ALIBABA CLOUD

Alibaba Cloud

Enterprise Distributed Application Service (EDAS)

Microservice Governance

Document Version: 20220708

Alibaba Cloud

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**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Spring Cloud service governance

1.1. Gracefully unpublish Spring Cloud applications

G111raceful unpublishing is imperceptible to the consumers of your online application when you restart or unpublish the application. Graceful unpublishing ensures business performance and continuity. By default, Enterprise Distributed Application Service (EDAS) supports the graceful unpublishing of Spring Cloud applications. You do not need to configure applications or perform operations in the EDAS console.

Reasons for graceful unpublishing

Graceful unpublishing ensures the normal processing of consumer service requests during the period from when applications are stopped to when services are recovered. The most secure and reliable solution is to update your application when no service requests exist. However, service requests exist even when the application is unpublished.

A traditional solution is to manually perform the following steps: (1) Manually remove traffic. (2) Stop your application. (3) Update your application and then restart the application. In this case, users are not notified about changes to the system. This applies to the update process and related manual operations.

An innovative solution is to use an automated mechanism at the container or framework level. This mechanism can be used to automatically remove traffic and process received requests. This makes the update process imperceptible to your business and improves the O&M efficiency. This mechanism is called graceful unpublishing.

Advantages of graceful unpublishing provided by EDAS

For open source Spring Cloud, graceful unpublishing can be implemented by using ShutDownHook, Spring Boot Actuator, or Ribbon. However, this requires further development efforts, and some service registries may cause temporary traffic loss.

EDAS integrates graceful unpublishing into the release process so that graceful unpublishing is automatically implemented when you stop, deploy, roll back, scale in, and reset applications. Compared with solutions that are provided by open source Spring Cloud, graceful unpublishing provided by EDAS has the advantages that are described in the following table.

Item	Open source Spring Cloud	EDAS
Version	When you call ServiceRegistryEndpoint, you must use Spring Boot Actuator and update it to the version that can be adapted to the version of Spring Cloud.	EDAS supports Spring Cloud Dalston or later. You do not need to perform extra operations.

Item	Open source Spring Cloud	EDAS
Service registries and traffic loss	Open source Spring Cloud depends on registries for service discovery. Some service registries may cause traffic loss. • If ZooKeeper is used, no traffic loss occurs. • If Eureka is used, traffic loss may occur for 3 seconds. • If Nacos is used, traffic loss may occur for up to 10 seconds because of the cache on clients.	EDAS does not depend on registries for service discovery. No traffic loss occurs.
Scenario	If you want to unpublish an application that is deployed in an Elastic Compute Service (ECS) cluster, you must view the change details of the application in the EDAS console. If you want to unpublish an application that is deployed in a Kubernetes cluster, you can use the preStop interface. However, you can configure only one action for the interface.	You can gracefully unpublish applications that are deployed in ECS clusters and Kubernetes clusters. The operations on and the configurations of the applications are not affected.
Cache on clients	You must configure an appropriate time range for Ribbon to refresh cache on clients. If the time range is too long, traffic loss occurs. If the time range is too short, service performance is affected.	When you unpublish applications, EDAS provides an enhanced mechanism for Ribbon to refresh cache. EDAS automatically refreshes the cache based on a reactive response method. You do not need to manually refresh the cache.

Check whether graceful unpublishing takes effect

You can check whether graceful unpublishing takes effect for applications based on your actual business. EDAS also provides two application demos. You can use these demos to check whether graceful unpublishing takes effect in a Kubernetes cluster.

You can perform the following steps to check whether graceful unpublishing takes effect:

1. Download application demos Provider and Consumer.
2. Deploy Provider and Consumer to a Kubernetes cluster.

   Provider has two instances deployed, and Consumer has one instance deployed. For more information, see Overview.
3. View the call status of Provider.

i. Log on to the pod where Consumer is deployed and run the following commands to continuously access the services of Provider:

```
#!/usr/bin/env bash
while true
do
    echo `curl -s -XGET http://localhost:18091/user/rest`
done
```

ii. View the response of the calls.

```
[root@sc-consumer-group-1-1-65fdddf668-s8ssk admin]# sh a.sh
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:23
Hello from [18084]172.20.0.223! 2020-03-23 10:44:23
Hello from [18084]172.20.0.221! 2020-03-23 10:44:23
Hello from [18084]172.20.0.223! 2020-03-23 10:44:23
Hello from [18084]172.20.0.223! 2020-03-23 10:44:23
```

The response shows that Consumer randomly accesses two instances of Provider. The IP addresses of the instances are 172.20.0.221 and 172.20.0.223.

> **Notice** Do not close the response window.

4. Scale in one instance from Provider and restart the instance. For more information, see Scale out and scale in an application.
5. View the response again to check whether graceful unpublishing takes effect.

```
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
```

View the call status of Consumer to check whether graceful unpublishing takes effect. View the logs of Consumer. The logs show that no exceptions occur, and the instance unavailability is imperceptible to Consumer.

The response shows that Consumer accesses the remaining instance of Provider. The IP address of the instance is 172.20.0.221. When Consumer accesses the remaining instance, no exceptions occur, and Consumer is not affected.

1.2. Publish Spring Cloud applications by using canary releases

1.2.1. Use the EDAS console to implement canary releases of applications in Kubernetes clusters

For Spring Cloud or Dubbo microservice-oriented applications that are deployed in a Kubernetes cluster, you can implement a canary release. The canary release allows you to verify a new application version on a small number of instances. If the verification is successful, you can update the application on all of your instances to a new version. This makes the update secure.

Limits

- High-Speed Service Framework (HSF) applications: Canary release is not supported.
- Dubbo applications: You can implement canary releases of Dubbo applications without limits.
- Spring Cloud applications: If you use Deployment.Metadata.Name or Deployment.Metadata.Uid to configure specific features of an application, do not implement a canary release of the application. Otherwise, the native features of the application may be abnormal after the canary release.

Procedure

1.
2. In the left-side navigation pane, click **Applications**. In the top navigation bar, select a region. In the upper part of the Applications page, select a microservice namespace.
3. In the left-side navigation pane, click **Applications**. In the top navigation bar, select a region. In the upper part of the Applications page, select a microservice namespace.
4. On the **Applications** page, select **Container Service or Serverless Kubernetes Cluster** from the **Cluster Type** drop-down list. Then, click the name of the application that you want to deploy.
5. In the upper-right corner of the **Application Overview** page, choose **Deploy > Deploy**.
6. In the **Canary Release (Phased)** section of the **Select Deployment Mode** page, click **Start Deployment** in the upper-right corner.
7. On the **Canary Release (Phased)** page, set the deployment parameters, release policy, and canary release rules. Then, click **OK**.
   i. Set the deployment parameters.

      Deployment parameters

Parameter	Description
Configure Image (applicable to only applications that are deployed by using images)	You can update the version of an image, but cannot change the image of an application.
Application Runtime Environment (applicable to applications that are deployed by using JAR packages or WAR packages)	The value must be the same as that used for the previous deployment. ■ JAR package: The application runtime environment is **Standard Java Application Runtime Environment**. You cannot change the type of the application runtime environment. ■ WAR package: The application runtime environment is **Apache Tomcat**. You cannot change the type of the application runtime environment. However, you can change the version of Apache Tomcat as needed.
Java Environment (applicable to applications that are deployed by using JAR packages or WAR packages)	Select a value from the drop-down list as needed.
Current Environment	The current runtime environment of the application. The current runtime environment is displayed only if the application is deployed by using JAR packages or WAR packages. Enterprise Distributed Application Service (EDAS) automatically upgrades the Java environment or application runtime environment of your application to the latest version.
File Uploading Method (applicable to applications that are deployed by using JAR packages or WAR packages)	The type of the deployment package must be the same as that used for the previous deployment. You can use a WAR package or a JAR package. This parameter value cannot be changed. Set the parameter based on your requirements. You can select **Upload Package** and upload a JAR or WAR package. You can also select **Package Address** and specify the address of a JAR or WAR package.
Version (applicable to applications that are deployed by using JAR packages or WAR packages)	The version number of the deployment package. You can use a timestamp as the version number.
Time Zone (applicable to applications that are deployed by using JAR packages or WAR packages)	The time zone for the application. Select a value from the drop-down list as needed.
Service Registration and Discovery	The O&M method of your service registry. For more information, see Select an O&M method for your service registry.

ii. In the **Release Policy** section, set the parameters for the release policy.

Parameters in the Release Policy section

Parameter	Description
Number of Instances for Canary Release	The number of application instances released in the first batch. The current number of instances for the application appears on the right side. The number of instances for the canary release cannot exceed 50% of the total number of instances. This makes the application stable. **Note** After the canary release is implemented, you must manually release the remaining batches.
Remaining Batches	After the release of the first batch is complete, the application is deployed to the remaining application instances based on the specified batches.
Batch Mode	The following processing methods are supported: ■ **Automatic**: automatically releases applications in batches based on the interval specified by the **Interval** parameter. **Interval**: the interval for releasing the remaining batches in minutes ■ **Manual**: manually triggers the release of the next batch. **Note** The **Batch Mode** parameter is available only if the value of the Remaining Batches parameter is greater than 1.
Deployment Interval Between Batches	If the number of instances in each batch is greater than 1, the application is deployed to the application instances at the specified interval. Unit: seconds.

The **Publish Policy Configuration** section on the right side shows the procedure for the canary release based on the configuration.

iii. Set canary release rules.

EDAS supports **Canary Release by Content** and **Canary Release by Ratio**.

Parameters for canary release rules

Tab	Parameter	Description
Canary Release by Content	**Protocol Type**	■ **Spring Cloud**: The **Path** parameter is required. ■ **Dubbo**: The **Select Service** and **Method** parameters are required.
	Conditional Mode	Select **Meet All Following Conditions** or **Meet Any of Following Conditions**.
	Conditions	■ **Spring Cloud**: Set the parameters based on **Cookie**, **Header**, or **Parameter**. ■ **Dubbo**: Set the **Parameter** and **Expression for Getting Parameter Values** parameters based on the actual values of your application.
Canary Release by Ratio	**Traffic Ratio**	Traffic is forwarded to the current instance group for the canary release based on the specified value.

> **Note** Click **Create Inbound Traffic Rule** to create multiple inbound traffic rules that can take effect at the same time.

iv. (Optional)Configure the advanced settings.

After the canary release is started, EDAS deploys the new application version to the specified instance group. The **Change List** page displays the deployment progress and status.

> **Note** You can check whether the traffic is distributed as expected.

8. After the traffic for the canary release is verified, click **Start Next Batch** on the right side of the **Change List** page. Complete the release of the subsequent batches.

   If problems are found during the verification process, you can click **Roll Back** in the upper-right corner of the **Change List** page. In the message that appears, click **OK**.

Verify the results

After the canary release is complete, check whether the **deployment** package is of the new version on the **Application Overview** page.

1.2.2. Release an application in canary mode in an ECS cluster in the EDAS console

To update a Spring Cloud or Dubbo microservice-oriented application that is deployed in an Elastic Compute Service (ECS) cluster, you can implement a canary release to verify the new version on a small number of instances. If the verification is successful, you can update the application on all instances.

Prerequisites

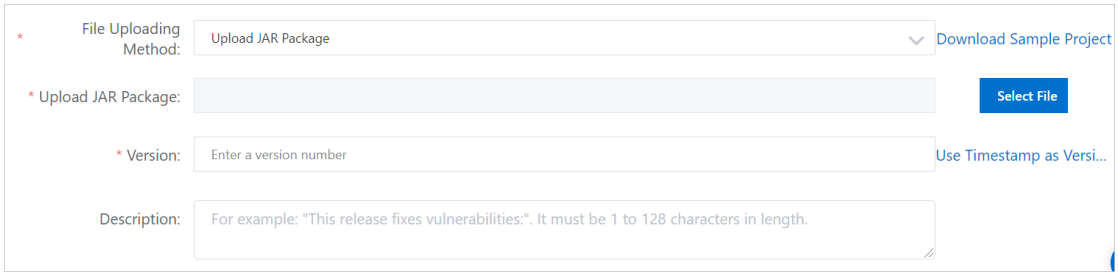
Before you implement a canary release, make sure that the application contains at least two instance groups and at least two groups contain instances. For more information about how to create instance groups and add ECS instances to the groups, see Manage instance groups for an application deployed in an ECS cluster in the EDAS console.

Limits

- High-Speed Service Framework (HSF) applications: Canary release is not supported.
- Dubbo applications: You can implement canary releases of Dubbo applications without limits.
- Spring Cloud applications: If you use Deployment.Metadata.Name or Deployment.Metadata.Uid to configure specific features of an application, do not implement a canary release of the application. Otherwise, the native features of the application may be abnormal after the canary release.

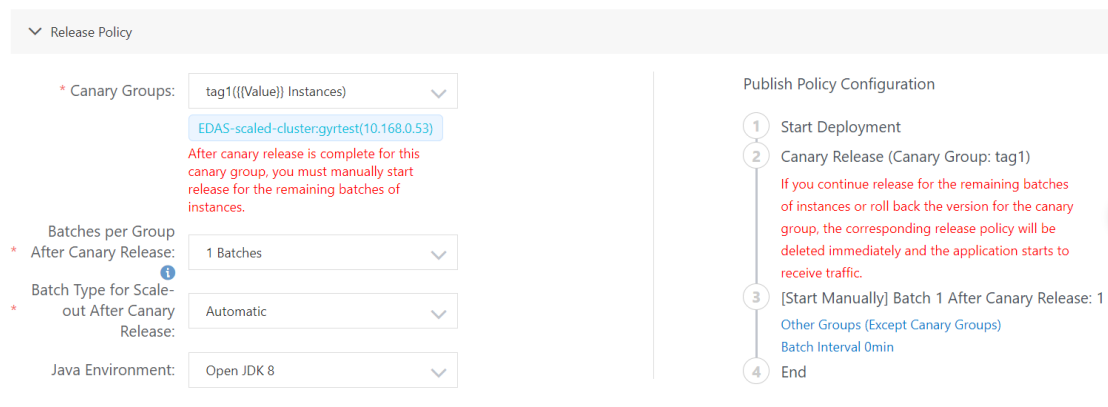
Procedure

1.
2. In the left-side navigation pane, click **Applications**.
3. In the top navigation bar, select a region. On the **Applications** page, select a microservice namespace and click the name of the application for which you want to implement a canary release.
4. On the Basic Information page, click **Deploy Application** in the upper-right corner.
5. On the **Select Deployment Mode** page, click **Start Deployment** in the upper-right corner of the **Canary Release (Phased)** section.
6. On the **Canary Release** page, upload the deployment package of the new application version, set the canary release policy and rules, and then click **OK**.
   i. Upload the deployment package of the new application version.



   ii. In the **Release Policy** section, set the parameters for the release policy.

   The **Publish Policy Configuration** section on the right side shows the procedure for the canary release based on the configuration.



The following table describes the parameters for configuring the release policy.

Parameter	Description
Canary Groups	The instance group for the canary release.
Batches per Group After Canary Release	After the canary release for the specified instance group is complete, the new version is deployed to application instances in other groups based on the preset batches. ■ If all groups are selected, the new version is deployed to the instances in each group based on the selected batch number. If the number of instances in a group is less than the selected batch number, the new version is deployed to the instances in the group based on the number of instances. ■ If you have specified a group, the new version is deployed to the instances in the specified group based on the selected batch number.
Batch Type for Scale-out After Canary Release	If you set the **Batches per Group After Canary Release** parameter to a value of at least 2, you must set this parameter. Valid values: *Automatic* and *Manual*. ■ *Automatic*: automatically deploys the new version to instances in batches based on the release interval. You must set the **Wait Time Before Next Batch** parameter. Valid values of **Wait Time Before Next Batch** are *Do Not Wait*, 1 Minutes, 2 Minutes, 3 Minutes, 4 Minutes, and 5 Minutes. ■ *Manual*: manually triggers the release of the next batch.

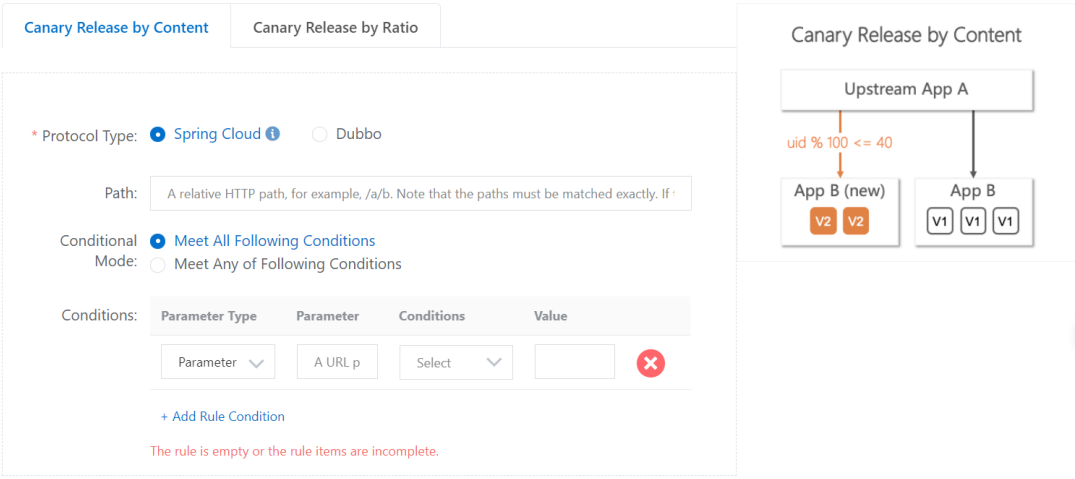
Parameter	Description
Java Environment	The runtime environment of the application. Select a runtime environment as needed.

iii. Configure canary release rules.

Enterprise Distributed Application Service (EDAS) supports **Canary Release by Content** and **Canary Release by Ratio**.

- **Canary Release by Content**: Click **Create Inbound Traffic Rule** and create a rule for inbound traffic.

> **Note** You can create multiple inbound traffic rules.



The following table describes the parameters for creating an inbound traffic rule on the Canary Release by Content tab.

Parameter	Description
Protocol Type	Valid values: *Spring Cloud* and *Dubbo*. Select one option based on the actual situation of the application. ■ *Spring Cloud*: The **Path** parameter is required. ■ *Dubbo*: The **Select Service** and **Method** parameters are required.
Conditional Mode	Select **Meet All Following Conditions** or **Meet Any of Following Conditions**.

Parameter	Description
Conditions	The conditions for *Spring Cloud* and *Dubbo* are different. Three methods are available: Cookie, Header, and Parameter. Set the parameters as needed. ■ *Spring Cloud*: *Cookie*, *Header*, and *Parameter* are available. Set the parameters as needed. ■ *Dubbo*: Set the **Parameter** and **Expression for Getting Parameter Values** parameters based on the actual values of your application.

- **Canary Release by Ratio**: Set the **Traffic Ratio** paramerter. Traffic is forwarded to the current instance group for the canary release based on this value.

After the canary release is started, EDAS deploys the new application version to the specified instance group. On the **Basic Information** page, the message `A change process is ongoing for this application. The application is in Executing state` appears. Click **View Details**. On the **Change Details** page that appears, view the deployment progress and status.

Stop a change: `The application is in the canary release state and this change has been stopped. Please roll back the application before you perform other operations.`

7. You can check whether the traffic is distributed as expected. For more information, see Monitor canary traffic.
8. After the traffic verification is complete, click **Start Next Batch** on the **Change Details** page. Complete the subsequent batch release.

   If an issue occurs during the verification process, click **Stop Change** in the upper-right corner of the **Change Details** page. After the change is stopped, on the **Basic Information** page, the message `The application is in the canary release state and this change has been stopped. Please roll back the application before you perform other operations.`

Verify the result

After the canary release is complete, check whether the **deployment package** is of the new version on the **Basic Information** page. On the **Instance Information** page, check whether the instances are in the **Normal** state.

1.3. Query Spring Cloud services

You can log on to the Enterprise Distributed Application Service (EDAS) console to query the service list and service details of Spring Cloud applications that are deployed in EDAS.

Limits

You can switch between the old and new versions of the **Service Query** page.

- In the new version, the system uses the EDAS agent to query services in the EDAS registry, Microservice Engine (MSE)-hosted registry, and self-managed registries, including ZooKeeper, Nacos, Eureka, and Consul.

- In the new version, you can query the services of Spring Cloud Edgware and later versions and the services in all registries.
- In the old version, you can query the services of Spring Cloud Dalston and later versions that are registered with the EDAS registry.
- In the old version, you can query only services in the EDAS registry.
- Outbound TCP connections over ports 8442, 8443, and 8883 must be allowed in the security group of your server. For more information about how to allow outbound connections in Elastic Compute Service (ECS), see Add a security group rule.

View services

1.
2. In the left-side navigation pane, choose **Microservices Governance > Spring Cloud**.
3. In the left-side navigation tree of **Spring Cloud**, click **Service Query**.
4. In the top navigation bar, select a region. On the **Service Query** page, select a microservice namespace to view the **Spring Cloud** services of your account.

   You can view the following information about a Spring Cloud service: the service name, application name, and number of instances.

   If a large number of services exist, you can filter services by service name, IP address, or application name. Filter keywords are not case-sensitive. The value of **IP** varies between ECS and Kubernetes clusters.

   - ECS cluster: The value is the IP address of an ECS instance.
   - Kubernetes cluster: The value is the IP address of a pod.

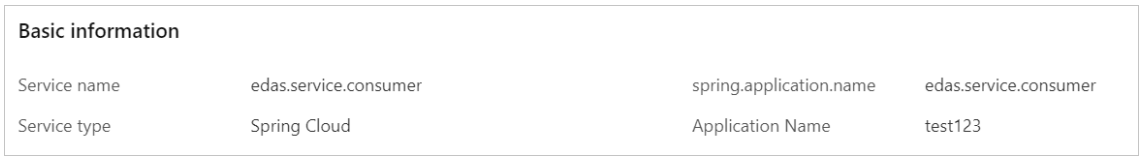
   > **Note** If you can query services of your applications on the old Service Query page but not on the new Service Query page, troubleshoot the problem by performing the following steps:
   >
   > i. The new version of the Service Query page was released at 00:00 of January 20, 2020. You must restart your applications after this point in time so that they can be automatically mounted with the latest EDAS agent. Therefore, you must restart your applications before you query services on the new Service Query page.
   >
   > ii. Check whether the microservice framework version is supported. For more information about the supported versions, see Limits.

View service details

1.
2. In the left-side navigation pane, choose **Microservices Governance > Spring Cloud**.
3. In the navigation tree of the **Spring Cloud** page, click **Service Query**.
4. In the top navigation bar, select a region. On the **Service Query** page, select a microservice namespace. Then, click a service name in the service list.
5. In the **Service Details** panel, view the details of the service.

   The **Service Details** panel contains the following sections: **Basic Information**, **Service invocation relationship**, and **Metadata**. The **Metadata** section contains **Interface metadata** and **Metadata Metadata**.

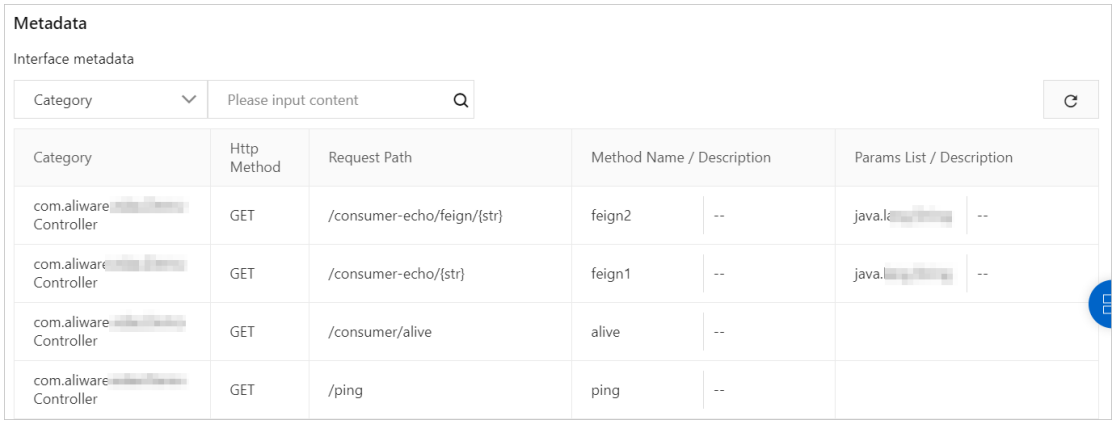
- **Basic Information**



- **Service Invocation Relationship**

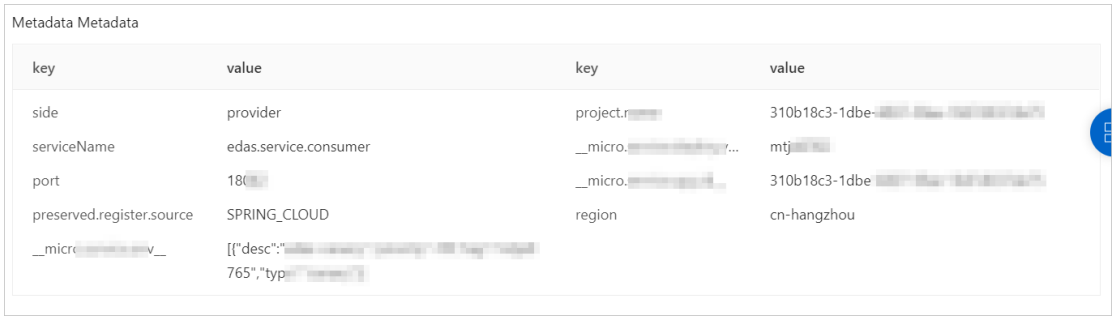
The Service Invocation Relationship section provides the **Service Provider** and **Service Consumer** tabs, which list information such as **IP** and **Port**.

- **Metadata**
  - **Interface Metadata**



The **Interface Metadata** section provides information about the class to which the service belongs, the request method, and the interface of the service.

  - **Metadata Metadata**



The **Metadata Metadata** section provides the metadata of the service and the EDAS-provided metadata for implementing microservice capabilities.

1.4. Query the traces of Spring Cloud service

You can log on to the Enterprise Distributed Application Service (EDAS) console to query the traces of Spring Cloud services that are deployed in EDAS.

EDAS is integrated with Application Real-Time Monitoring Service (ARMS). You can use ARMS to query service traces and holographic troubleshooting events.

1.5. Ensure the availability of Spring Cloud applications by using outlier instance removal

In a microservice framework, service calls are affected if consumers cannot detect the exceptions on the application instances of a provider. This further affects the performance and even availability of the services provided by the consumers. The outlier ejection feature monitors the availability of application instances and dynamically adjusts the instances. This ensures successful service calls and improves the service stability and quality of service (QoS).

Context

A system includes Applications A, B, C, and D, where Application A calls Applications B, C, and D. If the instances of Application B, C, or D become abnormal and Application A does not identify the abnormal instances, a part of calls initiated by Application A fail. Application B has one abnormal instance, and Applications C and D each have two abnormal instances. If Applications B, C, and D have a large number of abnormal instances, the service performance and availability of Application A may be affected.

To ensure the service performance and availability of Application A, you can configure an outlier application removal policy. After the policy is configured, Enterprise Distributed Application Service (EDAS) can monitor the instance status of Applications B, C, and D, and dynamically add or remove instances to ensure successful service calls.

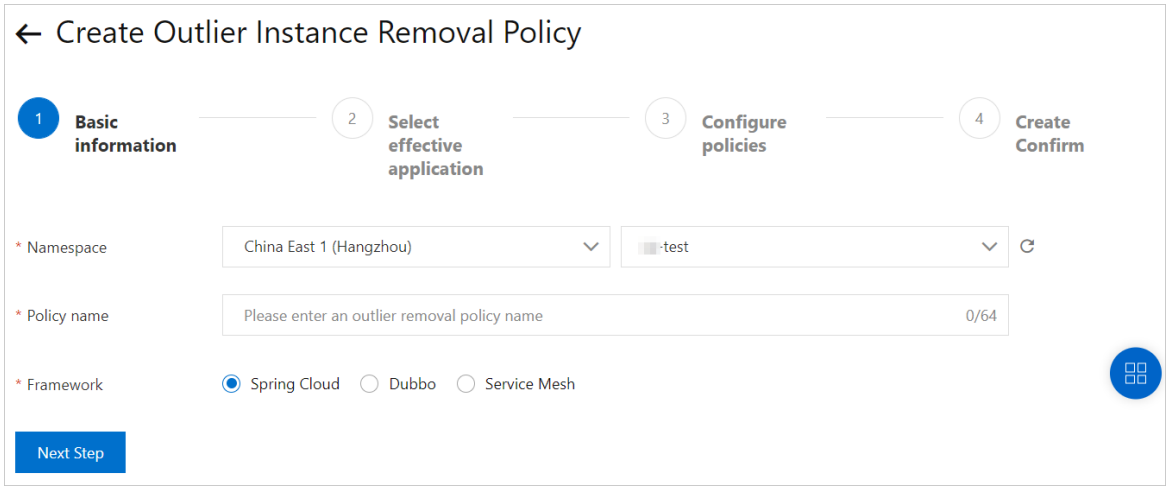
The following list describes the process of outlier instance removal:

1. EDAS detects whether Applications B, C, and D have abnormal instances. Then, EDAS determines whether to remove the abnormal instances from the applications based on the configured **Upper limit of instance removal ratio** parameter.
2. EDAS does not distribute the call requests of Application A to the removed instances.
3. EDAS detects whether the abnormal instances are recovered based on the configured **Recovery detection unit time** parameter.
4. The detection interval is proportional to the number of detection times and linearly increases by the value of the **Recovery detection unit time** parameter, which is 0.5 minutes by default. If the value of the **Maximum cumulative number of times not restored** parameter is reached, EDAS detects whether the abnormal instances are recovered at the maximum detection interval.
5. After the abnormal instances are recovered, they are added to the instance lists of the applications to continue processing call requests. The detection interval is reset to the value of the **Recovery detection unit time** parameter, such as 0.5 minutes.

> **Note**
>
> - If the provider has a large number of abnormal instances and the ratio of the abnormal instances exceeds the value of the Upper limit of instance removal ratio parameter, the number of actually removed instances equals the configured upper limit.
> - If the provider has only one instance available, this instance is not removed even if the error rate exceeds the configured limit.

Create an outlier instance removal policy

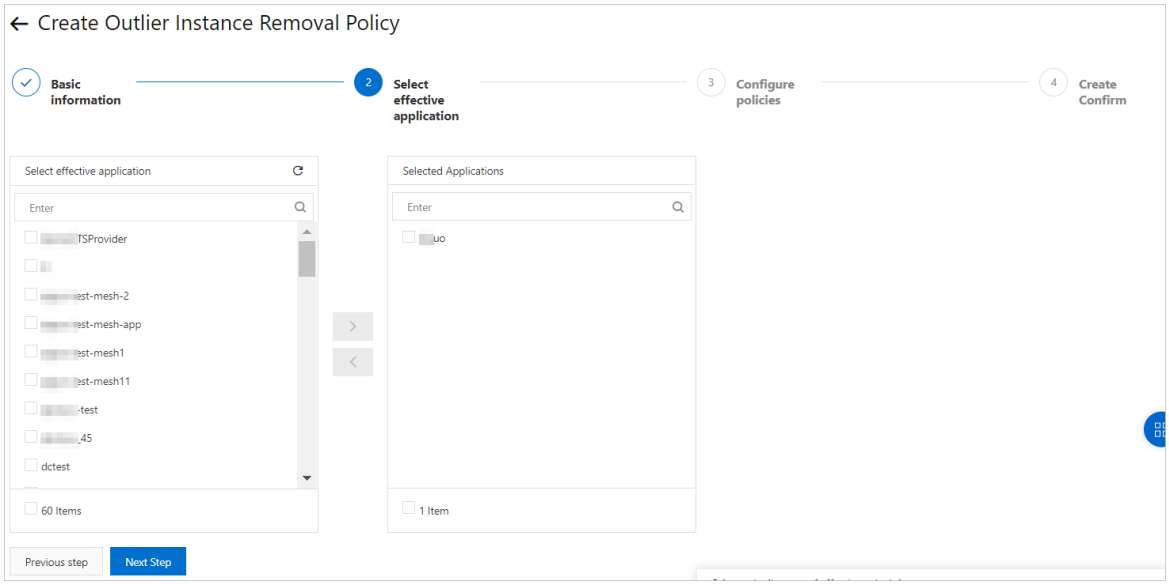
1.
2. In the left-side navigation pane, choose **Microservices Governance > Spring Cloud**.
3. Click **Outlier Instance Removal**.
4. On the **Outlier Instance Removal** page, select a region and a microservice namespace. Then, click **Create an outlier removal policy**.
5. In the **Basic information** step on the **Create Outlier Instance Removal Policy** page, configure the parameters and click **Next Step**.



The following table describes the parameters in the **Basic information** step.

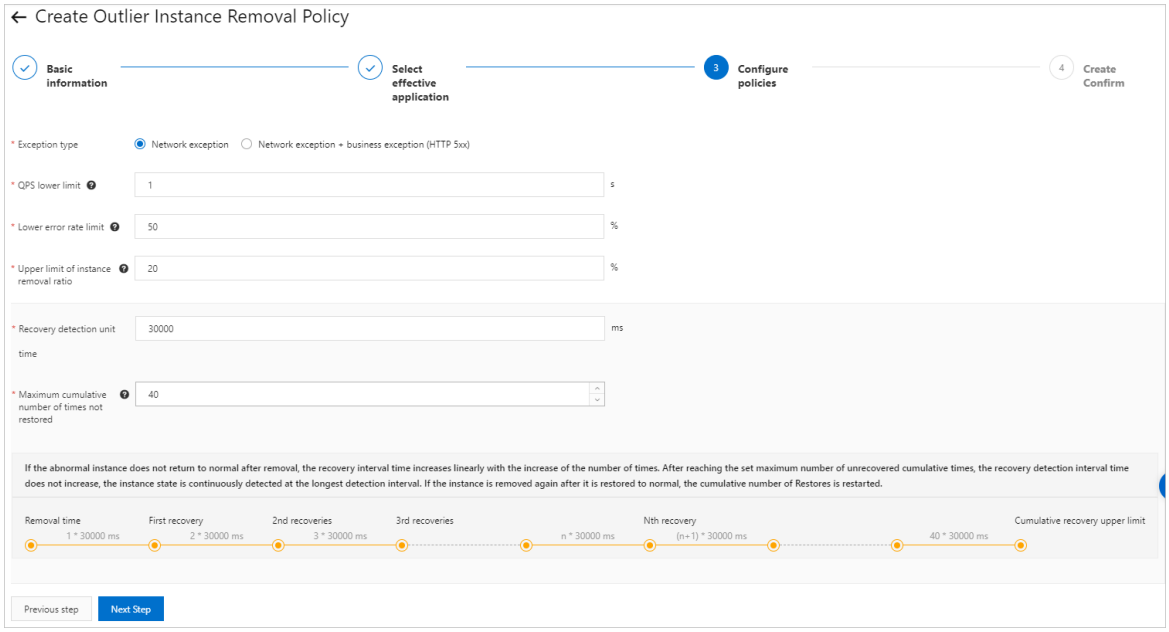
Parameter	Description
Microservice Space	Select a region and a namespace from the drop-down lists.
Policy name	Enter a name for the policy. The name can be up to 64 characters in length.
Framework	Select **Spring Cloud**.

6. In the **Select effective application** step on the **Create Outlier Instance Removal Policy** page, select the required application and click the **>** icon to add the application to Selected Applications. Then, click **Next Step**.



After the application is selected, all the abnormal instances of the applications that are called by this application are removed. Call requests from this application are not distributed to the removed instances.

7. In the **Configure policies** step on the **Create Outlier Instance Removal Policy** page, configure the parameters and click **Next Step**.



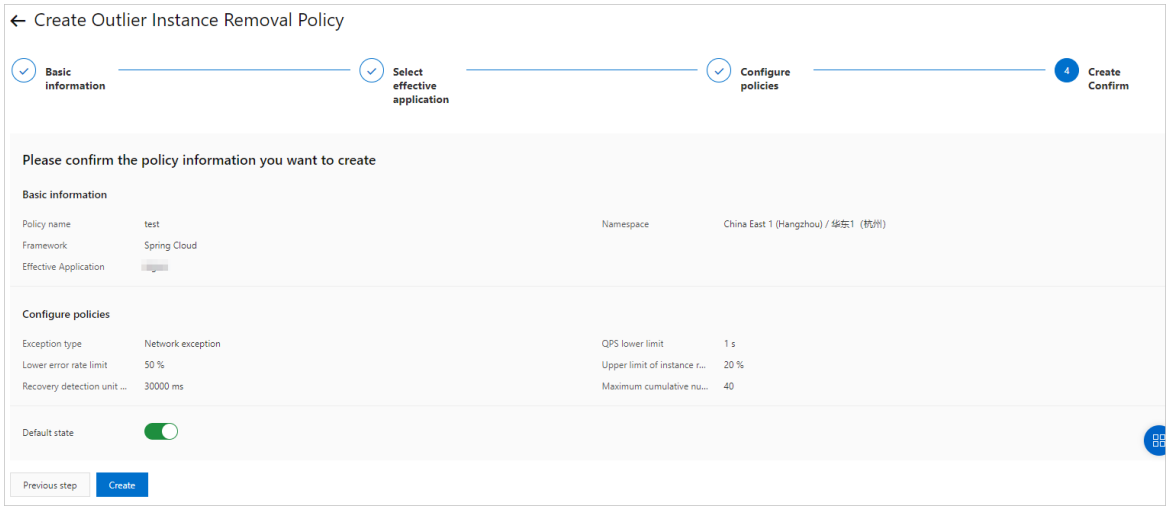
The following table describes the parameters in the **Configure policies** step.

