Task1: Create a GKE cluster

Task2: Get authentication credentials for the cluster

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ gcloud container clusters create "my-vpc-cluster" --region "asia-east1" --machine-type "g1-small" --num-nodes "1" --enable-ip-alias
WARNING: Starting in January 2021, clusters will use the Regular release channel by default when `--cluster-version`, `--release-channel`, `--no-enable-autoupgrade`, and `--no-enable-autorepair` flags are not specified.
WARNING: Starting with version 1.18, clusters will have shielded GKE nodes by default.
WARNING: The Pod address range limits the maximum size of the cluster. Please refer to https://cloud.google.com/kubernetes-engine/docs/how-to/flexible-pod-cidr to learn how to optimize IP address allocation.
WARNING: Starting with version 1.19, newly created clusters and node-pools will have COS_CONTAINERD as the default node image when no image type is specified.
Creating cluster my-vpc-cluster in asia-east1... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/firm-solution-315106/zones/asia-east1/clusters/my-vpc-cluster].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/asia-east1/my-vpc-cluster?project=firm-solution-315106
kubeconfig entry generated for my-vpc-cluster.
NAME           LOCATION     MASTER_VERSION   MASTER_IP      MACHINE_TYPE  NODE_VERSION      NUM_NODES  STATUS
my-vpc-cluster  asia-east1  1.19.9-gke.1400  34.80.23.219  g1-small      1.19.9-gke.1400  3          RUNNING
```

Task3: Quick Deploy an application to the cluster

1. To **create a new Deployment** hello-server from the hello-app container image, run the following kubectl create command:

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ kubectl create deployment hello-server --image=gcr.io/google-samples/hello-app:1.0
deployment.apps/hello-server created
```

2. To **create a Kubernetes Service**, which is a Kubernetes resource that lets you expose your application to external traffic, run the following kubectl expose command:

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ kubectl expose deployment hello-server --type=LoadBalancer --port 8080
service/hello-server exposed
```

3. To **inspect** the hello-server Service, run kubectl get:

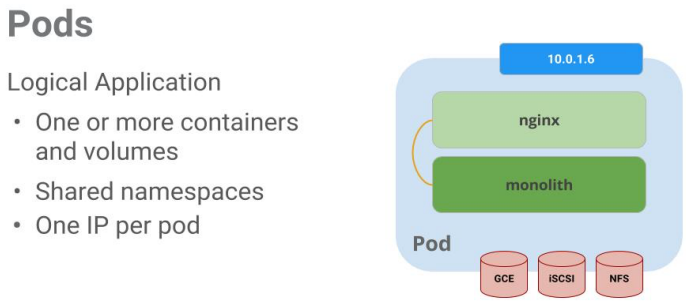
```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ kubectl get service
NAME           TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)          AGE
hello-server   LoadBalancer   10.4.4.157    <pending>     8080:30922/TCP   23s
kubernetes     ClusterIP      10.4.0.1      <none>        443/TCP          11m
```

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ kubectl get service
NAME           TYPE           CLUSTER-IP    EXTERNAL-IP      PORT(S)          AGE
hello-server   LoadBalancer   10.4.4.157    35.201.151.195   8080:30922/TCP   70s
kubernetes     ClusterIP      10.4.0.1      <none>           443/TCP          12m
```

4. To view the application from your web browser, open a new tab and enter the following address, replacing [EXTERNAL IP] with the EXTERNAL-IP for hello-server.


```
Hello, world!
Version: 1.0.0
Hostname: hello-server-76d47868b4-h4tcl
```

Task4: Creating Pods and Interacting with Pods



Task4-1: Creating Pods

Pods can be created using pod configuration files. Take a moment to explore the monolith pod configuration file.

1. Create the monolith pod using kubectl:

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ kubectl create -f pod.yaml
pod/monolith created
```

2. Examine your pods. Use the kubectl get pods command to list all pods running in the default namespace:

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
hello-server-76d47868b4-h4tcl   1/1     Running   0          25m
monolith                      1/1     Running   0          15s
```

   Once the pod is running, use kubectl describe command to get more information about the monolith pod:

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ kubectl describe pods monolith
Name:         monolith
Namespace:    default
Priority:     0
Node:         gke-my-vpc-cluster-default-pool-8ba9f9d8-pcbt/10.140.0.2
Start Time:   Fri, 28 May 2021 07:45:55 +0000
Labels:       app=monolith
Annotations:  <none>
Status:       Running
IP:           10.0.0.5
IPs:
  IP:  10.0.0.5
Containers:
  monolith:
    Container ID:  containerd://e8ab2895bb724ad481870fb898d64d073069f4a7003a52bf8d962c875db29d3d
    Image:         kelseyhightower/monolith:1.0.0
    Image ID:      sha256:980e09dd5c76f726e7369ac2c3aa9528fe3a8c92382b78e97aa54a4a32d3b187
    Ports:         80/TCP, 81/TCP
    Host Ports:    0/TCP, 0/TCP
    Args:
      -http=0.0.0.0:80
      -health=0.0.0.0:81
      -secret=secret
    State:          Running
      Started:      Fri, 28 May 2021 07:46:05 +0000
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     200m
      memory:  10Mi
    Requests:
      cpu:        200m
      memory:     10Mi
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-qm6z2 (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
```

```
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-qm6z2:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-qm6z2
    Optional:    false
QoS Class:       Guaranteed
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  3m52s  default-scheduler  Successfully assigned default/monolith to gke-my-vpc-cluster-default-pool-8ba9f9d8-pcbt
  Normal  Pulling    3m50s  kubelet            Pulling image "kelseyhightower/monolith:1.0.0"
  Normal  Pulled     3m43s  kubelet            Successfully pulled image "kelseyhightower/monolith:1.0.0" in 7.211818465s
  Normal  Created    3m42s  kubelet            Created container monolith
  Normal  Started    3m42s  kubelet            Started container monolith
```

Task4-2: Interacting with Pods

1. In the **2nd terminal**, run this command to set up port-forwarding:

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ kubectl port-forward monolith 10080:80
Forwarding from 127.0.0.1:10080 -> 80
```

2. Now in the **1st terminal** start talking to your pod using curl:

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ curl http://127.0.0.1:10080
{"message":"Hello"}
```

3. Try logging in to get an auth token back from the monolith:

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ curl -u user http://127.0.0.1:10080/login
Enter host password for user 'user':
{"token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6InVzZXJAZXhhbXBsZS5jb20iLCJleHAiOjE2MjI0NDc2MzYsImlhdCI6MTYyMjE4ODQzNiwiaXNzIjoiYXV0aC5zZXJ2aWNlIiwic3ViIjoidXNlciJ9.uDzAk
0WdHJtwaVOqEnWy-HCr55ktiUxMH8vdKFW11PA"}
```

At the login prompt, use the super-secret password "password" to login.

4. Logging in caused a JWT token to print out. Since Cloud Shell does not handle copying long strings well, create an environment variable for the token.

