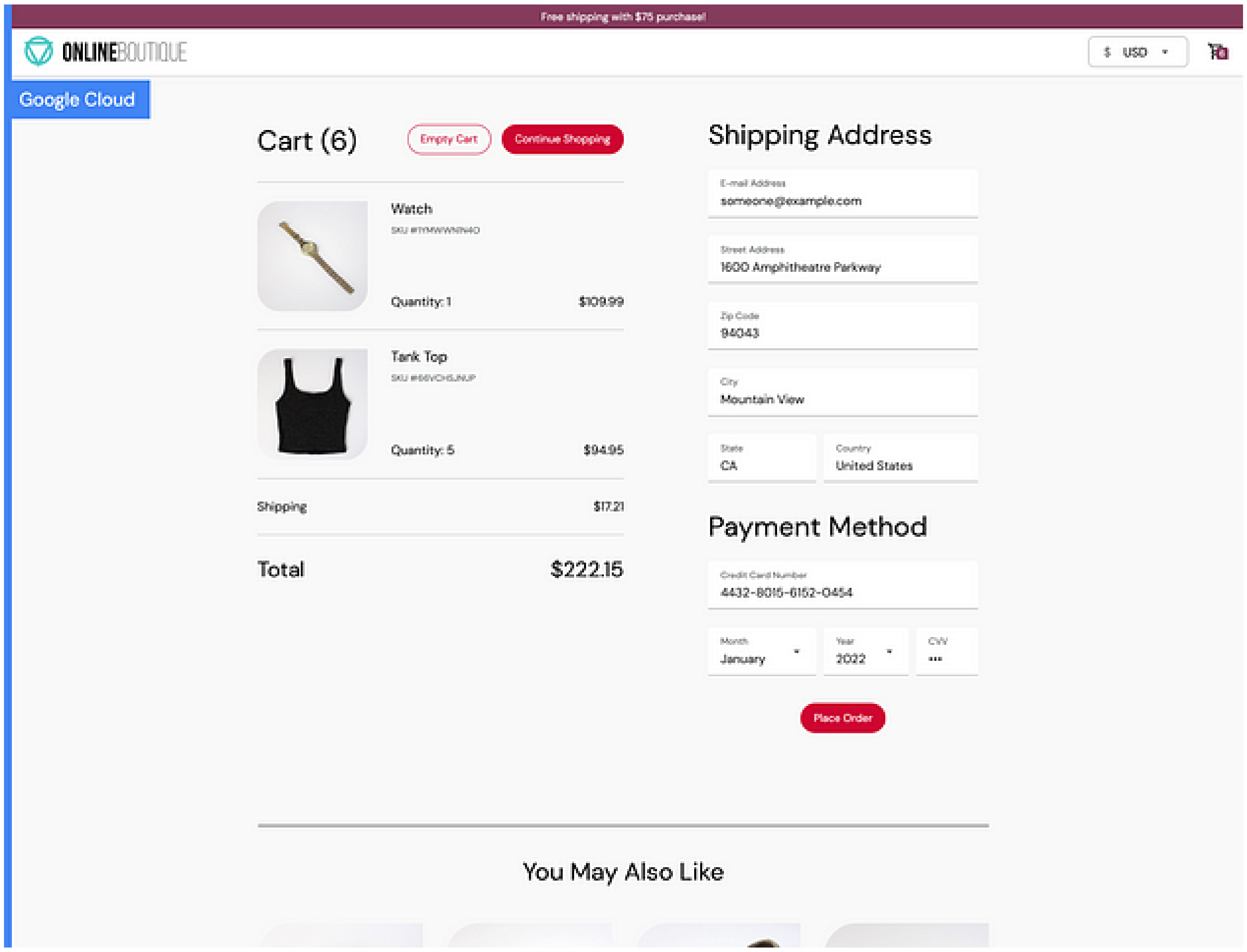
- kubectl get service frontend-external | awk '{print $4}'

8. Visit `http://EXTERNAL_IP` in a web browser to access your instance of Online Boutique.

Home Page Checkout Screen



9. Congrats! You’ve deployed the default Online Boutique. To deploy a different variation of Online Boutique



Cart (6)

Watch
SKU #1YMWWN1N4O
Quantity: 1 $109.99

Tank Top
SKU #66VCHSJNUP
Quantity: 5 $94.95

Shipping $17.21

Total $222.15

Shipping Address

E-mail Address
someone@example.com

Street Address
1600 Amphitheatre Parkway

Zip Code
94043

City
Mountain View

State
CA

Country
United States

Payment Method

Credit Card Number
4432-8015-6152-0454

Month
January

Year
2022

CVV
•••

You May Also Like

10. Once you are done with it, delete the GKE cluster.

Deleting the cluster may take a few minutes.