Parameter	Description
Exception type	Select **Network exception** or **Network exception + business exception (HTTP 5xx)** based on your business requirements.

Parameter	Description
QPS lower limit	Enter the lower limit of queries per second (QPS) based on the statistical time window. The time window is 15s for applications that run Dubbo 2.7, and is 10s for applications that run other Dubbo versions and Spring Cloud applications. If the QPS in a statistical time window, 15s for example, reaches the specified lower limit, EDAS starts to collect and analyze error rate statistics.
Lower error rate limit	Enter the lower limit of the error rate. If the error rate on an instance of a called application exceeds the limit, the instance is removed. Default value: 50. For example, an instance receives 10 call requests in the statistical time window, and 6 call requests fail. The error rate is 60%. If this parameter is set to 50, the instance is removed.
Upper limit of instance removal ratio	Enter the upper limit for the proportion of abnormal instances that can be removed. If the limit is reached, no more abnormal instances are removed. For example, an application has 6 instances in total. If you set this parameter to 60, the number of instances that can be removed is 3.6, which is rounded down to the nearest integer 3. The number is calculated by using the following formula: 6 × 60%. If the calculated result is less than 1, no instances are removed.
Recovery detection unit time	Set a unit interval to detect whether abnormal instances are recovered. After abnormal instances are removed, EDAS linearly increases the detection interval by the specified unit interval with the number of detection times. Default value: 30000. Unit: ms. The default value equals 0.5 minutes.

Parameter	Description
Maximum cumulative number of times not restored	Enter the maximum number of times that EDAS detects whether an abnormal instance is recovered. After the maximum number is reached, EDAS stops increasing the detection interval. For example, an abnormal instance remains unrecovered after being detected 20 times. If you set the **Recovery detection unit time** parameter to 30000 and the **Maximum cumulative number of times not restored** parameter to 20, EDAS detects whether the instance is recovered at an interval of 10 minutes, which is calculated by using the following formula: 20 × 30000 ms. If the instance is recovered before the maximum number is reached, the detection interval is reset to the value of the **Recovery detection unit time** parameter. **Note** We recommend that you do not set the **Maximum cumulative number of times not restored** parameter to a large value. A large value can result in a long detection interval. If an instance is recovered early before a long detection interval arrives, the recovery cannot be detected at earliest opportunity. This results in low resource utilization and postponed processing of service call requests.

8. In the **Create Confirm** step on the **Create Outlier Instance Removal Policy** page, confirm the settings and click **Create**.



Verify the result

The outlier ejection feature is enabled after you configure and create an outlier ejection policy. You can go to the details page of the application for which you have configured outlier ejection to view the application monitoring information. For example, you can check whether call requests are still forwarded to abnormal instances and whether the error rate per minute for application calls is higher than the value of the **Error Rate Threshold** parameter in a topology. This way, you can check whether the outlier ejection policy takes effect.

What to do next

On the **Outlier Instance Removal** page, you can click **Edit** or **Delete** in the Operation column to manage the policies.

1.6. Implement access control on Spring Cloud applications by using service authentication

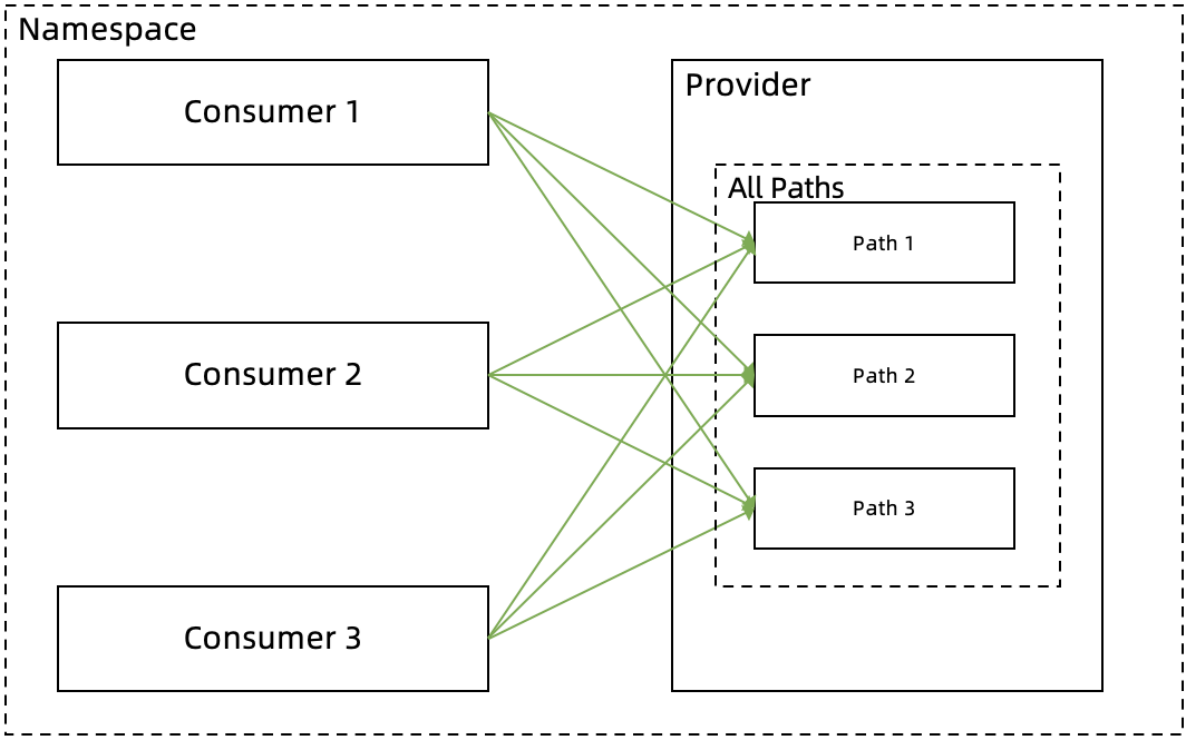
If a microservice-oriented application requires high security and you want to restrict access to it from other applications, you can authenticate the applications that call the microservice-oriented application. This ensures that only the applications that match the authentication rules can call the microservice-oriented application.

Context

This topic uses an example to introduce scenarios where Spring Cloud service authentication is performed.

- Do not configure service authentication

  Consumers 1, 2, and 3 and a service provider are deployed in the same namespace. By default, Consumers 1, 2, and 3 can call all the paths (Paths 1, 2, and 3) of the provider.

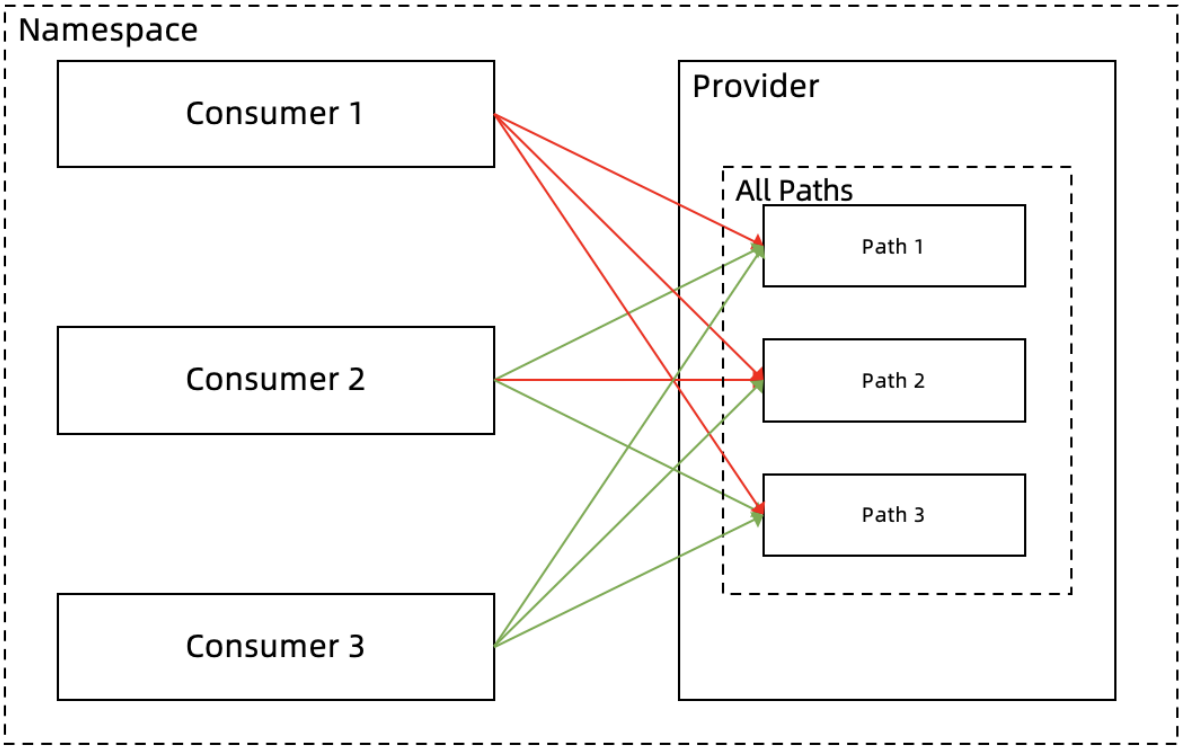


- Configure service authentication
  - Configure an authentication rule for all the paths.

    You can configure an authentication rule for all the paths of the provider. For example, you can configure a blacklist for Consumer 1 to prevent it from calling the paths of the provider, and configure a whitelist for Consumers 2 and 3 to allow them to call the paths of the provider.
  - Configure an authentication rule for a specific path.

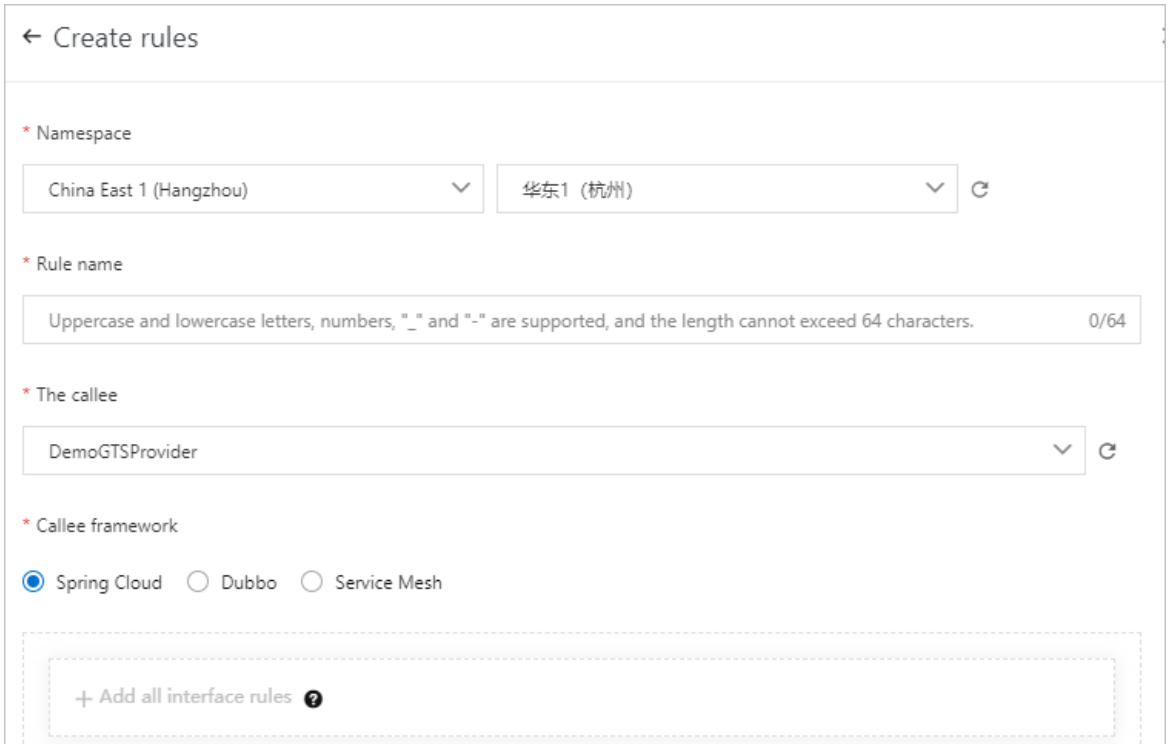
    You can also configure an authentication rule for a specific path of the provider. For example, you can configure a blacklist for Consumer 2 to prevent it from calling Path 2 of the provider because the path involves core business or core data. Then, Consumer 2 can call only Paths 1 and 3 of the provider.

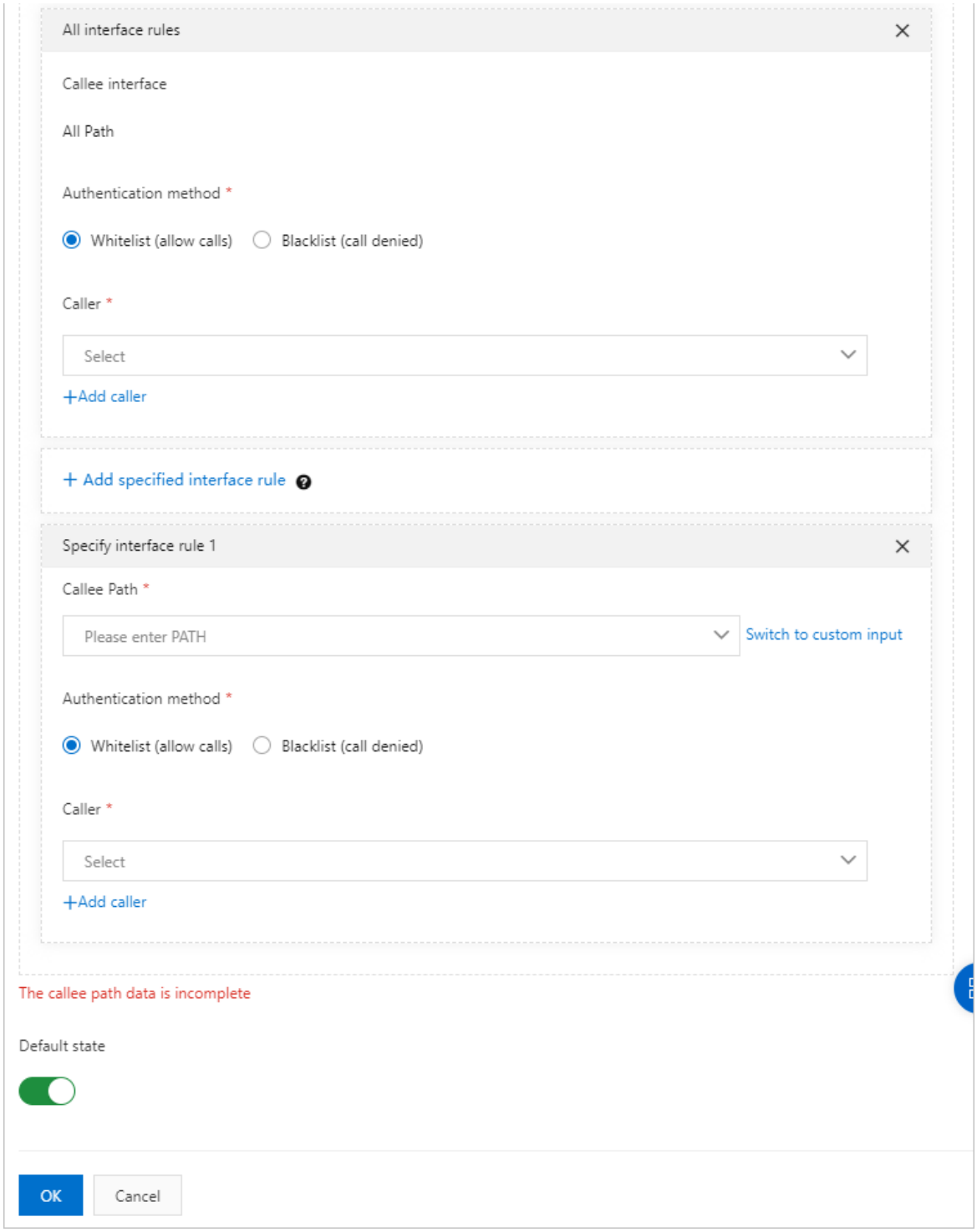
  The following figure shows the application call process after you configure the authentication rules.



Create a service authentication rule

1.
2.
3. In the left-side navigation tree of **Spring Cloud**, click **Service Authentication**.
4. On the **Service Authentication** page, click **Create rules**.
5. On the **Create rules** page, set the parameters for the service authentication rule , and click **OK**.





Parameters for the service authentication rule:

Parameter	Description
Microservice Namespaces	The region and the microservice namespace where the microservice is deployed.

Parameter	Description
Rule name	The name of the service authentication rule. The name can be a maximum of 64 characters in length, and can contain letters, digits, underscores (_), and hyphens (-).
The callee	The application to be called.
Callee framework	The framework that is used by the application. In this example, select **Spring Cloud**.
Add all interface rules **Notice** The global rule that applies to all paths. **You can create only one global rule for all paths.**	
Callee interface	The paths to which the rule applies. The value is fixed to **All Path**.
Authentication method	The service authentication method. Valid values: **Whitelist (allow calls)** and **Blacklist (call denied)**. Select an option as needed.
Caller	The caller application to be authenticated. Click **Add caller** to select multiple applications.
Add specified interface rule **Notice** The rule that applies to a specific path. **Such a rule is not appended. Instead, the rule overwrites the global rule for the paths. Exercise caution when you set this parameter.**	
Callee Path	The path of the application to be called.
Authentication method	The service authentication method. Valid values: **Whitelist (allow calls)** and **Blacklist (call denied)**. Select an option as needed.
Caller	The caller application to be authenticated. Click **Add caller** to select multiple applications.
Default State	Specifies whether to enable the rule. Valid values: ○ On: The rule is enabled immediately after you create it. This is the default value. ○ Off: The rule is not enabled after it is created. To enable the rule, find the rule on the **Service Authentication** page and click **Open** in the **Operation** column.

Verify the results

After the service authentication rule is created and enabled, check whether the rule takes effect.

What's next

After you create a service authentication rule, you can click **Edit**, **Close**, or **Open** in the Operation column to manage the rule. If the service authentication rule is no longer required, you can click **Delete** in the Operation column to delete the rule.

1.7. View the service contract and change records of a Spring Cloud application

A service contract refers to the description of microservice interfaces based on the OpenAPI specification. Microservice systems operate and run based on service contracts. After you deploy an application, you do not need to introduce dependencies to the application. You can view API information such as microservice interfaces and paths by using service contracts. You can use an easy method to query services and use features, such as service tests.

Context

A service contract includes the following three features:

- API query

  You can view important API information of a service provider or consumer by using a service contract. The information includes the method, parameter list, and return type. If you choose a Spring Cloud service, you can view the information such as the request method, request path, and class name.

- Swagger annotation parsing

  Swagger is the major contributor to the OpenAPI specification. Swagger is not the only tool that supports the OpenAPI specification. However, it is a basic standard tool that can be used to describe APIs.

  Service contracts support Swagger annotation parsing. The parsing results are displayed on the service contract page in the Enterprise Distributed Application Service (EDAS) console.

  - For Swagger 2.0, the values of annotations, such as @ApiOperation, @ApiParam, and @ApiImplicitParam, are parsed and displayed in the **Description** column.
  - For OpenAPI 3.0, the values of annotations, such as @Operation and @Parameter, are parsed and displayed in the **Description** column.

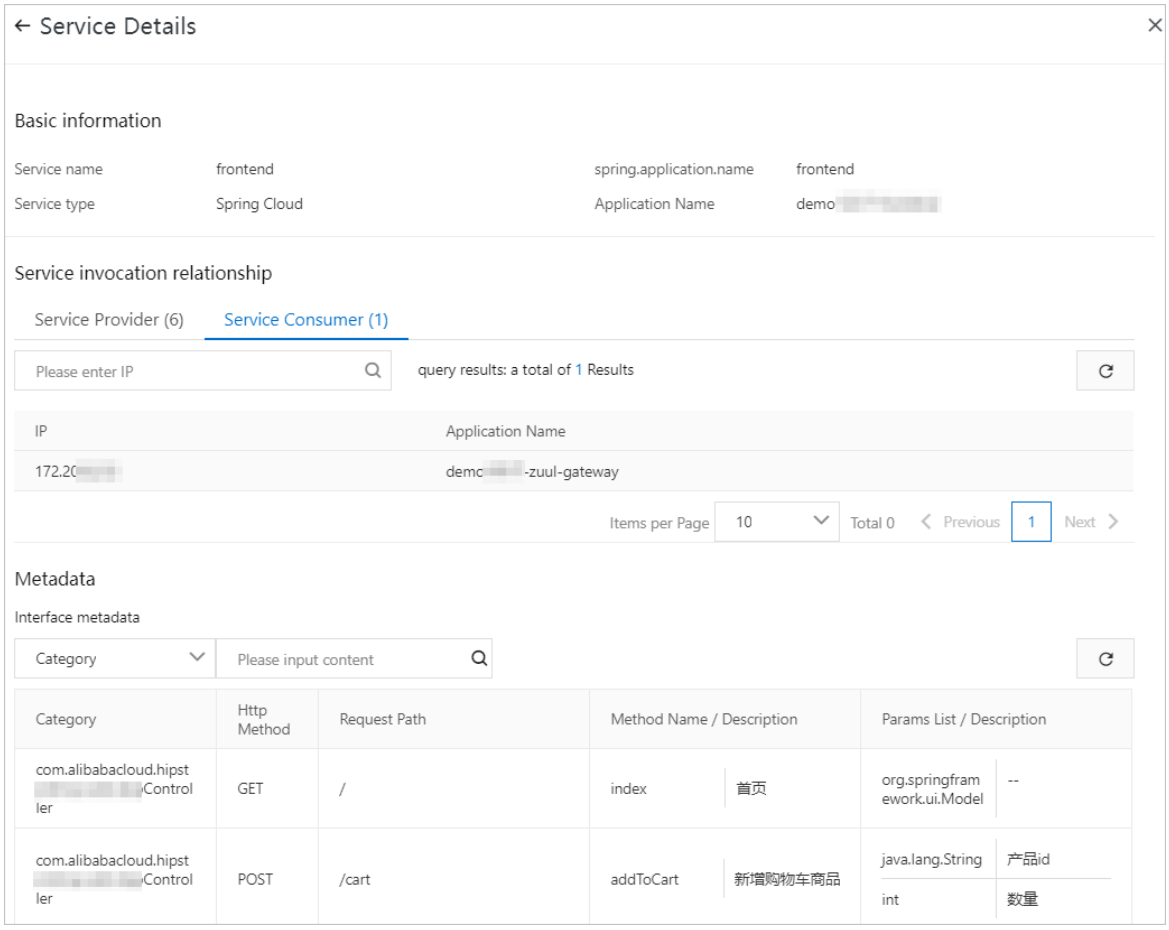
- Prerequisites of service test

  The service test feature tests a service interface or path based on the API information of the service. The API information is collected by using a service contract.

View the service contract of an application

1.
2. In the left-side navigation pane, choose **Microservices Governance > Spring Cloud**.
3. In the left-side navigation tree of **Spring Cloud**, click **Service Query**.
4. In the top navigation bar, select a region. On the Service Query page, select an option from the microservice namespace drop-down list. Then, click a service name in the service list.
5. In the **Metadata** section of the **Service Details** panel, view the API information in the interface metadata of the service.

If you use Swagger annotations, the parsed results of the annotations are displayed in the **Description** column.



View the change records of a service contract

1.
2. In the left-side navigation pane, click **Applications**. In the top navigation bar, select a region. On the Applications page, select an option from the microservice namespace drop-down list.
3. From the Cluster Type drop-down list, select **Container Service or Serverless Kubernetes Cluster** and click a specific application.
4. In the left-side navigation tree of **Application Overview**, click **Change List**.
5. On the **Change List** page, select **Deploy Application** from the **Change Type** drop-down list and click **View** in the **Actions** column.
6. On the Change List Details page, click **Click View Changes**.

   In the **Service Contract Changes** panel, you can view the change records of the service contract. The change records include the following types: **Add Methods**, **Delete Methods**, and **Modify Methods**.

1.8. Test a Spring Cloud service

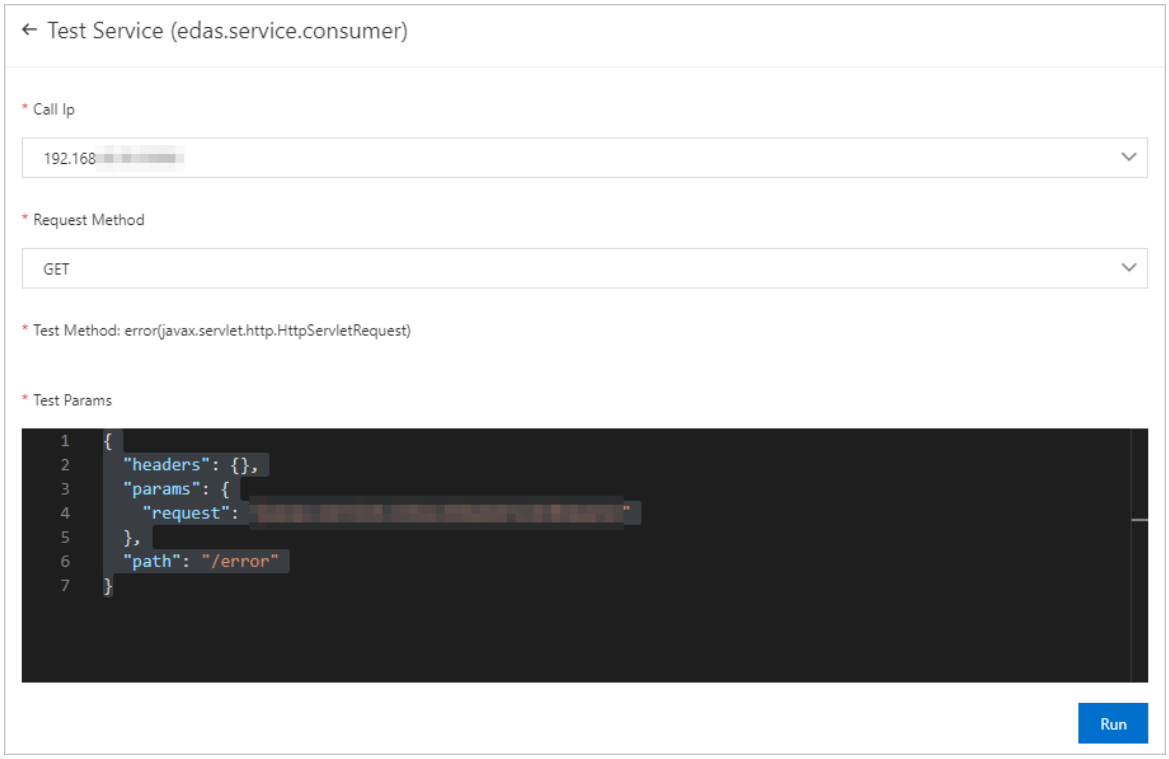
Developers or testers need to call online services to debug deployed services or query online data during the development process. The service testing feature allows you to set the parameters to call a service, initiate service calls, and obtain the results of the calls in the Enterprise Distributed Application Service (EDAS) console.

Context

- The service testing feature is in public preview. You can use this feature free of charge.
- If you test a service as a Resource Access Management (RAM) user, you must first grant the RAM user the permissions to test services in the RAM console. For more information, see Configure permissions for service testing in the RAM console.

Procedure

1.
2. In the left-side navigation pane, choose **Microservices Governance > Spring Cloud**.
3. In the left-side navigation pane of **Spring Cloud**, click **Service Testing**.
4. In the top navigation bar, select a region. On the **Service Testing - Select Service** page, select a microservice namespace from the Microservice Namespace drop-down list. Then, click the name of a specific service in the service list.
5. In the **Interface metadata** section of the **Select Test Method** panel, find the service that you want to test and click **Test** in the **Request Path** column.
6. In the **Test Service** panel, set the parameters and click **Run**.



The following table describes the parameters.

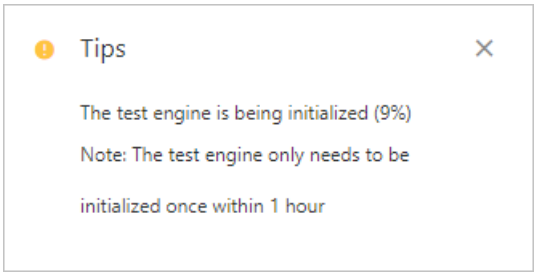
Parameter	Description

Parameter	Description
Call IP	The IP address of an instance on which the service is deployed. The instance can be an Elastic Compute Service (ECS) instance or a pod. If the service is deployed on multiple instances, you can select only the IP address of one instance.
Request Method	The request method of the class to which the service belongs. If the class contains multiple request methods, you can select only one request method.
Test Method	The test method to use. In the Test Params section, set the parameters based on the service code.

Verify the test result

In the **Result** section, you can check whether the test is successful. The following list describes the types of test results:

- **The test failed**, and the "`The test engine is being initialized.`" error message appears. When you perform a service test, the test engine requires 30 seconds to 50 seconds to initialize. Therefore, you must wait for 30 seconds to 50 seconds before you perform another test.

- **The test is successful**, and the response from the service appears.
- **The test failed**, and the response from the service appears. You can troubleshoot the issue based on the response to determine whether the issue is caused by the port, network, or code of the service.

1.9. Configure tag-based routing for a Spring Cloud application

The tag-based routing feature allows you to allocate one or more application instances to the same group by using tags. This way, you can forward traffic to application instances in specific groups. The tag-based routing feature can be used in scenarios such as multi-version development and testing, traffic adjustment for a multi-version application, and A/B testing.

Context

The tag-based routing feature is available only for applications that are deployed in Elastic Compute Service (ECS) clusters.

Scenarios

- Multi-version development and testing

  If multiple versions are developed at the same time, you must prepare a development environment for each version. The costs of development environments are high. To reduce costs, you can use tag-based routing to implement end-to-end traffic adjustment.

  End-to-end traffic adjustment is implemented based on the tag-based routing feature. End-to-end traffic adjustment allows you to route specific traffic to a specific development environment. For example, if only Application B and Application D are modified in Development Environment 1, you can create Tag 1 for the versions of the two applications in Development Environment 1, and create a tag-based routing rule. This way, when Application A calls Application B, the system checks whether the traffic meets the conditions of the tag-based routing rule. If yes, the traffic is routed to Application B V1.1 in Development Environment 1. If no, the traffic is routed to Application B V1 in the baseline environment. When Application C calls Application D, the traffic is routed to Application D V1 or Application D V1.1 based on the tag-based routing rule.

- Traffic adjustment for a multi-version application

  If multiple versions of an application run online at the same time and are deployed in different environments, you can use the tag-based routing feature to isolate the traffic that is destined for different versions in different environments. For example, you can route the traffic of flash sale orders or the traffic of orders from different channels to the special environment, and route common traffic to the common environment. This way, the traffic destined for the special environment is not routed to the common environment even if exceptions occur in the special environment, and the common environment is not affected.

- A/B testing

  Multiple versions of an application run online at the same time. To perform A/B testing on the different versions, you can use end-to-end traffic adjustment to route the traffic that is initiated from Region A, such as the China (Hangzhou) region, to Application V1, and route the traffic that is initiated from Region B, such as the China (Shanghai) region, to Application V1.1. Then, you can verify the different versions. This helps reduce risks when you publish new products or features and facilitate product innovation.

Procedure

In this example, Application A is deployed on ECS instances. Application A has a default group, and the group contains three application instances.

The following steps describe how to configure tag-based routing:

1. Create tags: Create Group 1 and Group 2 for Application A, allocate one application instance in the default group to Group 1 and one application instance to Group 2, and then create tags for Group 1 and Group 2.
2. Create tag-based routing rules: Create tag-based routing rules based on the tags of Group 1 and Group 2.

After tag-based routing is configured, when Application B calls Application A, traffic is routed to Group 1 and Group 2 based on the tag-based routing rules. Other traffic that does not meet the conditions of the tag-based routing rules is routed to the default group.

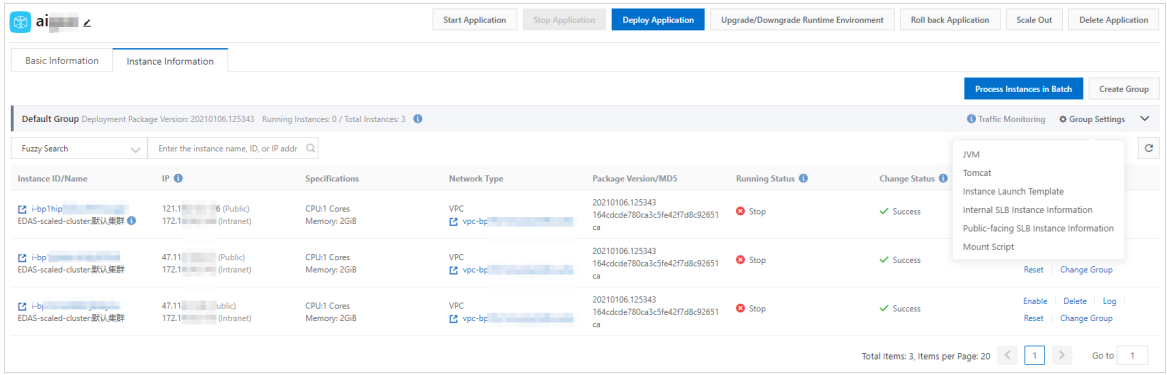
Create a tag

Create a group for an application and add an instance to the group. Then, select JVM from the Group Settings drop-down list to create a tag.

1.
2. Create a group for an application. For more information, see Create a group.
3. Add an application instance to the new group. You can allocate an application instance of the default group to the new group, or purchase a new instance. For more information, see Add an instance.

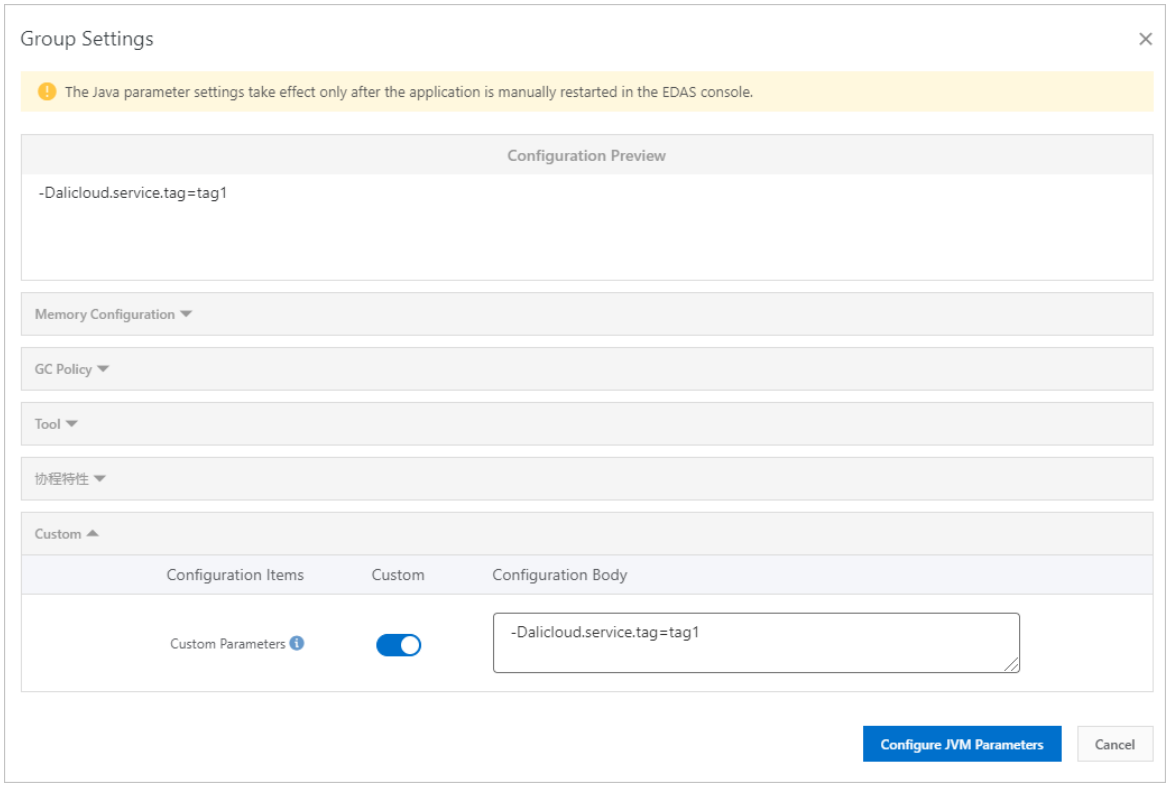
   > **Notice** After the application instance of the default group is allocated to the new group, you must restart the application instance. Otherwise, the system does not identify the tag. If you purchase a new instance for the new group, you do not need to restart the instance.

4. In the upper-right corner of the new group list, select **JVM** from the **Group Settings** drop-down list.



5. In the **Group Settings** dialog box, click **Custom**. In the **Custom** section, turn on the switch in the **Custom** column. In the **Configuration Body** field, configure a tag for the group and click **Configure JVM Parameters**.