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ TOKEN=$(curl http://127.0.0.1:10080/login -u user|jq -r '.token')
Enter host password for user 'user':
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   222  100   222    0     0    604      0 --:--:-- --:--:-- --:--:--   604
```

5. Use this command to copy and then use the token to hit the secure endpoint with curl:

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ curl -H "Authorization: Bearer $TOKEN" http://127.0.0.1:10080/secure
{"message":"Hello"}
```

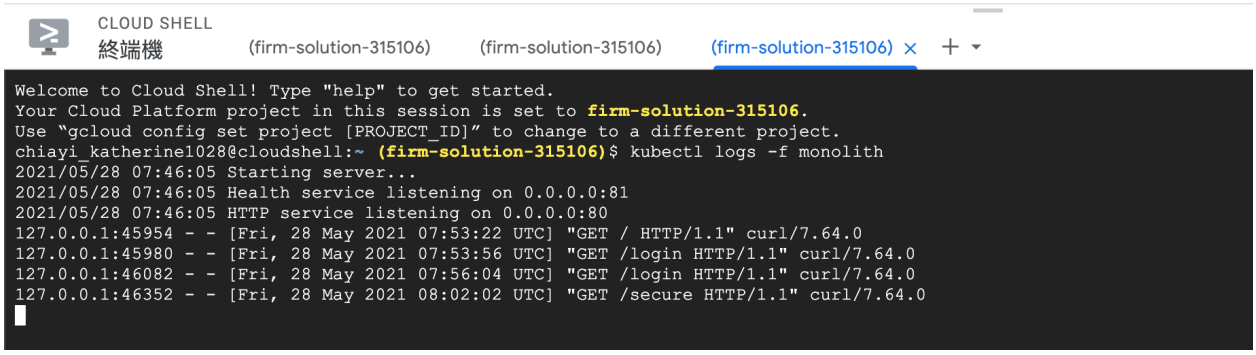
   At this point you should get a response back from our application, letting us know everything is right in the world again.

6. Use the kubectl logs command to view the logs for the monolith Pod.

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ kubectl logs monolith
2021/05/28 07:46:05 Starting server...
2021/05/28 07:46:05 Health service listening on 0.0.0.0:81
2021/05/28 07:46:05 HTTP service listening on 0.0.0.0:80
127.0.0.1:45954 - - [Fri, 28 May 2021 07:53:22 UTC] "GET / HTTP/1.1" curl/7.64.0
127.0.0.1:45980 - - [Fri, 28 May 2021 07:53:56 UTC] "GET /login HTTP/1.1" curl/7.64.0
127.0.0.1:46082 - - [Fri, 28 May 2021 07:56:04 UTC] "GET /login HTTP/1.1" curl/7.64.0
127.0.0.1:46352 - - [Fri, 28 May 2021 08:02:02 UTC] "GET /secure HTTP/1.1" curl/7.64.0
```

7. **Open a 3rd terminal** and use the -f flag to get a stream of the logs happening in real-time

8. Now if you use curl in the **1st terminal** to interact with the monolith, you can see the logs updating (in the **3rd terminal**):



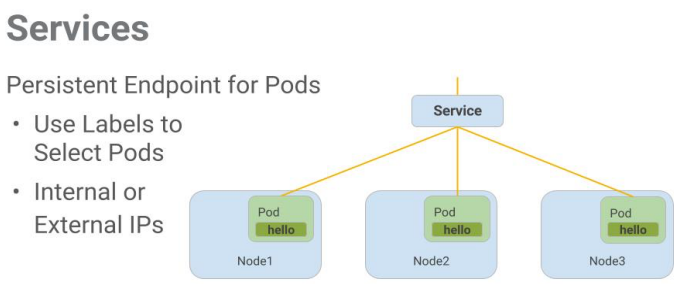
9. Use the kubectl exec command to run an interactive shell inside the Monolith Pod. This can come in handy when you want to troubleshoot from within a container:

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ kubectl exec monolith --stdin --tty -c monolith /bin/sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ #
```

10. Be sure to log out when you're done with this interactive shell.

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ kubectl exec monolith --stdin --tty -c monolith /bin/sh
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ #
```

Task5: Create a Service



Task5-1: Create a Service

Before you can create our services, first create a secure pod that can handle https traffic.

1. Explore the monolith service configuration file:

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ kubectl create -f service.yaml
service/monolith created
```

2. Examine your service. Use the kubectl get service command to list all pods running in the default namespace:

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ kubectl get service
NAME           TYPE           CLUSTER-IP    EXTERNAL-IP      PORT(S)          AGE
hello-server   LoadBalancer   10.4.4.157    35.201.151.195   8080:30922/TCP   43m
kubernetes     ClusterIP      10.4.0.1      <none>           443/TCP          54m
monolith       LoadBalancer   10.4.14.58    35.201.149.248   80:30266/TCP     42s
```

3. When the service is ready, you will see the monolith service's `EXTERNAL-IP` is not none. Copy the EXTERNAL-IP and paste it in your browser and see what happens.


```
{"message":"Hello"}
```

Final Task: Deleting the cluster

1. To **delete** the cluster, run the following command:
2. When prompted, type **Y** to confirm.

Istio on GKE

Task1: Activate Cloud Shell

Task2: Create a GKE cluster with Istio

1. If you haven't already, open a new Cloud Shell session and run the following to set environment variables for the zone and cluster name:

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ export CLUSTER_ZONE=us-central1-b
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ export CLUSTER_VERSION=latest
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ export CLUSTER_NAME=central
```

2. Now run the following command in Cloud Shell to create your GKE cluster using the *Istio on GKE* add-on:

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ gcloud beta container clusters create $CLUSTER_NAME --zone $CLUSTER_ZONE --num-nodes 4 --machine-type "n1-standard-2" --ima
ge-type "COS" --cluster-version=$CLUSTER_VERSION --enable-ip-alias --addons=Istio --istio-config=auth=MTLS_STRICT
WARNING: Starting with version 1.18, clusters will have shielded GKE nodes by default.
WARNING: The Pod address range limits the maximum size of the cluster. Please refer to https://cloud.google.com/kubernetes-engine/docs/how-to/flexible-pod-cidr to learn how to optim
ize IP address allocation.
Creating cluster central in us-central1-b... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1beta1/projects/firm-solution-315106/zones/us-central1-b/clusters/central].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-central1-b/central?project=firm-solution-315106
kubeconfig entry generated for central.
NAME     LOCATION       MASTER_VERSION    MASTER_IP      MACHINE_TYPE   NODE_VERSION      NUM_NODES  STATUS
central  us-central1-b  1.19.10-gke.1600  34.71.186.122  n1-standard-2  1.19.10-gke.1600  4          RUNNING
```

Once your cluster has been created, configure kubectl command line access by running the following:

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ export GCLOUD_PROJECT=$(gcloud config get-value project)
Your active configuration is: [cloudshell-8803]
```

3. Grant admin permissions in the cluster to the current gcloud user:

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ kubectl create clusterrolebinding cluster-admin-binding --clusterrole=cluster-admin --user=$(gcloud config get-value core/a
ccount)
Your active configuration is: [cloudshell-8803]
clusterrolebinding.rbac.authorization.k8s.io/cluster-admin-binding created
```

Note: although you can run the demo app without granting cluster admin permissions, the permissions are required if you want to access telemetry data and other Istio features.

4. Verify installation

a. Check that your cluster is up and running.

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ gcloud container clusters list
NAME            LOCATION       MASTER_VERSION    MASTER_IP       MACHINE_TYPE   NODE_VERSION      NUM_NODES  STATUS
my-vpc-cluster  asia-east1     1.19.9-gke.1400   34.80.23.219    g1-small       1.19.9-gke.1400   3          RUNNING
central         us-central1-b  1.19.10-gke.1600  34.71.186.122   n1-standard-2  1.19.10-gke.1600  4          RUNNING
```

b. Ensure the following Kubernetes services are deployed: istio-citadel, istio-galley, istio-pilot, istio-ingressgateway, istio-policy, istio-sidecar-injector, and istio-telemetry. You'll also see other deployed services:

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ kubectl get service -n istio-system
NAME                     TYPE           CLUSTER-IP      EXTERNAL-IP    PORT(S)                                                                                                                                AGE
istio-citadel            ClusterIP      10.108.8.178    <none>         8060/TCP,15014/TCP                                                                                                                     6m54s
istio-galley             ClusterIP      10.108.4.188    <none>         443/TCP,15014/TCP,9901/TCP                                                                                                             6m54s
istio-ingressgateway     LoadBalancer   10.108.4.156    35.193.19.69   15020:31478/TCP,80:32500/TCP,443:32666/TCP,31400:32392/TCP,15029:31904/TCP,15030:31374/TCP,15031:31211/TCP,15032:32354/TCP,15443:30337/TCP   6m54s
istio-pilot              ClusterIP      10.108.2.233    <none>         15010/TCP,15011/TCP,8080/TCP,15014/TCP                                                                                                 6m54s
istio-policy             ClusterIP      10.108.7.52     <none>         9091/TCP,15004/TCP,15014/TCP                                                                                                           6m54s
istio-sidecar-injector   ClusterIP      10.108.11.226   <none>         443/TCP,15014/TCP                                                                                                                      6m54s
istio-telemetry          ClusterIP      10.108.14.155   <none>         9091/TCP,15004/TCP,15014/TCP,42422/TCP                                                                                                 6m53s
istiod-istio-1611        ClusterIP      10.108.10.245   <none>         15010/TCP,15012/TCP,443/TCP,15014/TCP,853/TCP                                                                                          5m25s
prometheus               ClusterIP      10.108.12.169   <none>         9090/TCP                                                                                                                               5m9s
promsd                   ClusterIP      10.108.14.160   <none>         9090/TCP                                                                                                                               6m53s
```

c. Ensure the corresponding Kubernetes pods are deployed and all containers are up and running: istio-pilot, istio-galley, istio-policy, istio-telemetry, istio-ingressgateway, istio-sidecar-injector, and istio-citadel.

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ kubectl get pods -n istio-system
NAME                                      READY   STATUS      RESTARTS   AGE
istio-citadel-f5586dffb-rd89w             1/1     Running     0          7m25s
istio-galley-7975f77bbf-pjjqv             1/1     Running     0          7m24s
istio-ingressgateway-b9477dcdb-7s68f      1/1     Running     0          7m24s
istio-pilot-59d4884d67-5z8l5              2/2     Running     0          7m23s
istio-policy-6885cb4644-tl2ct             2/2     Running     2          7m23s
istio-security-post-install-1.4.10-gke.8-lrzgc   0/1     Completed   0          6m51s
istio-sidecar-injector-649d664b99-zrkjx   1/1     Running     0          7m23s
istio-telemetry-59b6bf55c7-rwxg6          2/2     Running     2          7m22s
istiod-istio-1611-6895859f65-9ffqt        1/1     Running     0          5m57s
prometheus-6655946b9f-tp6m9               2/2     Running     0          5m42s
promsd-574ccb9745-7ffd9                   2/2     Running     1          7m22s
```

Task3: Deploy an Istio-enabled multi-service application

In this task, you enable the istioctl tool, set up the **Bookinfo** sample microservices application, and explore the app.

1. Use Cloud Shell to download and extract the Istio release, with the istioctl tool, and sample code:

```
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ export LAB_DIR=$HOME/bookinfo-lab
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ export ISTIO_VERSION=1.4.6
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ mkdir $LAB_DIR
chiayi_katherine1028@cloudshell:~/gcp-practice/homework_samplecode (firm-solution-315106)$ cd $LAB_DIR
chiayi_katherine1028@cloudshell:~/bookinfo-lab (firm-solution-315106)$
```

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab (firm-solution-315106)$ curl -L https://git.io/getLatestIstio | ISTIO_VERSION=$ISTIO_VERSION sh -
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  4573  100  4573    0     0   3679      0  0:00:01  0:00:01 --:--:--  3679

Downloading istio-1.4.6 from https://github.com/istio/istio/releases/download/1.4.6/istio-1.4.6-linux.tar.gz ...
Istio 1.4.6 Download Complete!

Istio has been successfully downloaded into the istio-1.4.6 folder on your system.

Next Steps:
See https://istio.io/latest/docs/setup/install/ to add Istio to your Kubernetes cluster.

To configure the istioctl client tool for your workstation,
add the /home/chiayi_katherine1028/bookinfo-lab/istio-1.4.6/bin directory to your environment path variable with:
         export PATH="$PATH:/home/chiayi_katherine1028/bookinfo-lab/istio-1.4.6/bin"

Begin the Istio pre-installation check by running:
         istioctl x precheck

Need more information? Visit https://istio.io/latest/docs/setup/install/
```

2. Make the Istio tools visible to your environment, by adding bin to your **PATH**:

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab (firm-solution-315106)$ cd ./istio-*
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ export PATH=$PWD/bin:$PATH
```

Verify istioctl works:

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ istioctl version
client version: 1.4.6
citadel version: 1.4.10-gke.8
galley version: 1.4.10-gke.8
ingressgateway version: 1.4.10-gke.8
pilot version: 1.4.10-gke.8
policy version: 1.4.10-gke.8
sidecar-injector version: 1.4.10-gke.8
telemetry version: 1.4.10-gke.8
istiod version:
data plane version: 1.4.10-gke.8 (1 proxies)
```

Task4: Deploy Bookinfo

Bookinfo Overview

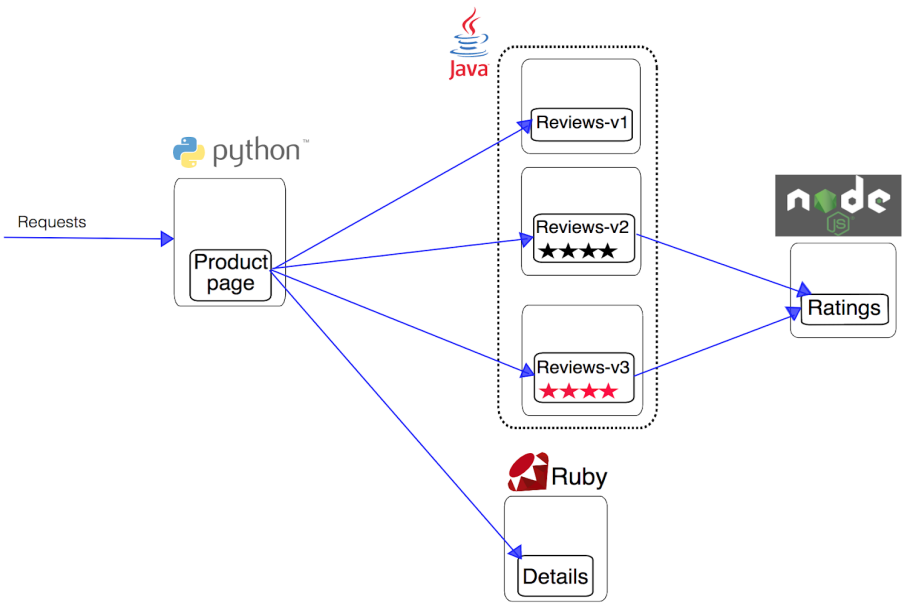
The microservices are:

- **productpage:** calls the details and reviews microservices to populate the page.
- **details**: contains book information.
- **reviews:** contains book reviews. It also calls the ratings microservice.
- **ratings**: contains book ranking information that accompanies a book review.

There are 3 versions of the **reviews** microservice:

- Reviews **v1** doesn't call the ratings service.
- Reviews **v2** calls the ratings service, and displays each rating as 1 - 5 black stars.
- Reviews **v3** calls the ratings service, and displays each rating as 1 - 5 red stars.

The end-to-end architecture of the application looks like this:



BookInfo Application without Istio

1. Look at the .yaml which describes the bookInfo application:

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ cat samples/bookinfo/platform/kube/bookinfo.yaml
# Copyright 2017 Istio Authors
#
#   Licensed under the Apache License, Version 2.0 (the "License");
#   you may not use this file except in compliance with the License.
#   You may obtain a copy of the License at
#
#       http://www.apache.org/licenses/LICENSE-2.0
#
#   Unless required by applicable law or agreed to in writing, software
#   distributed under the License is distributed on an "AS IS" BASIS,
#   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#   See the License for the specific language governing permissions and
#   limitations under the License.

##################################################################################################
# This file defines the services, service accounts, and deployments for the Bookinfo sample.
#
# To apply all 4 Bookinfo services, their corresponding service accounts, and deployments:
#
#   kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml
#
# Alternatively, you can deploy any resource separately:
#
#   kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml -l service=reviews # reviews Service
#   kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml -l account=reviews # reviews ServiceAccount
#   kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml -l app=reviews,version=v3 # reviews-v3 Deployment
##################################################################################################
```

```
##################################################################################################
# Details service
##################################################################################################
apiVersion: v1
kind: Service
metadata:
  name: details
  labels:
    app: details
    service: details
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: details
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfo-details
  labels:
    account: details
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: details-v1
  labels:
    app: details
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: details
      version: v1
```

```
  template:
    metadata:
      labels:
        app: details
        version: v1
    spec:
      serviceAccountName: bookinfo-details
      containers:
      - name: details
        image: docker.io/istio/examples-bookinfo-details-v1:1.15.0
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9080
---
##################################################################################################
# Ratings service
##################################################################################################
apiVersion: v1
kind: Service
metadata:
  name: ratings
  labels:
    app: ratings
    service: ratings
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: ratings
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfo-ratings
  labels:
    account: ratings
---
```

```
##################################################################################################
# Productpage services
##################################################################################################
apiVersion: v1
kind: Service
metadata:
  name: productpage
  labels:
    app: productpage
    service: productpage
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: productpage
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfo-productpage
  labels:
    account: productpage
---
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: productpage
    version: v1
  name: productpage-v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: productpage
      version: v1
```

Look for containers to see that each Deployment, has **one** container, for each version of each service in the Bookinfo application.

2. Look at the same .yaml with Istio proxy sidecars *injected* using *istioctl*:

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ istioctl kube-inject -f samples/bookinfo/platform/kube/bookinfo.yaml
```

3. In **Cloud Shell**, use the following command to *inject* the proxy sidecar along with each application Pod that is deployed.

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ kubectl apply -f <(istioctl kube-inject -f samples/bookinfo/platform/kube/bookinfo.yaml)
service/details created
serviceaccount/bookinfo-details created
deployment.apps/details-v1 created
service/ratings created
serviceaccount/bookinfo-ratings created
deployment.apps/ratings-v1 created
service/reviews created
serviceaccount/bookinfo-reviews created
deployment.apps/reviews-v1 created
deployment.apps/reviews-v2 created
deployment.apps/reviews-v3 created
service/productpage created
serviceaccount/bookinfo-productpage created
deployment.apps/productpage-v1 created
```

Task5: Enable external access using an Istio Ingress Gateway

Now that the Bookinfo services are up and running, you need to make the application accessible from outside of your Kubernetes cluster, e.g. from a browser. An **Istio Gateway** is used for this purpose.

1. Configure the **ingress gateway** for the application, which exposes an *external IP* you will use later:

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ kubectl apply -f samples/bookinfo/networking/bookinfo-gateway.yaml
gateway.networking.istio.io/bookinfo-gateway created
virtualservice.networking.istio.io/bookinfo created
```

Task6: Verify the Bookinfo deployments

1. Confirm that the application has been deployed correctly, review services, pods, and the ingress gateway:

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ kubectl get services
NAME          TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)    AGE
details       ClusterIP   10.108.11.224   <none>        9080/TCP   2m27s
kubernetes    ClusterIP   10.108.0.1      <none>        443/TCP    4h29m
productpage   ClusterIP   10.108.10.227   <none>        9080/TCP   2m23s
ratings       ClusterIP   10.108.4.43     <none>        9080/TCP   2m26s
reviews       ClusterIP   10.108.5.229    <none>        9080/TCP   2m25s
```

2. Review running application pods:

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
details-v1-569cb55f85-sdzjz       2/2     Running   0          2m48s
productpage-v1-6d8b866dcc-q64cl   2/2     Running   0          2m45s
ratings-v1-5b564db7cc-kz9zj       2/2     Running   0          2m48s
reviews-v1-589f9db7dc-w6bjq       2/2     Running   0          2m47s
reviews-v2-5bf7b8f6b-64zpk        2/2     Running   0          2m47s
reviews-v3-dc65f5d77-ztnmg        2/2     Running   0          2m46s
```

3. Confirm that the Bookinfo application is running by sending a curl request to it from some pod, within the cluster, for example from ratings:

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ kubectl exec -it $(kubectl get pod -l app=ratings -o jsonpath='{.items[0].metadata.name}') -c rati
ngs -- curl productpage:9080/productpage | grep -o "<title>.*</title>"
<title>Simple Bookstore App</title>
```

