

Alibaba Cloud

Alibaba Cloud Service Mesh Multi-cluster

Document Version: 20220207

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).
5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.
6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions









Style	Description	Example
 Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
 Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
 Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: If the weight is set to 0, the server no longer receives new requests.
 Note	A note indicates supplemental instructions, best practices, tips, and other content.	 Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click Settings> Network> Set network type .
Bold	Bold formatting is used for buttons, menus, page names, and other UI elements.	Click OK .
<code>Courier font</code>	Courier font is used for commands	Run the <code>cd /d C:/window</code> command to enter the Windows system folder.
<i>Italic</i>	Italic formatting is used for parameters and variables.	<code>bae log list --instanceid</code> <i>Instance_ID</i>
[] or [a b]	This format is used for an optional value, where only one item can be selected.	<code>ipconfig [-all -t]</code>
{ } or {a b}	This format is used for a required value, where only one item can be selected.	<code>switch {active stand}</code>

Table of Contents


1. Use ASM to manage applications in ECI pods running in ASK ...	05
2. Use ASM to manage applications in ECI pods running on virtu...	07
3. Use ASM to manage applications in registered external Kubern...	09
4. Use ASM to deploy an application in multiple clusters in the s...	12
5. Use ASM to implement cross-region disaster recovery and load...	23

1. Use ASM to manage applications in ECI pods running in ASK clusters

Container Service provides various serverless containers based on virtual nodes and Elastic Container Instance (ECI). For example, Serverless Kubernetes (ASK) clusters enable seamless integration of Kubernetes and ECI. This topic describes how to use Alibaba Cloud Service Mesh (ASM) to manage applications in ECI pods that run in ASK clusters.

Prerequisites

- An ASK cluster is created. For more information, see [ASK quick start](#).

 **Note** To use the service discovery feature, you must set the service discovery mode to PrivateZone or CoreDNS when you create an ASK cluster.

- The ASK cluster is added to your ASM instance. For more information, see [Add a cluster to an ASM instance](#).


Enable automatic sidecar injection

After you enable automatic sidecar injection for a namespace in the ASM console, an Envoy proxy is automatically injected as a sidecar into each pod that is created in the namespace. These Envoy proxies comprise the data plane of the ASM instance.

- Log on to the [ASM console](#).
- In the left-side navigation pane, choose **Service Mesh > Mesh Management**.
- On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.
- On the details page of the ASM instance, click **Namespace** in the left-side navigation pane.
- On the **Namespaces** page, find the namespace for which you want to enable automatic sidecar injection and click **Enable Automatic Sidecar Injection** in the **Automatic Sidecar Injection** column.
- In the **Submit** message, click **OK**.

Create an application that runs in an ECI pod

All pods that run in ASK clusters are ECI pods. You do not need to label these pods.

 **Note** After an application is created and run in an ECI pod, ASM can manage the application on the data plane by using sidecars.

- Run the following command to create an NGINX application:

```
kubectl run nginx -n default --image nginx
```

- Run the following command to view the information about the pods that run on virtual nodes:

```
kubectl get pod -n default -o wide|grep virtual-kubelet
```

FAQ

Why am I unable to enable service discovery for an ASK cluster?

If the logs of containers where sidecars are injected indicate that the `istiod.istio-system` service is resolved to an invalid IP address, the PrivateZone service of Alibaba Cloud DNS (DNS) is not activated. You can [submit a ticket](#) to enable the PrivateZone service or you can install CoreDNS.

2. Use ASM to manage applications in ECI pods running on virtual nodes of ACK clusters

Container Service provides various serverless containers based on virtual nodes and Elastic Container Instance (ECI). For example, virtual nodes can be deployed in Container Service for Kubernetes (ACK) clusters to enable seamless integration of Kubernetes and ECI. You can create ECI pods as needed to avoid the planning of cluster capacity. This topic describes how to use Alibaba Cloud Service Mesh (ASM) to manage applications in ECI pods that run on the virtual nodes of ACK clusters.

Prerequisites


- The version of your ASM instance is v1.7.5.41-ge61a01c3-aliyun or later.
- The ack-virtual-node component is deployed and properly runs in your ACK cluster. For more information, see [Step 1: Deploy ack-virtual-node in ACK clusters](#).
- The ACK cluster is added to your ASM instance. For more information, see [Add a cluster to an ASM instance](#).

Enable automatic sidecar injection

After you enable automatic sidecar injection for a namespace in the ASM console, an Envoy proxy is automatically injected as a sidecar into each pod that is created in the namespace. These Envoy proxies comprise the data plane of the ASM instance.

1. Log on to the [ASM console](#).
2. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.
3. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.
4. On the details page of the ASM instance, click **Namespace** in the left-side navigation pane.
5. On the **Namespaces** page, find the namespace for which you want to enable automatic sidecar injection and click **Enable Automatic Sidecar Injection** in the **Automatic Sidecar Injection** column. In this example, automatic sidecar injection is enabled for the namespaces that are named default and vk. If no namespaces are available, you must create at least two namespaces.
6. In the **Submit** message, click **OK**.

Create an application that runs in an ECI pod

 **Note** After an application is created and run in an ECI pod, ASM can manage the application on the data plane by using sidecars.

Use a pod label to create an application that runs in an ECI pod

You can add the `label alibabacloud.com/eci=true` label to a pod to make the pod an ECI pod that runs on a virtual node.