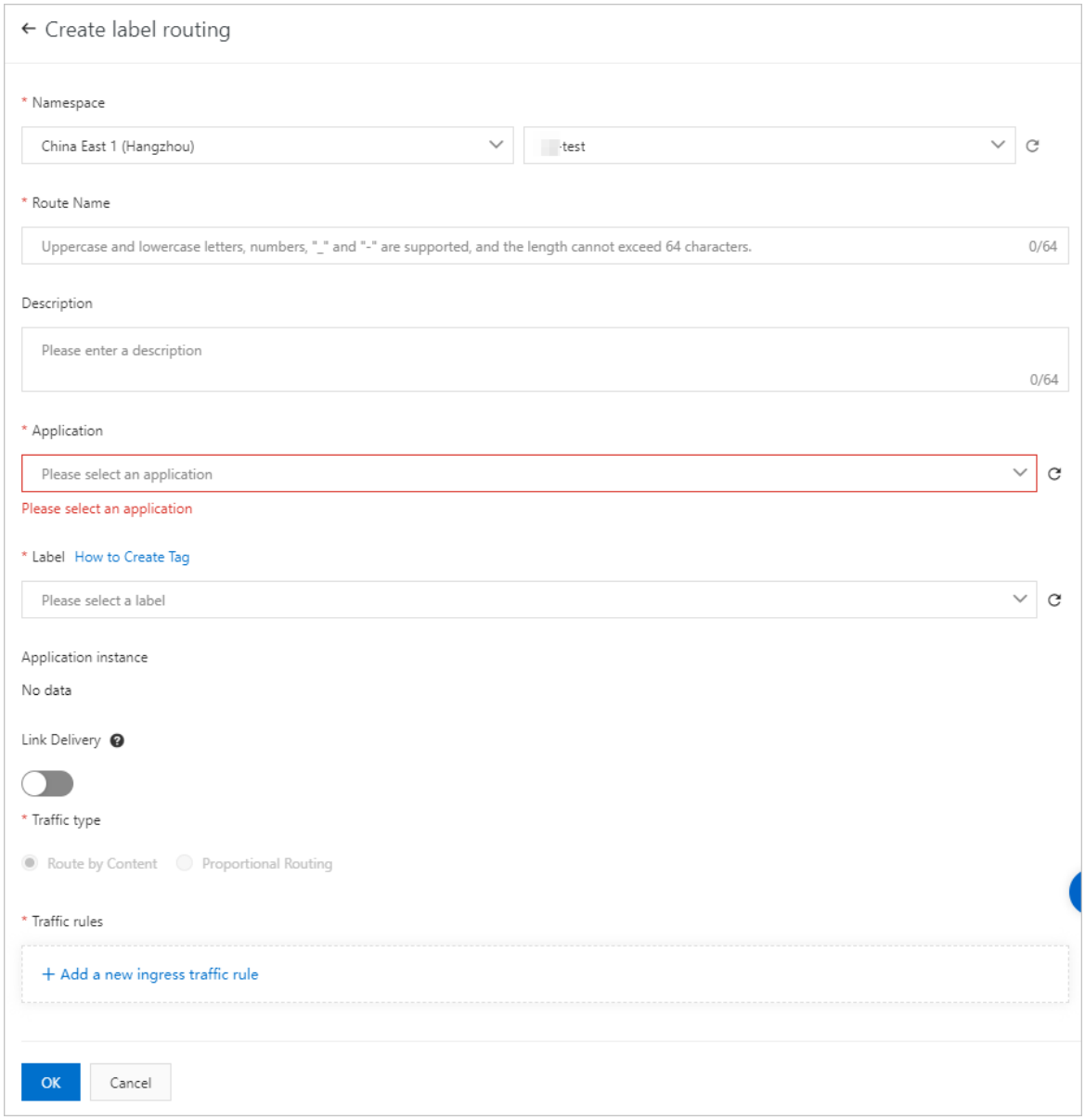
   You must configure a tag in the `Dalicloud.service.tag=tag1` format. *tag1* indicates the tag name. Set the tag name based on your business requirements.



Create a tag-based routing rule

After a tag is created, you can create a tag-based routing rule based on the tag.

1.
2. In the left-side navigation pane, choose **Microservices Governance > Spring Cloud**.
3. In the navigation tree of the **Spring Cloud** page, click **Tag-based Routing**.
4. On the **Tag-based Routing** page, select a region and a microservice namespace. Then, click **Create label routing**.
5. In the **Create label routing** panel, set the parameters and click **OK**.



The following table describes the parameters.

Parameter	Description
Namespace	Select a region and a microservice namespace based on your business requirements.
Route Name	Enter the name of the tag-based routing rule. For example, you can enter `lable-routing-group1`.
Description	Enter the description of the tag-based routing rule.
Application	Select an application from the drop-down list.

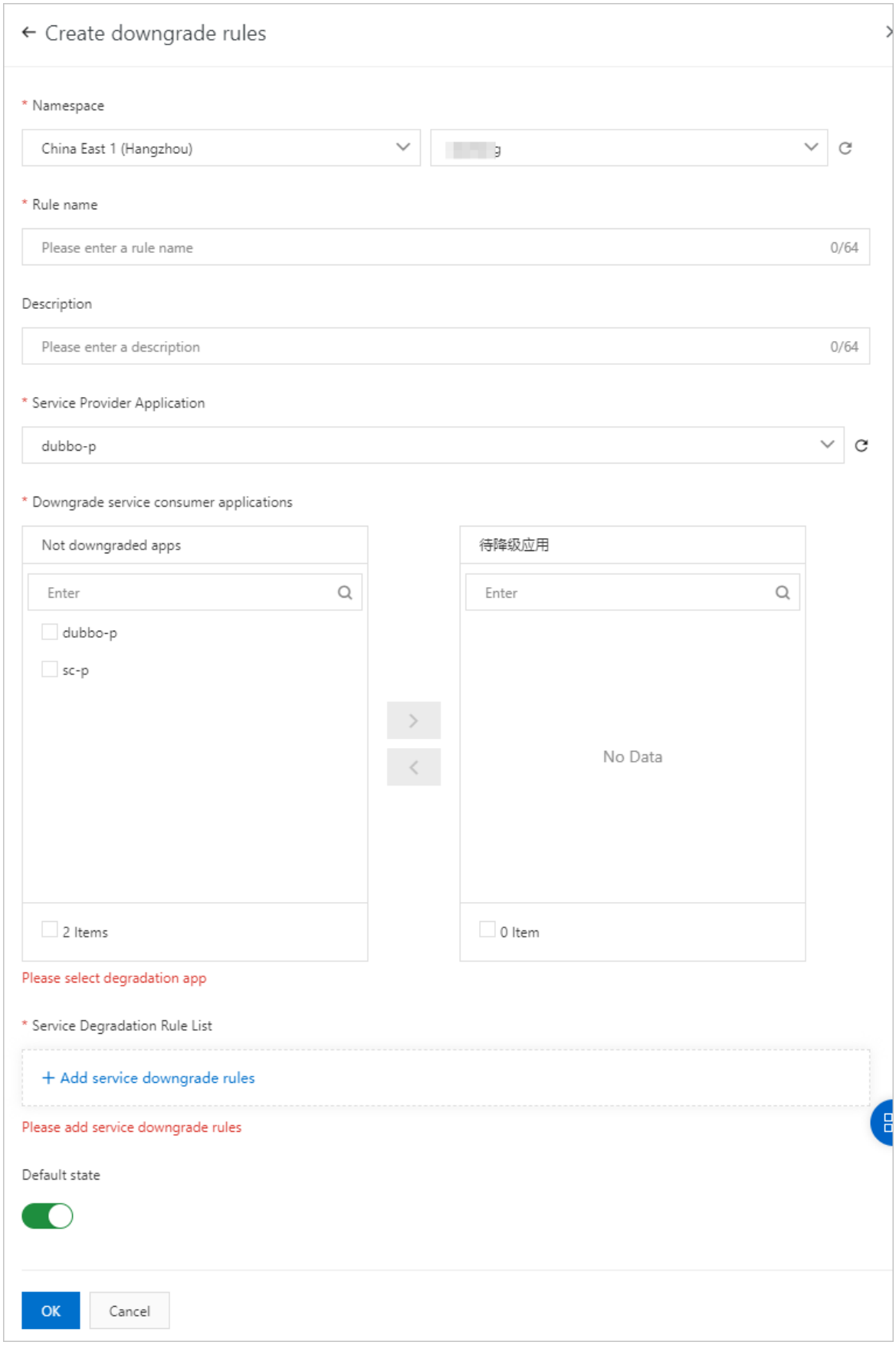
Parameter	Description
Label	Select a tag from the drop-down list, which is the value of the custom JVM parameter -Dalicloud.service.tag that you set for the new group. Then, the IP address and port number of the instance in the group are displayed in the **Application instance** section.
Link Delivery	Turn on **Link Delivery** if you want to use end-to-end traffic adjustment. **Note** The end-to-end traffic adjustment feature is in canary release. If you want to use end-to-end traffic adjustment, join the DingTalk group whose ID is 31723701 to contact technical support.
Traffic rules	
Frame type	Select **Spring Cloud** or **Dubbo** based on your business requirements. ○ Spring Cloud: You can specify only a URL, such as `/getIp`. ○ Dubbo: You can select a specific service and an interface.
Conditional mode	Select **At the same time meet the following conditions** or **Meet any of the following conditions** based on your business requirements.
Condition list	Select Parameter, Cookie, or Header from the drop-down list. Examples: ○ If you select Parameter, you must specify the value in the `name=xiaoming` format in the Value column. ○ If you select Cookie, you must specify the value in the `hello = "world"` or hello = "world2" format in the Value column.

1.10. Configure a service degradation rule for a Spring Cloud application

Context

Create a service degradation rule

1.
2. In the left-side navigation pane, choose **Microservices Governance > Spring Cloud**.
3. In the navigation tree of the **Spring Cloud** page, click **Service Degradation**.
4. On the **Service Degradation** page, select a region and a microservice namespace. Then, click **Create downgrade rules**.
5. In the **Create downgrade rules** panel, set the parameters and click **OK**.



The following table describes the parameters.

Parameter	Description
Microservice Space	Select the region and microservice namespace to which the application belongs.
Rule name	Enter the name of the service degradation rule. The name can be a maximum of 64 characters in length, and can contain letters, digits, underscores (_), and hyphens (-).
Description	Enter the description of the service degradation rule.
Service Provider Application	Select the application that can be called by other applications.
Downgrade service consumer applications	Select the application that you want to downgrade.
Service Degradation Rule List	Click **Add service downgrade rules** to create a service degradation rule.
Frame type	Select **Spring Cloud**.
Service Path	Select the application that you select for the Service Provider Application parameter and select the service path of the application.
Http Method	Select the request method for the application that you select for the Service Path parameter.
Effective strategy	Select the policy based on which the service degradation rule takes effect. Valid values: **Effective for all requests** and **Effective for abnormal requests**.
App to be downgraded	Select the policy for the service degradation rule. If the rule is triggered, the specified content is returned. Valid values: **Return Null**, **Return Exception**, and **Return custom Json data**.
Default state	Enable or disable the service degradation rule. ○ On: enables the rule after it is created. This is the default value. ○ Off: disables the rule after it is created. To enable the rule, find the rule on the **Service Degradation** page and click **Open** in the **Operation** column.

Result

What's next

1.11. End-to-end traffic adjustment

1.11.1. Overview

In Kubernetes clusters, Enterprise Distributed Application Service (EDAS) supports the end-to-end traffic adjustment feature for Spring Cloud microservice-oriented applications. The end-to-end traffic adjustment feature helps you create a traffic adjustment environment with ease and route traffic with specific characteristics to applications of a specified version.

Background information

In EDAS, part of Spring Cloud applications that are deployed in Kubernetes clusters may be updated to a specific version. In this case, traffic with specific characteristics may fail to be routed to applications of a desired version. This is because applications call each other randomly. The end-to-end traffic adjustment feature can help you isolate applications of a version from others in a lane, which is an independent runtime environment. You can configure traffic adjustment rules in the lane to route the request traffic that meets the rules to applications of the specified version.

This section describes how to use the end-to-end traffic adjustment feature in the order placement scenario of an e-commerce architecture.

After a customer places an order, the traffic comes in from the ingress application, which can also be a microservice gateway. The ingress application calls the transaction center, the transaction center calls the commodity center, and then the commodity center calls the downstream inventory center.

Both the transaction center and the commodity center are running in new versions V1.0 and V2.0. The two versions need to be verified during a canary release. At this time, you want to route the request traffic that meets specific traffic adjustment rules in the ingress application to applications of the new versions, and route all the remaining traffic to applications of the online version, which is the official version.

In the preceding flowchart, both the transaction center and the commodity center are running in new versions V1.0 and V2.0. Access requests are randomly forwarded to applications of each version, and the traffic cannot be controlled. You can use the end-to-end traffic adjustment feature to configure V1.0 as Lane *red* and V2.0 as Lane *blue* and configure traffic adjustment rules in the ingress application. When the request traffic in the ingress application meets the traffic adjustment rules of a lane, the request traffic is routed to the lane.

Terms

- Ingress application

  The ingress of traffic in a microservice system. An ingress application can be a service gateway that is built based on Spring Cloud Gateway or Spring Cloud Netflix Zuul, or a Spring Boot, Spring MVC, or Dubbo application.

- Lane

  An isolated environment that is defined for applications of the same version. Only the request traffic that meets the traffic adjustment rules of a lane can be routed to the applications that are configured to receive the marked traffic in the lane. An application can belong to multiple lanes. A lane can contain multiple applications. Applications have a many-to-many relationship with lanes.

- Lane group

  A collection of lanes. A lane group is used to distinguish different teams or different scenarios.

Limits

- After you configure the end-to-end traffic adjustment feature for applications, these applications no longer support a canary release.
- If you want to build an ingress gateway based on Spring Cloud Gateway, make sure that the version of Spring Cloud Gateway is 2.1.x or later.
- The quotas of lane groups and lanes vary based on the edition of EDAS. The following limits apply:
  - Standard Edition: All regions can contain only one lane group. This lane group can contain up to five lanes.

    All editions other than Professional Edition and Platinum Edition are Standard Edition.
  - Professional Edition: All regions can contain a maximum of 10 lane groups. Each lane group can contain up to 50 lanes.
  - Platinum Edition: All regions can contain a maximum of 10 lane groups. Each lane group can contain up to 50 lanes.
- The quotas of lane groups and lanes cannot be increased.

  If you want to increase the quotas of lane groups and lanes, submit a ticket.

1.11.2. Use the end-to-end traffic adjustment feature to monitor the traffic of an ingress application

After a lane group is created, you can directly access applications and monitor the inbound traffic of an ingress application.

Prerequisites

Before you configure end-to-end traffic adjustment for applications, make sure that the following prerequisites are met:

- Microservice-oriented applications are created. For more information, see Overview.

  An ingress application is available. For example, a service gateway is built based on Spring Cloud Gateway or Spring Cloud Netflix Zuul. The service gateway is associated with a microservice namespace of Enterprise Distributed Application Service (EDAS).

  If you want to build an ingress gateway based on Spring Cloud Gateway, make sure that the version of Spring Cloud Gateway is 2.1.x or later.
- A Server Load Balancer (SLB) instance is bound to the ingress application. For more information, see Bind SLB instances or Reuse an SLB instance.

Create a lane group

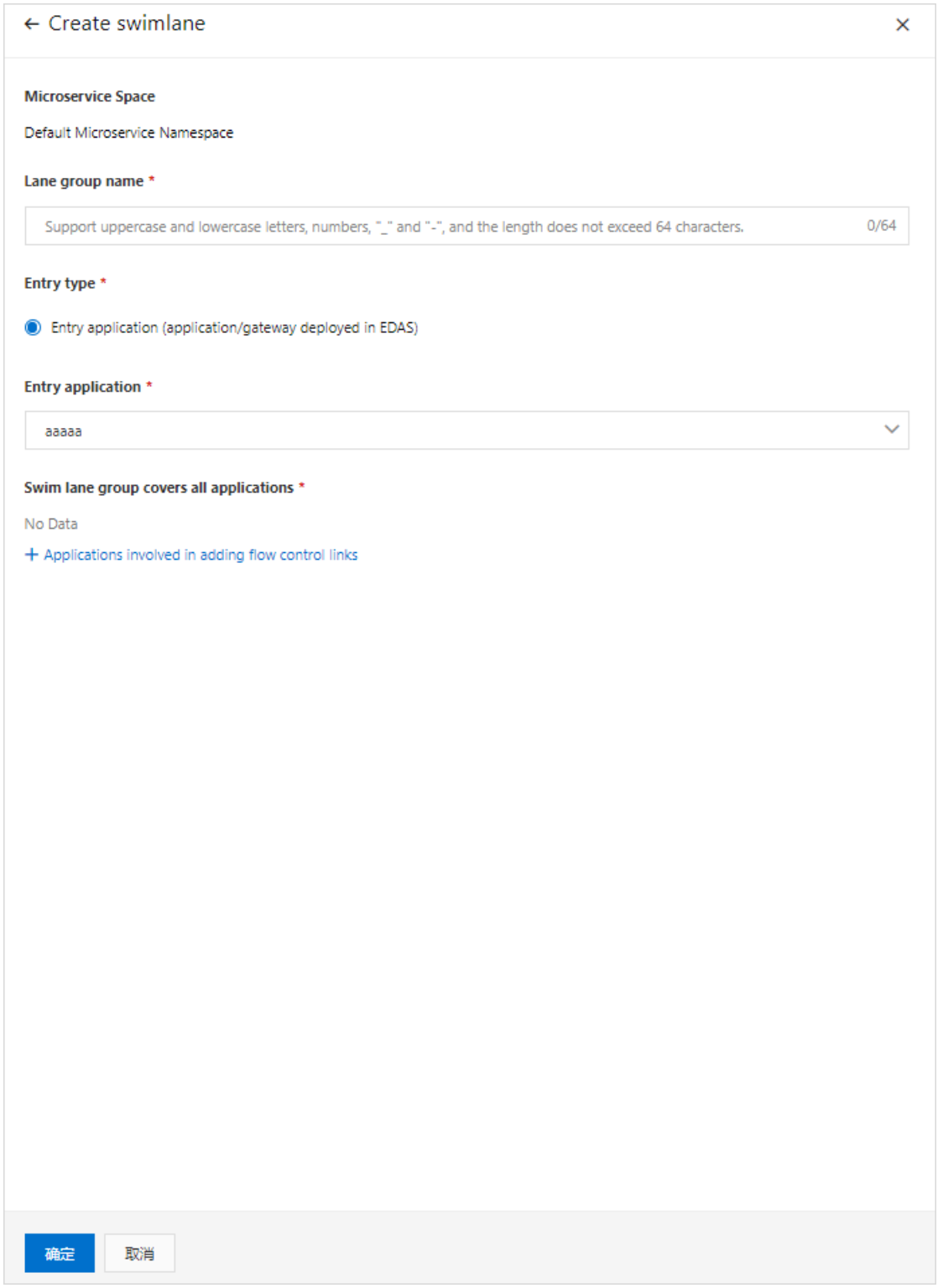
1.
2. In the left-side navigation pane, choose **Microservices Governance > End-to-end Traffic Adjustment**.
3. In the top navigation bar, select a region. On the **End-to-end Traffic Adjustment** page, select a microservice namespace.

4. In the lower part of the **End-to-end Traffic Adjustment** page, click **Create Lane Groups and Lanes**.

   If a lane group is created in the selected microservice namespace, click **Create** to the right of the **Select swim lane group** field.

   > **Note** A microservice namespace can contain up to two lane groups.

5. In the **Create swimlane** panel, set relevant parameters and click **OK**.



Parameter	Description
Microservice Space	The microservice namespace that you select on the **End-to-end Traffic Adjustment** page. The selected microservice namespace cannot be changed.

Parameter	Description
Lane group name	The name of the lane group. The name can be up to 64 characters in length, and can contain letters, digits, hyphens (-), and underscores (_).
Entry type	The type of the ingress application. Default value: **Entry application (application/gateway deployed in EDAS)**. **Note** EDAS allows you to use a service gateway that is built based on Spring Cloud Gateway or Spring Cloud Netflix Zuul as an ingress application. The service gateway must be associated with a microservice namespace of EDAS.
Entry application	This parameter is displayed only when the **Entry type** parameter is set to **Entry application (application/gateway deployed in EDAS)**.
Swim lane group covers all applications	Click **Applications involved in adding flow control links** and select all applications that are involved based on the ingress application or ingress gateway that you select.

After the lane group is created, the involved applications of the created lane group are displayed in the **Swim lane group involves applications** section of the **End-to-end Traffic Adjustment** page. Check whether the ingress application and the involved applications are properly selected. To modify the information about the lane group, click **Edit** and modify the information as needed.

Monitor the traffic of an ingress application

1. Obtain the endpoint of the SLB instance that is bound to the ingress application or ingress gateway that you want to manage.
   i. On the **Applications** page, click the name of the ingress application or ingress gateway.
   ii. In the **Access configuration** section of the **Application Overview** page, copy the endpoint of the SLB instance.
2. Use a browser or other tools to access an involved application of the lane group multiple times.

   In this example, the transaction center is accessed in a browser and the traffic is routed in different ways. For more information about how to route traffic to specific applications, see Use the end-to-end traffic adjustment feature to route traffic to specific applications.

```
Case 1: A [172.20.**.**] -> B1 [172.20.**.**] -> C [172.20.**.**]
Case 2: A [172.20.**.**] -> B [172.20.**.**] -> C [172.20.**.**]
Case 3: A2 [172.20.**.**] -> B [172.20.**.**] -> C [172.20.**.**]
.......
```

3. View the traffic monitoring chart of the ingress application.

i. On the **End-to-end Traffic Adjustment** page, select the lane group whose data you want to view.

ii. On the **End-to-end Traffic Adjustment** page, select a monitoring time range. The traffic monitoring chart is refreshed in the **Ingress Application Monitoring (total)** section.

In the traffic monitoring chart, you can view the queries per second (QPS) at a specific point in time.

1.11.3. Use the end-to-end traffic adjustment feature to route traffic to specific applications

In Enterprise Distributed Application Service (EDAS), you can configure the end-to-end traffic adjustment feature for Spring Cloud and Dubbo microservice-oriented applications that are deployed in Kubernetes clusters. This feature helps you route traffic that has specific characteristics to applications of a specified version.

Prerequisites

Before you configure end-to-end traffic adjustment for applications, make sure that the following prerequisites are met:

- Applications of a new version are deployed, or applications are updated. For more information, see Overview of application upgrades and rollbacks (applicable to Kubernetes clusters).
- An ingress application is available. For example, a service gateway is built based on Spring Cloud Gateway or Spring Cloud Netflix Zuul. The service gateway is associated with a microservice namespace of EDAS.

  If you want to build an ingress gateway based on Spring Cloud Gateway, make sure that the version of Spring Cloud Gateway is 2.1.x or later.
- A Server Load Balancer (SLB) instance is bound to the ingress application. For more information, see Bind SLB instances or Reuse an SLB instance.

Context

This section describes how to use the end-to-end traffic adjustment feature in the order placement scenario of an e-commerce architecture.

After a customer places an order, the traffic comes in from the ingress application, which can also be a microservice gateway. The ingress application calls the transaction center, the transaction center calls the commodity center, and then the commodity center calls the downstream inventory center.

Both the transaction center and the commodity center are running in new versions V1.0 and V2.0. The two versions need to be verified during a canary release. At this time, you want to route the request traffic that meets specific traffic adjustment rules in the ingress application to applications of the new versions, and route all the remaining traffic to applications of the online version, which is the official version.

Create a lane group

1.
2. In the left-side navigation pane, choose **Microservices Governance > End-to-end Traffic Adjustment**.
3. In the top navigation bar, select a region. On the **End-to-end Traffic Adjustment** page, select a

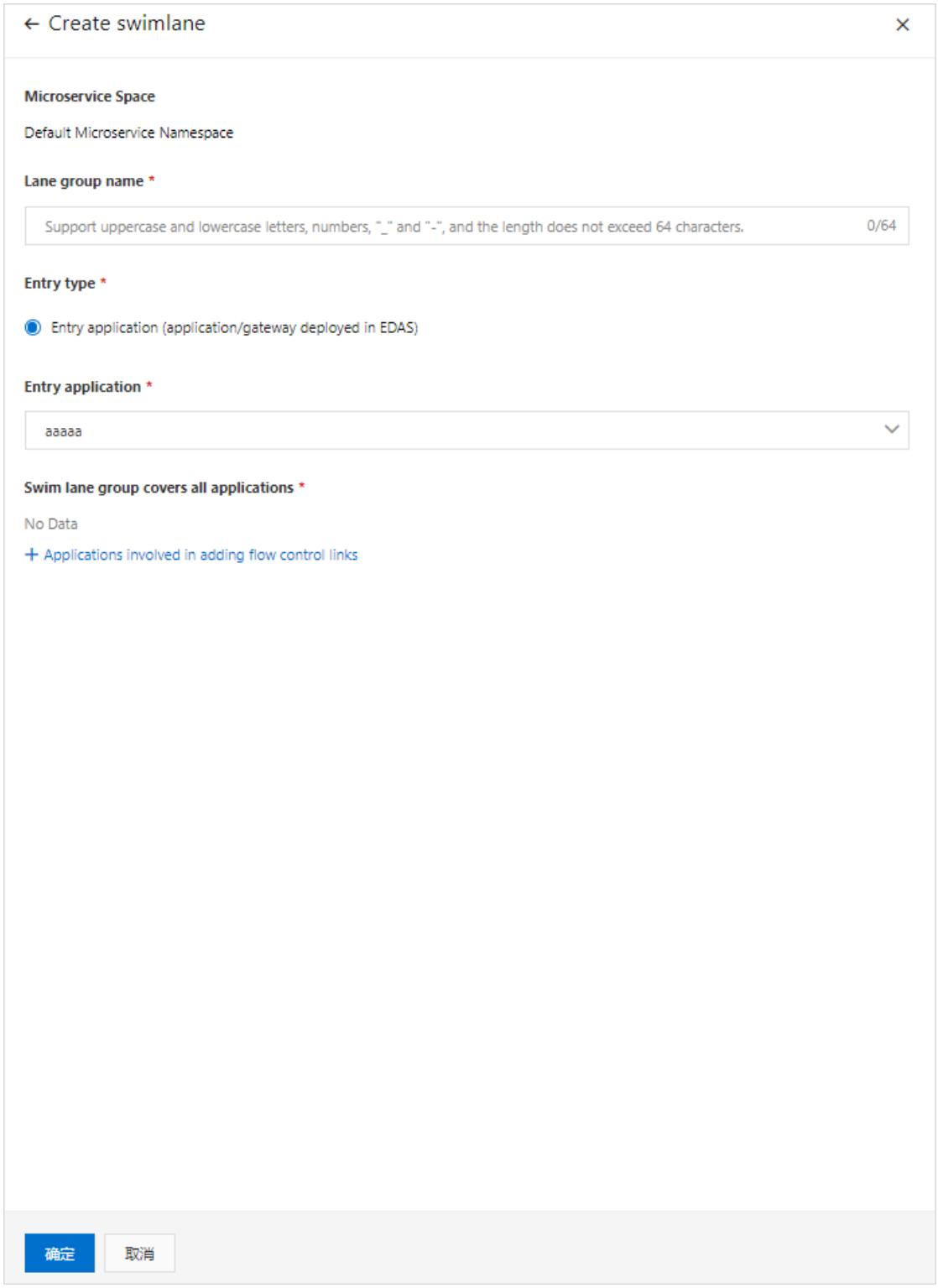
microservice namespace.

4. In the lower part of the **End-to-end Traffic Adjustment** page, click **Create Lane Groups and Lanes**.

   If a lane group is created in the selected microservice namespace, click **Create** to the right of the **Select swim lane group** field.

   > **Note** A microservice namespace can contain up to two lane groups.

5. In the **Create swimlane** panel, set relevant parameters and click **OK**.



Parameter	Description
Microservice Space	The microservice namespace that you select on the **End-to-end Traffic Adjustment** page. The selected microservice namespace cannot be changed.

Parameter	Description
Lane group name	The name of the lane group. The name can be up to 64 characters in length, and can contain letters, digits, hyphens (-), and underscores (_).
Entry type	The type of the ingress application. Default value: **Entry application (application/gateway deployed in EDAS)**. **Note** EDAS allows you to use a service gateway that is built based on Spring Cloud Gateway or Spring Cloud Netflix Zuul as an ingress application. The service gateway must be associated with a microservice namespace of EDAS.
Entry application	This parameter is displayed only when the **Entry type** parameter is set to **Entry application (application/gateway deployed in EDAS)**.
Swim lane group covers all applications	Click **Applications involved in adding flow control links** and select all applications that are involved based on the ingress application or ingress gateway that you select.

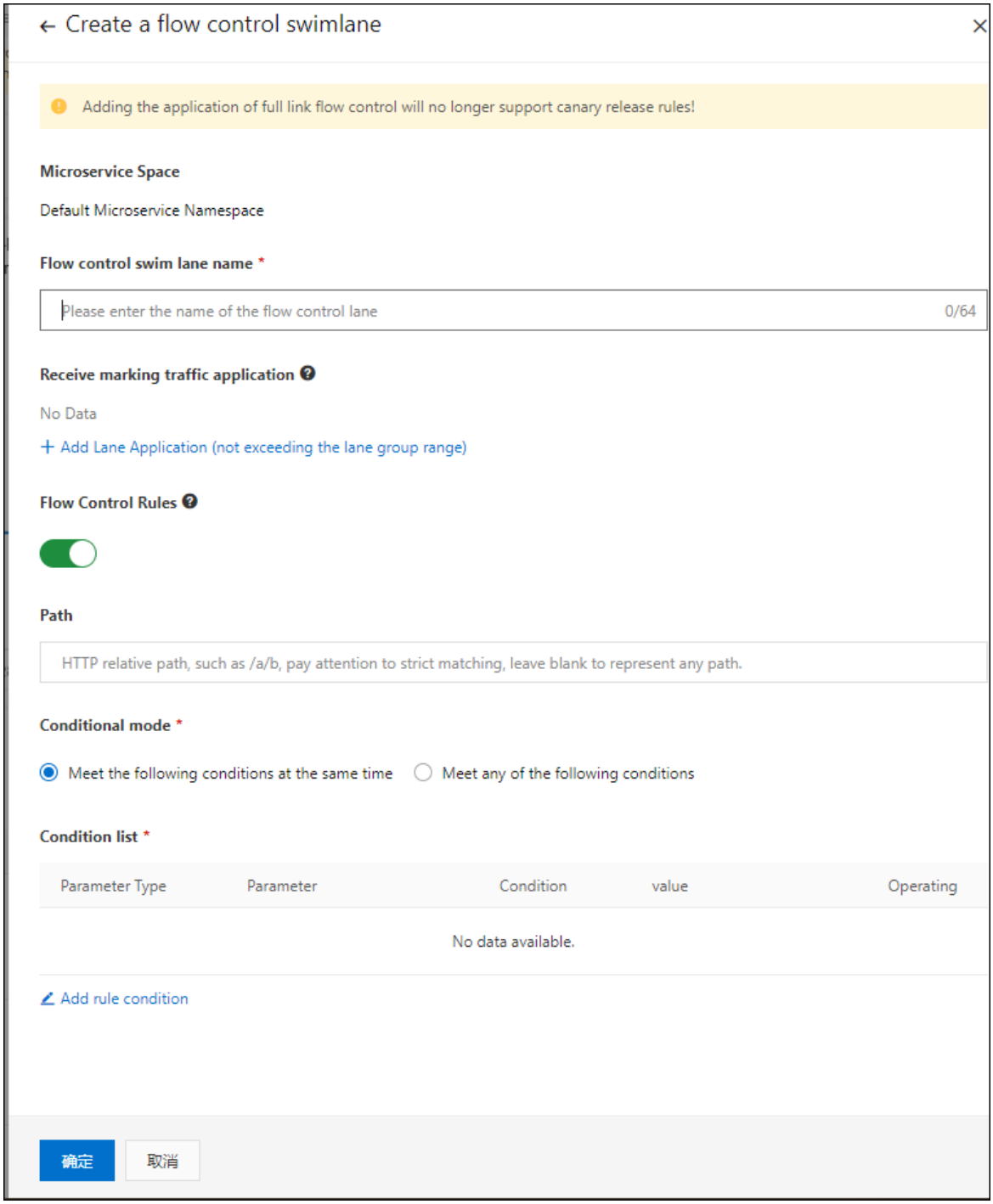
After the lane group is created, the involved applications of the created lane group are displayed in the **Swim lane group involves applications** section of the **End-to-end Traffic Adjustment** page. Check whether the ingress application and the involved applications are properly selected. To modify the information about the lane group, click **Edit** and modify the information as needed.

Create a lane

1. On the **End-to-end Traffic Adjustment** page, select the same microservice namespace as the lane group that you created. Then, click **Click to Create the First Split Lane** in the lower part of the page.

   Notice After you configure the end-to-end traffic adjustment feature for applications, these applications no longer support a canary release.

2. In the **Create a flow control swimlane** panel, set relevant parameters and click **OK**.



Parameter	Description
Microservice Space	The microservice namespace that you select on the **End-to-end Traffic Adjustment** page. Make sure that your lane group is created in the same microservice namespace. The selected microservice namespace cannot be changed.
Flow control swim lane name	The name of the lane. The name can be up to 64 characters in length, and can contain letters, digits, hyphens (-), and underscores (_).

Parameter	Description
(Optional) Receive marking traffic application	Click **Add Lane Application (not exceeding the lane group range)** and select an application in the lane group. **Note** ◦ You can select multiple applications for the same lane. You can also create a lane for each application. ◦ A lane group can contain up to five lanes. ◦ When you create a lane, you can skip the selection of applications that receive the marked traffic. Then, you can select such applications when you modify the created lane.
Flow Control Rules	
Switch	Specifies whether to enable traffic adjustment rules. By default, the switch is turned on.
Path	The HTTP relative path. If you leave this parameter empty, the rules take effect for all paths. Set this parameter based on your business requirements.
Conditional mode	The mode in which conditions are met. Select a mode based on your needs. Valid values: **Meet the following conditions at the same time** and **Meet any of the following conditions**. ◦ **Meet the following conditions at the same time**: The rules take effect when all the conditions are met at the same time. ◦ **Meet any of the following conditions**: The rules take effect when one of the conditions is met.

Parameter	Description
Condition list	Click **Add rule condition**. You can add multiple conditions as needed. You can set different types of conditions, such as *Cookie*, *Header*, *Parameter*, and *Body Content*. In this example, the following conditions are added: ○ The parameter type is *Parameter* and the condition is env=red. When the condition is met, the traffic is routed to the applications that are running in V1.0. ○ The parameter type is *Parameter* and the condition is env=blue. When the condition is met, the traffic is routed to the applications that are running in V2.0.

After the lane is created, the created lane appears in the **Flow Control Distribution** section of the **End-to-end Traffic Adjustment** page. Check whether the lane name, traffic adjustment rules, and applications that receive the marked traffic are correct. To modify the information about the lane, click **Edit** and modify the information as needed.

3. (Optional) To create more lanes, click **Create swimlane** in the **Flow Control Distribution** section and set relevant parameters.

> **Note** A lane group can contain up to five lanes.

Verify that traffic is routed to specific applications

1.
2. Use a browser or other tools to access an application in a lane of the lane group multiple times.

   For example, enter *http://ip:prt/**?env=red* in the address bar of a browser to access the transaction center. The traffic is routed in only one way. This indicates that the traffic that has specific characteristics is routed to the specified applications.

   In the URL that you entered, ** is the path that you specify in traffic adjustment rules, and *env=red* is a condition in the traffic adjustment rules.

```
A2[172.20.**.**] -> B2[172.20.**.**] -> C[172.20.**.**]
```

3. View the traffic monitoring charts of applications that receive the marked traffic.
   i. On the **End-to-end Traffic Adjustment** page, select the lane group whose data you want to view.
   ii. On the **End-to-end Traffic Adjustment** page, select a monitoring time range. The traffic monitoring data is refreshed in the **Ingress Application Monitoring (total)** and **Flow Control Distribution** sections.

   View the traffic monitoring charts. You can find that the request traffic is routed to the applications that receive the marked traffic and the following condition is met: *Total queries per second (QPS) in the ingress application = QPS in the applications that receive the unmarked traffic + QPS in the applications that receive the marked traffic*.

View the traffic monitoring charts of all applications

In addition to the traffic monitoring charts of the ingress application, applications that receive the unmarked traffic, and applications that receive the marked traffic, you can also view the traffic monitoring charts of all applications in the same lane group. You can compare the traffic monitoring charts of all applications to analyze more useful information. Examples:

- Find the applications that are called at the same time.
- Analyze traffic escape issues and find out the escaped traffic.

2.Dubbo service governance

2.1. Gracefully disconnect Dubbo applications

Online applications must be developed in a way to ensure normal service requests even during the period from when applications are stopped for service upgrade and deployment to when services are restarted and recovered. This means that the entire process must be imperceptible to clients. You must configure graceful disconnection to properly shut down applications for deployment, stop, rollback, scale-in, and reset.

Reasons for graceful unpublishing

Graceful unpublishing ensures the normal processing of consumer service requests during the period from when applications are stopped to when services are recovered. The most secure and reliable solution is to update your application when no service requests exist. However, service requests exist even when the application is unpublished.

A traditional solution is to manually perform the following steps: (1) Manually remove traffic. (2) Stop your application. (3) Update your application and then restart the application. In this case, users are not notified about changes to the system. This applies to the update process and related manual operations.

An innovative solution is to use an automated mechanism at the container or framework level. This mechanism can be used to automatically remove traffic and process received requests. This makes the update process imperceptible to your business and improves the O&M efficiency. This mechanism is called graceful unpublishing.

Advantages of graceful disconnection

For open source Dubbo, graceful disconnection can be implemented by using ShutDownHook and quality of service (QoS). However, this method has high development workloads and the version of Dubbo must meet requirements. In addition, some legacy issues affect the normal use of this feature.

EDAS integrates graceful disconnection into the publishing process so that graceful disconnection is automatically implemented when you stop, deploy, roll back, scale in, and reset the applications in Elastic Compute Service (ECS) or Kubernetes clusters. You do not need to perform graceful disconnection operations on applications or in the EDAS console, and traffic is not affected.

Check whether graceful unpublishing takes effect

You can check whether graceful unpublishing takes effect for applications based on your actual business. EDAS also provides two application demos. You can use these demos to check whether graceful unpublishing takes effect in a Kubernetes cluster.