4. Confirm the ingress gateway has been created:

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ kubectl get gateway
NAME               AGE
bookinfo-gateway   2m43s
```

5. Get the **external IP** address of the **ingress gateway**:

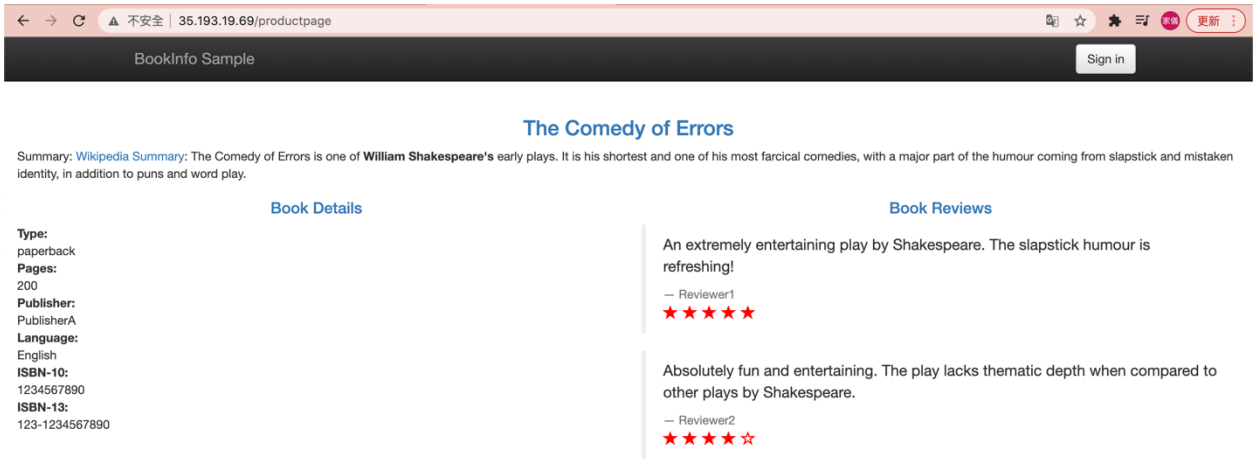
```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ kubectl get svc istio-ingressgateway -n istio-system
NAME                   TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)
                                                                                AGE
istio-ingressgateway   LoadBalancer   10.108.4.156   35.193.19.69    15020:31478/TCP,80:32500/TCP,443:32666/TCP,31400:32392/TCP,15029:31904/TCP,15030:31374/TCP,15031:31211/TCP,15032:
32354/TCP,15443:30337/TCP   4h31m
```

6. Check that the Bookinfo app is running by sending a curl request to it from *outside* the cluster (be sure to update [EXTERNAL-IP] with the output from the previous command):

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ export GATEWAY_URL=35.193.19.69
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ curl -I http://${GATEWAY_URL}/productpage
HTTP/1.1 200 OK
content-type: text/html; charset=utf-8
content-length: 5179
server: istio-envoy
date: Fri, 28 May 2021 12:47:27 GMT
x-envoy-upstream-service-time: 1045
```

Task7: Use the Bookinfo application

1. Point your browser to http://[$GATEWAY_URL]/productpage to see the BookInfo web page. Don't forget to replace [$GATEWAY_URL] with your working external IP address.



2. Refresh the page several times.

   Notice how you see three different versions of reviews, since we have not yet used Istio to control the version routing.

   There are three different book review services being called in a round-robin style:

   - No stars
   - Black stars
   - Red stars

   Switching among the three is normal Kubernetes routing/balancing behavior.

Final Task: Deleting the cluster

1. To **delete** the cluster, run the following command:
2. When prompted, type **Y** to confirm.

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ gcloud container clusters delete "central"
The following clusters will be deleted.
 - [central] in [us-central1-b]

Do you want to continue (Y/n)?  y

Deleting cluster central...done.
Deleted [https://container.googleapis.com/v1/projects/firm-solution-315106/zones/us-central1-b/clusters/central].
```

Installing Anthos Service Mesh on GKE

Before Task:

1. In Cloud Shell, verify your default **account** is configured.
2. Update **project** configuration if needed.

3. Enable APIs

   In Cloud Shell, enable the APIs required to use GKE and Anthos Service Mesh.

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ gcloud services enable \
>     container.googleapis.com \
>     compute.googleapis.com \
>     monitoring.googleapis.com \
>     logging.googleapis.com \
>     cloudtrace.googleapis.com \
>     meshca.googleapis.com \
>     meshtelemetry.googleapis.com \
>     meshconfig.googleapis.com \
>     iamcredentials.googleapis.com \
>     anthos.googleapis.com \
>     gkeconnect.googleapis.com \
>     gkehub.googleapis.com \
>     cloudresourcemanager.googleapis.com

Operation "operations/acf.p2-377767083518-38fbb2f2-a1ab-46d1-bc8c-b68ecb0eb1a1" finished successfully.
```

4. Configure environment variables

   Configure environment variables that will be used in the setup and installation commands.

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ export PROJECT_ID=$(gcloud config get-value project)
Your active configuration is: [cloudshell-6967]
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ export PROJECT_NUMBER=$(gcloud projects describe ${PROJECT_ID} \
>     --format="value(projectNumber)")
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ export CLUSTER_NAME=central
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ export CLUSTER_ZONE=us-central1-b
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ export WORKLOAD_POOL=${PROJECT_ID}.svc.id.goog
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ export MESH_ID="proj-${PROJECT_NUMBER}"
```

5. Verify sufficient permissions

   In Cloud Shell, verify that your user account has the **Owner** role assigned.

```
chiayi_katherine1028@cloudshell:~/bookinfo-lab/istio-1.4.6 (firm-solution-315106)$ gcloud projects get-iam-policy $PROJECT_ID \
>     --flatten="bindings[].members" \
>     --filter="bindings.members:user:$(gcloud config get-value core/account 2>/dev/null)"
---
bindings:
  members: user:chiayi.katherine1028@gmail.com
  role: roles/owner
etag: BwXDXoCxEHY=
version: 1
```

   Note: you also possess **viewer** privileges.

To complete setup, you need the permissions associated with these roles:

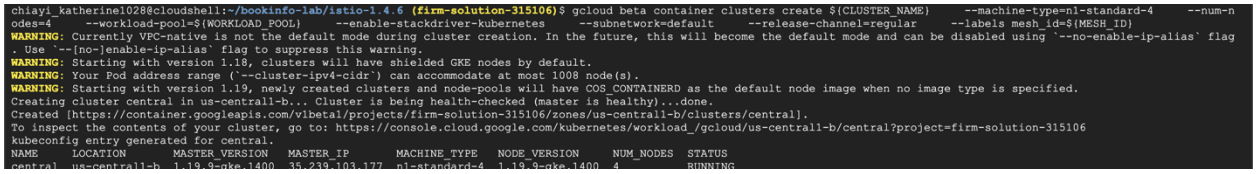
- Project Editor
- Kubernetes Engine Admin
- Project IAM Admin
- GKE Hub Admin
- Service Account Admin
- Service Account key Admin

If you have the Owner role, you have all these permissions and more, so you're ready to proceed.

Task1: Set up your GKE cluster

1. Create the cluster

   Now run the following command in Cloud Shell to create the Kubernetes cluster central:



2. Register the cluster

   a. In Cloud Shell, check that you have the **cluster-admin** RBAC role.

   b. Create a service account for use by GKE Connect.

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ kubectl auth can-i '*' '*' --all-namespaces
yes
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ kubectl create clusterrolebinding [BINDING_NAME] \
>     --clusterrole cluster-admin --user [USER]
clusterrolebinding.rbac.authorization.k8s.io/[BINDING NAME] created
```

c. Assign the **gkehub.connect** role to the newly created service account.