1. Run the following command to check whether the `istio-injection=enabled` label is added to the namespace that is named default:

```
kubectl get ns default --show-labels
```

The following output is expected:

NAME	STATUS	AGE	LABELS
default	Active	84d	istio-injection=enabled,provider=asm

2. Run the following command to create an NGINX application:

```
kubectl run nginx -n default --image nginx -l alibabacloud.com/eci=true
```

3. Run the following command to view the information about the pods that run on virtual nodes:

```
kubectl get pod -n default -o wide|grep virtual-kubelet
```

Use a namespace label to create an application that runs in an ECI pod

You can add the `label alibabacloud.com/eci=true` label to the namespace of a pod to make the pod an ECI pod that runs on a virtual node.

1. Run the following command to check whether the `istio-injection=enabled` label is added to the vk namespace:

```
kubectl get ns default --show-labels
```

The following output is expected:

NAME	STATUS	AGE	LABELS
default	Active	84d	istio-injection=enabled,provider=asm

2. Run the following command to add a label to the vk namespace:

```
kubectl label namespace vk alibabacloud.com/eci=true
```

3. Run the following command to create an NGINX application:

```
kubectl -n vk run nginx --image nginx
```

4. Run the following command to view the information about the pods that run on virtual nodes:

```
kubectl -n vk get pod -o wide|grep virtual-kubelet
```


3. Use ASM to manage applications in registered external Kubernetes clusters

Alibaba Cloud Service Mesh (ASM) allows you to manage applications in external Kubernetes clusters that are registered in the Container Service console.

Prerequisites

- An external Kubernetes cluster that can access the Internet is registered in the Container Service console. For more information, see [Register an external Kubernetes cluster](#).
- ASM is activated. For more information, see [Create an ASM instance](#).

Procedures

1. Log on to the [ASM console](#).
2. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.
3. On the **Mesh Management** page, click **Create ASM Instance**.
4. In the **Create ASM Instance** panel, enter an instance name, and select a region, a virtual private cloud (VPC), and a vSwitch.

Note

- Select the region where the registered external Kubernetes cluster resides or a region that is nearest to the cluster.
- Select the VPC where the registered external Kubernetes cluster resides.
- Select a vSwitch from the vSwitch drop-down list as required. If no vSwitch is available, click **Create vSwitch** to create one. For more information, see [Work with vSwitches](#).

5. Specify whether to allow Internet access to the API server.

Note

- An ASM instance runs on Kubernetes runtime. You can use the API server to define various mesh resources, such as virtual services, destination rules, and Istio gateways.
- If you allow Internet access to the API server, an elastic IP address (EIP) is created and bound to a Server Load Balancer (SLB) instance on the private network. Port 6443 of the API server is exposed. You can use the kubeconfig file of the cluster to connect to and manage the registered cluster to define mesh resources over the Internet.
 - If you do not allow Internet access to the API server, no EIP is created. You can use the kubeconfig file to connect to and manage the registered cluster to define mesh resource only through the VPC where the cluster resides.

6. Select **Expose Istio Pilot** in the Internet Access section.


Note

- If you do not select **Expose Istio Pilot**, the pod in the registered external cluster cannot connect to Istio Pilot, and applications in the pod cannot work as expected.


7. Keep the default settings for other parameters. Click **OK** to create the ASM instance.

 **Note** It takes 2 to 3 minutes to create an ASM instance.

8. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.
9. On the details page of the ASM instance, choose **Data Plane (Service Discovery) > Kubernetes Clusters** in the left-side navigation pane. On the Kubernetes Clusters page, click **Add**.
10. In the **Add Cluster** panel, select an external cluster as required and click **OK**.


 **Note** After you add a cluster to an ASM instance, the status of the ASM instance becomes **Updating**. Wait a few seconds and click **Refresh** in the upper-right corner. If the cluster is added to the instance, the status of the instance will become **Running**. The waiting duration may vary with the network speed. On the **Kubernetes Clusters** page, you can view the information about the added cluster.

11. On the details page of the ASM instance, click **ASM Gateways** in the left-side navigation pane. On the ASM Gateways page, click **Deploy Default Ingress Gateway**.
12. In the **Deploy Ingress Gateway** panel, set the parameters as required.
 - i. Select the cluster where you want to deploy an ingress gateway service from the **Cluster** drop-down list.
 - ii. Select **Internet Access** or **Internal Access** for the **SLB Instance Type** parameter.

 **Note** Different external clusters may support different types of SLB instances. For example, specific external clusters do not support internal SLB instances. Select the SLB instance type as required. If the registered external cluster does not support SLB instances, select **Internet Access** for **SLB Instance Type**. After the ingress gateway service is defined, edit the YAML file of the ingress gateway service to specify the service type, such as **Nodeport** or **ClusterIP**.

You can only create SLB instances instead of using existing ones for external clusters.

- iii. Configure port mappings.

 **Note**

- We recommend that you use the same port for the container and the service in a mapping and enable the port on the Istio gateway.
- ASM provides four default ports that are commonly used by Istio. You can keep or delete the default ports, or add new ports as required.

13. Click **OK** to deploy the ingress gateway service.

After you deploy the ingress gateway service, log on to the external cluster to view the details of the ingress gateway service.

Deploy applications in the external cluster

Deploy applications in the external cluster by running commands on the kubectl client or using the external cluster console. For more information, see [Deploy an application in an ASM instance](#).

Define Istio resources

Define Istio resources in the ASM console. For more information, see [Define Istio resources](#).

4. Use ASM to deploy an application in multiple clusters in the same VPC

Alibaba Cloud Service Mesh (ASM) allows you to deploy the microservices of an application in multiple clusters in the same Virtual Private Cloud (VPC). This topic uses the Bookinfo application as an example to describe how to deploy an application in two clusters that share the same VPC and are added to the same ASM instance.