You can perform the following steps to check whether graceful unpublishing takes effect:

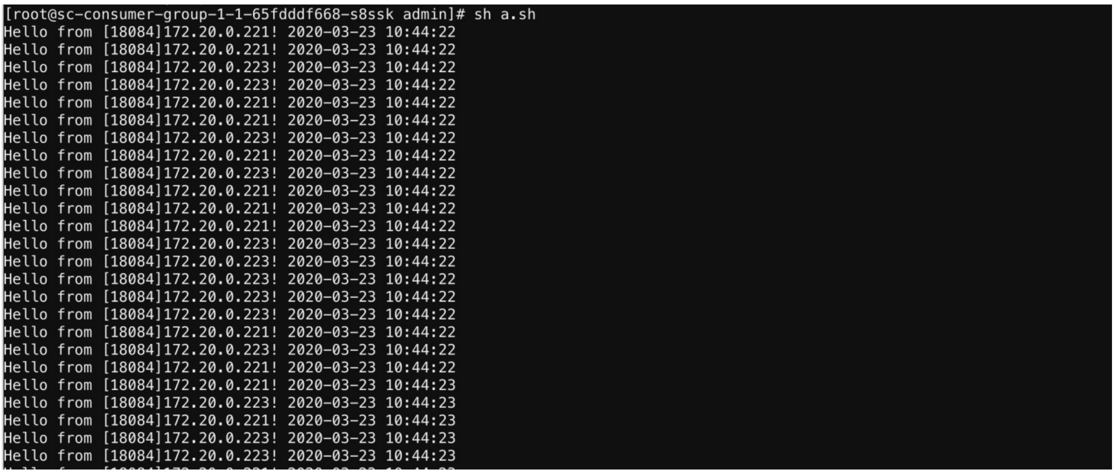
1. Download application demos Provider and Consumer.
2. Deploy Provider and Consumer to a Kubernetes cluster.

   Provider has two instances deployed, and Consumer has one instance deployed. For more information, see Overview.
3. View the call status of Provider.

i. Log on to the pod where Consumer is deployed and run the following commands to continuously access the services of Provider:

```
#!/usr/bin/env bash
while true
do
    echo `curl -s -XGET http://localhost:18091/user/rest`
done
```

ii. View the response of the calls.

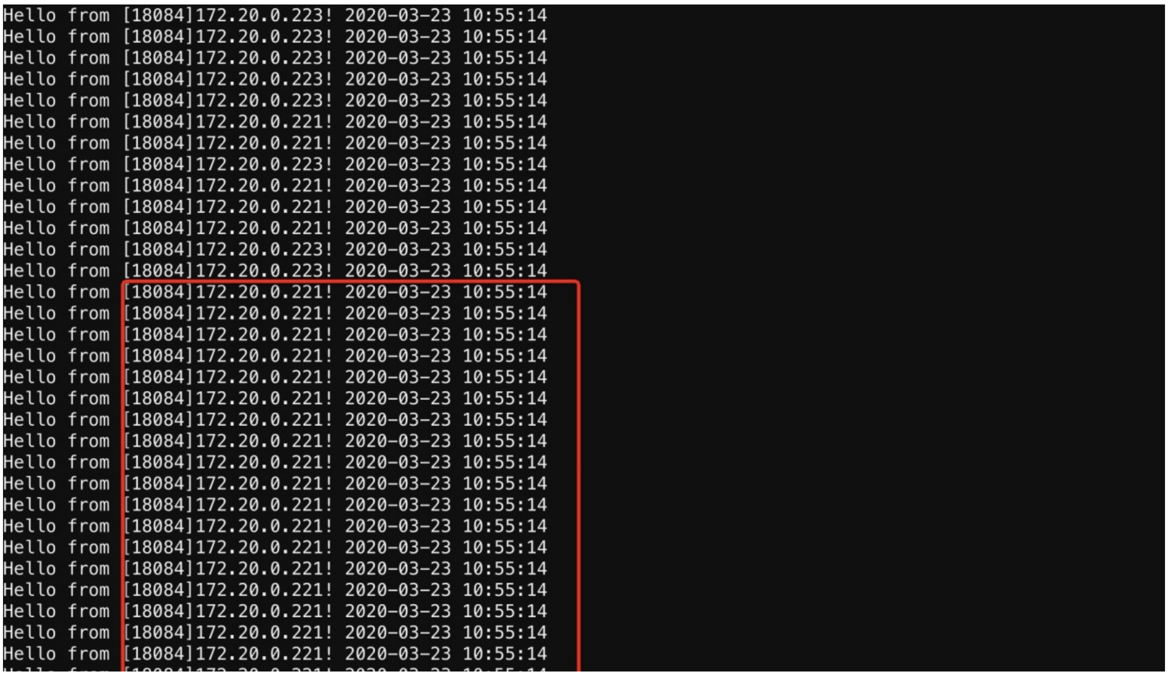


```
[root@sc-consumer-group-1-1-65fdddf668-s8ssk admin]# sh a.sh
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.223! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:22
Hello from [18084]172.20.0.221! 2020-03-23 10:44:23
Hello from [18084]172.20.0.223! 2020-03-23 10:44:23
Hello from [18084]172.20.0.221! 2020-03-23 10:44:23
Hello from [18084]172.20.0.223! 2020-03-23 10:44:23
Hello from [18084]172.20.0.223! 2020-03-23 10:44:23
```

The response shows that Consumer randomly accesses two instances of Provider. The IP addresses of the instances are 172.20.0.221 and 172.20.0.223.

> **Notice** Do not close the response window.

4. Scale in one instance from Provider and restart the instance. For more information, see Scale out and scale in an application.
5. View the response again to check whether graceful unpublishing takes effect.



```
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.223! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
Hello from [18084]172.20.0.221! 2020-03-23 10:55:14
```

View the call status of Consumer to check whether graceful unpublishing takes effect. View the logs of Consumer. The logs show that no exceptions occur, and the instance unavailability is imperceptible to Consumer.

The response shows that Consumer accesses the remaining instance of Provider. The IP address of the instance is 172.20.0.221. When Consumer accesses the remaining instance, no exceptions occur, and Consumer is not affected.

2.2. Publish Dubbo applications by using canary releases

2.2.1. Use the EDAS console to implement canary releases of applications in Kubernetes clusters

For Spring Cloud or Dubbo microservice-oriented applications that are deployed in a Kubernetes cluster, you can implement a canary release. The canary release allows you to verify a new application version on a small number of instances. If the verification is successful, you can update the application on all of your instances to a new version. This makes the update secure.

Limits

- High-Speed Service Framework (HSF) applications: Canary release is not supported.
- Dubbo applications: You can implement canary releases of Dubbo applications without limits.
- Spring Cloud applications: If you use Deployment.Metadata.Name or Deployment.Metadata.Uid to configure specific features of an application, do not implement a canary release of the application. Otherwise, the native features of the application may be abnormal after the canary release.

Procedure

1.
2. In the left-side navigation pane, click **Applications**. In the top navigation bar, select a region. In the upper part of the Applications page, select a microservice namespace.
3. In the left-side navigation pane, click **Applications**. In the top navigation bar, select a region. In the upper part of the Applications page, select a microservice namespace.
4. On the **Applications** page, select **Container Service or Serverless Kubernetes Cluster** from the **Cluster Type** drop-down list. Then, click the name of the application that you want to deploy.
5. In the upper-right corner of the **Application Overview** page, choose **Deploy > Deploy**.
6. In the **Canary Release (Phased)** section of the **Select Deployment Mode** page, click **Start Deployment** in the upper-right corner.
7. On the **Canary Release (Phased)** page, set the deployment parameters, release policy, and canary release rules. Then, click **OK**.
   i. Set the deployment parameters.

      Deployment parameters

Parameter	Description
Configure Image (applicable to only applications that are deployed by using images)	You can update the version of an image, but cannot change the image of an application.
Application Runtime Environment (applicable to applications that are deployed by using JAR packages or WAR packages)	The value must be the same as that used for the previous deployment. ■ JAR package: The application runtime environment is **Standard Java Application Runtime Environment**. You cannot change the type of the application runtime environment. ■ WAR package: The application runtime environment is **Apache Tomcat**. You cannot change the type of the application runtime environment. However, you can change the version of Apache Tomcat as needed.
Java Environment (applicable to applications that are deployed by using JAR packages or WAR packages)	Select a value from the drop-down list as needed.
Current Environment	The current runtime environment of the application. The current runtime environment is displayed only if the application is deployed by using JAR packages or WAR packages. Enterprise Distributed Application Service (EDAS) automatically upgrades the Java environment or application runtime environment of your application to the latest version.
File Uploading Method (applicable to applications that are deployed by using JAR packages or WAR packages)	The type of the deployment package must be the same as that used for the previous deployment. You can use a WAR package or a JAR package. This parameter value cannot be changed. Set the parameter based on your requirements. You can select **Upload Package** and upload a JAR or WAR package. You can also select **Package Address** and specify the address of a JAR or WAR package.
Version (applicable to applications that are deployed by using JAR packages or WAR packages)	The version number of the deployment package. You can use a timestamp as the version number.
Time Zone (applicable to applications that are deployed by using JAR packages or WAR packages)	The time zone for the application. Select a value from the drop-down list as needed.
Service Registration and Discovery	The O&M method of your service registry. For more information, see Select an O&M method for your service registry.

ii. In the **Release Policy** section, set the parameters for the release policy.

Parameters in the Release Policy section

Parameter	Description
Number of Instances for Canary Release	The number of application instances released in the first batch. The current number of instances for the application appears on the right side. The number of instances for the canary release cannot exceed 50% of the total number of instances. This makes the application stable. **Note** After the canary release is implemented, you must manually release the remaining batches.
Remaining Batches	After the release of the first batch is complete, the application is deployed to the remaining application instances based on the specified batches.
Batch Mode	The following processing methods are supported: ■ **Automatic**: automatically releases applications in batches based on the interval specified by the **Interval** parameter. **Interval**: the interval for releasing the remaining batches in minutes ■ **Manual**: manually triggers the release of the next batch. **Note** The **Batch Mode** parameter is available only if the value of the Remaining Batches parameter is greater than 1.
Deployment Interval Between Batches	If the number of instances in each batch is greater than 1, the application is deployed to the application instances at the specified interval. Unit: seconds.

The **Publish Policy Configuration** section on the right side shows the procedure for the canary release based on the configuration.

iii. Set canary release rules.

EDAS supports **Canary Release by Content** and **Canary Release by Ratio**.

Parameters for canary release rules

Tab	Parameter	Description
Canary Release by Content	**Protocol Type**	■ **Spring Cloud**: The **Path** parameter is required. ■ **Dubbo**: The **Select Service** and **Method** parameters are required.
	Conditional Mode	Select **Meet All Following Conditions** or **Meet Any of Following Conditions**.
	Conditions	■ **Spring Cloud**: Set the parameters based on **Cookie**, **Header**, or **Parameter**. ■ **Dubbo**: Set the **Parameter** and **Expression for Getting Parameter Values** parameters based on the actual values of your application.
Canary Release by Ratio	**Traffic Ratio**	Traffic is forwarded to the current instance group for the canary release based on the specified value.

> **Note** Click **Create Inbound Traffic Rule** to create multiple inbound traffic rules that can take effect at the same time.

iv. (Optional)Configure the advanced settings.

After the canary release is started, EDAS deploys the new application version to the specified instance group. The **Change List** page displays the deployment progress and status.

> **Note** You can check whether the traffic is distributed as expected.

8. After the traffic for the canary release is verified, click **Start Next Batch** on the right side of the **Change List** page. Complete the release of the subsequent batches.

   If problems are found during the verification process, you can click **Roll Back** in the upper-right corner of the **Change List** page. In the message that appears, click **OK**.

Verify the results

After the canary release is complete, check whether the **deployment** package is of the new version on the **Application Overview** page.

2.2.2. Canary release for an application in an ECS cluster

To update a Spring Cloud or Dubbo microservice-oriented application that is deployed in an Elastic Compute Service (ECS) cluster, you can implement a canary release to verify the new version on a small number of instances. If the verification is successful, you can update the application on all instances.

Prerequisites

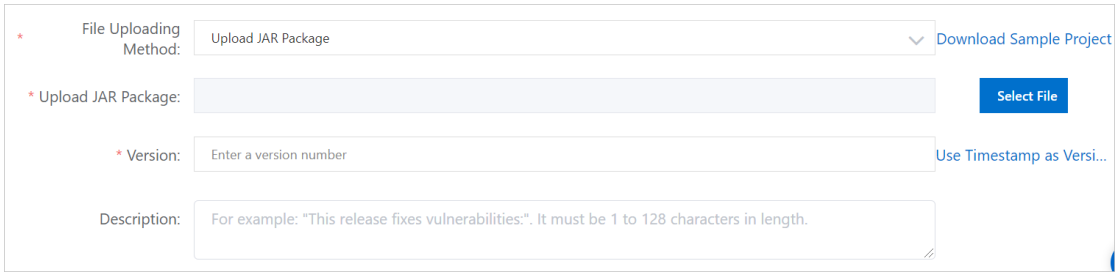
Before you implement a canary release, make sure that the application contains at least two instance groups and at least two groups contain instances. For more information about how to create instance groups and add ECS instances to the groups, see Manage instance groups for an application deployed in an ECS cluster in the EDAS console.

Limits

- High-Speed Service Framework (HSF) applications: Canary release is not supported.
- Dubbo applications: You can implement canary releases of Dubbo applications without limits.
- Spring Cloud applications: If you use Deployment.Metadata.Name or Deployment.Metadata.Uid to configure specific features of an application, do not implement a canary release of the application. Otherwise, the native features of the application may be abnormal after the canary release.

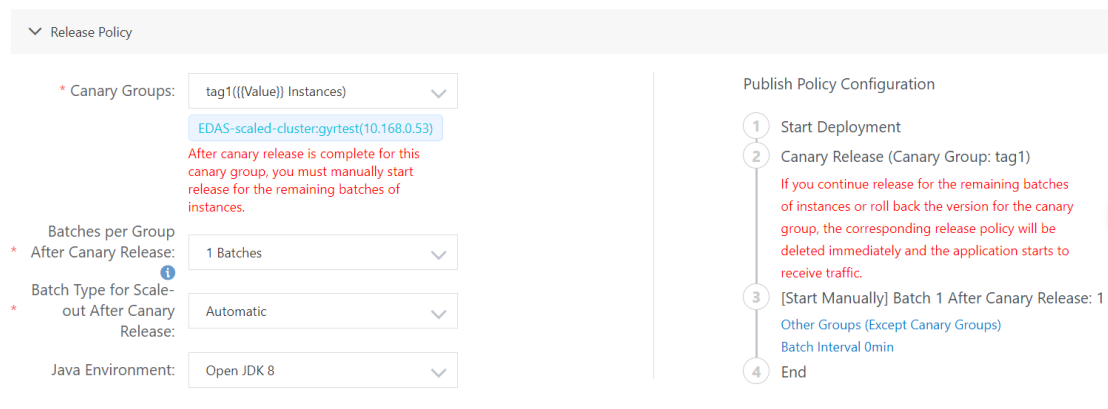
Procedure

1.
2. In the left-side navigation pane, click **Applications**.
3. In the top navigation bar, select a region. On the **Applications** page, select a microservice namespace and click the name of the application for which you want to implement a canary release.
4. On the Basic Information page, click **Deploy Application** in the upper-right corner.
5. On the **Select Deployment Mode** page, click **Start Deployment** in the upper-right corner of the **Canary Release (Phased)** section.
6. On the **Canary Release** page, upload the deployment package of the new application version, set the canary release policy and rules, and then click **OK**.
   i. Upload the deployment package of the new application version.



   ii. In the **Release Policy** section, set the parameters for the release policy.

   The **Publish Policy Configuration** section on the right side shows the procedure for the canary release based on the configuration.



The following table describes the parameters for configuring the release policy.

Parameter	Description
Canary Groups	The instance group for the canary release.
Batches per Group After Canary Release	After the canary release for the specified instance group is complete, the new version is deployed to application instances in other groups based on the preset batches. ■ If all groups are selected, the new version is deployed to the instances in each group based on the selected batch number. If the number of instances in a group is less than the selected batch number, the new version is deployed to the instances in the group based on the number of instances. ■ If you have specified a group, the new version is deployed to the instances in the specified group based on the selected batch number.
Batch Type for Scale-out After Canary Release	If you set the **Batches per Group After Canary Release** parameter to a value of at least 2, you must set this parameter. Valid values: *Automatic* and *Manual*. ■ *Automatic*: automatically deploys the new version to instances in batches based on the release interval. You must set the **Wait Time Before Next Batch** parameter. Valid values of **Wait Time Before Next Batch** are *Do Not Wait*, 1 Minutes, 2 Minutes, 3 Minutes, 4 Minutes, and 5 Minutes. ■ *Manual*: manually triggers the release of the next batch.

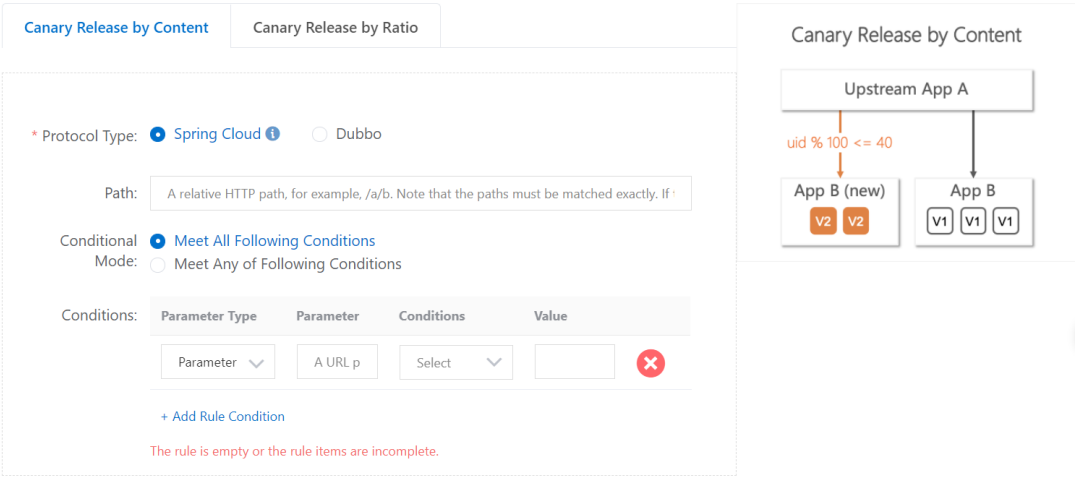
Parameter	Description
Java Environment	The runtime environment of the application. Select a runtime environment as needed.

iii. Configure canary release rules.

Enterprise Distributed Application Service (EDAS) supports **Canary Release by Content** and **Canary Release by Ratio**.

- **Canary Release by Content**: Click **Create Inbound Traffic Rule** and create a rule for inbound traffic.

> **Note** You can create multiple inbound traffic rules.



The following table describes the parameters for creating an inbound traffic rule on the Canary Release by Content tab.

Parameter	Description
Protocol Type	Valid values: *Spring Cloud* and *Dubbo*. Select one option based on the actual situation of the application. ■ *Spring Cloud*: The **Path** parameter is required. ■ *Dubbo*: The **Select Service** and **Method** parameters are required.
Conditional Mode	Select **Meet All Following Conditions** or **Meet Any of Following Conditions**.

Parameter	Description
Conditions	The conditions for *Spring Cloud* and *Dubbo* are different. Three methods are available: Cookie, Header, and Parameter. Set the parameters as needed. ■ *Spring Cloud*: *Cookie*, *Header*, and *Parameter* are available. Set the parameters as needed. ■ *Dubbo*: Set the **Parameter** and **Expression for Getting Parameter Values** parameters based on the actual values of your application.

- **Canary Release by Ratio**: Set the **Traffic Ratio** paramerter. Traffic is forwarded to the current instance group for the canary release based on this value.

After the canary release is started, EDAS deploys the new application version to the specified instance group. On the **Basic Information** page, the message `A change process is ongoing for this application. The application is in Executing state` appears. Click **View Details**. On the **Change Details** page that appears, view the deployment progress and status.

Stop a change: `The application is in the canary release state and this change has been stopped. Please roll back the application before you perform other operations.`

7. You can check whether the traffic is distributed as expected. For more information, see Monitor canary traffic.
8. After the traffic verification is complete, click **Start Next Batch** on the **Change Details** page. Complete the subsequent batch release.

   If an issue occurs during the verification process, click **Stop Change** in the upper-right corner of the **Change Details** page. After the change is stopped, on the **Basic Information** page, the message `The application is in the canary release state and this change has been stopped. Please roll back the application before you perform other operations.`

Verify the result

After the canary release is complete, check whether the **deployment package** is of the new version on the **Basic Information** page. On the **Instance Information** page, check whether the instances are in the **Normal** state.

2.3. Query Dubbo services

You can log on to the Enterprise Distributed Application Service (EDAS) console to query the service list and service details of Dubbo applications that are deployed in EDAS.

Limits

You can switch between the old and new versions of the **Service Query** page.

- In the new version, the system uses the EDAS agent to query services in the EDAS registry, Microservice Engine (MSE)-hosted registry, and self-managed registries, including ZooKeeper, Nacos, Eureka, and Consul.

- In the new version, you can query the services of all Dubbo versions and the services in all registries.
- In the old version, you can query only the services of Dubbo 2.7.x that are registered at the EDAS registry through Nacos.
- In the old version, you can query the services only in the EDAS registry.
- Outbound TCP connections over ports 8442, 8443, and 8883 must be allowed in the security group of your server. For more information about how to allow outbound connections in Elastic Compute Service (ECS), see Add a security group rule.

View the service list

1.
2. In the left-side navigation pane, choose **Microservices Governance > Dubbo**.
3. In the left-side navigation tree of **Dubbo**, click **Service Query**.
4. In the top navigation bar, select a region. On the **Service Query** page, select an option from the microservice namespace drop-down list and view the **Dubbo** services within your account.

   You can view the following information about a Dubbo service: **Service name**, **Version**, **Grouping**, **Application Name**, and **Number of instances**.

   If a large number of services exist, you can filter services by service name, IP address, or application name. Filter keywords are not case-sensitive. The value of **IP** varies between ECS and Kubernetes clusters.

   - ECS cluster: The value is the IP address of an ECS instance.
   - Kubernetes cluster: The value is the IP address of a pod.

   > **Note** If you can query services of your applications on the old Service Query page but not on the new Service Query page, troubleshoot the problem by performing the following steps:
   >
   > i. The new version of the Service Query page is released at 00:00:00 of January 20, 2020. You must restart your applications after this point in time so that they can be automatically mounted with the latest EDAS agent. Therefore, you must restart your applications before you query services on the new Service Query page.
   >
   > ii. Check whether the microservice framework version is supported. For more information about the supported versions, see Limits.

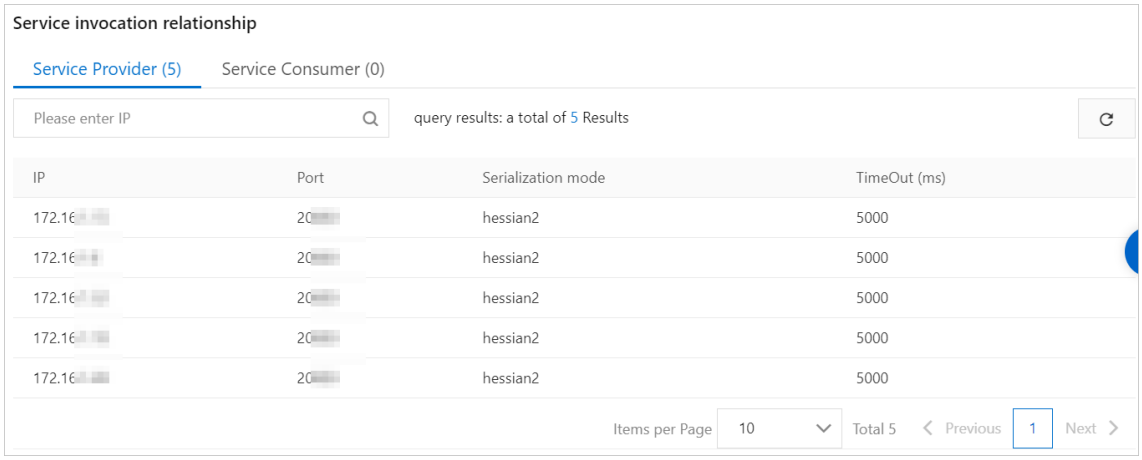
View service details

1.
2. In the left-side navigation pane, choose **Microservices Governance > Dubbo**.
3. In the left-side navigation tree of **Dubbo**, click **Service Query**.
4. In the top navigation bar, select a region. On the **Service Query** page, select an option from the microservice namespace drop-down list. Then, click a service name in the service list.
5. In the **Service Details** panel, view the details of the service.

   The **Service Details** panel provides the following information: **Basic information**, **Service invocation relationship**, and **Metadata**.

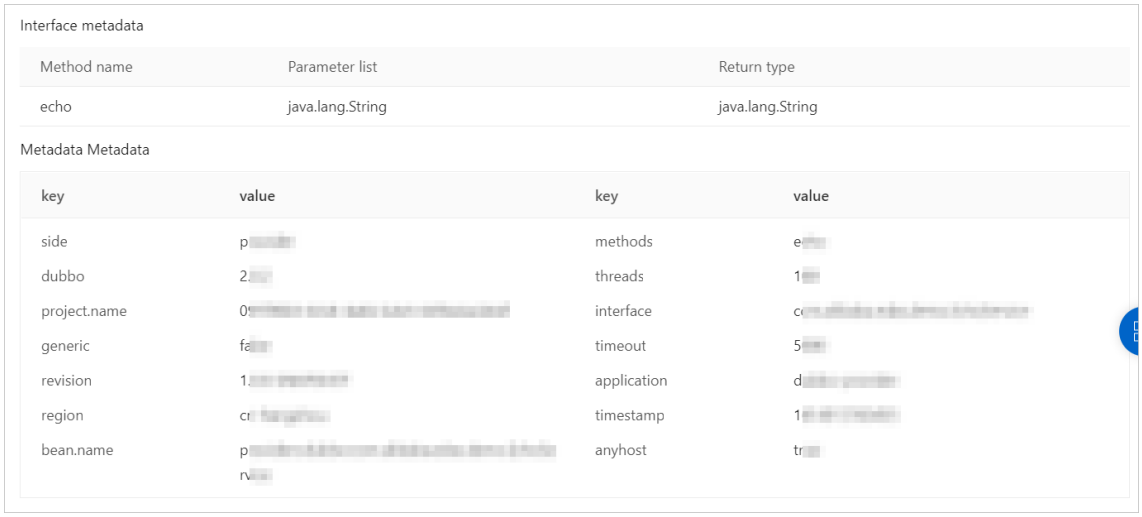
   - **Basic information**

- **Service invocation relationship**



The Service invocation relationship section provides the **Service Provider** and **Service Consumer** tabs, which list information such as **IP**, **Port**, **Serialization mode**, and **TimeOut (ms)**.

- **Metadata**



The Metadata section provides **Metadata Metadata** and **Interface metadata**.

- The **Metadata Metadata** section provides the metadata of the service and the EDAS-provided metadata for implementing microservice capabilities.
- The **Interface Metadata** section provides **Method name**, **Parameter list**, and **Return type**.

2.4. Query Dubbo service traces

You can log on to the EDAS console to the query the traces of Dubbo services that are deployed in EDAS.

EDAS is integrated with Application Real-Time Monitoring Service (ARMS). You can use ARMS to query service traces and holographic troubleshooting events.

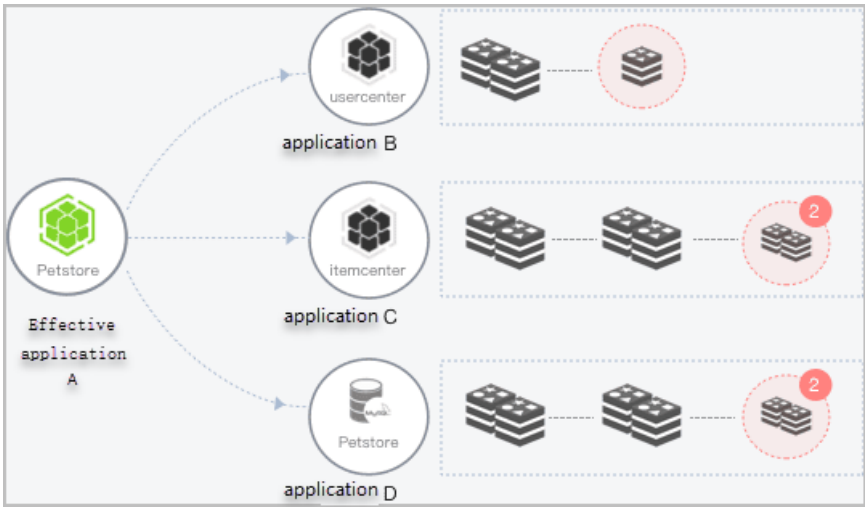
2.5. Ensure the availability of Dubbo applications by using outlier ejection

In a microservice framework, service calls are affected if consumers cannot detect the exceptions on the application instances of a provider. This further affects the performance and even availability of the services provided by the consumers. The outlier ejection feature monitors the availability of application instances and dynamically adjusts the instances. This ensures successful service calls and improves the service stability and quality of service (QoS).

Context

The following figure shows a system that requires outlier ejection. In this example, the system has Applications A, B, C, and D, among which Application A calls the instances of Applications B, C, and D. If some instances of Application B, C, or D become abnormal but are not identified by Application A, some calls initiated by Application A may fail. In this example, Application B has one abnormal instance, Application C has two abnormal instances, and Application D also has two abnormal instances. If a large number of instances are abnormal in Applications B, C, and D, the service performance and availability of Application A may be affected.

To ensure the service performance and availability of Application A, you can configure an outlier ejection policy for Application A. After the policy is configured, Enterprise Distributed Application Service (EDAS) can monitor the instance status of Applications B, C, and D, and dynamically add or remove instances to ensure successful service calls.



The following content describes the process of outlier ejection:

1. EDAS detects whether Application B, C, or D has abnormal instances. If abnormal instances are found, EDAS determines whether to remove the abnormal instances from the application based on the **Instance Removal Rate Threshold** parameter.
2. After the abnormal instances are removed, the call requests of Application A are no longer distributed to the removed instances.

3. EDAS detects whether the abnormal instances are recovered based on the **Recovery Detection Unit Time** parameter.
4. The detection interval linearly increases with the value of the **Recovery Detection Unit Time** parameter. The default value of Recovery Detection Unit Time is 30000 ms, which equals 0.5 minutes. If the threshold specified by the **Max Number of Instance Checked Before Restoration** parameter is reached, EDAS detects whether the abnormal instances are recovered at the maximum detection interval.
5. After the abnormal instances are recovered, EDAS adds the instances back to the application to process call requests. The detection interval is reset to the value of the **Recovery Detection Unit Time** parameter, such as 30000 ms.

> **Note**
> - If the ratio of abnormal instances of a provider exceeds the threshold that is specified by the Instance Removal Rate Threshold parameter, EDAS removes abnormal instances based on this threshold.
> - If the provider has only one instance available, EDAS does not remove this instance even if the threshold specified by the Error Rate Threshold parameter is exceeded.

Video

Create an outlier ejection policy

Verify the result

The outlier ejection feature is enabled after you configure and create an outlier ejection policy. You can go to the details page of the application for which you have configured outlier ejection to view the application monitoring information. For example, you can check whether call requests are still forwarded to abnormal instances and whether the error rate per minute for application calls is higher than the value of the **Error Rate Threshold** parameter in a topology. This way, you can check whether the outlier ejection policy takes effect.

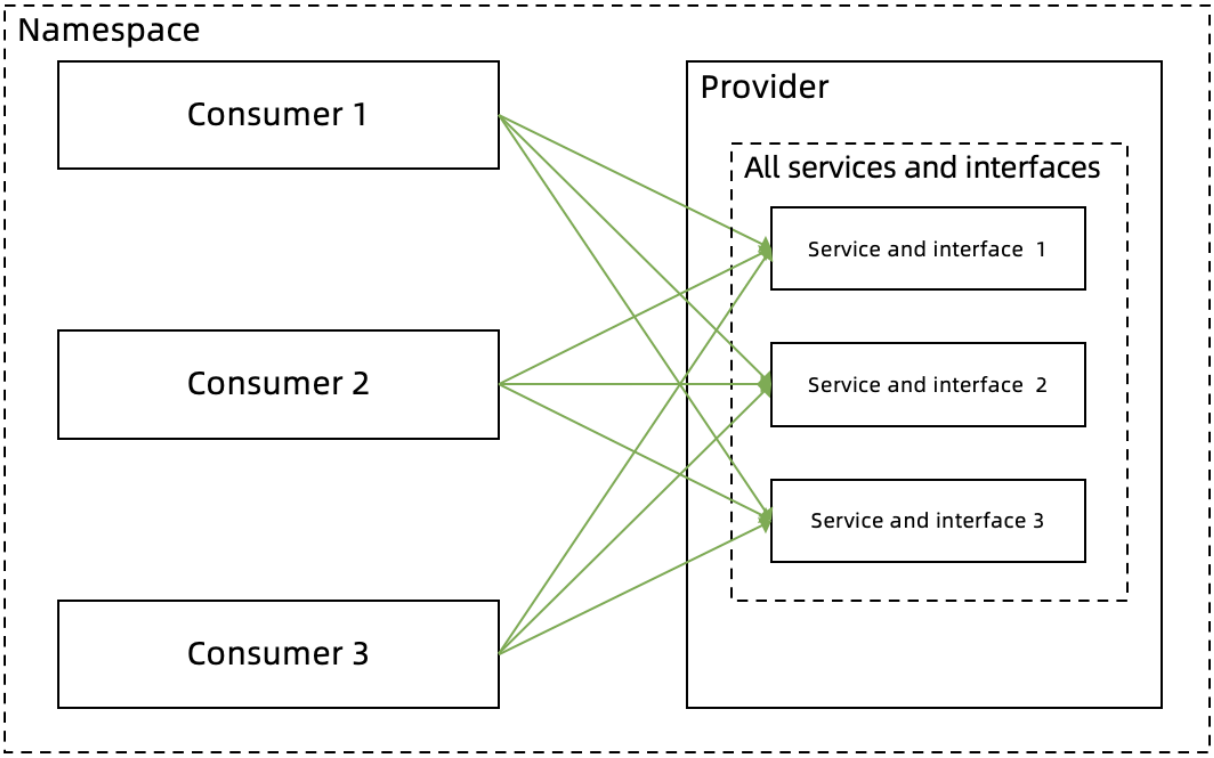
2.6. Implement access control of Dubbo applications through service authentication

If a microservice-oriented application requires high security and you want to restrict access to it from other applications, you can authenticate the applications that call the microservice-oriented application. This ensures that only the applications that match the authentication rules can call the microservice-oriented application.

Context

This topic uses an example to introduce scenarios where Dubbo service authentication is performed.

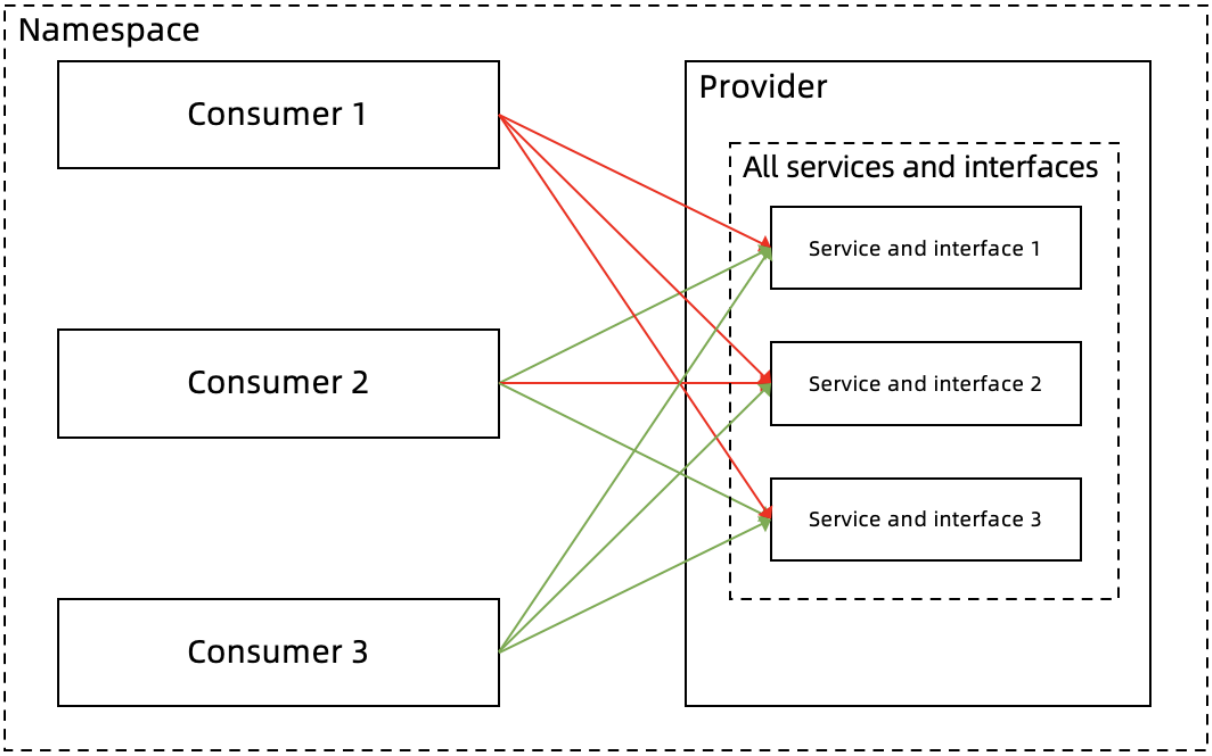
Consumers 1, 2, and 3 and a service provider are deployed in the same namespace. By default, Consumers 1, 2, and 3 can call all the services and interfaces of the provider.