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ gcloud projects add-iam-policy-binding ${PROJECT_ID} \
>  --member="serviceAccount:connect-sa@${PROJECT_ID}.iam.gserviceaccount.com" \
>  --role="roles/gkehub.connect"

Updated IAM policy for project [firm-solution-315106].
bindings:
- members:
  - serviceAccount:service-377767083518@compute-system.iam.gserviceaccount.com
  role: roles/compute.serviceAgent
- members:
  - serviceAccount:service-377767083518@container-engine-robot.iam.gserviceaccount.com
  role: roles/container.serviceAgent
- members:
  - serviceAccount:service-377767083518@containerregistry.iam.gserviceaccount.com
  role: roles/containerregistry.ServiceAgent
- members:
  - serviceAccount:377767083518-compute@developer.gserviceaccount.com
  - serviceAccount:377767083518@cloudservices.gserviceaccount.com
  role: roles/editor
- members:
  - serviceAccount:connect-sa@firm-solution-315106.iam.gserviceaccount.com
  role: roles/gkehub.connect
- members:
  - user:chiayi.katherine1028@gmail.com
  role: roles/owner
- members:
  - serviceAccount:service-377767083518@gcp-sa-pubsub.iam.gserviceaccount.com
  role: roles/pubsub.serviceAgent
etag: BwXDchKSagw=
version: 1
```

d. Create and download a service account key file.

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ gcloud iam service-accounts keys create connect-sa-key.json \
>    --iam-account=connect-sa@${PROJECT_ID}.iam.gserviceaccount.com
created key [6755bdae484006c52952a6b4a2b5286805277bfd] of type [json] as [connect-sa-key.json] for [connect-sa@firm-solution-315106.iam.gserviceaccount.com]
```

e. Use the key file to register the cluster.

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ gcloud container hub memberships register ${CLUSTER_NAME}-connect \
>     --gke-cluster=${CLUSTER_ZONE}/${CLUSTER_NAME}  \
>     --service-account-key-file=./connect-sa-key.json
kubeconfig entry generated for central.

Waiting for membership to be created...done.
Created a new membership [projects/firm-solution-315106/locations/global/memberships/central-connect] for the cluster [central-connect]
Generating the Connect Agent manifest...
Deploying the Connect Agent on cluster [central-connect] in namespace [gke-connect]...
Deployed the Connect Agent on cluster [central-connect] in namespace [gke-connect].
Finished registering the cluster [central-connect] with the Hub.
```

As part of registering your cluster, the Connect Agent is deployed to your cluster.

Task2: Prepare to install Anthos Service Mesh

1. Initialize your project for service mesh installation.

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ curl --request POST \
>    --header "Authorization: Bearer $(gcloud auth print-access-token)" \
>    --data '' \
>    https://meshconfig.googleapis.com/v1alpha1/projects/${PROJECT_ID}:initialize

{}
```

This command creates a service account for Istio components, among other things.

The command responds with empty curly braces: {}.

2. Download the installation file.

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ curl -LO https://storage.googleapis.com/gke-release/asm/istio-1.6.11-asm.1-linux-amd64.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 57.0M  100 57.0M    0     0  21.3M      0  0:00:02  0:00:02 --:--:-- 21.3M
```

3. Download the signature file and verify the signature.

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ curl -LO https://storage.googleapis.com/gke-release/asm/istio-1.6.11-asm.1-linux-amd64.tar.gz.1.sig
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    71  100    71    0     0    125      0 --:--:-- --:--:-- --:--:--   125
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ openssl dgst -verify /dev/stdin -signature istio-1.6.11-asm.1-linux-amd64.tar.gz.1.sig istio-1.6.11-asm.1-linux-amd64.tar.g
z <<'EOF'
> -----BEGIN PUBLIC KEY-----
> MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEWZrGCUaJJr1H8a36sG4UUoXvlXvZ
> wQfk16sxprI2gOJ2vFFggdq3ixF2h4qNBt0kI7ciDhgpwS8t+/960IsIgw==
> -----END PUBLIC KEY-----
> EOF
Verified OK
```

4. Extract the content of the file.
5. Change directories into the installation directory.
6. For convenience, add the tools in the **/bin** directory to your PATH.

```
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ tar xzf istio-1.6.11-asm.1-linux-amd64.tar.gz
chiayi_katherine1028@cloudshell:~ (firm-solution-315106)$ cd istio-1.6.11-asm.1
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ export PATH=$PWD/bin:$PATH
```

You can now use the **istioctl** and **asmctl** tools.

Task3: Preparing Resource Configuration Files

1. Install **kpt**

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ sudo apt-get install google-cloud-sdk-kpt
********************************************************************************
You are running apt-get inside of Cloud Shell. Note that your Cloud Shell
machine is ephemeral and no system-wide change will persist beyond session end.

To suppress this warning, create an empty ~/.cloudshell/no-apt-get-warning file.
The command will automatically proceed in 5 seconds or on any key.

Visit https://cloud.google.com/shell/help for more information.
********************************************************************************
Reading package lists... Done
Building dependency tree
Reading state information... Done
google-cloud-sdk-kpt is already the newest version (340.0.0-0).
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.
```

2. Create a new dierectory for the Anthos Service Mesh package resource configuration files.
3. Download the Anths Service Mesh package.

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ mkdir ${CLUSTER_NAME}
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ cd ${CLUSTER_NAME}
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central (firm-solution-315106)$ kpt pkg get https://github.com/GoogleCloudPlatform/anthos-service-mesh-packages@1.6.8-asm.9 asm
fetching package "/" from "https://github.com/GoogleCloudPlatform/anthos-service-mesh-packages" to "asm"
automatically set 30 field(s) for setter "gcloud.core.project" to value "firm-solution-315106" in package "asm/asm" derived from gcloud config
automatically set 1 field(s) for setter "gcloud.project.projectNumber" to value "377767083518" in package "asm/asm" derived from gcloud config
automatically set 27 field(s) for setter "gcloud.core.project" to value "firm-solution-315106" in package "asm/asm-citadel" derived from gcloud config
automatically set 1 field(s) for setter "gcloud.project.projectNumber" to value "377767083518" in package "asm/asm-citadel" derived from gcloud config
automatically set 30 field(s) for setter "gcloud.core.project" to value "firm-solution-315106" in package "asm/asm-patch" derived from gcloud config
automatically set 1 field(s) for setter "gcloud.project.projectNumber" to value "377767083518" in package "asm/asm-patch" derived from gcloud config
automatically set 27 field(s) for setter "gcloud.core.project" to value "firm-solution-315106" in package "asm/asm-patch-citadel" derived from gcloud config
automatically set 1 field(s) for setter "gcloud.project.projectNumber" to value "377767083518" in package "asm/asm-patch-citadel" derived from gcloud config
```