Prerequisites

- Two Container Service for Kubernetes (ACK) clusters are created in the same VPC. For more information, see [Create an ACK dedicated cluster](#). In this topic, the two clusters are m1c1 and m1c2.
- An ASM instance is created. For more information, see [Create an ASM instance](#). In this topic, the ASM instance is mesh1.

Step 1: Change the security group names for the two clusters

Change the security group names for the two clusters. Make sure that users can deduce the corresponding clusters from the new security group names. In this example, change the security group names to m1c1-sg and m1c2-sg.

- Log on to the [ECS console](#).
- In the left-side navigation pane, choose **Network & Security > Security Groups**.
- In the top navigation bar, select a region.
- On the **Security Groups** page, find the security group to be modified and click **Modify** in the **Actions** column.
- In the dialog box that appears, modify **Security Group Name** and **Description**.
- Click **OK**.

The following figure shows the new security group names.

<input type="checkbox"/>	Security Group ID/Name	Tag	VPC	Related Instances	Available IP Addresses	Network Type(All)	Security Group Type(All)	Creation Time	Description	Actions
<input type="checkbox"/>	sg-8vbgxqxdwanq4wh23zq m1c2-sg		vpc-8vbgmrtbx1jzrcleenuh5 meshvpc	6	1994	VPC	Basic Security Group	2020-08-11 10:00:00	This is used by kubern...	Modify Clone Restore Rules Manage Instances Add Rules Manage ENIs
<input type="checkbox"/>	sg-8vben21m59x4nm6pgkg m1c1-sg		vpc-8vbgmrtbx1jzrcleenuh5 meshvpc	3	1976	VPC	Basic Security Group	2020-08-11 10:00:00	security group of ACS ...	Modify Clone Restore Rules Manage Instances Add Rules Manage ENIs

Step 2: Set security group rules to allow mutual access between the two clusters

To enable the two clusters to access each other, you must set rules for accessing the security groups of the two clusters.

- On the configuration page of the m1c1-sg group, create a rule to allow the access from m1c2-sg. For more information, see [Add security group rules](#).
- On the configuration page of the m1c2-sg group, create a rule to allow the access from m1c1-sg.

Step 3: Add the two clusters to the ASM instance and deploy an ingress gateway

The two clusters can access each other. After you add the two clusters to the ASM instance, you only need to deploy an ingress gateway for one of the two clusters.

1. Add the two clusters to the ASM instance. For more information, see [Add a cluster to an ASM instance](#).
2. Deploy an ingress gateway for the m1c1 cluster. For more information, see [Deploy an ingress gateway service](#).

Step 4: Deploy the Bookinfo application

ASM allows you to deploy an application across clusters. You can deploy the microservices of the Bookinfo application in the two clusters.

1. Deploy the Bookinfo application excluding the v3 version of the reviews microservice in the m1c2 cluster. For more information, see [Deploy an application in an ASM instance](#).

 **Note** The v3 version of the reviews microservice displays ratings as red stars.

The following code shows the content of the YAML file:

```
#####
#####
# 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
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ratings-v1
  labels:
    app: ratings
    version: v1
spec:
```

```

spec:
  replicas: 1
  selector:
    matchLabels:
      app: ratings
      version: v1
  template:
    metadata:
      labels:
        app: ratings
        version: v1
    spec:
      serviceAccountName: bookinfo-ratings
      containers:
      - name: ratings
        image: docker.io/istio/examples-bookinfo-ratings-v1:1.15.0
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9080
---
#####
#####
# Reviews service
#####
#####
apiVersion: v1
kind: Service
metadata:
  name: reviews
  labels:
    app: reviews
    service: reviews
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: reviews
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfo-reviews
  labels:
    account: reviews
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reviews-v1
  labels:
    app: reviews
    version: v1
spec:
  replicas: 1

```

```
selector:
  matchLabels:
    app: reviews
    version: v1
template:
  metadata:
    labels:
      app: reviews
      version: v1
  spec:
    serviceAccountName: bookinfo-reviews
    containers:
      - name: reviews
        image: docker.io/istio/examples-bookinfo-reviews-v1:1.15.0
        imagePullPolicy: IfNotPresent
        ports:
          - containerPort: 9080
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reviews-v2
  labels:
    app: reviews
    version: v2
spec:
  replicas: 1
  selector:
    matchLabels:
      app: reviews
      version: v2
  template:
    metadata:
      labels:
        app: reviews
        version: v2
    spec:
      serviceAccountName: bookinfo-reviews
      containers:
        - name: reviews
          image: docker.io/istio/examples-bookinfo-reviews-v2:1.15.0
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 9080
# ---
# apiVersion: apps/v1
# kind: Deployment
# metadata:
#   name: reviews-v3
#   labels:
#     app: reviews
#     version: v3
# spec:
#   replicas: 1
```



```

# selector:
#   matchLabels:
#     app: reviews
#     version: v3
# template:
#   metadata:
#     labels:
#       app: reviews
#       version: v3
# spec:
#   serviceAccountName: bookinfo-reviews
#   containers:
#   - name: reviews
#     image: docker.io/istio/examples-bookinfo-reviews-v3:1.15.0
#     imagePullPolicy: IfNotPresent
#     ports:
#     - containerPort: 9080
---
#####
#####
# 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:
  name: productpage-v1
  labels:
    app: productpage
    version: v1
spec:
  replicas: 1
  selector:

```

```
matchLabels:
  app: productpage
  version: v1
template:
  metadata:
    labels:
      app: productpage
      version: v1
  spec:
    serviceAccountName: bookinfo-productpage
    containers:
      - name: productpage
        image: docker.io/istio/examples-bookinfo-productpage-v1:1.15.0
        imagePullPolicy: IfNotPresent
        ports:
          - containerPort: 9080
---
```

2. Deploy the v3 version of the reviews microservice and the rating microservice on which the reviews microservice depends in the m1c1 cluster.