You can specify an authentication method for all the services and interfaces of the provider. For example, set the authentication method to Blacklist (call denied) for Consumer 1 and set the authentication method to Whitelist (allow calls) for Consumer 2 and Consumer 3.

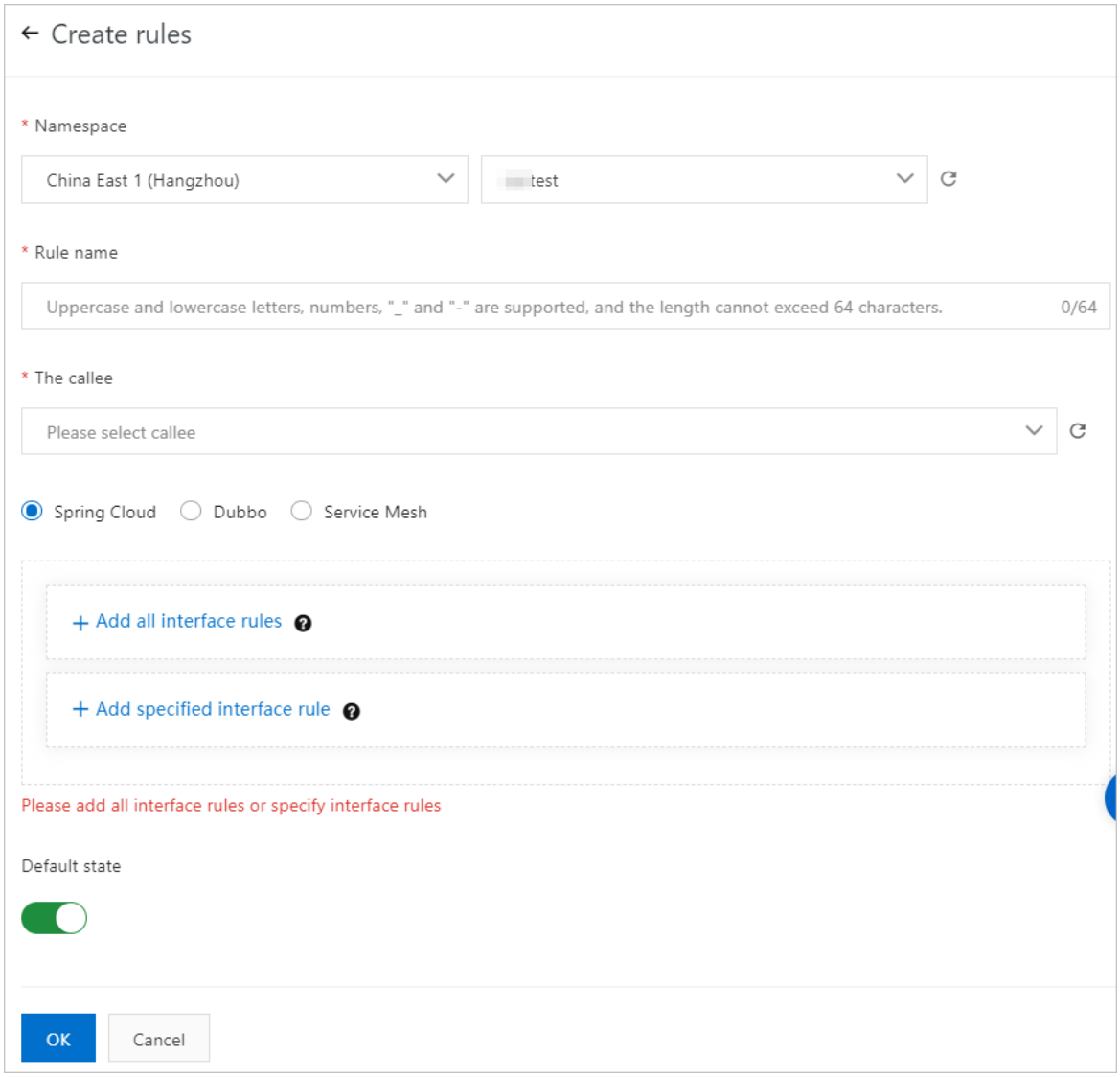
Then, you can also set an authentication method for specified services and interfaces of the provider. For example, after you apply the preceding settings, Consumer 2 and Consumer 3 can access all services and interfaces of the provider. However, Service and Interface 2 of the provider involves core business and data. To disable Consumer 2 from accessing Service and Interface 2, set the authentication method of Service and Interface 2 to Blacklist (call denied) for Consumer 2. This way, Consumer 2 can access only Service and Interface 1 and Service and Interface 3 of the provider.

The following figure shows the application call process after you configure the authentication rules.



Create a service authentication rule

1.
2.
3. In the left-side navigation pane of **Dubbo**, click **Service Authentication**.
4. On the **Service Authentication** page, click **Create rules**.
5. On the **Create rules** page, set service authentication parameters, and click **OK**.



Service authentication rule parameters:

Parameter	Description
Microservice Namespaces	The region and the microservice namespace where the service is deployed.
Rule name	The name of the service authentication rule. The name can be a maximum of 64 characters in length, and can contain letters, digits, underscores (_), and hyphens (-).
The callee	The called application.
Callee framework	The framework that is used by the called application. For this example, select **Dubbo**.
Add all interface rules **Notice** You can add only one global rule for all interfaces.	

Parameter	Description
Callee Path	Default value: **All services/all interfaces**. You cannot change the value of this parameter.
Authentication method	The service authentication method. Valid values: **Whitelist (allow calls)** and **Blacklist (call denied)**. Select an option as needed.
Caller	The caller application to be authenticated for calling the service. Click **Add caller** to select multiple applications.
Add specified interface rule **Notice** The rule added for a specific interface is not appended. Instead, the rule overwrites the common rule added for the interface. Exercise caution when you configure this parameter.	
Callee Interface	Specify the services and interfaces of the called application.
Authentication method	The service authentication method. Valid values: **Whitelist (allow calls)** and **Blacklist (call denied)**. Select an option as needed.
Caller	The caller application to be authenticated for calling the service. Click **Add caller** to select multiple applications.
Default State	Specifies whether to enable the rule. ○ On: The rule is enabled immediately after you create it. This is the default value. ○ Off: The rule is not enabled after it is created. To enable the rule, find the rule on the **Service Authentication** page and click **Open** in the **Operation** column.

Verify the results

After the service authentication rule is created and enabled, check whether the rule takes effect.

What's next

After you create a service authentication rule, you can click **Edit**, **Close**, or **Open** in the Operation column to manage the rule. If the service authentication rule is no longer required, you can click **Delete** in the Operation column to delete the rule.

2.7. Test a Dubbo service

Developers or testers need to call online services to debug deployed services or query online data during the development process. The service testing feature allows you to set the parameters to call a service, initiate service calls, and obtain the results of the calls in the Enterprise Distributed Application Service (EDAS) console.

Context

- The service testing feature is in public preview. You can use this feature free of charge.

- If you test a service as a Resource Access Management (RAM) user, you must first grant the RAM user the permissions to test services in the RAM console. For more information, see Configure permissions for service testing in the RAM console.

Procedure

1.
2. In the left-side navigation pane, choose **Microservices Governance > Dubbo**.
3. In the left-side navigation pane of **Dubbo**, click **Service Testing**.
4. In the top navigation bar, select a region. On the **Service Testing - Select Service** page, select a microservice namespace from the Microservice Namespace drop-down list. Then, click the name of a specific service in the service list.
5. In the **Interface metadata** section of the **Select Test Method** panel, find the service that you want to test and click **Test** in the **Request Path** column.
6. In the **Test Service** panel, set the parameters and click **Run**.

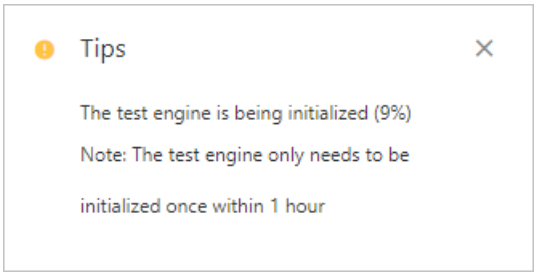
   The following table describes the parameters.

Parameter	Description
Call IP	The IP address of an instance on which the service is deployed. The instance can be an Elastic Compute Service (ECS) instance or a pod. If the service is deployed on multiple instances, you can select only the IP address of one instance.
Test Method	The test method to use. In the Script section, set the parameters based on the service code.

Verify the test result

In the **Result** section, you can check whether the test is successful. The following list describes the types of test results:

- **The test failed**, and the "`The test engine is being initialized.`" error message appears. When you perform a service test, the test engine requires 30 seconds to 50 seconds to initialize. Therefore, you must wait for 30 seconds to 50 seconds before you perform another test.

- **The test is successful**, and the response from the service appears.
- **The test failed**, and the response from the service appears. You can troubleshoot the issue based on the response to determine whether the issue is caused by the port, network, or code of the service.

2.8. Configure tag-based routing for a Dubbo application

The tag-based routing feature allows you to allocate one or more application instances to the same group by using tags. This way, you can forward traffic to application instances in specific groups. You can use the tag-based routing feature for blue-green release and canary release.

Context

The tag-based routing feature is available only for applications that are deployed in Elastic Compute Service (ECS) clusters.

Scenarios

- Multi-version development and testing

  If multiple versions are developed at the same time, you must prepare a development environment for each version. The costs of development environments are high. To reduce costs, you can use tag-based routing to implement end-to-end traffic adjustment.

  End-to-end traffic adjustment is implemented based on the tag-based routing feature. End-to-end traffic adjustment allows you to route specific traffic to a specific development environment. For example, if only Application B and Application D are modified in Development Environment 1, you can create Tag 1 for the versions of the two applications in Development Environment 1, and create a tag-based routing rule. This way, when Application A calls Application B, the system checks whether the traffic meets the conditions of the tag-based routing rule. If yes, the traffic is routed to Application B V1.1 in Development Environment 1. If no, the traffic is routed to Application B V1 in the baseline environment. When Application C calls Application D, the traffic is routed to Application D V1 or Application D V1.1 based on the tag-based routing rule.

- Traffic adjustment for a multi-version application

  If multiple versions of an application run online at the same time and are deployed in different environments, you can use the tag-based routing feature to isolate the traffic that is destined for different versions in different environments. For example, you can route the traffic of flash sale orders or the traffic of orders from different channels to the special environment, and route common traffic to the common environment. This way, the traffic destined for the special environment is not routed to the common environment even if exceptions occur in the special environment, and the common environment is not affected.

- A/B testing

  Multiple versions of an application run online at the same time. To perform A/B testing on the different versions, you can use end-to-end traffic adjustment to route the traffic that is initiated from Region A, such as the China (Hangzhou) region, to Application V1, and route the traffic that is initiated from Region B, such as the China (Shanghai) region, to Application V1.1. Then, you can verify the different versions. This helps reduce risks when you publish new products or features and facilitate product innovation.

Procedure

In this example, Application A is deployed in an ECS cluster. Application A has a default group, and the group contains three application instances.

Perform the following steps to configure tag-based routing:

1. Create tags: Create Group 1 and Group 2 for Application A, allocate one application instance in the default group to Group 1 and one application instance to Group 2, and then create tags for Group 1 and Group 2.
2. Create tag-based routing rules: Create tag-based routing rules based on the tags of Group 1 and Group 2.

After tag-based routing is configured, when Application B calls Application A, traffic is routed to Group 1 and Group 2 based on the tag-based routing rules. Other traffic that does not meet the conditions of the tag-based routing rules is routed to the default group.

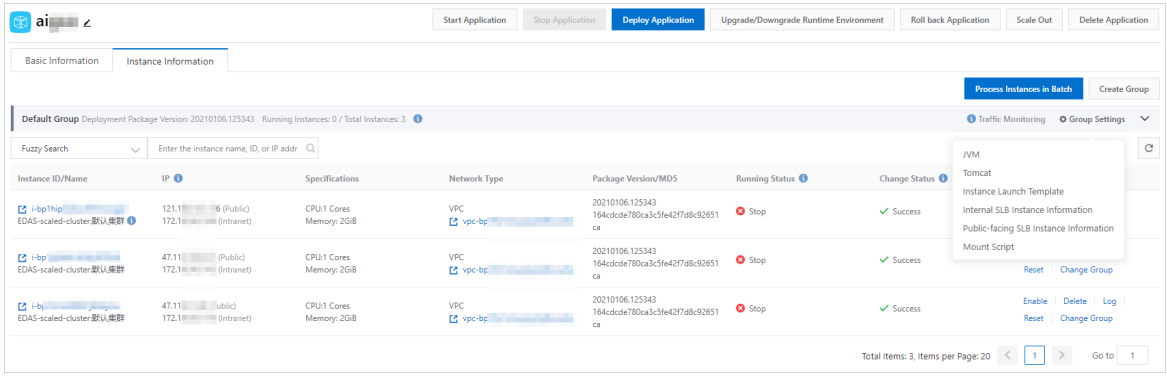
Create a tag

Create a group for an application and add an instance to the group. Then, select JVM from the Group Settings drop-down list to create a tag.

1.
2. Create a group for an application. For more information, see Create a group.
3. Add an application instance to the new group. You can allocate an application instance of the default group to the new group, or purchase a new instance. For more information, see Add an instance.

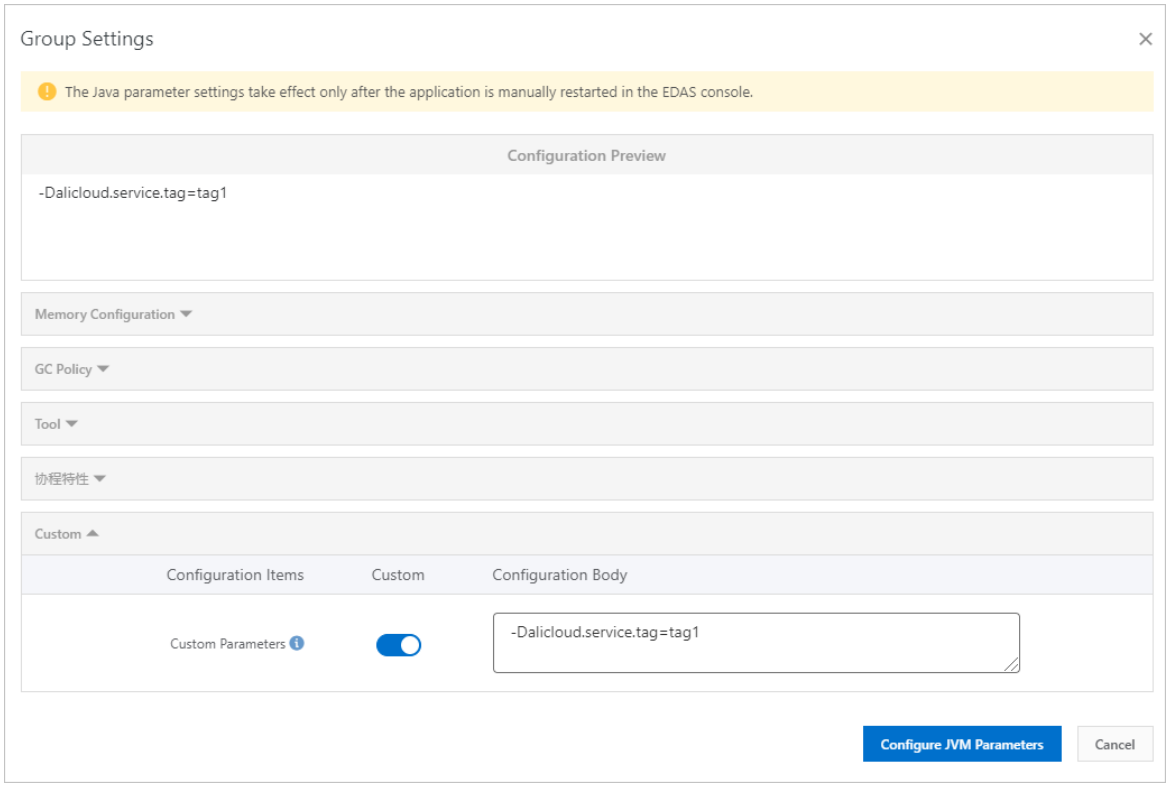
   > **Notice** After the application instance of the default group is allocated to the new group, you must restart the application instance. Otherwise, the system does not identify the tag. If you purchase a new instance for the new group, you do not need to restart the instance.

4. In the upper-right corner of the new group list, select **JVM** from the **Group Settings** drop-down list.



5. In the **Group Settings** dialog box, click **Custom**. In the **Custom** section, turn on the switch in the **Custom** column. In the **Configuration Body** field, configure a tag for the group and click **Configure JVM Parameters**.

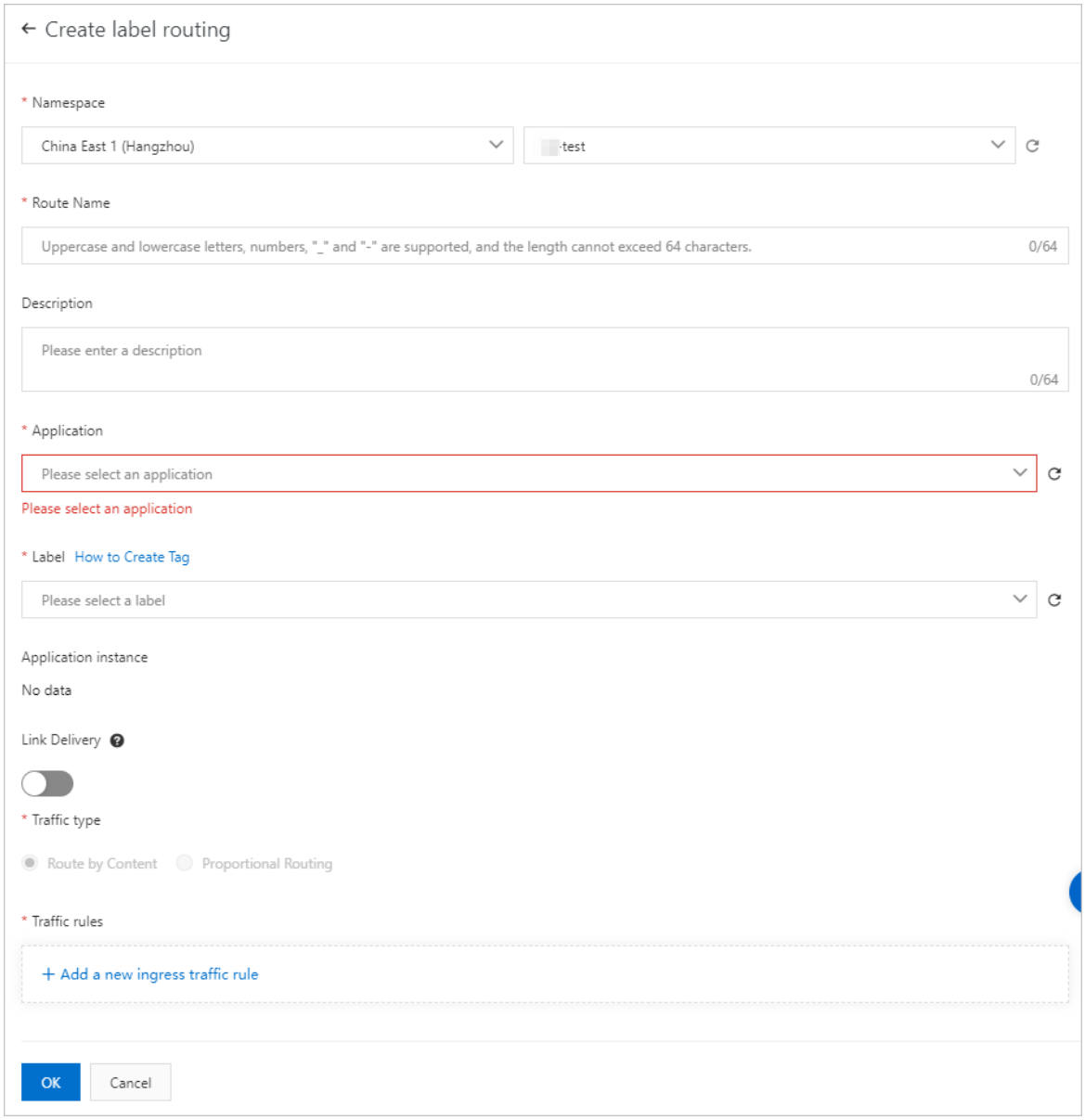
   You must configure a tag in the `Dalicloud.service.tag=tag1` format. *tag1* indicates the tag name. Set the tag name based on your business requirements.



Create a tag-based routing rule

After a tag is created, you can create a tag-based routing rule based on the tag.

1.
2. In the left-side navigation pane, choose **Microservices Governance > Dubbo**.
3. In the navigation tree of the **Dubbo** page, click **Tag-based Routing**.
4.
5. In the **Create label routing** panel, set the parameters and click **OK**.



The following table describes the parameters.

Parameter	Description
Namespace	Select a region and a namespace based on your business requirements.
Route Name	Enter a name for the tag-based routing rule. For example, you can enter `test`.
Description	Enter a description for the tag-based routing rule.
Application	Select an application from the drop-down list.

Parameter	Description
Label	Select a tag from the drop-down list. The tag is the value that you specify for the custom Java virtual machine (JVM) parameter -Dalicloud.service.tag when you create a group for the application. After you select the tag, the IP address and port number of the application instance in the group appear in the **Application instance** section.
Link Delivery	Turn on **Link Delivery** if you want to enable end-to-end traffic adjustment. **Note** The end-to-end traffic adjustment feature is in canary release. If you need to use the end-to-end traffic adjustment feature, join the DingTalk group whose ID is 31723701 to contact EDAS technical support.
Traffic rules	
Frame type	Select the framework of the application. Valid values: **Spring Cloud** and **Dubbo**. ○ Spring Cloud: You can specify only a URL path, such as `/getIp`. ○ Dubbo: You can select a specific service and an interface.
Conditional mode	Select **At the same time meet the following conditions** or **Meet any of the following conditions** based on your business requirements.
Condition list	Set the Condition and Value parameters based on the **Parameters** and **Parameter value get expression** parameters.

2.9. Configure a dynamic timeout period for Dubbo services

Enterprise Distributed Application Service (EDAS) allows you to configure a dynamic timeout period at the method level for a service. This way, the timeout period dynamically changes based on the interface response time. This improves service governance. This topic describes how to configure a dynamic timeout period for Dubbo services.

Prerequisites

The Dubbo services can be queried in the EDAS console. For more information, see Query Dubbo

services.

Context

You may configure timeout periods in various scenarios. If the business logic changes, you may need to adjust the existing calling relationship. The response time of a service interface can be determined only after the service is published. EDAS allows you to configure a dynamic timeout period for Dubbo services at the method level. This way, the timeout period for the services dynamically changes. This improves service availability.

Procedure

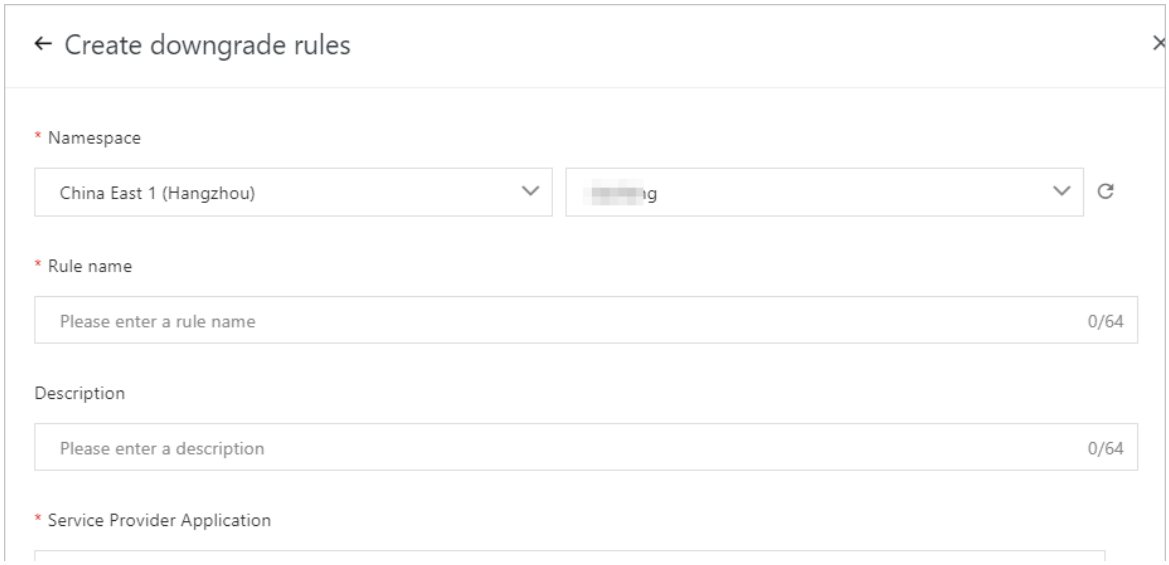
1.
2. In the left-side navigation pane, choose **Microservices Governance > Dubbo**.
3. In the top navigation bar, select a region. On the **Service Query** page, select an option from the microservice namespace drop-down list. Then, click a service name in the service list.
4.
5.

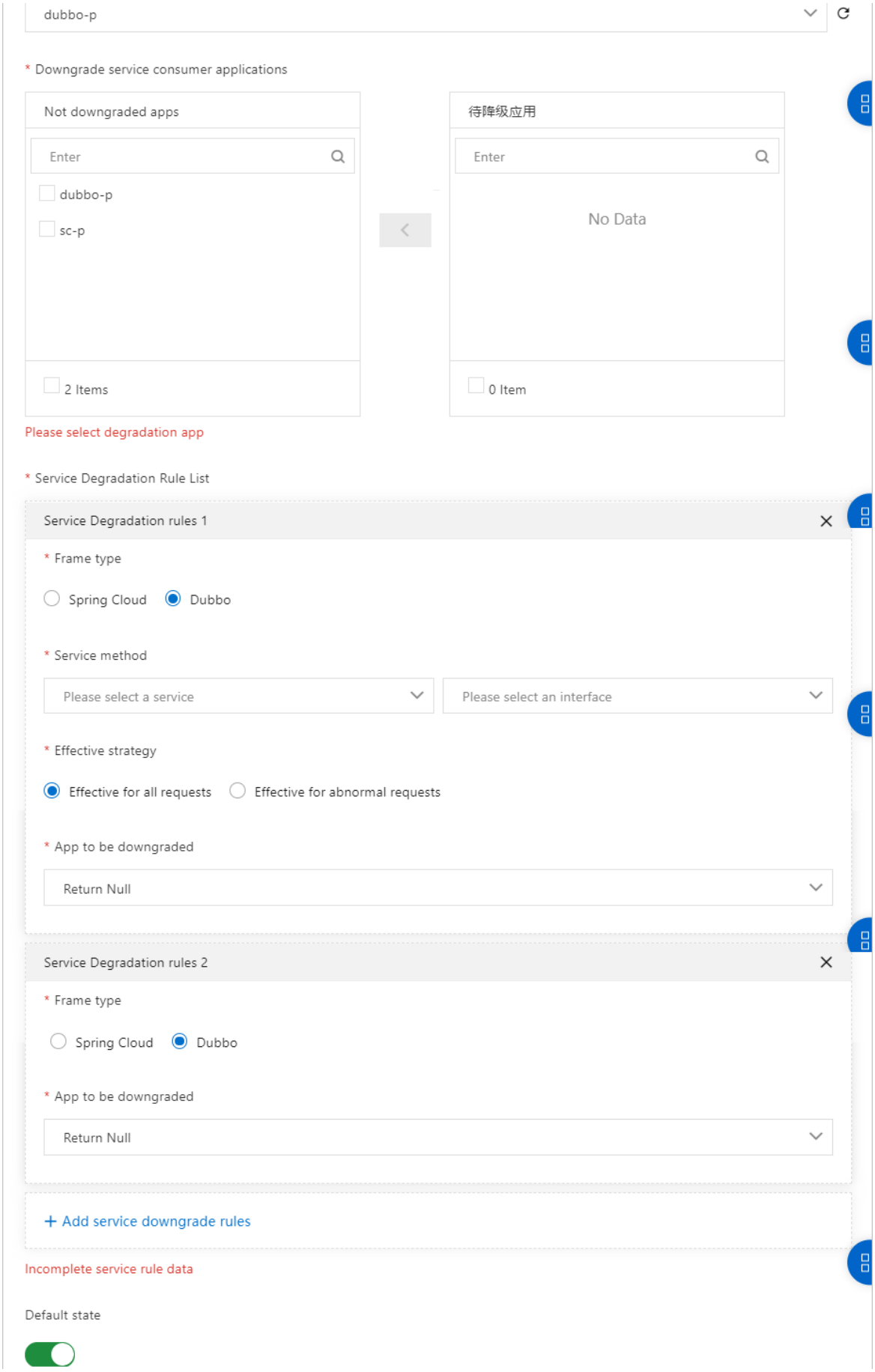
2.10. Configure a service degradation rule for a Dubbo application

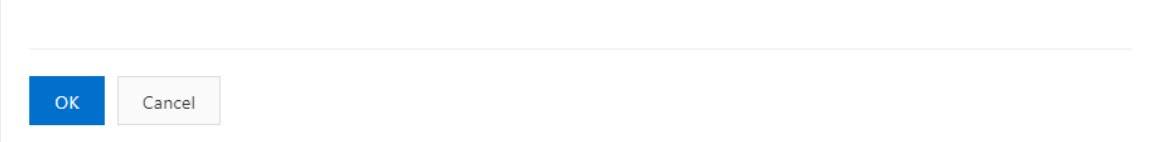
Context

Create a service degradation rule

1.
2. In the left-side navigation pane, choose **Microservices Governance > Dubbo**.
3. In the navigation tree of the **Dubbo** page, click **Service Degradation**.
4. In the top navigation bar, select a region. On the **Service Degradation** page, select a microservice namespace. Then, click **Create downgrade rules**.
5. In the **Create downgrade rules** panel, set the parameters and click **OK**.




dubbo-p
* Downgrade service consumer applications
Not downgraded apps
Enter
dubbo-p
sc-p
2 Items
待降级应用
Enter
No Data
0 Item
Please select degradation app
* Service Degradation Rule List
Service Degradation rules 1
* Frame type
Spring Cloud
Dubbo
* Service method
Please select a service
Please select an interface
* Effective strategy
Effective for all requests
Effective for abnormal requests
* App to be downgraded
Return Null
Service Degradation rules 2
* Frame type
Spring Cloud
Dubbo
* App to be downgraded
Return Null
+ Add service downgrade rules
Incomplete service rule data
Default state



The following table describes the parameters.

Parameter	Description
Microservice Space	Select the region and microservice namespace where the application resides.
Rule name	Enter a name for the service degradation rule. The name can be up to 64 characters in length, and can contain letters, digits, underscores (_), and hyphens (-).
Description	Enter a description for the service degradation rule.
Service Provider Application	Select the application that can be called by other applications.
Downgrade service consumer applications	Select the application that you want to downgrade.
Service Degradation Rule List	Click **Add service downgrade rules** to create a service degradation rule.
Frame type	Select **Dubbo**.
Service method	Select the application that you select from the Service Provider Application drop-down list and select the interface that is used to call the application.
Effective strategy	Select the policy based on which the service degradation rule takes effect. Valid values: **Effective for all requests** and **Effective for abnormal requests**.
App to be downgraded	Select the policy for the service degradation rule. If the rule is triggered, the specified content is returned. Valid values: **Return Null**, **Return Exception**, and **Return custom Json data**.
Default state	Turn on or off the switch to enable or disable the rule. ○ On: enables the rule after it is created. By default, the switch is turned on. ○ Off: disables the rule after it is created. To enable the rule, find the rule on the **Service Degradation** page and click **Enable** in the **Operation** column.

Result

What's next

2.11. End-to-end traffic adjustment

2.11.1. Overview

In Kubernetes clusters, Enterprise Distributed Application Service (EDAS) supports the end-to-end traffic adjustment feature for Dubbo microservice-oriented applications. The end-to-end traffic adjustment feature helps you create a traffic adjustment environment with ease and route traffic that has specific characteristics to applications of a specified version.

Background information

In EDAS, some Dubbo applications that are deployed in Kubernetes clusters may be updated to a specific version. In this case, traffic that has specific characteristics may fail to be routed to applications of a specified version. This is because applications call each other randomly. The end-to-end traffic adjustment feature can help you isolate applications of a version from others in a lane, which is an independent runtime environment. You can configure traffic adjustment rules in the lane to route the request traffic that meets the rules to applications of the specified version.

This section describes how to use the end-to-end traffic adjustment feature in the order placement scenario of an e-commerce architecture.

After a customer places an order, the traffic comes in from the ingress application, which can also be a microservice gateway. The ingress application calls the transaction center, the transaction center calls the commodity center, and then the commodity center calls the downstream inventory center.

Both the transaction center and the commodity center are running in new versions V1.0 and V2.0. The two versions need to be verified during a canary release. At this time, you want to route the request traffic that meets specific traffic adjustment rules in the ingress application to applications of the new versions, and route all the remaining traffic to applications of the online version, which is the official version.

Terms

- Ingress application

  The ingress of traffic in a microservice system. An ingress application can be a service gateway that is built based on Spring Cloud Gateway or Spring Cloud Netflix Zuul, or a Spring Boot, Spring MVC, or Dubbo application.

- Lane

  An isolated environment that is defined for applications of the same version. Only the request traffic that meets the traffic adjustment rules of a lane can be routed to the applications that are configured to receive the marked traffic in the lane. An application can belong to multiple lanes. A lane can contain multiple applications. Applications have a many-to-many relationship with lanes.

- Lane group

  A collection of lanes. A lane group is used to distinguish different teams or different scenarios.

Limits

- After you configure the end-to-end traffic adjustment feature for applications, these applications no longer support a canary release.
- If you want to build an ingress gateway based on Spring Cloud Gateway, make sure that the version of Spring Cloud Gateway is 2.1.x or later.
- The quotas of lane groups and lanes vary based on the edition of EDAS. The following limits apply:
  - Standard Edition: All regions can contain only one lane group. This lane group can contain up to five lanes.

    All editions other than Professional Edition and Platinum Edition are Standard Edition.
  - Professional Edition: All regions can contain a maximum of 10 lane groups. Each lane group can contain up to 50 lanes.
  - Platinum Edition: All regions can contain a maximum of 10 lane groups. Each lane group can contain up to 50 lanes.
- The quotas of lane groups and lanes cannot be increased.

  If you want to increase the quotas of lane groups and lanes, submit a ticket.

2.11.2. Use the end-to-end traffic adjustment feature to monitor the traffic of an ingress application

After a lane group is created, you can directly access applications and monitor the inbound traffic of an ingress application.

Prerequisites

Before you configure end-to-end traffic adjustment for applications, make sure that the following prerequisites are met:

- Microservice-oriented applications are created. For more information, see Overview.

  An ingress application is available. For example, a service gateway is built based on Spring Cloud Gateway or Spring Cloud Netflix Zuul. The service gateway is associated with a microservice namespace of Enterprise Distributed Application Service (EDAS).

  If you want to build an ingress gateway based on Spring Cloud Gateway, make sure that the version of Spring Cloud Gateway is 2.1.x or later.
- A Server Load Balancer (SLB) instance is bound to the ingress application. For more information, see Bind SLB instances or Reuse an SLB instance.

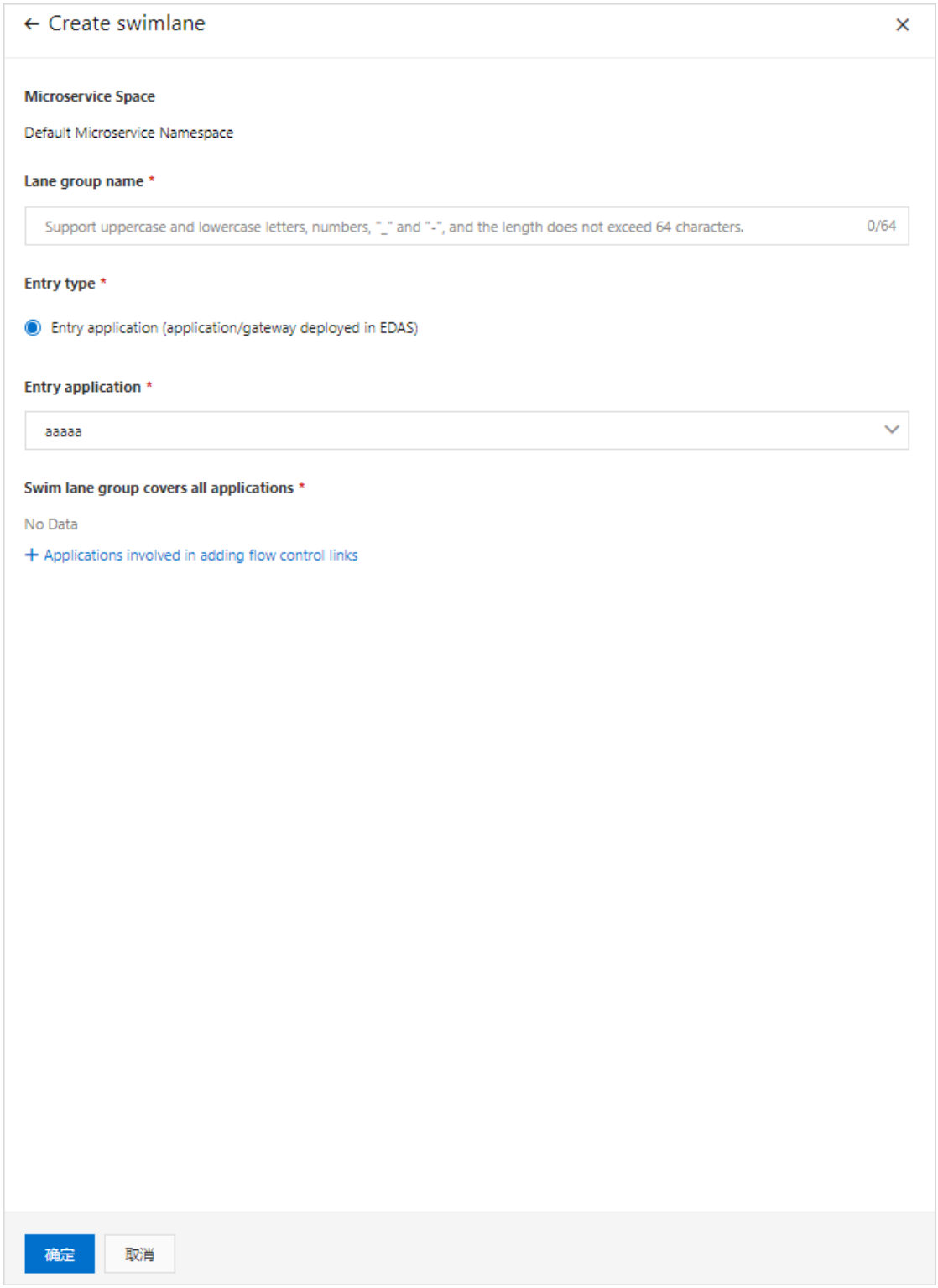
Create a lane group

1.
2. In the left-side navigation pane, choose **Microservices Governance > End-to-end Traffic Adjustment**.
3. In the top navigation bar, select a region. On the **End-to-end Traffic Adjustment** page, select a microservice namespace.
4. In the lower part of the **End-to-end Traffic Adjustment** page, click **Create Lane Groups and Lanes**.