4. Go to the newly created asm directory

5. Use kpt to customize installation settings in the configuration files:

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central (firm-solution-315106)$ cd asm
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central/asm (firm-solution-315106)$ kpt cfg set asm gcloud.container.cluster ${CLUSTER_NAME}
asm/
set 17 field(s) of setter "gcloud.container.cluster" to value "central"
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central/asm (firm-solution-315106)$ kpt cfg set asm gcloud.project.environProjectNumber ${PROJECT_NUMBER}
asm/
set 2 field(s) of setter "gcloud.project.environProjectNumber" to value "377767083518"
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central/asm (firm-solution-315106)$ kpt cfg set asm gcloud.core.project ${PROJECT_ID}
asm/
set 30 field(s) of setter "gcloud.core.project" to value "firm-solution-315106"
asm/
set 1 field(s) of setter "gcloud.project.projectNumber" to value "377767083518"
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central/asm (firm-solution-315106)$ kpt cfg set asm gcloud.compute.location ${CLUSTER_ZONE}
asm/
set 16 field(s) of setter "gcloud.compute.location" to value "us-central1-b"
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central/asm (firm-solution-315106)$ kpt cfg set asm anthos.servicemesh.profile asm-gcp
asm/
set 1 field(s) of setter "anthos.servicemesh.profile" to value "asm-gcp"
```

Task4: Install Anthos Service Mesh

Auto mutual TLS (auto mTLS) is enabled by default. With auto mTLS, a client sidecar proxy automatically detects if the server has a sidecar. The client sidecar sends mTLS to workloads with sidecars and sends plain text traffic to workloads without sidecars..

1. Use istioctl to install ASM.

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central/asm (firm-solution-315106)$ istioctl install -f asm/cluster/istio-operator.yaml
! global.mtls.enabled is deprecated; use the PeerAuthentication resource instead
✔ Istio core installed
✔ Istiod installed
✔ Ingress gateways installed
✔ Installation complete
```

2. Use **kubectl** to check on completion of the installation.

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central/asm (firm-solution-315106)$ kubectl wait --for=condition=available --timeout=600s deployment \
>     --all -n istio-system
deployment.apps/istio-ingressgateway condition met
deployment.apps/istiod condition met
```

   It can take several minutes for installation to complete.

3. When all conditions are met, check that the control plane Pods are **running**.
4. Validate the installation with the following command.

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central/asm (firm-solution-315106)$ kubectl get pod -n istio-system
NAME                                    READY   STATUS    RESTARTS   AGE
istio-ingressgateway-58b4844495-fskhb   1/1     Running   0          9m22s
istio-ingressgateway-58b4844495-rbkbh   1/1     Running   0          9m6s
istiod-b786dd59c-bzfsg                  1/1     Running   0          9m41s
istiod-b786dd59c-jc5gn                  1/1     Running   0          9m41s
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central/asm (firm-solution-315106)$ asmctl validate
[asmctl version 0.4.1]
Using Kubernetes context: firm-solution-315106_us-central1-b_central
To change the context, use the --context flag
Validating enabled APIs
OK
Validating ingressgateway configuration
OK
Validating istio system
OK
```

Task5: Enable sidecar injection

Anthos Service Mesh uses sidecar proxies to enhance network security, reliability, and observability. With Anthos Service Mesh, these functions are abstracted away from an application's primary container and implemented in a common out-of-process proxy delivered as a separate container in the same Pod.

Before you deploy workloads, make sure to configure sidecar proxy injection so that Anthos Service Mesh can monitor and secure traffic.

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central/asm (firm-solution-315106)$ kubectl label namespace default istio-injection=enabled --overwrite
namespace/default labeled
```

Note: Sidecar proxy injection is now enabled for future workloads.

Task6: Deploy Bookinfo, an Istio-enabled multi-service application

Bookinfo detail see here. We now use **kubectl** to deploy the application, not **istioctl**.

1. Look at the .yaml which describes the bookInfo application:

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1/central/asm (firm-solution-315106)$ cd ~/istio-1.6.11-asm.1
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ cat samples/bookinfo/platform/kube/bookinfo.yaml
# Copyright 2017 Istio Authors
#
#   Licensed under the Apache License, Version 2.0 (the "License");
#   you may not use this file except in compliance with the License.
#   You may obtain a copy of the License at
#
#       http://www.apache.org/licenses/LICENSE-2.0
#
#   Unless required by applicable law or agreed to in writing, software
#   distributed under the License is distributed on an "AS IS" BASIS,
#   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
#   See the License for the specific language governing permissions and
#   limitations under the License.

################################################################################################
# This file defines the services, service accounts, and deployments for the Bookinfo sample.
#
# To apply all 4 Bookinfo services, their corresponding service accounts, and deployments:
#
#   kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml
#
# Alternatively, you can deploy any resource separately:
#
#   kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml -l service=reviews # reviews Service
#   kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml -l account=reviews # reviews ServiceAccount
#   kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml -l app=reviews,version=v3 # reviews-v3 Deployment
################################################################################################

################################################################################################
# Details service
################################################################################################
apiVersion: v1
kind: Service
```

```
apiVersion: v1
kind: Service
metadata:
  name: details
  labels:
    app: details
    service: details
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: details
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfo-details
  labels:
    account: details
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: details-v1
  labels:
    app: details
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: details
      version: v1
  template:
```

Look for containers to see that each Deployment, has **one** container, for each version of each service in the Bookinfo application.

2. In **Cloud Shell**, use the following command to inject the proxy sidecar along with each application Pod that is deployed.

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ kubectl apply -f samples/bookinfo/platform/kube/bookinfo.yaml
service/details created
serviceaccount/bookinfo-details created
deployment.apps/details-v1 created
service/ratings created
serviceaccount/bookinfo-ratings created
deployment.apps/ratings-v1 created
service/reviews created
serviceaccount/bookinfo-reviews created
deployment.apps/reviews-v1 created
deployment.apps/reviews-v2 created
deployment.apps/reviews-v3 created
service/productpage created
serviceaccount/bookinfo-productpage created
deployment.apps/productpage-v1 created
```

Task7: Enable external access using an Istio Ingress Gateway

Now that the Bookinfo services are up and running, you need to make the application accessible from outside of your Kubernetes cluster, e.g. from a browser. An **Istio Gateway** is used for this purpose.

1. Configure the **ingress gateway** for the application, which exposes an *external IP* you will use later:

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ kubectl apply -f samples/bookinfo/networking/bookinfo-gateway.yaml
gateway.networking.istio.io/bookinfo-gateway created
virtualservice.networking.istio.io/bookinfo created
```

Task8: Verify the Bookinfo deployments

1. Confirm that the application has been deployed correctly, review services, pods, and the ingress gateway:
2. Review running application pods:

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ kubectl get services
NAME          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
details       ClusterIP   10.107.242.139   <none>        9080/TCP   77s
kubernetes    ClusterIP   10.107.240.1     <none>        443/TCP    17h
productpage   ClusterIP   10.107.241.221   <none>        9080/TCP   74s
ratings       ClusterIP   10.107.253.16    <none>        9080/TCP   76s
reviews       ClusterIP   10.107.249.63    <none>        9080/TCP   75s
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
details-v1-79c697d759-4ntj6       2/2     Running   0          85s
productpage-v1-65576bb7bf-t4mf9   2/2     Running   0          81s
ratings-v1-7d99676f7f-vg2dq       2/2     Running   0          84s
reviews-v1-987d495c-jrglt         2/2     Running   0          83s
reviews-v2-6c5bf657cf-xmv9h       2/2     Running   0          82s
reviews-v3-5f7b9f4f77-nxhp5       2/2     Running   0          82s
```

3. Confirm that the Bookinfo application is running by sending a curl request to it from some pod, within the cluster, for example from ratings:

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ kubectl exec -it $(kubectl get pod -l app=ratings -o jsonpath='{.items[0].metadata.name}') \
>     -c ratings -- curl productpage:9080/productpage | grep -o "<title>.*</title>"
<title>Simple Bookstore App</title>
```