The following code shows the content of the YAML file:

```
#####
#####
# Reviews service
#####
#####
apiVersion: v1
kind: Service
metadata:
  name: reviews
  labels:
    app: reviews
    service: reviews
spec:
  ports:
    - port: 9080
      name: http
  selector:
    app: reviews
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfo-reviews
  labels:
    account: reviews
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reviews-v3
  labels:
    app: reviews
```

```

    version: v3
spec:
  replicas: 1
  selector:
    matchLabels:
      app: reviews
      version: v3
  template:
    metadata:
      labels:
        app: reviews
        version: v3
    spec:
      serviceAccountName: bookinfo-reviews
      containers:
        - name: reviews
          image: docker.io/istio/examples-bookinfo-reviews-v3:1.15.0
          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

```

Step 5: Define a virtual service and an Istio gateway

1. In the namespace that is named default of the ASM instance, define a virtual service that is named bookinfo. For more information, see [Define Istio resources](#).

The following code shows the content of the YAML file:

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: bookinfo
spec:
  hosts:
  - "*"
  gateways:
  - bookinfo-gateway
  http:
  - match:
    - uri:
        exact: /productpage
    - uri:
        prefix: /static
    - uri:
        exact: /login
    - uri:
        exact: /logout
    - uri:
        prefix: /api/v1/products
    route:
    - destination:
        host: productpage
        port:
          number: 9080
```

2. In the namespace that is named default of the ASM instance, define an Istio gateway that is named bookinfo-gateway. For more information, see [Define Istio resources](#).

The following code shows the content of the YAML file:

```
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: bookinfo-gateway
spec:
  selector:
    istio: ingressgateway # use istio default controller
  servers:
  - port:
      number: 80
      name: http
      protocol: HTTP
    hosts:
    - "*"

```

You can refresh the product page to view the effect of the three versions of the reviews microservice in turn. The v3 version of the reviews microservice can take effect normally though it does not reside in the same cluster as other microservices.

Step 6: Make the v3 version of the reviews microservice take effect all the time (Optional)

You can define a destination rule and a virtual service to set a policy for deploying the microservices of the Bookinfo application. The following example specifies that the v3 version of the reviews microservice always takes effect.

1. In the namespace that is named default of the ASM instance, define a destination rule that is named reviews.

The following code shows the content of the YAML file:

```
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: reviews
spec:
  host: reviews
  subsets:
    - name: v1
      labels:
        version: v1
    - name: v2
      labels:
        version: v2
    - name: v3
      labels:
        version: v3
```

2. In the namespace that is named default of the ASM instance, define a virtual service that is named reviews.

The following code shows the content of the YAML file:

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: reviews
spec:
  hosts:
    - reviews
  http:
    - route:
        - destination:
            host: reviews
            subset: v3
```

When you access the product page, the v3 version of the reviews microservice takes effect all the time. In this case, ratings are displayed as red stars.

The Comedy of Errors

Summary: [Wikipedia Summary](#): The Comedy of Errors is one of **William Shakespeare's** early plays. It is his shortest and one of his most farcical comedies, with a major part of the humour coming from slapstick and mistaken identity, in addition to puns and word play.

Book Details

Type:
paperback
Pages:
200
Publisher:
PublisherA
Language:
English
ISBN-10:
1234567890
ISBN-13:
123-1234567890

Book Reviews

An extremely entertaining play by Shakespeare. The slapstick humour is refreshing!

— Reviewer1

★★★★★

Absolutely fun and entertaining. The play lacks thematic depth when compared to other plays by Shakespeare.

— Reviewer2

★★★★☆

5. Use ASM to implement cross-region disaster recovery and load balancing

Alibaba Cloud Service Mesh (ASM) provides cross-region traffic distribution and failover capabilities for applications. The cross-region traffic distribution feature implements cross-region load balancing by routing traffic to multiple clusters based on their weights. The cross-region failover feature implements cross-region disaster recovery by transferring traffic from a faulty region to another region. This topic describes how to use cross-region failover and traffic distribution features to implement cross-region disaster recovery and load balancing. The Bookinfo application is used as an example.


Prerequisites

An ASM instance is created. For more information, see [Create an ASM instance](#).

Plan a network

Before you use ASM, you must plan a network for ASM. This involves the CIDR blocks and names of the vSwitches, virtual private clouds (VPCs), and clusters. In this example, create a network based on the following plan:

- Network plan for the vSwitches and VPCs
 - vSwitches

 **Notice** To prevent route conflicts when you use Cloud Enterprise Network (CEN) to connect to a VPC, use different CIDR blocks for each vSwitch.

vSwitch	VPC	IPv4 CIDR block
vpc-hangzhou-switch-1	vpc-hangzhou	192.168.0.0/24
vpc-shanghai-switch-1	vpc-shanghai	192.168.2.0/24

- VPCs

VPC	Region	IPv4 CIDR block
vpc-hangzhou	cn-hangzhou	192.168.0.0/16
vpc-shanghai	cn-shanghai	192.168.0.0/16

- Network plan for the clusters

Cluster	Region	VPC	Pod CIDR	Service CIDR
ack-hangzhou	cn-hangzhou	vpc-hangzhou	10.45.0.0/16	172.16.0.0/16
ack-shanghai	cn-shanghai	vpc-shanghai	10.47.0.0/16	172.18.0.0/16

Step 1: Create clusters in different regions

1. Create two vSwitches in the China (Hangzhou) and China (Shanghai) regions based on the preceding plan, and then create VPCs that are associated with the vSwitches. For more information, see [Create a vSwitch](#) and [Create a VPC](#).
2. Use the VPCs that you created and the preceding network plan to create clusters in the China (Hangzhou) and China (Shanghai) regions. For more information, see [Create an ACK managed cluster](#).