If a lane group is created in the selected microservice namespace, click **Create** to the right of the **Select swim lane group** field.

> **Note** A microservice namespace can contain up to two lane groups.

5. In the **Create swimlane** panel, set relevant parameters and click **OK**.



Parameter	Description
Microservice Space	The microservice namespace that you select on the **End-to-end Traffic Adjustment** page. The selected microservice namespace cannot be changed.
Lane group name	The name of the lane group. The name can be up to 64 characters in length, and can contain letters, digits, hyphens (-), and underscores (_).
Entry type	The type of the ingress application. Default value: **Entry application (application/gateway deployed in EDAS)**. **Note** EDAS allows you to use a service gateway that is built based on Spring Cloud Gateway or Spring Cloud Netflix Zuul as an ingress application. The service gateway must be associated with a microservice namespace of EDAS.
Entry application	This parameter is displayed only when the **Entry type** parameter is set to **Entry application (application/gateway deployed in EDAS)**.
Swim lane group covers all applications	Click **Applications involved in adding flow control links** and select all applications that are involved based on the ingress application or ingress gateway that you select.

After the lane group is created, the involved applications of the created lane group are displayed in the **Swim lane group involves applications** section of the **End-to-end Traffic Adjustment** page. Check whether the ingress application and the involved applications are properly selected. To modify the information about the lane group, click **Edit** and modify the information as needed.

Monitor the traffic of an ingress application

1. Obtain the endpoint of the SLB instance that is bound to the ingress application or ingress gateway that you want to manage.
   i. On the **Applications** page, click the name of the ingress application or ingress gateway.
   ii. In the **Access configuration** section of the **Application Overview** page, copy the endpoint of the SLB instance.
2. Use a browser or other tools to access an involved application of the lane group multiple times.

   In this example, the transaction center is accessed in a browser and the traffic is routed in different ways. For more information about how to route traffic to specific applications, see Use the end-to-end traffic adjustment feature to route traffic to specific applications.

```
Case 1: A [172.20.**.**] -> B1 [172.20.**.**] -> C [172.20.**.**]
Case 2: A [172.20.**.**] -> B [172.20.**.**] -> C [172.20.**.**]
Case 3: A2 [172.20.**.**] -> B [172.20.**.**] -> C [172.20.**.**]
.......
```

3. View the traffic monitoring chart of the ingress application.
   i. On the **End-to-end Traffic Adjustment** page, select the lane group whose data you want to view.
   ii. On the **End-to-end Traffic Adjustment** page, select a monitoring time range. The traffic monitoring chart is refreshed in the **Ingress Application Monitoring (total)** section.

   In the traffic monitoring chart, you can view the queries per second (QPS) at a specific point in time.

2.11.3. Use the end-to-end traffic adjustment feature to route traffic to specific applications

In Enterprise Distributed Application Service (EDAS), you can configure the end-to-end traffic adjustment feature for Spring Cloud and Dubbo microservice-oriented applications that are deployed in Kubernetes clusters. This feature helps you route traffic that has specific characteristics to applications of a specified version.

Prerequisites

Before you configure end-to-end traffic adjustment for applications, make sure that the following prerequisites are met:

- Applications of a new version are deployed, or applications are updated. For more information, see Overview of application upgrades and rollbacks (applicable to Kubernetes clusters).
- An ingress application is available. For example, a service gateway is built based on Spring Cloud Gateway or Spring Cloud Netflix Zuul. The service gateway is associated with a microservice namespace of EDAS.

  If you want to build an ingress gateway based on Spring Cloud Gateway, make sure that the version of Spring Cloud Gateway is 2.1.x or later.
- A Server Load Balancer (SLB) instance is bound to the ingress application. For more information, see Bind SLB instances or Reuse an SLB instance.

Context

This section describes how to use the end-to-end traffic adjustment feature in the order placement scenario of an e-commerce architecture.

After a customer places an order, the traffic comes in from the ingress application, which can also be a microservice gateway. The ingress application calls the transaction center, the transaction center calls the commodity center, and then the commodity center calls the downstream inventory center.

Both the transaction center and the commodity center are running in new versions V1.0 and V2.0. The two versions need to be verified during a canary release. At this time, you want to route the request traffic that meets specific traffic adjustment rules in the ingress application to applications of the new versions, and route all the remaining traffic to applications of the online version, which is the official version.

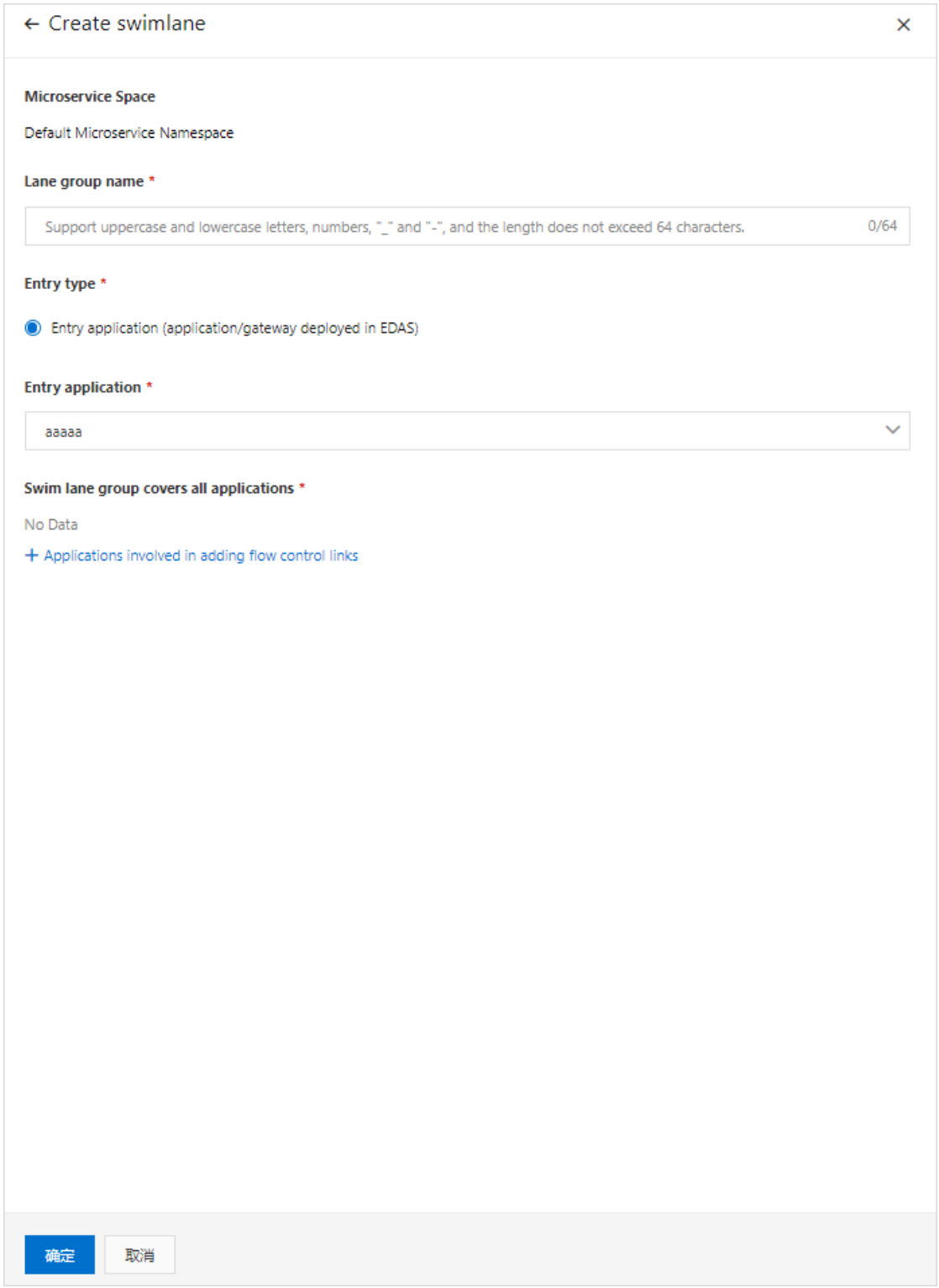
Create a lane group

1.
2. In the left-side navigation pane, choose **Microservices Governance > End-to-end Traffic Adjustment**.
3. In the top navigation bar, select a region. On the **End-to-end Traffic Adjustment** page, select a microservice namespace.
4. In the lower part of the **End-to-end Traffic Adjustment** page, click **Create Lane Groups and Lanes**.

   If a lane group is created in the selected microservice namespace, click **Create** to the right of the **Select swim lane group** field.

   > **Note** A microservice namespace can contain up to two lane groups.

5. In the **Create swimlane** panel, set relevant parameters and click **OK**.



Parameter	Description
Microservice Space	The microservice namespace that you select on the **End-to-end Traffic Adjustment** page. The selected microservice namespace cannot be changed.

Parameter	Description
Lane group name	The name of the lane group. The name can be up to 64 characters in length, and can contain letters, digits, hyphens (-), and underscores (_).
Entry type	The type of the ingress application. Default value: **Entry application (application/gateway deployed in EDAS)**. **Note** EDAS allows you to use a service gateway that is built based on Spring Cloud Gateway or Spring Cloud Netflix Zuul as an ingress application. The service gateway must be associated with a microservice namespace of EDAS.
Entry application	This parameter is displayed only when the **Entry type** parameter is set to **Entry application (application/gateway deployed in EDAS)**.
Swim lane group covers all applications	Click **Applications involved in adding flow control links** and select all applications that are involved based on the ingress application or ingress gateway that you select.

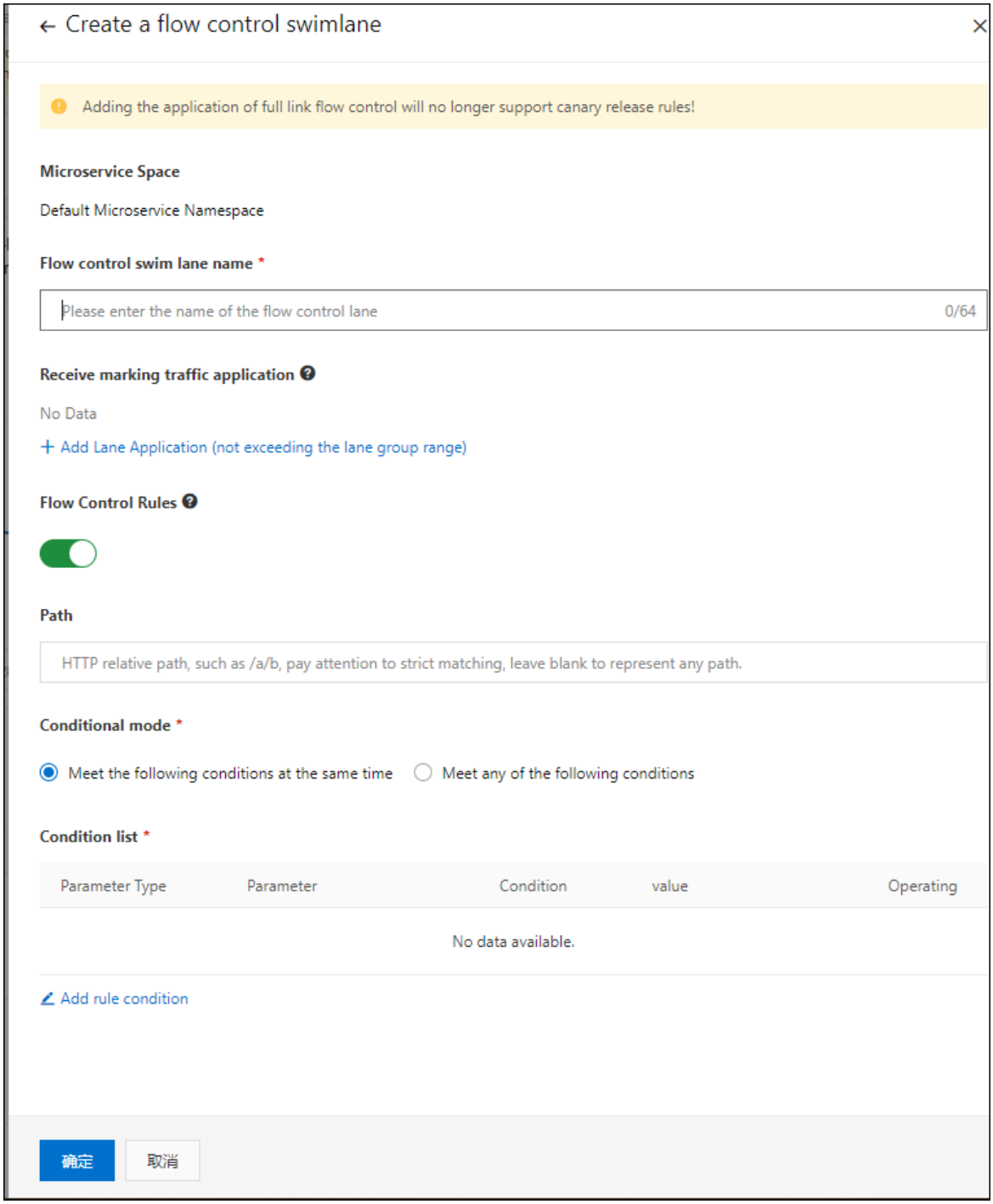
After the lane group is created, the involved applications of the created lane group are displayed in the **Swim lane group involves applications** section of the **End-to-end Traffic Adjustment** page. Check whether the ingress application and the involved applications are properly selected. To modify the information about the lane group, click **Edit** and modify the information as needed.

Create a lane

1. On the **End-to-end Traffic Adjustment** page, select the same microservice namespace as the lane group that you created. Then, click **Click to Create the First Split Lane** in the lower part of the page.

   Notice After you configure the end-to-end traffic adjustment feature for applications, these applications no longer support a canary release.

2. In the **Create a flow control swimlane** panel, set relevant parameters and click **OK**.



Parameter	Description
Microservice Space	The microservice namespace that you select on the **End-to-end Traffic Adjustment** page. Make sure that your lane group is created in the same microservice namespace. The selected microservice namespace cannot be changed.
Flow control swim lane name	The name of the lane. The name can be up to 64 characters in length, and can contain letters, digits, hyphens (-), and underscores (_).

Parameter	Description
(Optional) Receive marking traffic application	Click **Add Lane Application (not exceeding the lane group range)** and select an application in the lane group. **Note** - You can select multiple applications for the same lane. You can also create a lane for each application. - A lane group can contain up to five lanes. - When you create a lane, you can skip the selection of applications that receive the marked traffic. Then, you can select such applications when you modify the created lane.
Flow Control Rules	
Switch	Specifies whether to enable traffic adjustment rules. By default, the switch is turned on.
Path	The HTTP relative path. If you leave this parameter empty, the rules take effect for all paths. Set this parameter based on your business requirements.
Conditional mode	The mode in which conditions are met. Select a mode based on your needs. Valid values: **Meet the following conditions at the same time** and **Meet any of the following conditions**. - **Meet the following conditions at the same time**: The rules take effect when all the conditions are met at the same time. - **Meet any of the following conditions**: The rules take effect when one of the conditions is met.

Parameter	Description
Condition list	Click **Add rule condition**. You can add multiple conditions as needed. You can set different types of conditions, such as *Cookie*, *Header*, *Parameter*, and *Body Content*. In this example, the following conditions are added: ○ The parameter type is *Parameter* and the condition is env=red. When the condition is met, the traffic is routed to the applications that are running in V1.0. ○ The parameter type is *Parameter* and the condition is env=blue. When the condition is met, the traffic is routed to the applications that are running in V2.0.

After the lane is created, the created lane appears in the **Flow Control Distribution** section of the **End-to-end Traffic Adjustment** page. Check whether the lane name, traffic adjustment rules, and applications that receive the marked traffic are correct. To modify the information about the lane, click **Edit** and modify the information as needed.

3. (Optional) To create more lanes, click **Create swimlane** in the **Flow Control Distribution** section and set relevant parameters.

> **Note** A lane group can contain up to five lanes.

Verify that traffic is routed to specific applications

1.
2. Use a browser or other tools to access an application in a lane of the lane group multiple times.

   For example, enter *http://ip:prt/**?env=red* in the address bar of a browser to access the transaction center. The traffic is routed in only one way. This indicates that the traffic that has specific characteristics is routed to the specified applications.

   In the URL that you entered, ** is the path that you specify in traffic adjustment rules, and *env=red* is a condition in the traffic adjustment rules.

```
A2[172.20.**.**] -> B2[172.20.**.**] -> C[172.20.**.**]
```

3. View the traffic monitoring charts of applications that receive the marked traffic.
   i. On the **End-to-end Traffic Adjustment** page, select the lane group whose data you want to view.
   ii. On the **End-to-end Traffic Adjustment** page, select a monitoring time range. The traffic monitoring data is refreshed in the **Ingress Application Monitoring (total)** and **Flow Control Distribution** sections.

   View the traffic monitoring charts. You can find that the request traffic is routed to the applications that receive the marked traffic and the following condition is met: *Total queries per second (QPS) in the ingress application = QPS in the applications that receive the unmarked traffic + QPS in the applications that receive the marked traffic*.

View the traffic monitoring charts of all applications

In addition to the traffic monitoring charts of the ingress application, applications that receive the unmarked traffic, and applications that receive the marked traffic, you can also view the traffic monitoring charts of all applications in the same lane group. You can compare the traffic monitoring charts of all applications to analyze more useful information. Examples:

- Find the applications that are called at the same time.
- Analyze traffic escape issues and find out the escaped traffic.

3.HSF service governance

3.1. Query HSF services

You can log on to the Enterprise Distributed Application Service (EDAS) console to query the service list and service details of High-Speed Service Framework (HSF) applications that are deployed in EDAS.

View services

1.
2. In the left-side navigation pane, choose **Microservices Governance > HSF**.
3. In the navigation tree of the **HSF** page, click **Service Query**.
4. In the top navigation bar, select a region. On the **Service Query** page, select a microservice namespace and view the **HSF** services within the current account.

   The following information of HSF services is displayed: **Service name**, **Version**, **Grouping**, **Application Name**, and **Number of instances**.

   If a large number of services exist, you can filter services by service name, IP address, or application name. Filter keywords are not case-sensitive. The value of **IP** varies between ECS and Kubernetes clusters.

   - ECS cluster: The value is the IP address of an ECS instance.
   - Kubernetes cluster: The value is the IP address of a pod.

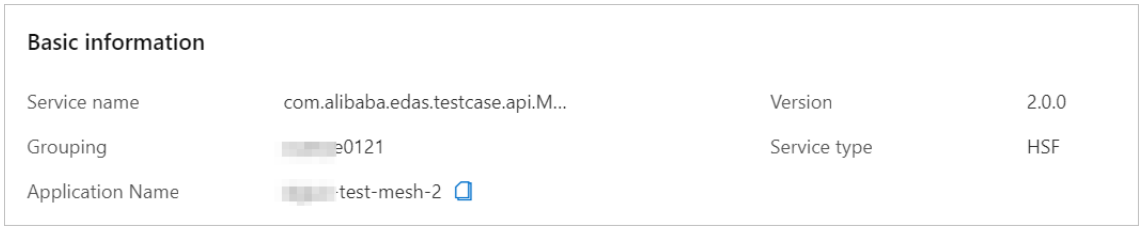
   > **Note** If you can query the services of your applications on the earlier Service Query page but cannot query the services on the new Service Query page, perform the following steps to troubleshoot the issue:
   >
   > The new Service Query page was released at 00:00:00 on January 20, 2020. You must restart your applications after this point in time so that they can be automatically mounted with the latest EDAS Agent. Therefore, restart your applications before you query services on the new Service Query page.

View service details

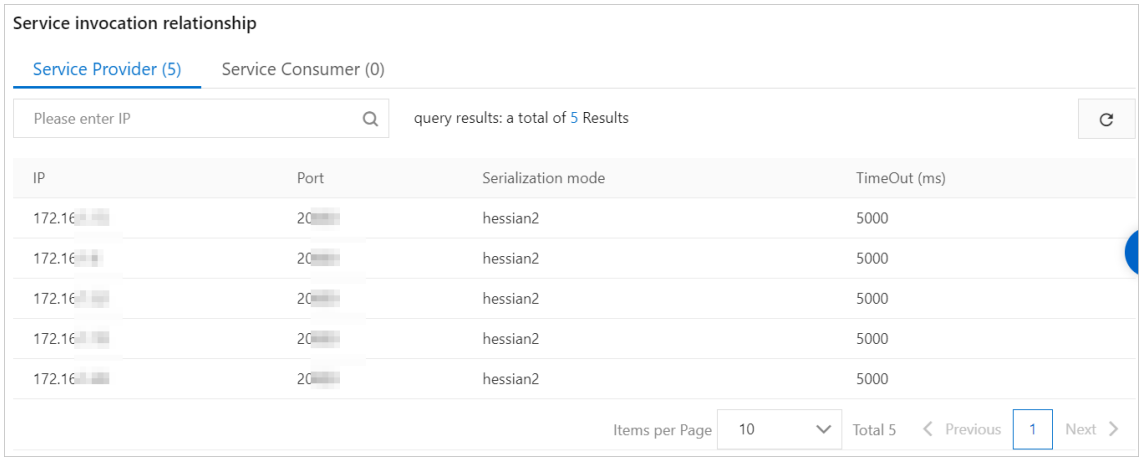
1.
2. In the left-side navigation pane, choose **Microservices Governance > HSF**.
3. In the navigation tree of the **HSF** page, click **Service Query**.
4. In the top navigation bar, select a region. On the **Service Query** page, select a microservice namespace and click a specific service name in the service list.
5. On the **Service Details** page, view the details of the service.

   The **Service Details** page contains the **Basic information** and **Service invocation relationship** sections.

   - **Basic information**



- **Service invocation relationship**



The Service invocation relationship section contains the **Service Provider** and **Service Consumer** tabs. The following information is displayed on the tabs: **IP**, **Port**, **Serialization mode**, and **TimeOut**.

3.2. Query the traces of HSF services

You can log on to the Enterprise Distributed Application Service (EDAS) console to query the traces of High-Speed Service Framework (HSF) services that are deployed in EDAS.

EDAS is integrated with Application Real-Time Monitoring Service (ARMS). You can use ARMS to query service traces and holographic troubleshooting events.

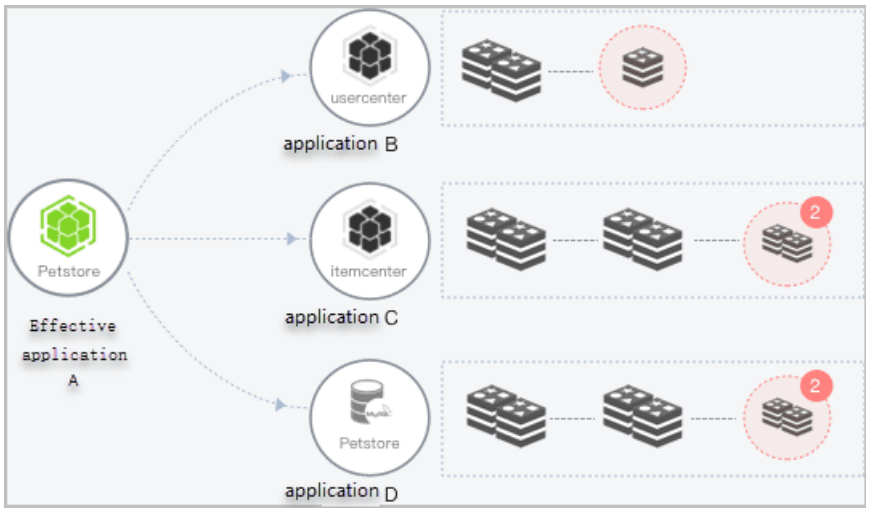
3.3. Ensure the availability of HSF applications by removing outlier instances

In a microservice framework, service calls are affected when service consumers cannot perceive abnormal application instances of service providers. This further affects the serviceability and availability of service consumers. The outlier instance removal feature monitors the availability of High-Speed Service Framework (HSF) applications and service instances and dynamically adjusts them. This ensures successful service calls and improves service stability and quality of service (QoS).

Context

The following figure shows a system that requires outlier ejection. In this example, the system has Applications A, B, C, and D, among which Application A calls the instances of Applications B, C, and D. If some instances of Application B, C, or D become abnormal but are not identified by Application A, some calls initiated by Application A may fail. In this example, Application B has one abnormal instance, Application C has two abnormal instances, and Application D also has two abnormal instances. If a large number of instances are abnormal in Applications B, C, and D, the service performance and availability of Application A may be affected.

To ensure the service performance and availability of Application A, you can configure an outlier ejection policy for Application A. After the policy is configured, Enterprise Distributed Application Service (EDAS) can monitor the instance status of Applications B, C, and D, and dynamically add or remove instances to ensure successful service calls.



The following content describes the process of outlier ejection:

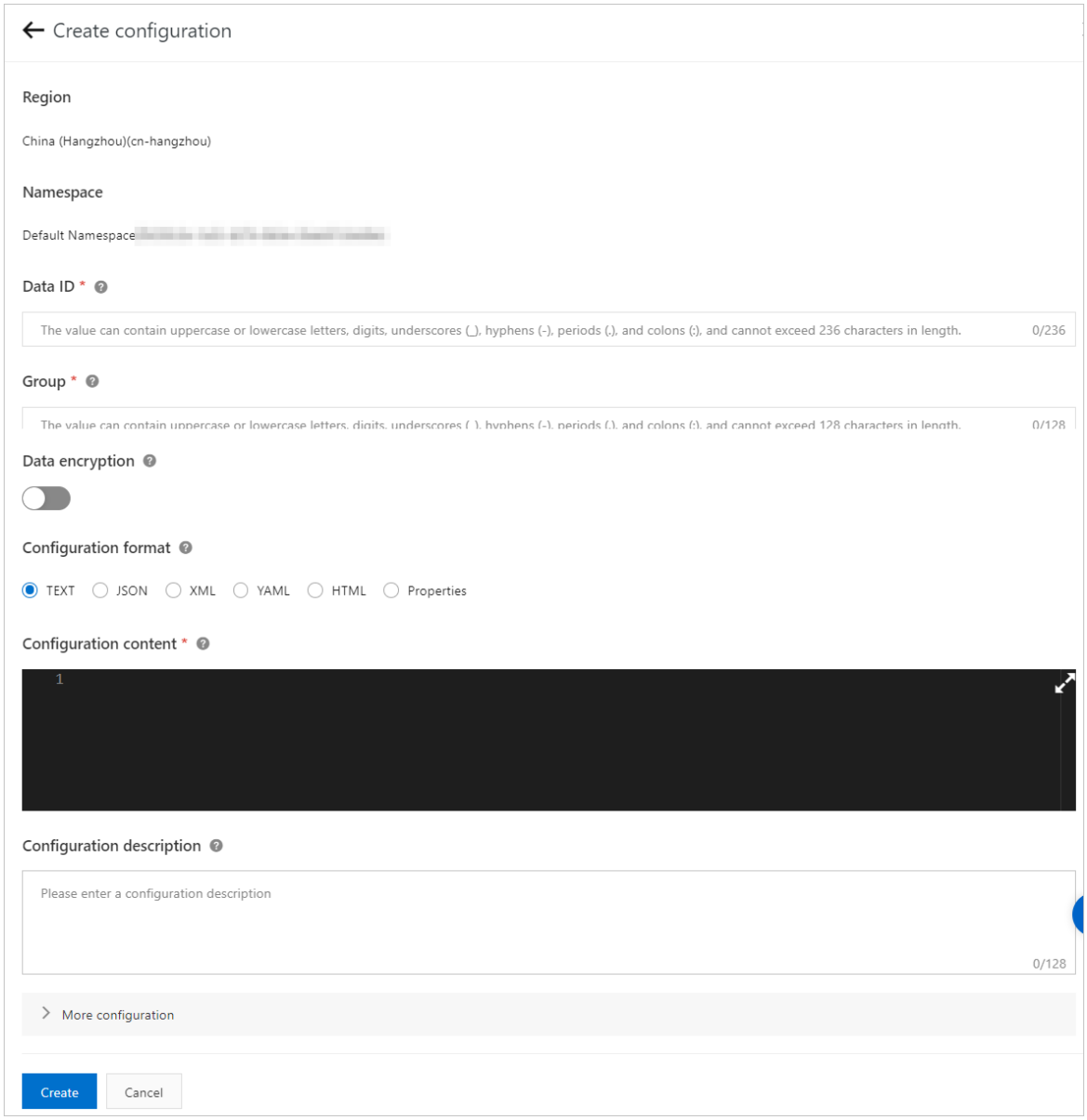
1. EDAS detects whether Application B, C, or D has abnormal instances. If abnormal instances are found, EDAS determines whether to remove the abnormal instances from the application based on the **Instance Removal Rate Threshold** parameter.
2. After the abnormal instances are removed, the call requests of Application A are no longer distributed to the removed instances.
3. EDAS detects whether the abnormal instances are recovered based on the **Recovery Detection Unit Time** parameter.
4. The detection interval linearly increases with the value of the **Recovery Detection Unit Time** parameter. The default value of Recovery Detection Unit Time is 30000 ms, which equals 0.5 minutes. If the threshold specified by the **Max Number of Instance Checked Before Restoration** parameter is reached, EDAS detects whether the abnormal instances are recovered at the maximum detection interval.
5. After the abnormal instances are recovered, EDAS adds the instances back to the application to process call requests. The detection interval is reset to the value of the **Recovery Detection Unit Time** parameter, such as 30000 ms.

> **Note**
>
> - If the ratio of abnormal instances of a provider exceeds the threshold that is specified by the Instance Removal Rate Threshold parameter, EDAS removes abnormal instances based on this threshold.
> - If the provider has only one instance available, EDAS does not remove this instance even if the threshold specified by the Error Rate Threshold parameter is exceeded.

Create an outlier instance removal policy

For HSF applications, you can create application- and service-level outlier instance removal policies.

1.
2. In the left-side navigation pane, choose **Microservice Configurations > Configurations**.
3. In the top navigation bar, select a region. On the **Configurations** page, select a microservice namespace from the Microservice Namespace drop-down list. Then, click **Create configuration**.
4. In the **Create configuration** panel, set the parameters. Then, click **Create** in the lower part of the panel.



The following section describes the parameters for creating an outlier instance removal policy:

- **Region**: The value is the region that you select before you create the outlier instance removal policy and cannot be changed.
- **Micro service space**: The value is the namespace that you select before you create the outlier instance removal policy and cannot be changed.
- **Data ID**: Enter an ID for the outlier instance removal policy in the format of `<Application ID>.QOSCONFIG`. You can obtain the ID of an application on the details page of the application.
- **Group**: The value is HSF and cannot be changed.
- **Data encryption**: Turn on or off the switch to specify whether to encrypt the data. If the outlier instance removal policy contains sensitive data, we recommend that you turn on Data encryption to reduce the risk of data leaks.
- **Configuration format**: Select a data format for the content of the outlier instance removal policy. The system verifies the data based on the format that you select.
- **Configuration content**: Enter the content of the outlier instance removal policy.

You can create an outlier instance removal policy for an HSF application at the application or service level by using the related properties and the values that you specify for them. The following examples show how to create outlier instance removal policies at these two levels.

> **Note** A service-level outlier instance removal policy takes precedence over an application-level outlier instance removal policy.

- Example on how to create an application-level outlier instance removal policy

```
{
"DEFAULT": {
"errorRateThreshold":0.5,
"isolationTime":60000,
"maxIsolationRate":0.2,
"maxIsolationTimeMultiple":15,
"qosEnabled":true,
"requestThreshold":20,
"timeWindowInSeconds":10,
"ipDimension":true
}
}
```

- Example on how to create a service-level outlier instance removal policy

```
{
"DEFAULT": {
"errorRateThreshold":0.5,
"isolationTime":60000,
"maxIsolationRate":0.2,
"maxIsolationTimeMultiple":15,
"qosEnabled":true,
"requestThreshold":20,
"timeWindowInSeconds":10
},
"service:version": {
"errorRateThreshold":0.5,
"isolationTime":60000,
"maxIsolationRate":0.2,
"maxIsolationTimeMultiple":15,
"qosEnabled":true,
"requestThreshold":20,
"timeWindowInSeconds":10
}
}
```

If you have other requirements, see Parameters for creating an outlier instance removal policy.

Parameters for creating an outlier instance removal policy

You can create an outlier instance removal policy by using related properties in configuration management, or by using -D parameters for Java Virtual Machine (JVM). Outlier instance removal policies created in configuration management take precedence over those created by using the -D parameters. We recommend that you create an outlier instance removal policy in configuration management.

Parameter	Property	-D parameter	Description	Default value
Maximum number of calls	requestThreshold	-Dhsf.qos.request.threshold	The maximum number of calls. An outlier instance is removed only when the number of calls in the most recent statistics window exceeds the threshold.	10
Lower error rate limit	errorRateThreshold	-Dhsf.qos.error.rate.threshold	The lower limit of the error rate. When the error rate of an instance deployed with the called application or service exceeds the lower limit, the instance is removed.	0.5

Parameter	Property	-D parameter	Description	Default value
Upper limit of instance removal ratio	maxIsolationRate	-Dhsf.qos.max.isolation.rate	The maximum proportion of abnormal instances to be removed. If the threshold is reached, no more abnormal instances are removed. For example, an application has six instances in total. If you set this parameter to 60%, the maximum number of instances that can be removed is 3.6, which is rounded down to the nearest integer 3. The number is calculated by using the following formula: 6 × 60% = 3.6. If the calculation result is less than 1, one instance is removed.	0.2
Recovery detection unit time	isolationTime	-Dhsf.qos.isolation.time	The unit time used to detect whether abnormal instances are recovered. After abnormal instances are removed, EDAS continuously detects whether abnormal instances are recovered at an interval that accumulates by the specified unit time. The unit is ms.	60 × 1,000 ms (1 minute)
			The maximum	

Parameter	Property	-D parameter	Description	Default value
Maximum cumulative number of times not restored	maxIsolationTimeMultiple	-Dhsf.qos.max.isolation.time.multiple	number of detections. EDAS continuously detects abnormal instances, and the detection interval linearly increases with the number of detections by the recovery detection unit time. When the specified maximum number of detections is reached, EDAS continuously detects whether abnormal instances are recovered based on the longest detection interval. For example, the recovery detection unit time is set to 60,000 ms, and the maximum cumulative number of times not recovered is set to 60. If an abnormal instance remains abnormal after it is detected 60 times, the instance is subsequently detected at intervals of 60 minutes, which is calculated by using the following formula: 60 × 60,000 ms = 60 minutes. If the instance is recovered before the specified maximum number of detections is reached, the detection interval is reset to the	60

Parameter	Property	-D parameter	Description	Default value
			initial interval, which is the value of the recovery detection unit time.	
Enable outlier instance removal	qosEnabled	-Dhsf.qos.enable	Specifies whether to enable outlier instance removal for the application or service.	false
Time window for statistics	timeWindowInSeconds	-Dhsf.qos.time.window.in.seconds	The time window for statistics on the maximum number of calls. This time window is the statistical period.	10s

Parameter	Property	-D parameter	Description	Default value
Exception type	bizExceptionPredicateClassName	-Dhsf.qos.biz.exception.class.name	The exception type of the instances of the application or service. By default, all service exceptions are considered as exceptions. You can also define specific service exceptions by using custom interfaces. For example, you can define exceptions in the following ways: • Define all service exceptions as exceptions: com.taobao.hsf.exception.CountBizExceptionPredicate. • Ignore all service exceptions: com.taobao.hsf.exception.IgnoreBizExceptionPredicate. • Configure the instance deployed with the application whose code contains bizExceptionPredicate and com.taobao.hsf.Predicate. com.taobao.hsf.Predicate is the implementation of bizExceptionPredicate.	com.taobao.hsf.exception.CountBizExceptionPredicate: defines all service exceptions as exceptions.