4. Confirm the ingress gateway has been created:

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ kubectl get gateway
NAME               AGE
bookinfo-gateway   96s
```

5. Get the **external IP** address of the **ingress gateway**:

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ kubectl get svc istio-ingressgateway -n istio-system
NAME                   TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)                                                                      AGE
istio-ingressgateway   LoadBalancer   10.107.255.199   35.225.171.74   15021:31681/TCP,80:31398/TCP,443:32244/TCP,15443:30415/TCP   15m
```

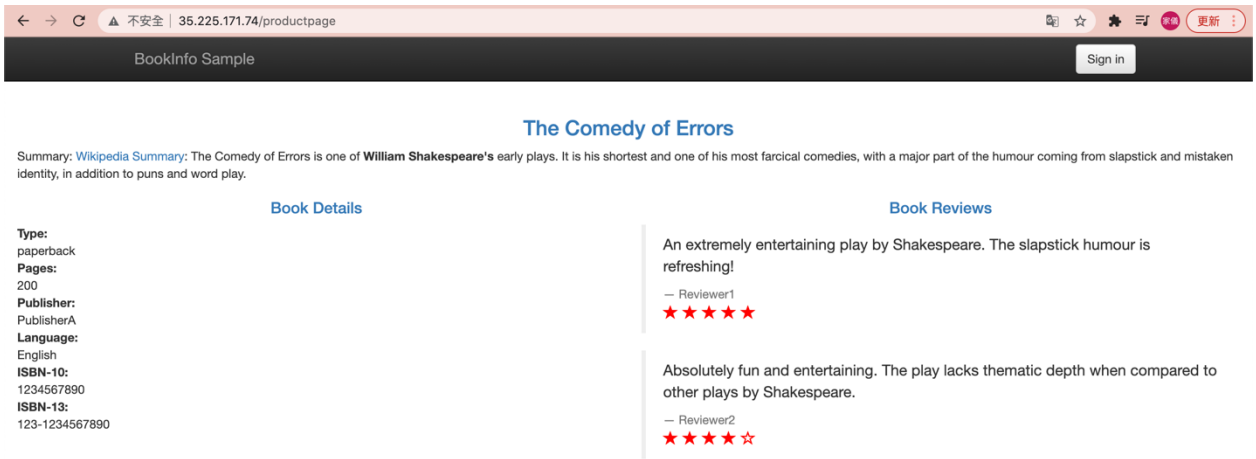
6. Now run the following command, replacing [EXTERNAL-IP] with the external IP that was outputted from the previous command:

```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ export GATEWAY_URL=35.225.171.74
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ curl -I http://${GATEWAY_URL}/productpage
HTTP/1.1 200 OK
content-type: text/html; charset=utf-8
content-length: 4183
server: istio-envoy
date: Sat, 29 May 2021 06:52:26 GMT
x-envoy-upstream-service-time: 610
```

Task9: Use the Bookinfo application

Try the application in your web browser

1. Point your browser to http://[$GATEWAY_URL]/productpage to see the BookInfo web page. Don't forget to replace [$GATEWAY_URL] with your working external IP address.



2. Refresh the page several times.

   Notice how you see three different versions of reviews, since we have not yet used Istio to control the version routing.

   There are three different book review services being called in a round-robin style:

   - No stars

- Black stars
- Red stars

Switching among the three is normal Kubernetes routing/balancing behavior.

Task10: Generate a steady background load

Run the siege utility to simulate traffic to Bookinfo.

1. In Cloud Shell, install **siege**.

   Siege is a utility for generating load against Web sites.

2. Use **siege** to create traffic against your services.

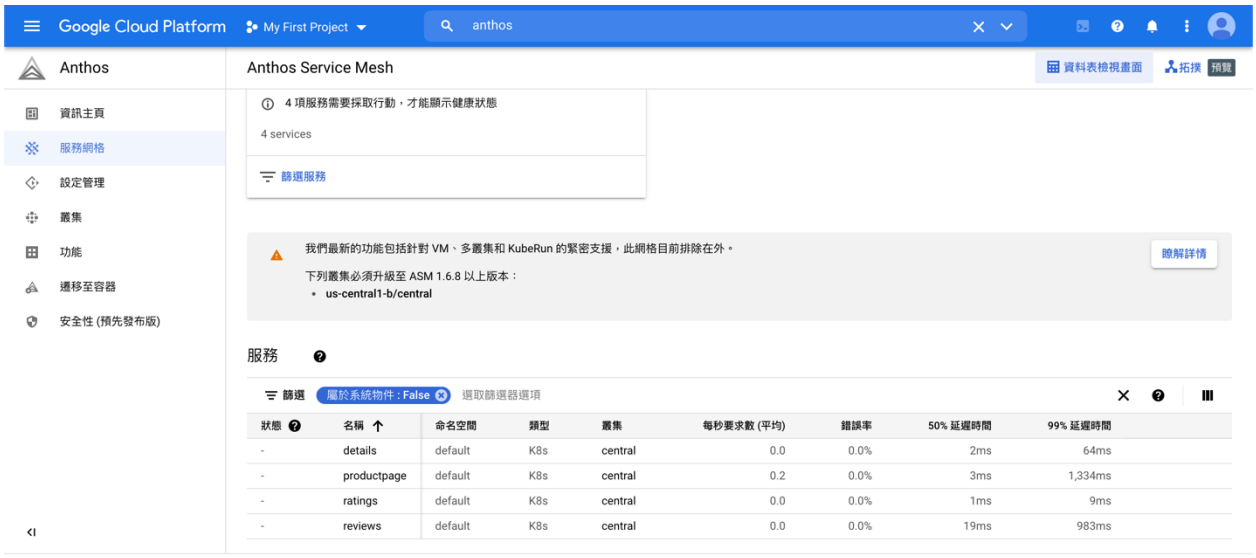
```
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ sudo apt install siege
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  siege
0 upgraded, 1 newly installed, 0 to remove and 7 not upgraded.
Need to get 103 kB of archives.
After this operation, 276 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian buster/main amd64 siege amd64 4.0.4-1 [103 kB]
Fetched 103 kB in 0s (1,095 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package siege.
(Reading database ... 116603 files and directories currently installed.)
Preparing to unpack .../siege_4.0.4-1_amd64.deb ...
Unpacking siege (4.0.4-1) ...
Setting up siege (4.0.4-1) ...
Processing triggers for man-db (2.8.5-2) ...
chiayi_katherine1028@cloudshell:~/istio-1.6.11-asm.1 (firm-solution-315106)$ siege http://${GATEWAY_URL}/productpage
New configuration template added to /home/chiayi_katherine1028/.siege
Run siege -C to view the current settings in that file
** SIEGE 4.0.4
** Preparing 25 concurrent users for battle.
The server is now under siege...
```

Task11: Evaluate service performance using the Anthos Service Mesh dashboard

Gather data from the services table view.

1. In the Console, go to **Navigation menu** > **Anthos** > **Dashboard**.

2. Click **View Service Mesh**.



3. On the bottom half of the window, you will see a **Service** section.

4. Click on the **productpage** service to drill down and see more details.