Step 2: Use CEN to implement cross-region VPC communication

1. Create a CEN instance.
 - i. Log on to the [CEN console](#).
 - ii. On the **Instances** page, click **Create CEN Instance**.
 - iii. In the **Create CEN Instance** panel, set the parameters and click **OK**.

Parameter	Description
Name	The name of the CEN instance. The name must be 2 to 128 characters in length and can contain digits, underscores (_), and hyphens (-). It must start with a letter.
Description	The description of the instance.
Network Type	The type of the network. In this example, VPC is used.
Region	The region where the instance resides. In this example, China (Hangzhou) is used.
Networks	The instance that you want to attach. In this example, the VPC that you created in the China (Hangzhou) region is used.

2. Attach a network instance.
 - i. On the **Instances** page, find the CEN instance that you created and click its ID.
 - ii. Click the **Networks** tab and then click **Attach Network**.
 - iii. In the **Attach Network** panel, set the parameters on the **Your Account** tab and click **OK**.

Parameter	Description
Network Type	The type of the network. In this example, VPC is used.
Region	The region where the instance resides. In this example, China (Shanghai) is used.
Networks	The instance that you want to attach. In this example, the VPC that you created in the China (Shanghai) region is used.

3. Purchase the bandwidth plan. For more information, see [Purchase a bandwidth plan](#).
4. Set the cross-region connection bandwidth.
 - i. On the **Instances** page, find the CEN instance that you created and click its ID.
 - ii. Click the **Region Connections** tab and then click **Set Region Connection**.
 - iii. In the **Set Region Connection** panel, select the bandwidth plan that you purchased from the **Bandwidth Plans** drop-down list, set the **Connected Regions** to China (Shanghai) and China (Hangzhou), and then click **OK**.

5. Add rules to the security groups.

Add the pod network CIDR of the ack-shanghai cluster to the security group of the ack-hangzhou cluster and vice versa. This allows the pod CIDR of a cluster to access the localhost of another cluster.

- i. Log on to the [ACK console](#).
- ii. On the **Clusters** page, find the ack-shanghai cluster and click **Details** in the **Actions** column.
- iii. On the **Cluster Information** page, click the **Basic Information** tab.
View the pod network CIDR of the ack-shanghai cluster and go back to the **Clusters** page.
- iv. On the **Clusters** page, find the ack-hangzhou cluster and click **Details** in the **Actions** column.
- v. On the **Cluster Information** page, click the **Cluster Resources** tab. Then, click the security group ID next to **Security Group**.
- vi. On the **Security Group Rules** page, click **Add Rule** on the **Inbound** tab.
- vii. Set the **Protocol Type** parameter to **All** and the **Source** parameter to the pod network CIDR of the ack-shanghai cluster. Then, click **Save** in the **Actions** column.
- viii. Repeat the preceding substeps to view the pod network CIDR of the ack-hangzhou cluster and add the pod network CIDR to the security group of the ack-shanghai cluster.

Step 3: Publish the pod route information to CEN

1. Log on to the [ACK console](#).
2. On the **Clusters** page, find the ack-hangzhou cluster and click **Details** in the **Actions** column.
3. On the cluster details page, click the **Cluster Resources** tab and click the VPC ID next to **VPC**.
4. On the **Information** tab, view the router ID in the **vRouter Basic Information** section.
5. In the left-side navigation pane of the VPC console, click **Route Tables**.
6. On the **Route Tables** page, find the router ID and click the name of the route instance.
7. On the **Route Entry List** tab, click **Custom**.
8. Click **Publish** next to the pod CIDR block. This topic uses a CIDR block of 10.45.0.0/16.
9. In the **Publish Route Entry** dialog box, click **OK**.
10. Repeat the preceding substeps to publish the pod route information about the ack-shanghai cluster to CEN.
11. Verify whether the pod route information about the ack-hangzhou and ack-shanghai clusters is published to CEN.

On the details page of the route table in the China (Hangzhou) region, click **Dynamic** on the **Routes** tab. You can view the route information of the pod CIDR block in the ack-shanghai cluster and the route information about the IPv4 CIDR block of vpc-shanghai-switch-1. On the details page

of the route table in the China (Shanghai) region, you can also view the route information of the pod CIDR block in the ack-hangzhou cluster and the route information about the IPv4 CIDR block of vpc-hangzhou-switch-1. This indicates that the pod route information about the ack-hangzhou and ack-shanghai clusters is published to CEN.

Step 4: Add clusters to an ASM instance

Add the ack-hangzhou and ack-shanghai clusters that you created to an ASM instance.

1. Log on to the [ASM console](#).
2. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.
3. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.
4. On the details page of the ASM instance, choose **Data Plane (Service Discovery) > Kubernetes Clusters** in the left-side navigation pane. On the Kubernetes Clusters page, click **Add**.
5. In the **Add Cluster** panel, select the ack-hangzhou cluster and click **OK**.
6. In the **Note** dialog box, click **OK**.
7. Repeat the preceding substeps to add the ack-shanghai cluster to the same ASM instance.