Verify the result

The outlier ejection feature is enabled after you configure and create an outlier ejection policy. You can go to the details page of the application for which you have configured outlier ejection to view the application monitoring information. For example, you can check whether call requests are still forwarded to abnormal instances and whether the error rate per minute for application calls is higher than the value of the **Error Rate Threshold** parameter in a topology. This way, you can check whether the outlier ejection policy takes effect.

3.4. Gracefully release HSF applications

This topic describes how to gracefully release High-Speed Service Framework (HSF) applications in Enterprise Distributed Application Service (EDAS).

Prerequisites

- Your EDAS Container version is V3.5.7 or later. If your EDAS Container version is earlier than V3.5.7, update it. For more information, see Upgrade or downgrade the runtime environment.
- Your application is configured with a health check URL.

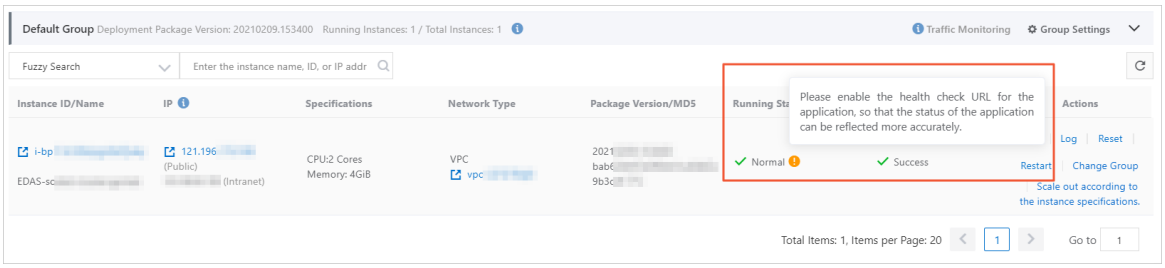
  If you want to gracefully release an HSF application, you must configure a health check URL for the application. This way, an automatic release script can be mounted to inform EDAS when the application is started. The script is automatically executed after the application is started.

  By default, the health check URL is not configured in EDAS. You must manually create and configure the required controller in the application code.

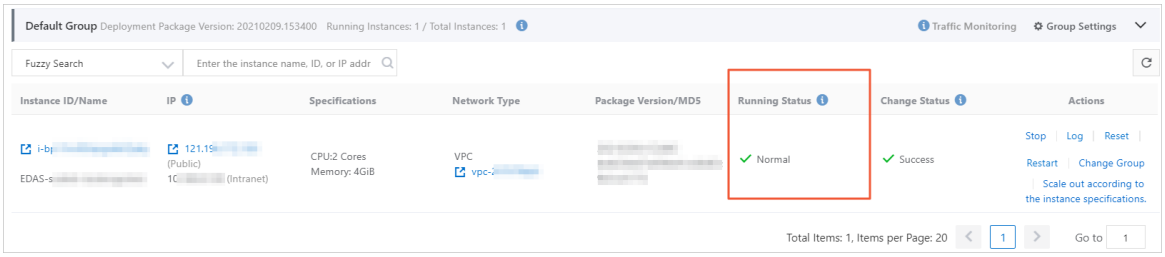
```
@RestController
public class HealthCheckController {
    @RequestMapping("/health")
    public String healthCheck(){
        return "success";
    }
}
```

  The URL-based health check reflects the health status of an application more accurately than the port-based health check.

  - Before a health check URL is configured



- After a health check URL is configured



Context

During application startup, a service is registered with the registry. After a service consumer receives a notification of successful registration, the consumer initiates a call to the service provider. However, the application startup is a continuous process. During this process, the service may have been released, but its dependent components, such as Redis or database resources, are still not initialized. If inbound traffic is generated in this situation, the call fails. To prevent call failures, you can gracefully release the HSF application.

All provider beans of HSF are not registered with the registry during initialization. Instead, they are registered after all the beans in the Spring container are initialized and RefreshEvent is sent. In addition, Pandora sets the status to true after all services are registered. O&M is also required. After the `app server (Tomcat)` is started and before the web server is started, you can run the **curl localhost:12201/hsf/status** command to check whether the service is initialized. If so, start the Apache or NGINX web server.

Configure delayed release for HSF applications

1.
2. In the left-side navigation pane, click **Applications**.
3. In the top navigation bar, select a region. On the **Applications** page, select the microservice namespace to which the application that you want to gracefully release belongs. Then, click the name of the application.
4. On the **Basic Information** page of the application, click **Edit** to the right of **JVM Parameters** in the **Application Settings** section.
5. In the **Application Settings** dialog box, click **Custom**. In the **Custom Parameters** field, enter *-Dhsf.publish.delayed=true* and click **Configure JVM Parameters**.

   After delayed release is configured, the HSF application is not immediately released. Instead, the HSF application is released after it receives a notification sent by the release script.
6. Log on to the Elastic Compute Service (ECS) instance on which the HSF application is deployed to verify the delayed release.
   i. Run the **telnet localhost 12201** command to log on to the ECS instance.
   ii. Run the **cd hsf** command to go to the HSF directory.
   iii. Run the **ls** command to view the service release status.

Mount the automatic release script

1.
2. In the left-side navigation pane, click **Applications**.
3. In the top navigation bar, select a region. On the **Applications** page, select the microservice

namespace to which the application that you want to gracefully release belongs. Then, click the name of the application.

4. On the **Basic Information** page of the application, click **Mount Script** in the **Application Settings** section.
5. In the **Mount Script** dialog box, click **Post-launch Script** and enter the following script:

```
grep "PANDORA QOS PORT" /home/admin/edas-assist/edas-assist.pid | sed 's/\x0D$//' | a
wk -F":" '{ print "curl localhost:"$2"/hsf/online?k=hsf"}'| sh
```

The following list describes the script:

- Content of the *edas-assist.pid* file

```
PID:19426
HSF PORT:12200
PANDORA QOS PORT:12203
MONITOR PORT:8006
CSP PORT:8719
```

- */home/admin/edas-assist/edas-assist.pid* is the file that records the port number of Pandora Boot. The port number of Pandora Boot is randomly generated after EDAS Container is started. In most cases, the port number is 12201. If the port is occupied, the next port is used.
- The **curl localhost:"$2"/hsf/online?k=hsf** command is used to release the HSF application and notify EDAS Container that the HSF application is released. You can also manually run this command.

Verify the result

You can use the quality of service (QoS) method or the log method to check whether the HSF application is gracefully released.

- QoS

  After the script is configured, you can gracefully release the HSF application when you perform operations such as deploying or resetting the application. You can log on to the ECS instance on which the application is deployed and check the service release status.

- Log

  Check whether the */home/admin/logs/hsf/hsf.log* file contains the following logs. If the logs exist, the HSF application has received the release command.

```
01 2019-11-26 16:23:03.456 INFO [qos-worker-3-1:t.hsf] [38ef6d01-10a8-405d-8725-bd7bf121*
***] [] [] Receive online command.Do HSF online.
```

3.5. View HSF service reports

HSF service reports provide the information about the runtime status of all the services that are deployed in all applications within the current tenant in the last 24 hours. The information includes Total Calls, Average Call RT, and Total Call Errors. These reports allow you to compare all services in the system.

Procedure

1.

2. In the left-side navigation pane, choose **Microservices Governance > HSF**.
3. In the left-side navigation pane of the **Service Query** page, click **Service Statistics**. On the **Service Statistics** page, view the runtime data of the services.

3.6. End-to-end traffic adjustment

3.6.1. Overview

Canary release helps developers smoothly update applications to a later version. Enterprise Distributed Application Service (EDAS) supports canary traffic adjustment on a single application and end-to-end traffic adjustment on multiple applications.

Scenarios

You can implement traffic adjustment on a single application and multiple applications based on the following two methods: HTTP and High-Speed Service Framework (HSF).

- Update of a single application

  New versions are continuously released in the application iteration process. Before a new version is officially released, you can use canary traffic adjustment to verify the new version on a small number of instances by collecting the user experience data. This way, you can check the metrics, such as features, performance, and stability, of the new application version before you perform a full update.

- Multi-application troubleshooting

  When your HSF microservice-oriented applications that are deployed in EDAS fail to work, you can perform end-to-end traffic adjustment to route specific traffic to an application for troubleshooting problems in the application. This ensures that all microservice-oriented applications can run as expected.

Ingress application and traffic adjustment rules

In end-to-end traffic adjustment, you must first specify the ingress application. Then, you must specify a traffic adjustment rule based on HTTP or HSF.

- For HTTP traffic, canary traffic is identified only after the traffic accesses the ingress application. The traffic that complies with the rule is marked as canary traffic. In simple terms, EDAS only identifies canary traffic.

  > **Notice**
  > - EDAS identifies canary traffic only for the allocated HTTP traffic. End-to-end traffic adjustment cannot route HTTP canary traffic. EDAS identifies canary traffic based on HTTP traffic on the ingress application and routes HSF canary traffic that is generated in each subsequent step.
  > - The HTTP traffic adjustment rule for a single application is different from the end-to-end traffic adjustment rule for multiple applications. The former is used for application instance groups. EDAS uses this rule to identify canary traffic and routes the canary traffic to the application instance groups.

- For HSF traffic, canary traffic is identified and routed before the traffic reaches the ingress application. If the ingress application has an instance in the current canary instance group, the canary

traffic is directly routed to the canary instance group. Otherwise, the canary traffic is routed to the default group of this ingress application. In simple terms, EDAS identifies canary traffic and routes it.

You can specify a canary traffic adjustment rule for multiple applications in a similar way as the traffic adjustment rule for a single application. The difference is that you can specify multiple rules for canary traffic adjustment on multiple applications.

- You must specify the traffic protocol type for each rule. The traffic supports two protocol types: HTTP and HSF.
- Each rule can have multiple rule conditions that are in the AND or OR relationship.

Traffic adjustment environment

EDAS manages canary releases by defining a traffic adjustment environment. A traffic adjustment environment consists of the ingress application and identification rules. It is also a logical space that contains application instance groups that are deployed in the traffic adjustment environment. Therefore, you can add or remove an instance group that is a non-default group of an application to or from a traffic adjustment environment.

Flexible features

The end-to-end traffic adjustment solution of EDAS implements canary release and traffic adjustment by using the EDAS console. The solution provides the following flexible features:

- You need only to prepare instance resources for the applications that require canary release. You do not need to build an entire service system.
- The solution allows you to enable canary release for multiple applications and specify different rules of canary traffic adjustment for different applications. The solution even allows one application to be involved in multiple rules for canary traffic adjustment at the same time.
- The solution supports link-based canary release. This indicates that the solution allows multiple applications to be deployed in the same traffic adjustment environment. The canary traffic identified by the upstream process can pass through immediate application instances that do not require canary release. The canary traffic can still be routed to the corresponding canary application instance in the downstream process.

3.6.2. Upgrade a single application by using end-to-end throttling

During application iterations, you can use end-to-end throttling to verify a new version on a small number of instances. After the verification succeeds, you can upgrade the applications on all instances to the new version.

Scenario

Web Application A V1 is deployed on two Elastic Compute Service (ECS) instances by using WAR packages. After Web Application A V2 is available, you must verify it on one instance. After the verification succeeds, upgrade Web Application A V1 on the other instance to V2.

Canary release process

1. Create a canary instance group.
2. In the canary instance group, configure and enable a throttling rule.

3. Deploy V2 in the canary instance group and check whether the specified traffic is distributed to the instance in the canary instance group.
4. Verify V2 based on the traffic that is distributed to the canary instance group.
5. After the verification succeeds, upgrade the application on the instance in the default group to V2.

   If issues occur during the verification, disable the throttling rule for the canary instance group and move the instance from the canary instance group to the default group. After you troubleshoot the issues, enable the throttling rule for the canary instance group again. Then, deploy and verify V2 in the canary instance group.
6. Disable the throttling rule for the canary instance group and delete the canary instance group.

Step 1: Create a canary instance group

In ECS clusters, different application versions are deployed based on instance groups, and throttling rules are configured based on instance groups. Therefore, you must create a canary instance group first.

1.
2. In the left-side navigation pane, click **Applications**. On the **Applications** page, select the namespace to which the application to upgrade belongs and click the name of the application.
3. On the application details page, click the **Instance Information** tab. Then, click **Create Group** in the upper-right corner.
4. In the **Create Group** dialog box, set **Group Name** to *Canary Instance Group* and click **Create**. After the group is created, the message **The group is created.** appears in the upper part of the page.

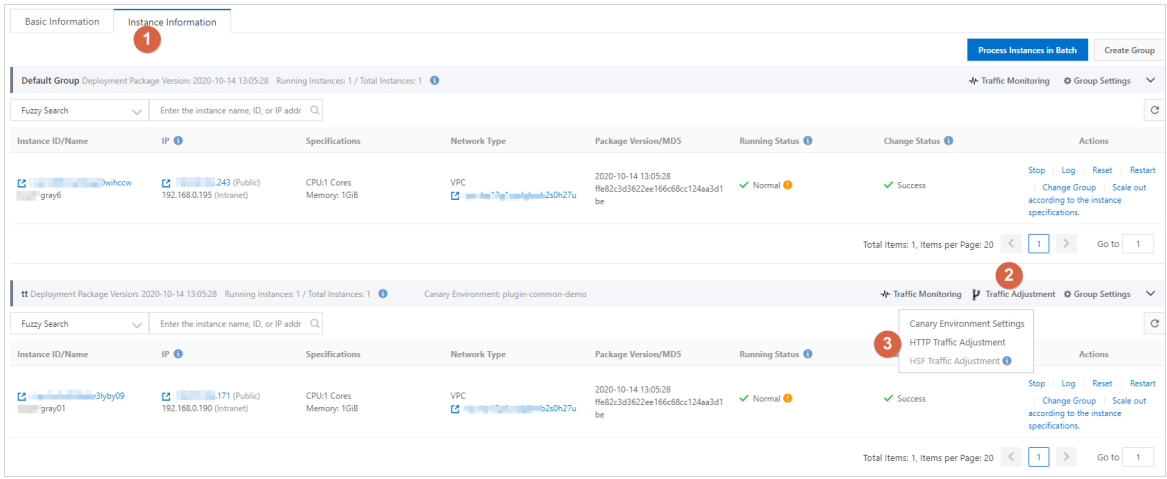
Step 2: Configure and enable a throttling rule

You can configure HSF and HTTP throttling rules.

- To configure an HTTP throttling rule, click the **Basic Information** tab and enable **Traffic Management**.
- To configure an HSF throttling rule, you do not need to enable **Traffic Management**. However, you must make sure that Enterprise Distributed Application Service (EDAS) Container 3.5.3 or later is used.

The following procedure describes how to configure an HTTP throttling rule:

1. In the **Application Settings** section of the **Basic Information** tab, click **Enable** next to **Traffic Management**.
2. On the **Instance Information** tab, click **Traffic Adjustment** in the upper-right corner of the Canary Instance Group section and select **HTTP Traffic Adjustment**.

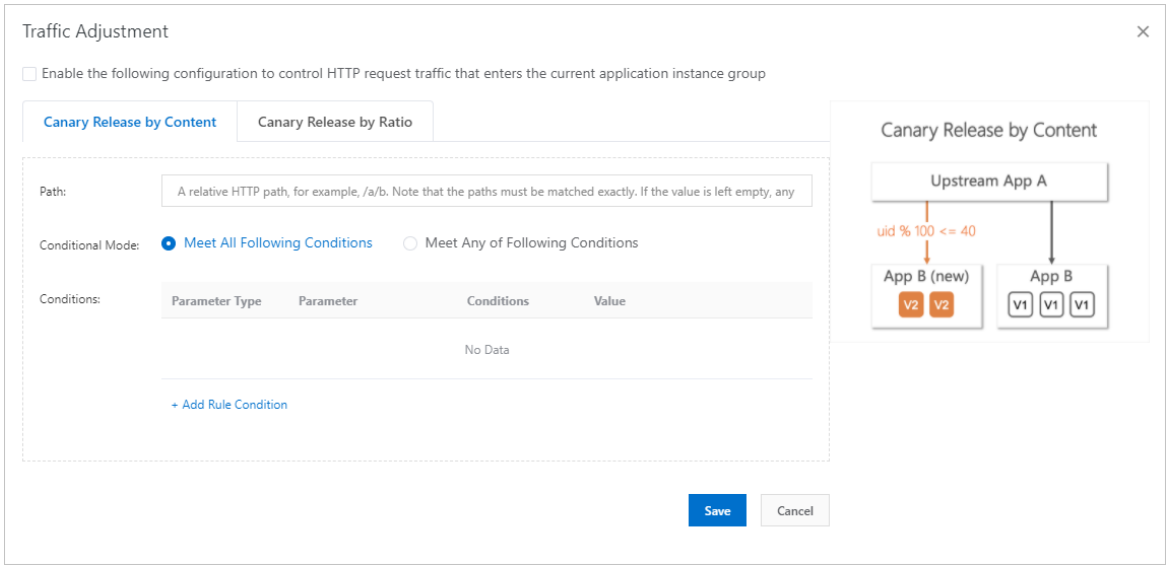


3. In the **Traffic Adjustment** dialog box, configure the parameters and select **Enable the following configuration to control HTTP request traffic that enters the current application instance group**. Then, click **Save**.

   You can configure one of the following throttling rules for inbound traffic: **Canary Release by Content** and **Canary Release by Ratio**.

   - Canary Release by Content: distributes traffic that meets the configured rule to the canary instance group.
   - Canary Release by Ratio: distributes traffic to the canary instance group based on a specific proportion.

   The Canary Release by Ratio rule is easy to configure. The following sections describe how to configure the Canary Release by Content rule for inbound traffic.



   Parameters required for a throttling rule:

   - **Conditional Mode**: Select **Meet All Following Conditions**.
   - **Conditions**: Valid values are Cookie, Header, and Parameter. In this example, Parameter is used.
     - **Parameter Type**: Select **Parameter**.
     - **Parameter**: Enter *version*.
     - **Conditions**: Select **=**.

- **Value**: Enter *1*.

> **Notice** After the parameters are configured, select **Enable the following configuration to control HTTP request traffic that enters the current application instance group** to make the throttling rule take effect.

If the canary instance group has no instances to receive the canary traffic after the throttling rule takes effect, the default group is automatically used.

Step 3: Deploy and verify the new version

1. Move an instance to the canary instance group.
   i. Open the **Instance Information** tab. In the Default Group section, find the instance that you want to move to the canary instance group and click **Change Group** in the Actions column.
   ii. In the **Change Group** dialog box, select **Canary Instance Group** from the **Target Group** drop-down list and click **OK**.

   After the instance is moved to the canary instance group, the version of the application on this instance is V1.
2. Deploy the new version.

   You can deploy the new version by using the console or tools. In this example, the new version is deployed in the console. For more information about how to deploy the new version by using other tools, see Overview.
   i. In the upper part of the application details page, click **Deploy Application**.
   ii. On the **Select Deployment Mode** page, click **Start Deployment** next to **Regular Release (Single-batch/Multi-batch)**.

   > **Note** In this example, V1 is deployed by using a WAR package. Therefore, you must select the WAR package to deploy V2. If you have deployed the application by using a JAR package, configure the parameters for the **JAR** deployment method.

iii. Configure the parameters for application deployment.

- **Deployment Method**: Select **JAR**.
- **File Uploading Method**: Select **Upload JAR Package** and click **Select File**. In the dialog box that appears, select the local JAR package for V2.
- **Version**: Enter *V2*.
- **Group**: Select the canary instance group that you create.
- **Batches per Group**: Select **1 Batches**.
- **Batch Mode**: Select **Automatic**.

After the application is deployed, the Change Details page appears. You can view the deployment progress on this page. If the value of **Change Status** is **Success**, the deployment is successful. If the deployment fails, the related log appears on the Change Details page. You can check the log for troubleshooting. For more information, see How do I troubleshoot issues in a change process?.

Open the application details page and click the **Instance Information** tab. On the Instance Information tab, the value of **Package Version/MD5** is changed to **V2**, and the value of **Running Status** is changed to **Normal** in the canary instance group.

3. Verify the throttling rule.
   i. In the browser address bar, enter `http://<IP address of the instance in the default group>:<service port>` and press Enter.
   The web page of Web Application V1 appears.
   ii. In the browser address bar, enter `http://<IP address of the instance in the default group>:<service port>?version=1` and press Enter.
   The web page of Web Application V2 appears.

   The results show that the throttling rule has taken effect, and the specified traffic is distributed to the instance in the canary instance group.

Step 4: Verify the new version

You can verify the new version based on your business requirements.

If issues occur during the verification, disable the throttling rule for the canary instance group and move the instance from the canary instance group to the default group. After you troubleshoot the issues, enable the throttling rule for the canary instance group again. Then, deploy and verify V2 in the canary instance group.

Step 5: Upgrade the application in the default group

After the verification succeeds, upgrade the application in the default group to V2.

To upgrade the application, you must deploy the application again. For more information about the procedure, see Step 3: Deploy and verify the new version.

Step 6: Disable the throttling rule and delete the canary instance group

After Web Application A V1 on the instances in both groups is upgraded to V2, you must disable the throttling rule and delete the canary instance group.

1. On the **Instance Information** tab, click **Traffic Adjustment** in the upper-right corner of the

Canary Instance Group section and select **HTTP Traffic Adjustment**.

2. In the **Traffic Adjustment** dialog box, clear **Enable the following configuration to control HTTP request traffic that enters the current application instance group** and click **Save**.
3. Open the **Instance Information** tab, find the instance in the canary instance group and click **Change Group** in the Actions column.
4. In the **Change Group** dialog box, select **Default Group** from the **Target Group** drop-down list and click **OK**.
5. Open the **Instance Information** tab, and click **Delete Group** in the upper-right corner of the Canary Instance Group section.

3.6.3. Use the end-to end throttling feature to troubleshoot application issues

When your High-speed Service Framework (HSF) microservice-oriented application that is deployed in Enterprise Distributed Application Service (EDAS) fails, you can troubleshoot the specific application by using the end-to end throttling feature. This improves the troubleshooting efficiency and ensures the normal operation of the entire microservice-oriented application. This topic uses an example to describe how to troubleshoot application issues by using the end-to end throttling feature.

Limits

If you use the end-to end throttling feature to troubleshoot application issues during canary releases, take note of the following limits:

- The application must be deployed in an Elastic Compute Service (ECS) cluster.
- The application must run in EDAS Container, which means that the application is an HSF application.
- The application cannot be deployed by using images.

Scenario

Applications A, B, C, and D are deployed on an HSF microservice-oriented application. The versions are A1, B1, C1, and D1. The HSF microservice-oriented application fails. The preliminary troubleshooting indicates that issues exist in Application B and Application D. The end-to end throttling feature is used to route the specific traffic to the throttling groups of Application B and Application D for troubleshooting.

Troubleshooting process

The troubleshooting process is divided into two phases: troubleshoot Application D first and then Application B.

Procedure

You can perform the following steps to troubleshoot the issues.

1. Create a throttling group for Application D. For more information, see Manage instance groups for an application deployed in an ECS cluster in the EDAS console.
2. Create a throttling environment, specify Application A as the inbound application, and then configure a throttling rule for Application A. Add the throttling group of Application D to the throttling environment and enable the throttling environment. For more information, see Create a traffic adjustment environment for multiple applications.

The method that is used to identify and test canary traffic on inbound HTTP applications is convenient and general. In this example, the inbound HTTP application is Application A, and the application that requires troubleshooting is Application D. The canary traffic is identified and processed on different applications. Therefore, the end-to-end throttling capability is required.

> **Note** If no instances are deployed in the throttling group, the group cannot receive the canary traffic, and canary degradation is triggered. The canary traffic is destined for the applications in the default group. After an instance is added to the throttling group, canary degradation is disabled, and the canary traffic is destined for the instance in the throttling group.

3. Add an instance to the throttling group of Application D. For more information, see Add an instance.

   Add the instance based on the version of the instance in the default group.

4. Check whether the traffic distribution meets your requirements and troubleshoot the issues in the throttling group of Application D.
   i. To determine whether the throttling rule takes effect and whether the traffic distribution meets your requirements, click **Flow Monitoring** next to the throttling group. For more information, see Monitor canary traffic.
   ii. Troubleshoot the issues based on the canary traffic in the throttling group of Application D.
   - If the troubleshooting goes smoothly, create a throttling group for Application B for troubleshooting.

     The throttling group of Application D is created after you create the throttling environment and configure the throttling rule. You can first configure a throttling rule or first add instances to throttling groups. The sequence of the two operations is flexible. For more information, see End-to-end traffic adjustment policy.
   - If the configurations are invalid or you must update the version of the application, disable the throttling environment and remove instances from the throttling group of Application D. After the configuration is modified or the new version is available, enable the throttling environment and redeploy the application for verification.

5. Create a throttling group for Application B. For more information, see Create a group.
6. Add the throttling group of Application B to the throttling environment. For more information, see Create a traffic adjustment environment for multiple applications.
7. Add an instance to the throttling group of Application B. For more information, see Add an instance.
8. Check whether the traffic distribution meets your requirements and troubleshoot the issues in the throttling groups of Application B and Application D.
   i. To determine whether the throttling rule takes effect and whether the traffic distribution meets your requirements, click **Flow Monitoring** next to the throttling group. For more information, see Monitor canary traffic.

ii. Troubleshoot the issues based on the canary traffic in the throttling groups of Application B and Application D.

- If the troubleshooting process is completed, disable the throttling environment and delete the throttling groups of Application B and Application D. You can delete the throttling environment based on your business requirements.
- If configurations are invalid or you must update the version of an application, update the application or configurations and verify again after redeployment.

  The troubleshooting is not complete. The throttling environment is not stopped, but the instance is removed from the throttling group of Application B.

3.6.4. Enable throttling for a single application

New versions are released continuously during application iterations. Before you release a new version, you can use throttling rules to verify the new version on a small number of instances. After the verification succeeds, you can upgrade the applications on all instances to the new version. You can use HSF and HTTP throttling rules. This topic describes the two throttling rules.

Prerequisites

- A group is created.
- If you use a RAM user to perform end-to-end traffic adjustment, the RAM user is granted the permissions to view clusters, applications, and services, and manage applications and services. For more information, see Replace EDAS-defined permissions with RAM policies.

Configure an HTTP throttling rule

You can specify the URL and use the Cookie value, Header value, or URL parameter to distribute the traffic based on the whitelist or the range of the remainder of Mod 100.

> **Notice** If you configure HTTP throttling rules for multiple groups of an application, only the last throttling rule takes effect.

1.
2. In the left-side navigation pane, click **Applications**. In the top navigation bar, select a region. On the Applications page, select the namespace to which the application to upgrade belongs from the Namespace drop-down list. Then, click the name of the application.
3. On the application details page, click the **Basic Information** tab. In the **Application Settings** section, click **Enable** next to **Traffic Management**.

   If throttling is not enabled, you cannot configure HTTP throttling rules.
4. On the **Instance Information** tab, click **Traffic Adjustment** in the upper-right corner of the canary instance group section and click **HTTP Traffic Adjustment**.
5. In the **Traffic Adjustment** dialog box, select **Enable the following configuration to control HTTP request traffic that enters the current application instance group**.
6. In the **Traffic Adjustment** dialog box, configure the parameters. Then, click **Save**.

   You can configure one of the following throttling rules for inbound traffic: **Canary Release by Content** and **Canary Release by Ratio**.

   - Canary Release by Content: distributes traffic that meets the configured rule to the canary

instance group.

- Canary Release by Ratio: distributes traffic to the canary instance group based on a specific proportion.

The following list describes how to configure the two rules:

- Configure the Canary Release by Content rule
  a. In the **Traffic Adjustment** dialog box, click the **Canary Release by Content** tab.
  b. On the tab, configure the parameters.

Parameter	Description	
path	Enter the path in the HTTP requests for the service.	
Conditional Mode	If multiple rules are configured, you must configure the conditions for these rules to take effect. Valid values: **Meet All Following Conditions** and **Meet Any of Following Conditions**.	
Conditions	Parameter Type	Valid values: Cookie, Header, and Parameter.
	Parameter	Enter the parameter name. The name can be up to 64 characters in length. For more information about the writing specifications and examples, see throttling rule parameters.
	Conditions	Select a condition. Valid values: =, !, =, >, <, >=, <=, White List, and Mod 100.
	Value	Enter a value for the mod operation or list.

- Configure the Canary Release by Ratio rule
  a. In the **Traffic Adjustment** dialog box, click the **Canary Release by Ratio** tab.
  b. On the tab, specify **Traffic Ratio**. The value must be a positive integer. Valid values: 1 to 100. Example: 40.

Configure an HSF throttling rule

You can specify a service or a method, use the parameter value of the method, and use the remainder range of Mod 100 or the list as the condition to distribute the traffic.

1. On the **Instance Information** tab, click **Traffic Adjustment** in the upper-right corner of the canary instance group section and click **HSF Traffic Adjustment**.
2. In the **Traffic Adjustment** dialog box, select **Enable the following configuration to control the HSF request volume that comes into the current application instance group**.
3. In the **Traffic Adjustment** dialog box, configure the parameters. Then, click **Save**.

   You can configure one of the following throttling rules for inbound traffic: **Canary Release by Content** and **Canary Release by Ratio**.

- **Canary Release by Content**: distributes traffic that meets the configured rule to the canary instance group.
  a. In the **Traffic Adjustment** dialog box, click the **Grayscale by Content** tab.
  b. On the tab, configure the parameters.

Parameter	Description	
Select Service	Select a service in the ingress application from the drop-down list.	
Method	Select a method in the service that you select in the ingress application. If you do not select a method, all requests that access the service are matched based on the rule.	
Conditional Mode	If multiple rules are configured, you must configuire the conditions for these rules to take effect. Valid values: **Meet All Following Conditions** and **Meet Any of Following Conditions**.	
Conditions	Parameter	Parameters in the service and method are automatically listed and named in the format of parameteri. i indicates the serial number of the parameter. The serial number starts from 0.
	Expression for Getting Parameter Values	The expression is concatenated by two fields in the format of argsi.xxx. The first field argsi depends on the parameter you select. If you select parameter0, the argsi field is args0. The second field .xxx is customized based on your requirements. For more information about the writing specifications and examples, see throttling rule parameters.
	Conditions	Select a condition. Valid values: =, !, =, >, <, >=, <=, White List, and Mod 100.
	Value	Enter a parameter value. A string must be enclosed in double quotation marks (""). If the value is of the BOOLEAN type, the valid value is true or false.

- **Canary Release by Ratio**: distributes traffic to the canary instance group based on a specific

proportion.

a. In the **Traffic Adjustment** dialog box, click the **Canary Release by Ratio** tab.

b. On the tab, specify **Traffic Ratio**. The value must be a positive integer. Valid values: 1 to 100. Example: 40.

Verify the throttling rule

After you configure and enable a throttling rule, you can monitor the canary traffic to verify whether the traffic destined for the canary instance group meets your expectations. For more information, see Monitor canary traffic.

3.6.5. Create a traffic adjustment environment for multiple applications

The traffic adjustment environment is the core of canary release. You must perform traffic adjustment for multiple applications in a traffic adjustment environment. This topic describes how to create a traffic adjustment environment.

Prerequisites

- A group is created.
- If you use a RAM user to perform end-to-end traffic adjustment, the RAM user is granted the permissions to view clusters, applications, and services, and manage applications and services. For more information, see Replace EDAS-defined permissions with RAM policies.

Create a traffic adjustment environment

To configure traffic adjustment rules for High-Speed Service Framework (HSF) applications, you must use Enterprise Distributed Application Service (EDAS) Container V3.5.3 or later.

1.
2. In the left-side navigation pane, choose **Microservices Governance > HSF**. In the navigation tree of the **Service Query** page, click **End-to-end Traffic Adjustment**.
3. On the **End-to-end Traffic Adjustment** page, click **Create Environment** in the upper-right corner.
4. In the **Basic Information** step, set the **Microservice Namespace** parameter to specify a region and a microservice namespace, set the **Canary Environment Name**, **Canary Identifier**, and **Environment Description** parameters, and then click **Next**.
5. In the **Set Inbound Traffic Rule** step, select an ingress application and configure traffic adjustment rules. Then, click **Next**.

   The following table describes the parameters for traffic adjustment rules.

Parameter	Description
Entrance Application	Select an ingress application for the traffic adjustment environment from the drop-down list based on your business requirements.
Protocol Type	Select the protocol type based on your business requirements. Valid values: HTTP and HSF.

Parameter	Description	
If you set the Protocol Type parameter to **HTTP**, set the following parameters:		
Path	Enter the path in the HTTP requests for the service.	
Conditional Mode	If you configure multiple rules, you must configure the conditions for these rules to take effect. Valid values: Meet All Following Conditions and Meet Any of Following Conditions.	
Conditions	Parameter Type	Select the type of the parameter. Valid values: Cookie, Header, and Parameter.
	Parameter	Enter a parameter. The parameter can be up to 64 characters in length. For more information about how to configure this parameter, see Throttling rule parameters.
	Conditions	Select a condition. Examples: Mod 100 and Whitelist.
	Value	Enter a value for the mod operation or list.
If you set the Protocol Type parameter to **HSF**, set the following parameters:		
Select Service	Select a service in the ingress application. For example, the value is in the format of `com.alibaba.edas.demo.api.DempSevice:1.0.0(Service group name)`.	
Method	Select a method in the ingress application. For example, the value is in the format of `echoTime(java.lang.String, java.util.List<java.lang.Integer>)`.	
Conditional Mode	If you configure multiple rules, you must configure the conditions for these rules to take effect. Valid values: Meet All Following Conditions and Meet Any of Following Conditions.	
	Parameter	Select a parameter in the method. For more information about the parameters for HSF traffic adjustment rules, see Throttling rule parameters.

Parameter	Description	
Conditions	Expression for Getting Parameter Values	Enter a parameter expression to obtain a property of the parameter. Examples: ◦ Empty: obtains the value of the parameter. ◦ .name: obtains the name property of the parameter. This expression is equivalent to args0.getName(). ◦ .isEnabled(): obtains the enabled property of the parameter. This expression is equivalent to args0.isEnabled(). ◦ [0]: indicates that the value of the parameter is an array and obtains the first value of the array. This expression is equivalent to args0[0]. Take note that this expression is not prefixed with a period (.). ◦ .get(0): indicates that the value of the parameter is a list and obtains the first value of the list. This expression is equivalent to args0.get(0). ◦ .get("key"): indicates that the value of the parameter is a map and obtains the value of the key in the map. This expression is equivalent to args0.get("key").

Parameter	Description	
	Conditions	Select a condition. ○ `=` : supports comparison between strings, numbers, BOOLEAN values, and CHAR values. ○ `!=` : supports comparison between strings, numbers, BOOLEAN values, and CHAR values. ○ `>` : supports comparison between numbers. ○ `>=` : supports comparison between numbers. ○ `<` : supports comparison between numbers. ○ `<=` : supports comparison between numbers.
	Value	Enter the value of the parameter.

> **Note**
> - To configure multiple conditions, click **Add Rule Condition**.
> - To create multiple inbound traffic adjustment rules, click **Create Inbound Traffic Rule**.

6. In the **Select Application** step, select specific applications in the Select Application section based on your business requirements, click the **>** icon to add the applications to the **Selected Applications** section, select the application instance group, and then click **Next**.

> **Note**
> - Some applications in the Select Application section cannot be selected and have the
>
> icon next to them. This indicates that these applications are in the default group. In this case, the applications cannot be added to the traffic adjustment environment.
> - Some application instance groups in the Selected Applications section also have the
>
> icon next to them. This indicates that such a group has no instances. In this case, you must add at least one instance to the group. For more information, see Add an instance.

7. In the **Created** step, verify the settings of the traffic adjustment environment and click **Submit**.

Enable the traffic adjustment environment

After the traffic adjustment environment is created, you can turn on Entrance Flow to apply traffic adjustment rules to the ingress application that runs in the traffic adjustment environment. To enable the traffic adjustment environment, perform the following steps:

1. Go to the **End-to-end Traffic Adjustment** page.
2. Select a region and a microservice namespace where the traffic adjustment environment resides.
3. Find the traffic adjustment environment that you created and turn on **Entrance Flow**.

Verify the result

After you configure and enable the traffic adjustment environment, you can monitor the canary traffic to verify whether the environment meets your business requirements. For more information, see Monitor canary traffic.

3.6.6. Monitor canary traffic

You can monitor canary traffic to ensure a successful canary release and monitor the traffic of applications and instances.

Context

You can monitor the canary traffic of a single application and the end-to-end canary traffic of multiple applications.

Monitor the canary traffic of a single application

You can perform a canary release for a single application by application instance group. After the canary release is complete, you can monitor the traffic of the application and instances. The following procedure describes how to monitor the traffic.

1.
2. In the left-side navigation pane, click **Applications**.
3. On the **Applications** page, click the name of the application for which a canary release has been performed.
4. On the application details page, click the **Instance Information** tab. On the right side of the application instance group that you want to manage, click **Traffic Monitoring**.
5. In the **Traffic Monitoring** dialog box, select **Instance Perspective** or **Service Perspective**. Then, select the start time and end time for monitoring in the date and time picker.
6. Monitor the traffic of the application.
   - Instance Perspective: On the **Overview** tab, you can view the upstream and downstream traffic, response time (RT), number of requests, and number of errors of the application and instances. You can select an application or an instance to monitor in the left-side list. EDAS also supports other common monitoring features, such as Java virtual machine (JVM) monitoring, host monitoring, and interface snapshot.
   - Service Perspective: On the **Overview** tab, you can view the upstream and downstream traffic, RT, number of requests, and number of errors of a service provided by the application. You can select a service to monitor in the left-side list. EDAS also supports other common monitoring features, such as interface snapshot.