Step 5: Configure an ingress gateway in ASM

1. View the ID of the ack-shanghai cluster.
 - i. Log on to the [ACK console](#).
 - ii. On the **Clusters** page, find the ack-shanghai cluster and click **Details** in the **Actions** column.
 - iii. On the Cluster Information page, click the **Basic Information** tab.

In the **Basic Information** section, view the ID of the ack-shanghai cluster.

2. Log on to the [ASM console](#).
3. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.
4. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.
5. On the details page of the ASM instance, click **ASM Gateways** in the left-side navigation pane.
6. On the **ASM Gateways** page, click **Create**.
7. On the **Create** page, select the ack-hangzhou cluster from the **Cluster** drop-down list, set the **SLB Instance Type** parameter to **Internet Access**, and then select an SLB instance from the **Create SLB Instance** drop-down list. Keep the default values of other parameters. Click **Create**.
8. On the **ASM Gateways** page, find the gateway named ingressgateway and click **YAML** in the **Actions** column.
9. In the **Edit** panel, enter the ID of the ack-shanghai cluster and click **OK**.

```
spec:
  clusterIds:
    - ack-hangzhou cluster-id
    - ack-shanghai cluster-id
```

Step 6: Deploy the Bookinfo application

1. Use kubectl to connect to the ack-hangzhou cluster. For more information, see [Connect to ACK](#)

clusters by using `kubectl`.

2. Create an `ack-hangzhou-k8s.yaml` file that contains the following content:

☐ View the content of the YAML file.

```
# 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-v1:1.16.2
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 9080
          securityContext:
            runAsUser: 1000
```

```
runAsUser: 1000
---
# 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
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ratings-v1
  labels:
    app: ratings
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ratings
      version: v1
  template:
    metadata:
      labels:
        app: ratings
        version: v1
    spec:
      serviceAccountName: bookinfo-ratings
      containers:
        - name: ratings
          image: docker.io/istio/examples-bookinfo-ratings-v1:1.16.2
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 9080
          securityContext:
            runAsUser: 1000
---
# Reviews service
apiVersion: v1
```

```
kind: Service
metadata:
  name: reviews
  labels:
    app: reviews
    service: reviews
spec:
  ports:
    - port: 9080
      name: http
  selector:
    app: reviews
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfo-reviews
  labels:
    account: reviews
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reviews-v1
  labels:
    app: reviews
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: reviews
      version: v1
  template:
    metadata:
      labels:
        app: reviews
        version: v1
    spec:
      serviceAccountName: bookinfo-reviews
      containers:
        - name: reviews
          image: docker.io/istio/examples-bookinfo-reviews-v1:1.16.2
          imagePullPolicy: IfNotPresent
          env:
            - name: LOG_DIR
              value: "/tmp/logs"
          ports:
            - containerPort: 9080
          volumeMounts:
            - name: tmp
              mountPath: /tmp
            - name: wlp-output
              mountPath: /opt/ibm/wlp/output
```

```
        securityContext:
          runAsUser: 1000
        volumes:
        - name: wlp-output
          emptyDir: {}
        - name: tmp
          emptyDir: {}
    ---
# 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:
  name: productpage-v1
  labels:
    app: productpage
    version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: productpage
      version: v1
  template:
    metadata:
      labels:
        app: productpage
        version: v1
    spec:
      serviceAccountName: bookinfo-productpage
      containers:
      - name: productpage
        image: docker.io/istio/examples-bookinfo-productpage-v1:1.16.2
        imagePullPolicy: IfNotPresent
```

```


    ports:
      - containerPort: 9080
    volumeMounts:
      - name: tmp
        mountPath: /tmp
    securityContext:
      runAsUser: 1000
  volumes:
    - name: tmp
      emptyDir: {}
---

```

3. Run the following command to deploy the Bookinfo application in the ack-hangzhou cluster:

```
kubectl apply -f ack-hangzhou-k8s.yaml
```

4. Use `kubectl` to connect to the ack-shanghai cluster. For more information, see [Connect to ACK clusters by using kubectl](#).

 **Note** When you use `kubectl` to connect to the ack-shanghai cluster, you must switch the kubeconfig of the ack-hangzhou cluster to that of the ack-shanghai cluster.

5. Create an `ack-shanghai.yaml` file that contains the following content:

[View the content of the YAML file.](#)

```

# 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-v1:1.16.2
          imagePullPolicy: IfNotPresent
          ports:
            - containerPort: 9080
          securityContext:
            runAsUser: 1000
---
# 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
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ratings-v1
  labels:
    app: ratings
    version: v1
spec:
  replicas: 1
```



```

selector:
  matchLabels:
    app: ratings
    version: v1
template:
  metadata:
    labels:
      app: ratings
      version: v1
  spec:
    serviceAccountName: bookinfo-ratings
    containers:
    - name: ratings
      image: docker.io/istio/examples-bookinfo-ratings-v1:1.16.2
      imagePullPolicy: IfNotPresent
      ports:
      - containerPort: 9080
      securityContext:
        runAsUser: 1000
---
# Reviews service
apiVersion: v1
kind: Service
metadata:
  name: reviews
  labels:
    app: reviews
    service: reviews
spec:
  ports:
  - port: 9080
    name: http
  selector:
    app: reviews
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: bookinfo-reviews
  labels:
    account: reviews
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: reviews-v2
  labels:
    app: reviews
    version: v2
spec:
  replicas: 1
  selector:
    matchLabels:
      app: reviews
      version: v2

```

```
version: v2
template:
  metadata:
    labels:
      app: reviews
      version: v2
  spec:
    serviceAccountName: bookinfo-reviews
    containers:
      - name: reviews
        image: docker.io/istio/examples-bookinfo-reviews-v2:1.16.2
        imagePullPolicy: IfNotPresent
        env:
          - name: LOG_DIR
            value: "/tmp/logs"
        ports:
          - containerPort: 9080
        volumeMounts:
          - name: tmp
            mountPath: /tmp
          - name: wlp-output
            mountPath: /opt/ibm/wlp/output
        securityContext:
          runAsUser: 1000
    volumes:
      - name: wlp-output
        emptyDir: {}
      - name: tmp
        emptyDir: {}
---
# 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:
```

```


name: productpage-v1
labels:
  app: productpage
  version: v1
spec:
  replicas: 1
  selector:
    matchLabels:
      app: productpage
      version: v1
  template:
    metadata:
      labels:
        app: productpage
        version: v1
    spec:
      serviceAccountName: bookinfo-productpage
      containers:
      - name: productpage
        image: docker.io/istio/examples-bookinfo-productpage-v1:1.16.2
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 9080
        volumeMounts:
        - name: tmp
          mountPath: /tmp
        securityContext:
          runAsUser: 1000
      volumes:
      - name: tmp
        emptyDir: {}
---

```

6. Run the following command to deploy the Bookinfo application in the ack-shanghai cluster:

```
kubectl apply -f ack-shanghai.yaml
```

7. Use `kubectl` to connect to the ASM instance. For more information, see [Use kubectl to connect to an ASM instance](#).

 **Note** When you use `kubectl` to connect to the ASM instance, you must switch the kubeconfig of the ack-shanghai cluster to that of the ASM instance.

8. Create an `asm.yaml` file that contains the following content:

☐ View the content of the YAML file.

```

apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
  name: bookinfo-gateway
spec:
  selector:
    istio: ingressgateway # use istio default controller

```

```
servers:
- port:
  number: 80
  name: http
  protocol: HTTP
  hosts:
  - "*"
---
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: bookinfo
spec:
  hosts:
  - "*"
  gateways:
  - bookinfo-gateway
  http:
  - match:
    - uri:
      exact: /productpage
    - uri:
      prefix: /static
    - uri:
      exact: /login
    - uri:
      exact: /logout
    - uri:
      prefix: /api/v1/products
    route:
    - destination:
      host: productpage
      port:
        number: 9080
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: productpage
spec:
  host: productpage
  subsets:
  - name: v1
    labels:
      version: v1
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: reviews
spec:
  host: reviews
  subsets:
  - name: v1
```

```

    labels:
      version: v1
  - name: v2
    labels:
      version: v2
  - name: v3
    labels:
      version: v3
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: ratings
spec:
  host: ratings
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
  - name: v2-mysql
    labels:
      version: v2-mysql
  - name: v2-mysql-vm
    labels:
      version: v2-mysql-vm
---
apiVersion: networking.istio.io/v1alpha3
kind: DestinationRule
metadata:
  name: details
spec:
  host: details
  subsets:
  - name: v1
    labels:
      version: v1
  - name: v2
    labels:
      version: v2
---

```

9. Run the following command to create a routing rule in the ASM instance:

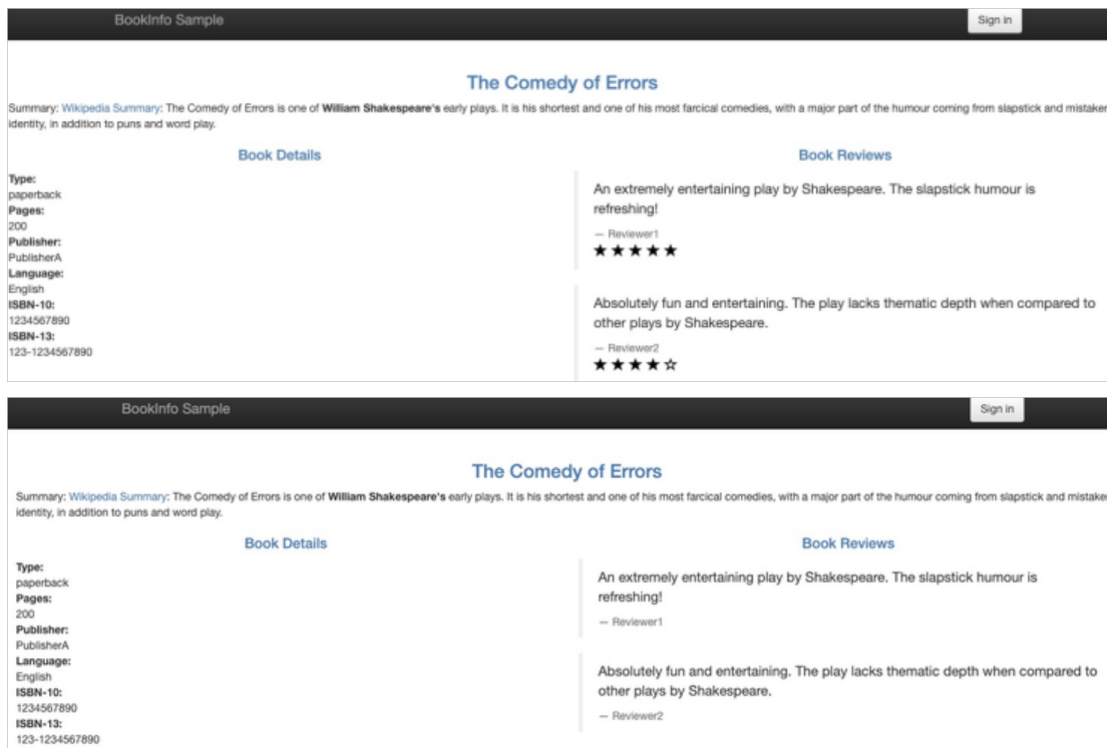
```
kubectl apply -f asm.yaml
```

10. Verify whether the Bookinfo application is deployed.

- i. Log on to the [ACK console](#).
- ii. In the left-side navigation pane of the ACK console, click **Clusters**.
- iii. On the **Clusters** page, find the ack-hangzhou cluster and click **Details** in the **Actions** column.

- iv. In the left-side navigation pane of the details page, choose **Network > Services**
- v. At the top of the **Services** page, select **istio-system** from the **Namespace** drop-down list. Find the ingress gateway that is named **istio-ingressgateway** and view the IP address whose port is 80 in the **External Endpoint** column.
- vi. Enter the *<IP address of the ingress gateway>/productpage* in the address bar of your browser.

Refresh the page multiple times. The following images alternately appear on the screen.






Step 7: Use the cross-region traffic distribution and failover features

Configure cross-region traffic distribution

1. Log on to the [ASM console](#).
2. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.
3. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.
4. In the **Basic Information** section, click **Enable locality traffic distribution** on the right of **Locality-Failover**.

 **Note** If you have enabled cross-region failover, you must disable cross-region failover before you can enable cross-region traffic distribution.

5. In the **Locality-Traffic-Distribution** dialog box, set the **Policy** parameter to **cn-hangzhou** and click **New Policy**.
6. Click the  icon and the  icon. Set the **To** parameter to **cn-hangzhou** and the **Weight** parameter to 90%.

7. Click the  icon, set the **To** parameter to `cn-shanghai` and the **Weight** parameter to 10%, and then click **Submit**.
8. Run the following command to request the `BookInfo` application 10 times to verify whether the cross-region traffic distribution is successful:

```
for ((i=1;i<=10;i++));do curl http://<Ingress gateway endpoint of port 80 in the ack-hangzhou cluster>/productpage 2>&1|grep full.stars;done
```

Expected output:

```
<!-- full stars: -->
<!-- full stars: -->
```

You can find that 10 access requests are made and two rows of `full stars` output are returned. This indicates that 9 of the 10 requests are routed to the v1 reviews service in the ack-hangzhou cluster and 1 request is routed to the v2 reviews service in the ack-shanghai cluster. Traffic is routed to different clusters based on the weights of the clusters.

Configure cross-region failover

1. Disables the reviews service in the ack-hangzhou cluster.
 - i. Log on to the [ACK console](#).
 - ii. In the left-side navigation pane of the ACK console, click **Clusters**.
 - iii. In the left-side navigation pane of the details page, choose **Workloads > Deployments**.
 - iv. On the **Deployments** page, set the **Namespace** parameter to default, find `reviews-v1`, and then click **Scale** in the **Actions** column.
 - v. In the **Scale** dialog box, set the **Desired Number of Pods** parameter to 0 and click **OK**.
2. Configure a destination rule.

Configure a destination rule. If the reviews service cannot be requested within 1 second, the reviews service will be ejected for 1 minute.

- i. Log on to the [ASM console](#).
- ii. In the left-side navigation pane, choose **Service Mesh > Mesh Management**.
- iii. On the **Mesh Management** page, find the ASM instance that you want to configure. Click the name of the ASM instance or click **Manage** in the **Actions** column of the ASM instance.
- iv. On the details page of the ASM instance, choose **Traffic Management > DestinationRule** in the left-side navigation pane.
- v. On the **DestinationRule** page, find the reviews service and click **YAML** in the **Actions** column.


vi. In the **Edit** panel, copy the following content to the code editor and click **OK**.

```
spec:
  .....
  trafficPolicy:
    connectionPool:
      http:
        maxRequestsPerConnection: 1
    outlierDetection:
      baseEjectionTime: 1m
      consecutive5xxErrors: 1
      interval: 1s
```

- `maxRequestsPerConnection`: specifies the maximum number of requests per connection.
- `baseEjectionTime`: specifies the minimum ejection duration.
- `consecutive5xxErrors`: specifies the number of consecutive errors.
- `interval`: specifies the time interval for ejection analysis.

3. Enable cross-region failover.

i. In the **Basic Information** section, click **Enable Locality-Failover** on the right of **Locality-Failover**.

 **Note** If you have enabled cross-region traffic distribution, you must disable cross-region traffic distribution before you can enable cross-region failover.

ii. In the **Locality-Failover** dialog box, set the Failover parameter to `cn-hangzhou` if the **From** parameter is set to `cn-shanghai`. Set the Failover parameter to `cn-shanghai` if the **From** parameter is set to `cn-hangzhou`. Then, click **Submit**.

4. Run the following command to request the BookInfo application 10 times and record the number of successful routes to the v2 reviews service:

```
for ((i=1;i<=10;i++));do curl http://<Ingress gateway endpoint of port 80 in the ack-hangzhou cluster>/productpage 2>&1|grep full.stars;done|wc -l
```

Expected output:

```
20
```

You can find that 10 access requests are made and 20 rows of results are returned. This is because a two-row result that contains `full stars` is returned each time a route to the v2 reviews service succeeds. This indicates that all 10 requests are routed to the v2 reviews service in the ack-shanghai cluster, and the cross-region failover is successful.