Monitor end-to-end canary traffic

If a canary release involves multiple applications, it is performed based on a traffic adjustment environment. After the canary release is complete, you can monitor the traffic of each application in the traffic adjustment environment. To monitor end-to-end canary traffic, perform the following steps:

1.
2. In the left-side navigation pane, choose **Microservices Governance > HSF**. In the navigation tree of the **HSF** page, click **End-to-end Traffic Adjustment**.
3. On the **End-to-end Traffic Adjustment** page, select a microservice namespace from the Microservice Namespace drop-down list and click the name of the specified traffic adjustment environment.
4. On the details page of the traffic adjustment environment, click the **Monitoring Details** tab.
5. On the **Monitoring Details** tab, select an application from the **Application in Canary Environment** drop-down list, and select a monitoring perspective such as **Instance Perspective** or **Service Perspective**. Then, select the start time and end time for monitoring in the date and time picker.
6. Monitor the canary traffic of a specific application in the traffic adjustment environment.

   The canary traffic data of a specific application is the same as the data collected during the monitoring of a single application. For more information, see the Monitor the canary traffic of a single application section of this topic.

3.6.7. Limits on end-to-end throttling

End-to-end throttling enables flexible canary release methods. It also poses some limits and leads to specific conventions.

If an application instance group belongs to multiple throttling environments, throttling conflicts may occur. Therefore, during end-to-end throttling, one application instance group can belong to only one throttling environment.

> **Note** If a High-speed Service Framework (HSF) rule is enabled for the application instance group during single application throttling, a throttling environment is created for this group.

Uniqueness of the throttling property

After a request is marked by a throttling rule, the request is never marked by another throttling rule even if this request matches another throttling rule.

Priorities of throttling rules

An application may be used as the ingress application for multiple throttling environments, and a request may match multiple throttling rules at the same time. You must set priorities for multiple throttling rules because a request can be marked by only one throttling rule. The throttling rules that are created or modified later preferentially take effect.

Limits on the joint use of single application throttling and end-to-end throttling

An application may use both single application throttling and end-to-end throttling at the same time. An HSF rule created for single application throttling is equivalent to a throttling rule. An application instance group cannot belong to different throttling environments. If an application instance group has been added to a throttling environment, you can no longer create an HSF rule for throttling of the single application. Similarly, if an application instance group has an HSF rule, it cannot be added to another throttling environment.

Unique rule for the same ingress endpoint of an ingress application in single application throttling mode

In single application throttling mode, the HTTP or HSF rules of an application are subject to the uniqueness limit. One ingress endpoint of an application can be defined by only one HTTP or HSF rule. The following list defines an ingress endpoint:

- For the HTTP protocol, an endpoint refers to an application, and only one group can be set for an application.
- For the HSF protocol, an endpoint refers to an interface method of an application.

During single application throttling, only one HTTP rule can be defined for one application. Furthermore, an interface method of an application can be used only in one HSF rule.

3.6.8. End-to-end traffic adjustment policy

In canary release, you can specify a traffic adjustment rule and deploy a new version by using multiple policies. This is because the order of these two operations is not limited. The following table describes several available combination methods for canary release. You can evaluate the methods and choose an appropriate method based on your requirements.

Method	Benefit	Problem
Deploy the new version > Specify a traffic adjustment rule	You can verify the maximum traffic for a canary group. Application deployment and canary rule configuration need to be joined only once.	Before the canary release rule takes effect, the inbound traffic of the instances for the application of the new version may not be canary traffic.
Deploy the earlier version > Specify a traffic adjustment rule > Upgrade to the new version	You can verify the maximum traffic for a canary group.	The smooth upgrade is also important. You need to check whether the upgrade of an application on relevant instances affects the requests that are being processed during the upgrade. Two application deployments are performed.
Specify a traffic adjustment rule > Deploy the new version	Application deployment and canary rule configuration need to be joined only once.	Before the application is deployed, canary traffic is downgraded and routed to the non-canary release environment. However, after the first batch of application instances of the new version is released, the instances may be affected by all the canary traffic.

Method	Benefit	Problem
Specify an invalid traffic adjustment rule > Deploy the new version > Specify a traffic adjustment rule	This method has the optimal controllability. The application instances that are newly deployed receive canary traffic only after the new valid canary release rule takes effect.	Two canary rule configurations are performed.

Note

- We recommend that you use the third method. This method is simple and can ensure that only canary traffic enters the canary release environment. However, only a few canary application instances exist in the early phase of the deployment, and application instances may be affected by heavy canary traffic after the canary release. Therefore, make sure that the traffic received by instances is in the expected range.
- End-to-end traffic adjustment is similar to single-application traffic adjustment. The difference is that specifying a traffic adjustment rule is changed to creating a canary release environment in end-to-end traffic adjustment.

3.6.9. Throttling rule parameters

A canary release allows you to create throttling rules by HTTP and HSF, whose parameter settings are different.

You can configure parameters for HTTP throttling rules based on the cookie, HTTP header, and URL parameter values. You can determine traffic based on the remainder range or list after the mod operation (mod 100). This type of rule is easy to configure and is not described in detail in this topic. If a parameter value contains a non-digit character, this character is converted to a digit by using the hash algorithm. If you use complex parameters, we recommend that you use lists to determine the traffic.

This topic describes the parameters of HSF throttling rules.

The end-to-end canary release allows you to obtain a property of a parameter by using a parameter expression. The following table describes supported parameter expressions.

Expression	Description	Remarks
args0	The value of the current parameter.	None.
args0.name	The name property of the parameter.	It can be translated into the `arg.getName()` Java statement.
args0.isEnabled()	The enabled property of the parameter, which is of the BOOLEAN type.	In Java specifications, getter for the BOOLEAN type must be `isXXX()` .
args0[0]	The first value of the arg array.	None.
args0.get(0)	The first value of the arg list.	None.

Expression	Description	Remarks
args0.get("key")	The value of the key in the arg map.	None.

If you select the first parameter, EDAS automatically generates an args0 prefix on the page.

The preceding expressions can be combined. Example:

`args0.persons[0].meta.get("name")` : retrieves the first parameter in the persons array and obtains the meta property of persons, which is a map. Then, the value of name in this map is retrieved.

Supported conditions

- `=` : supports comparison between STRING, NUMBER, BOOLEAN, and CHAR values.
- `!=` : supports comparison between STRING, NUMBER, BOOLEAN, and CHAR values.
- `>` : supports comparison between NUMBER values.
- `>=` : supports comparison between NUMBER values.
- `<` : supports comparison between NUMBER values.
- `<=` : supports comparison between NUMBER values.

Supported value expressions

A value expression for an HSF parameter matching condition represents a Java value. Only the basic data types of Java are supported, including STRING, NUMBER, BOOLEAN, and CHAR. Complex and custom data types are not supported.

EDAS supports the following types of value expressions:

- **Standard Java STRING type**

  A standard Java STRING expression represents a string that is enclosed in double quotation marks (`"..."`). Example:

  - `"tom"` : the tom string
  - `"10"` : the 10 string
  - `"abc"` : the abc string followed by a space
  - `"a"` : the a string
  - `"\n"` : the line feed
  - `""abc""` : the `"abc"` string
  - `"a\bc"` : the a\bc string

  This type of expression can represent arbitrary strings.

- **NUMBER type**

  If you want to express a NUMBER value, just enter the number. Example:

  - 100
  - 1.23
  - -3.14
  - 1.23f

> **Note** In Java, 1.23 is a DOUBLE value by default. The FLOAT value for 1.23 is 1.23f. This is determined by the precision in Java specifications.

- **BOOLEAN type**

  The BOOLEAN type has only two valid values: true and false.

- **CHAR type**

  A CHAR value represents a character that is enclosed in single quotation marks (''), Example: 'a'.

- **NULL type**

  The NULL type indicates the null value in Java. You can directly enter null.

- **String literals**

  A string literal exactly represents a string so that no characters in this string need to be escaped.

String literal	Java string
tom	"tom"
"	"\""
\	"\"
a\b	"a\b"

The following table lists the value expressions for all types of values.

Value type	Value	Value expression (to be entered in the EDAS console)
java.lang.String	"tom"	"tom" or tom
java.lang.String	"true"	"true"
java.lang.String	"10"	"10" **Note** To represent 10 of the STRING type, you must enclose 10 in double quotation marks (""). If double quotation marks ("") are absent, this string is parsed as 10 of the NUMBER type.
java.lang.String	Line feed	"\n"
java.lang.String	'	"\'"
java.lang.String	"	"\""
java.lang.String	\	"\"

Value type	Value	Value expression (to be entered in the EDAS console)
java.lang.String	aa'bb	"aa'bb"
int	10	10
java.lang.Integer	10	10
byte	10	10
boolean	true	true
java.lang.Boolean	true	true
short	10	10
long	100	100
java.lang.Long	100	100
float	1.23f	1.23f **Note** If arg uses a FLOAT value, you must suffix this value with f. For `1.23f==1.23`, false is returned.
java.lang.Float	1.23f	1.23f
double	1.23	1.23
java.lang.Double	1.23	1.23
char	'a'	'a'
null	null	null

Examples

- **Parameters of the STRING type**

  You do not need to enter anything in the field. If the field is left empty, this represents the parameter itself.

- **Parameters of the ARRAY type**

  Assume that the parameter is a string array.

  You can enter `[0]` in the field to retrieve the first element of the array.

- **Parameters of the LIST type**

  Assume that the parameter is a `list<String>`.

  You can enter .get(0) in the field to retrieve the first element of the list. Make sure that the period (.) is contained.

- **Parameters of complex types**

Assume that the first parameter of a method is of the following type:

```
public class Person {
    private String name;
    private int age;
    private String[] array;
    private List<String> list;
    private Map<String,String> map;
}
```

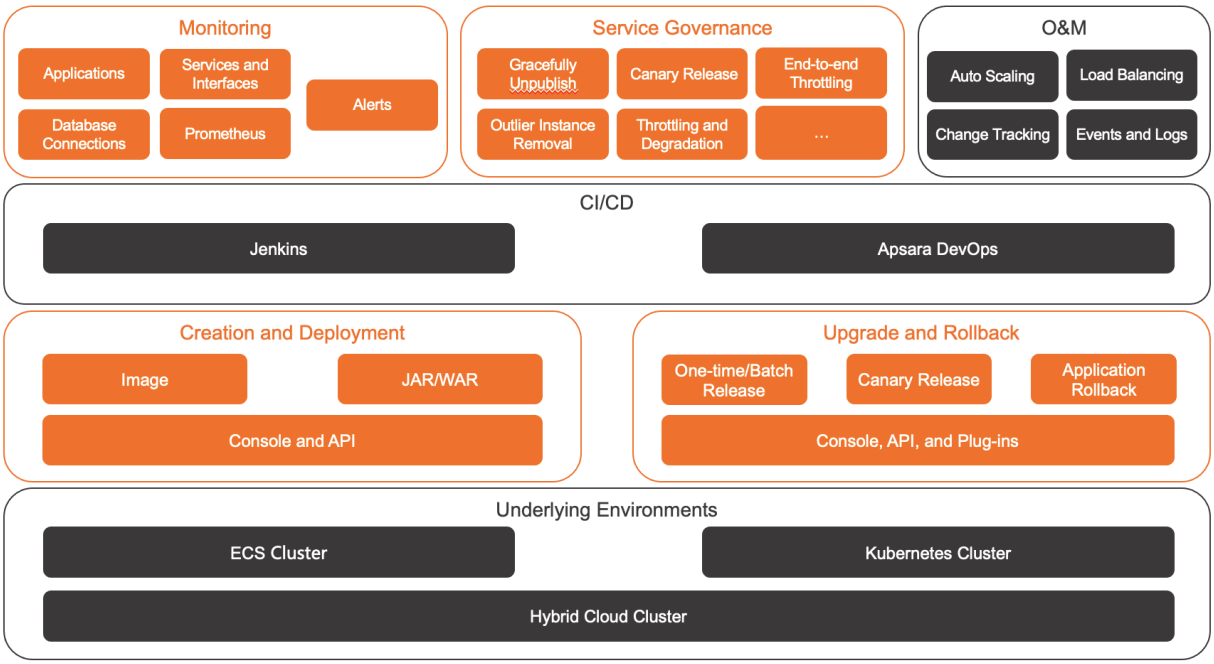
4.Multilingual service governance

4.1. Cross language interoperability in EDAS

This topic describes how to use Enterprise Distributed Application Service (EDAS) to implement interoperability between multi-language applications and Java Spring Cloud applications.

Context

As non-Java applications are widely used, the interoperability between Java applications and non-Java applications becomes an urgent need. To meet this need, EDAS provides an architecture that allows you to implement interoperability between multi-language applications and Java Spring Cloud applications. The following figure shows the architecture:



- Java applications use an agent to retrieve the data of multi-language applications that are deployed in an Alibaba Cloud Service Mesh (ASM) instance.
- Java applications use the service names of multi-language applications to call the multi-language applications.

Prerequisites

The following conditions must be met:

- A multi-language application is deployed in EDAS.
- A Java Spring Cloud application is deployed in EDAS.
- The required environment variables are configured for the interoperability between multi-language applications and Java Spring Cloud applications.

For more information, see Configure environment variables.

Scenarios

- Scenario 1: use a Java Spring Cloud application to call a multi-language application

  The method for a Java Spring Cloud application to call a multi-language application is the same as that for the Java Spring Cloud application to call another Java application. For example, you can run the following command to use the restTemplate class to call the A API operation of the Java Spring Cloud application named go-sc-a.

  ```
  restTemplate.getForObject("http://go-sc-a/A", String.class)
  ```

  You can also use other methods to call the API operation. You can access the application without specifying the port number.

- Scenario 2: use a multi-language application to call a Java Spring Cloud application

  You can use a multi-language application to call the Kubernetes service provided by a Java Spring Cloud application to access the Java Spring Cloud application.

Deploy a Java Spring Cloud application and a multi-language application

1. Log on to the Container Service for Kubernetes (ACK) console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Applications** in the **Actions** column.
4. In the left-side navigation pane, choose **Workloads > Deployments**, select a namespace from the **Namespace** drop-down list, and then click **Create from YAML**.
5. On the **Create** page, select a template from the **Sample Template** drop-down list, modify the content of the YAML file in the **Template** section, and then click **Create**.

   You can use the following YAML template to deploy a Java Spring Cloud application:

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     labels:
       app: sc-c
     name: sc-c
     namespace: default
   spec:
     progressDeadlineSeconds: 600
     replicas: 1
     revisionHistoryLimit: 10
     selector:
       matchLabels:
         app: sc-c
     strategy:
       rollingUpdate:
         maxSurge: 25%
         maxUnavailable: 25%
       type: RollingUpdate
     template:
   ```

```
    metadata:
      annotations:
        sidecar.istio.io/inject: "false"
        msePilotAutoEnable: "on"
        msePilotCreateAppName: "sc-c"
      labels:
        app: sc-c
    spec:
      containers:
        - env:
            - name: throwException
              value: 'true'
            - name: JAVA_TOOL_OPTIONS
              value: '-Dspring.cloud.nacos.discovery.server-addr=127.0.0.1:8848' // Rep
lace the IP address with the IP address of the Nacos service registry in use.
            - name: profile.micro.service.envoy.xds.server
              value: 'istiod.istio-system:15012'
            - name: profile.micro.service.envoy.xds.enable
              value: 'true'
          image: >-
            registry.cn-hangzhou.aliyuncs.com/alibabacloud-microservice-demo/sc-c:demo
          imagePullPolicy: Always
          name: sc-c
          resources:
            requests:
              cpu: 250m
              memory: 512Mi
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
-----
apiVersion: v1
kind: Service
metadata:
  name: sc-c
  labels:
    app: sc-c
    service: sc-c
spec:
  ports:
  - port: 20003
    name: http
  selector:
    app: sc-c
```

You can view the created Java Spring Cloud application on the **Deployments** page.

1. In the left-side navigation pane, choose **Workloads > Deployments**, select a namespace from the **Namespace** drop-down list, and then click **Create from YAML**.
2. On the **Create** page, select a template from the **Sample Template** drop-down list, modify the

content of the YAML file in the **Template** section, and then click **Create**.

You can use the following YAML template to deploy a multi-language application:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    version: v1
  name: go-sc-a-v1
  namespace: default
spec:
  progressDeadlineSeconds: 600
  replicas: 1
  revisionHistoryLimit: 10
  selector:
    matchLabels:
      version: v1
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: go-sc-a
        version: v1
    spec:
      containers:
        - env:
            - name: LOG_DIR
              value: /tmp/logs
          image: 'registry.cn-hangzhou.aliyuncs.com/edas_test1/helloa:demo'
          imagePullPolicy: IfNotPresent
          name: go-sc-a
          ports:
            - containerPort: 8085
              protocol: TCP
          resources: {}
          terminationMessagePath: /dev/termination-log
          terminationMessagePolicy: File
          volumeMounts:
            - mountPath: /tmp
              name: tmp
            - mountPath: /opt/ibm/wlp/output
              name: wlp-output
      dnsPolicy: ClusterFirst
      restartPolicy: Always
      schedulerName: default-scheduler
      securityContext: {}
      terminationGracePeriodSeconds: 30
      volumes:
        - emptyDir: {}
          name: wlp-output
```

```
        - emptyDir: {}
          name: tmp
-----
apiVersion: v1
kind: Service
metadata:
  labels:
    app: go-sc-a
    service: go-sc-a
  name: go-sc-a
  namespace: default
spec:
  ports:
  - port: 8085
    name: http
  selector:
    app: go-sc-a
```

You can view the created multi-language application on the **Deployments** page.

Deploy a Java Spring Cloud application

Deploy a multi-language application

Configure environment variables

1. Log on to the ACK console.
2. In the left-side navigation pane, click **Clusters**.
3. On the **Clusters** page, find the cluster that you want to manage and click **Applications** in the **Actions** column.
4. In the left-side navigation pane, choose **Workloads > Deployments**. On the Deployments page, select **ARMS-pilot** from the **Namespace** drop-down list, find the created multi-language application, and then click **Edit** in the Actions column.
5. In the **Environment Variable** section, click **Add**. Create the following environment variable, and click **Update**.

   Environment variable:

   profile.micro.service.envoy.xds.enable: `true`
6. On the **Deployments** page, select the namespace of the created Java Spring Cloud application from the **Namespace** drop-down list, find the application, and then click **Edit** in the **Actions** column.
7. In the **Environment Variable** section, click **Add**. Create the following environment variable and startup parameter, and click **Update**.

   Environment variable and startup parameter:

   - profile.micro.service.envoy.xds.server: `istiod.istio-system:15012`
   - profile.micro.service.envoy.xds.enable: `true`

Implement interoperability between the Java Spring Cloud application and multi-language application

Java Spring Cloud applications can interoperate with multi-language applications that are deployed in an ASM instance. The method for a Java Spring Cloud application to call a multi-language application is the same as that for the Java Spring Cloud application to call another Java application. You do not need to modify the code.

Run the following command to use the created Java Spring Cloud application to call the created multi-language application:

```
curl localhost:20003/go
```

> **Note** The Java application can use its internal API to access the multi-language application.

The following result is returned:

```
[Java Spring Cloud ] -> [Service Mesh APP10.191.XX.XX ]
```

Run the following command to use the created multi-language application to call the created Java Spring Cloud application:

```
curl localhost:8085/java
```

The following result is returned:

```
[ Service Mesh APP ] -> [Java Spring Cloud10.191.XX.XX]
```

Use the Java Spring Cloud application to call the multi-language application

Use the multi-language application to call the Java Spring Cloud application

4.2. Release multilingual applications by using canary releases

If you want to update a multilingual microservice-oriented application that is deployed in a Kubernetes cluster, you can use canary releases. You can update the application and verify the new version on a small number of instances. After the verification succeeds, you can update the application on all instances to the new version. This ensures service stability and performance during the update.

Procedure

1.
2. In the left-side navigation pane, click **Applications**. In the top navigation bar, select a region.
3. On the **Applications** page, select a microservice namespace. Then, select **Container Service or Serverless Kubernetes Cluster** from the **Cluster Type** drop-down list, and click the name of the application that you want to release.
4. In the upper-right corner of the **Application Overview** page, choose **Deploy > Deploy**.

5. On the **Select Deployment Mode** page, click **Start Deployment** in the upper-right corner of the **Canary Release (Phased)** section.
6. On the page that appears, set the parameters in the Service Mesh and Release Policy sections, and configure canary release rules. Then, click **OK**.
   i. Set the parameters in the **Service Mesh** section.

   Parameters in the Service Mesh section

Parameter	Description
Protocol	The protocol that is used by the service. Valid values: http, http2, gRPC, and TCP.
Service Name	The name of the service that is provided by the application.
Service Port	The port number of the service that is provided by the application.

> **Note** The values of the **Service Name** and **Service Port** parameters must be the same as those in the application code. Otherwise, the service cannot be registered or called.

ii. Set the parameters in the **Release Policy** section.

Parameters in the Release Policy section

Parameter	Description
Number of Instances for Canary Release	The number of application instances for the release in the first phase. You can view the total number of instances on the right side of this parameter. To ensure application stability, we recommend that you set this parameter to a value less than half of the total number of instances.
Remaining Batches	The number of remaining phases to release the application. After the application is released in the first phase, the application is released on the remaining instances based on the value of this parameter. **Note** The **Batch Mode** parameter is displayed only when the number of remaining phases is greater than 1.
Deployment Interval Between Batches	If the number of application instances in each phase is greater than 1, the application is deployed to the application instances at the specified interval. Unit: seconds.

The **Publish Policy Configuration** section on the right side shows the procedure for the canary release based on the configuration.

iii. Configure canary release rules.

EDAS supports the following canary release rules: **Canary Release by Content** and **Canary Release by Ratio**.

Parameters for canary release rules

Tab	Parameter	Description
Canary Release by Content	**Protocol Type**	The value is fixed to **Service Mesh**.
	Path	The path in the HTTP requests for the canary multilingual application.
	Conditions	You can set this parameter based on the HTTP header value.
Canary Release by Ratio	**Traffic Ratio**	The proportion based on which the traffic is distributed to the canary application.

> **Note** You can click **Create Inbound Traffic Rule** to create multiple inbound traffic adjustment rules. The rules can take effect at the same time.

iv. (Optional)Configure advanced settings.

After a canary release is started, EDAS deploys the new version of the application to the specified canary instance group. The deployment progress and status are displayed on the **Upgrade History** page.

> **Note** You can monitor the canary traffic to check whether the traffic is distributed to the canary group as expected. For more information, see Monitor canary traffic.

7. After the traffic for the canary release is verified, click **Start the Next Batch** on the right side of the **Upgrade History** page. Complete the release of the subsequent phases.

   If issues occur during the verification, you can click **Undo** in the upper-right corner of the **Upgrade History** page. In the **Undo the Verification** message, click **OK**.

Verify the result

After the canary release is complete, check whether the deployment package is of the new version on the **Application Overview** page.

4.3. Control access to multilingual applications by using service authentication

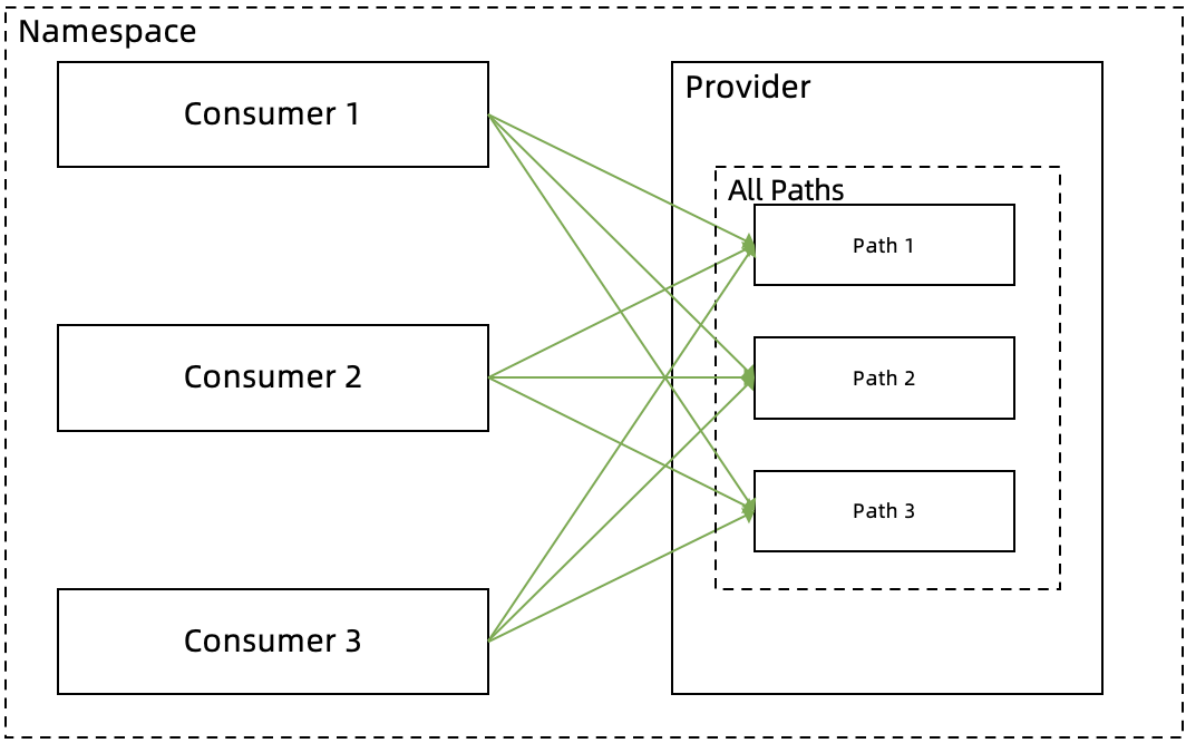
If a microservice-oriented application requires high security and you want to restrict access to it from other applications, you can authenticate the applications that call the microservice-oriented application. This ensures that only the applications that match the authentication rules can call the microservice-oriented application.

Context

This topic uses an example to introduce scenarios where Spring Cloud service authentication is performed.

- Do not configure service authentication

  Consumers 1, 2, and 3 and a service provider are deployed in the same namespace. By default, Consumers 1, 2, and 3 can call all the paths (Paths 1, 2, and 3) of the provider.

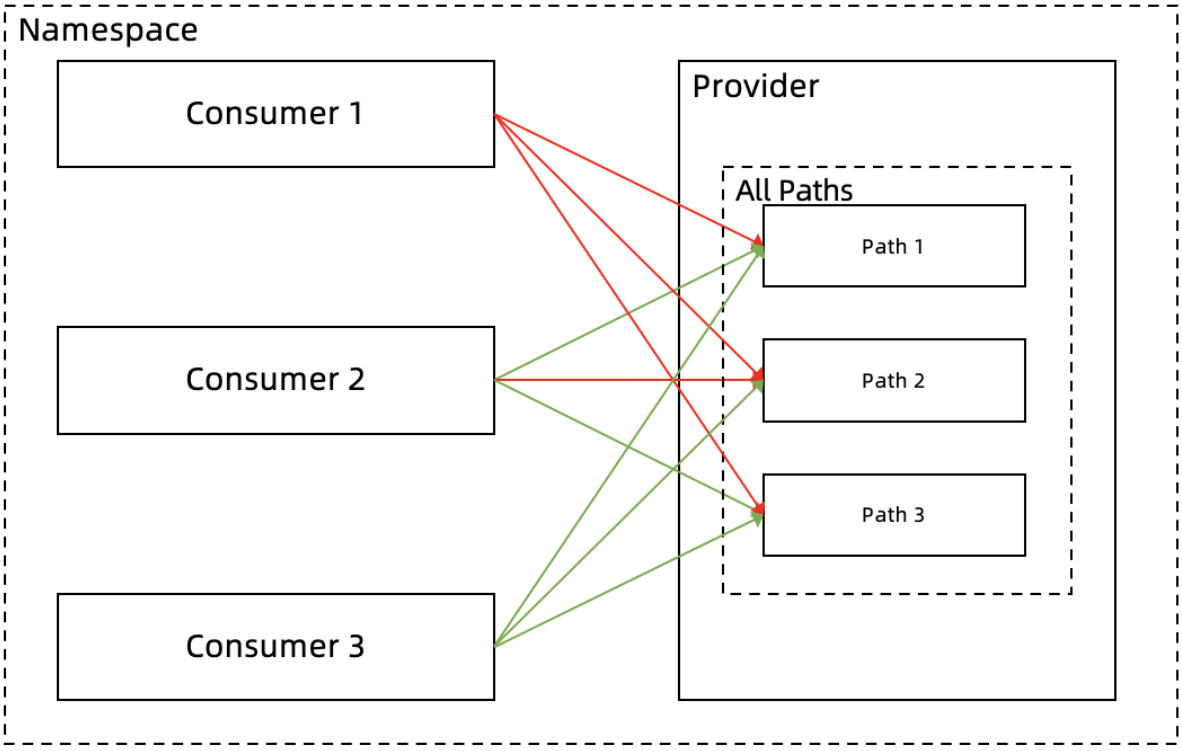


- Configure service authentication
  - Configure an authentication rule for all the paths.

    You can configure an authentication rule for all the paths of the provider. For example, you can configure a blacklist for Consumer 1 to prevent it from calling the paths of the provider, and configure a whitelist for Consumers 2 and 3 to allow them to call the paths of the provider.
  - Configure an authentication rule for a specific path.

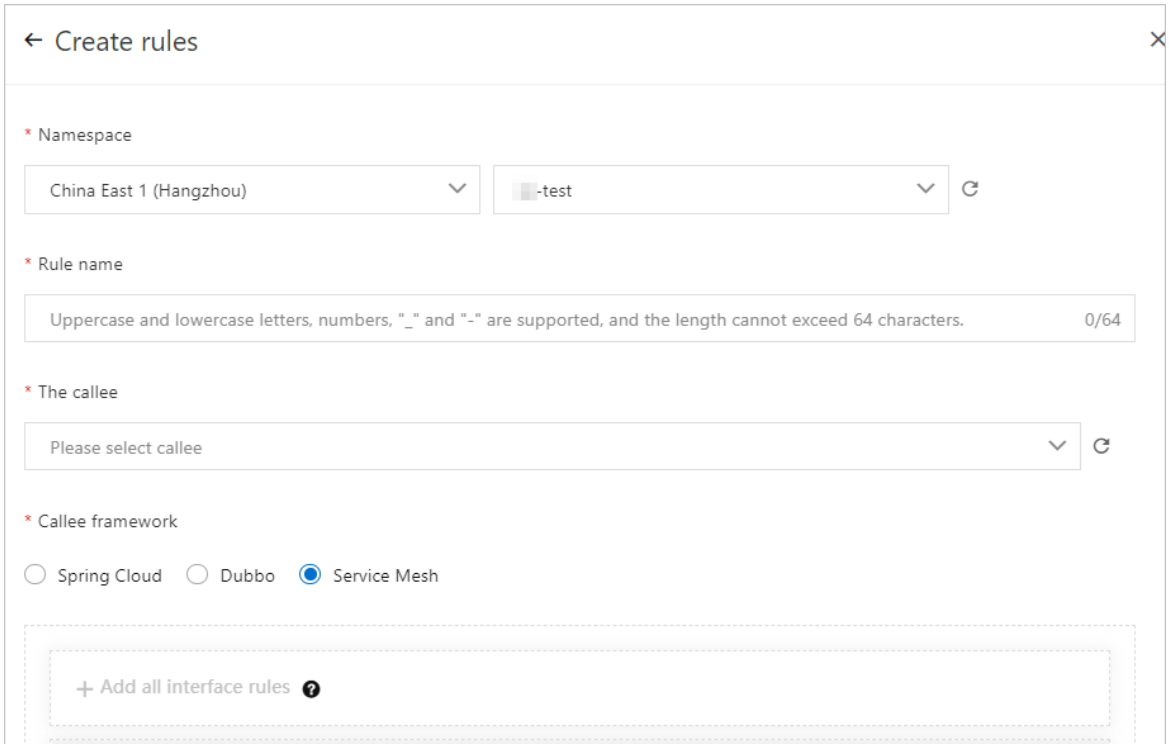
    You can also configure an authentication rule for a specific path of the provider. For example, you can configure a blacklist for Consumer 2 to prevent it from calling Path 2 of the provider because the path involves core business or core data. Then, Consumer 2 can call only Paths 1 and 3 of the provider.

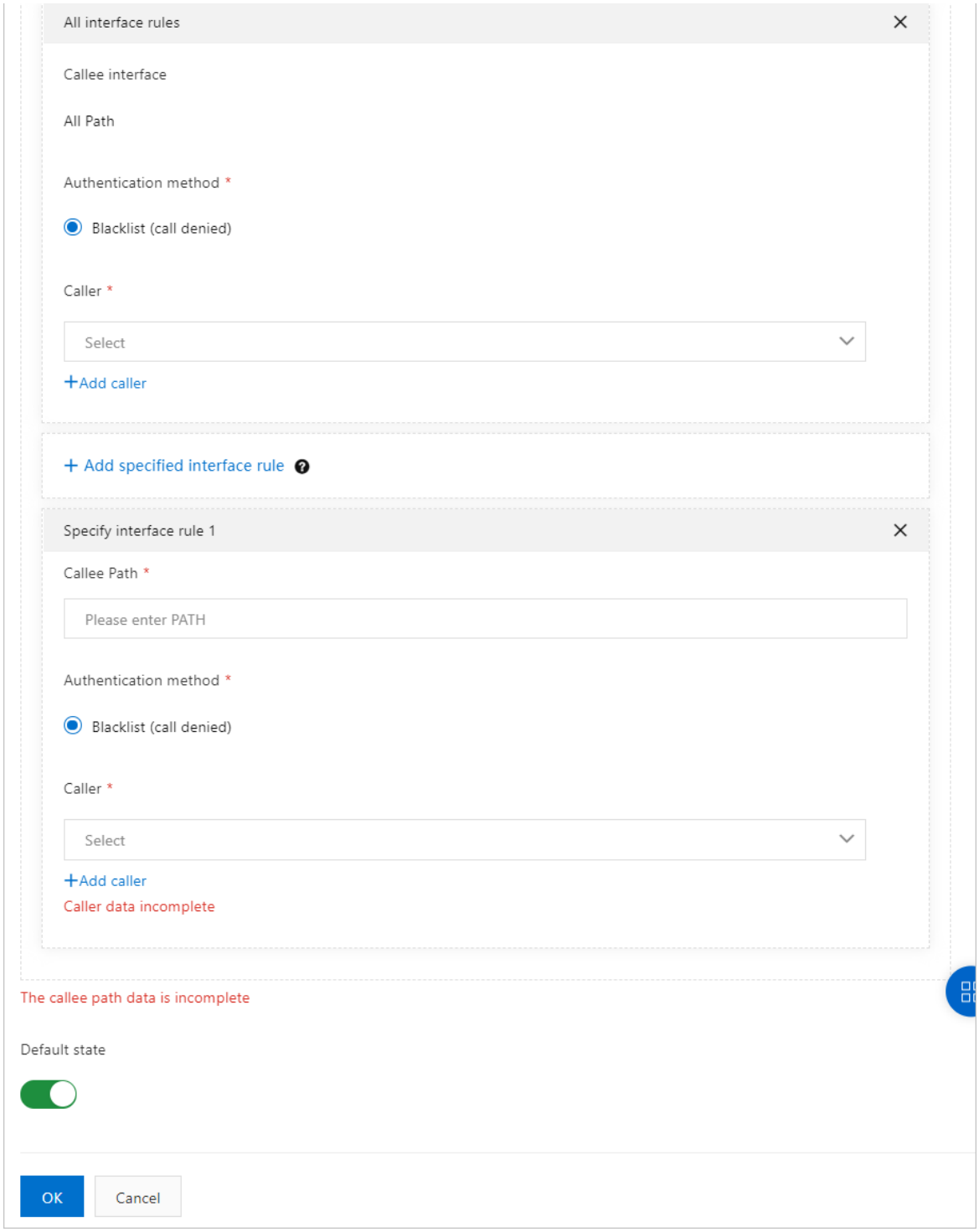
  The following figure shows the application call process after you configure the authentication rules.



Create a service authentication rule

1.
2. In the left-side navigation pane, choose **Microservices Governance > Service Mesh**.
3. In the navigation tree of the **Service Mesh** page, click **Service Authentication**.
4. On the **Service Authentication** page, click **Create rules**.
5. In the **Create rules** panel, set the parameters and click **OK**.





The following table describes the parameters.

Parameter	Description
Microservice Namespaces	The region and microservice namespace where the service resides.
Rule name	The name of the service authentication rule. The name can be up to 64 characters in length, and can contain letters, digits, underscores (_), and hyphens (-).

Parameter	Description
The callee	The called application.
Callee framework	The framework that is used by the called application. In this example, select **Service Mesh**.
Add all interface rules Notice You can create a common rule for all interfaces only once.	
Callee interface	Default value: **Callee interface**. You cannot change the value of this parameter.
All Path	Default value: **All Path**. You cannot change the value of this parameter.
Authentication method	The method that is used for service authentication. Only **Blacklist (call denied)** is supported.
Caller	The application that must be authenticated before it can call the service. To add multiple applications, click **Add caller**.
Add specified interface rule Notice The rule created for a specific interface is not appended. Instead, the rule overwrites the common rule for all interfaces. Exercise caution when you configure this type of rule.	
Callee Path	The path of the called application.
Authentication method	The method that is used for service authentication. Only **Blacklist (call denied)** is supported.
Caller	The application that must be authenticated before it can call the service. To add multiple applications, click **Add caller**.
Default state	Specifies whether to enable the rule. ○ On: enables the rule after it is created. By default, the switch is turned on. ○ Off: disables the rule after it is created. To enable the rule, find the rule on the **Service Authentication** page and click **Open** in the **Operation** column.

Verify the results

After the service authentication rule is created and enabled, check whether the rule takes effect.

What's next

After you create a service authentication rule, you can click **Edit**, **Close**, or **Open** in the Operation column to manage the rule. If the service authentication rule is no longer required, you can click **Delete** in the Operation column to delete the rule.

4.4. Ensure the availability of multilingual applications by removing outlier instances

In a microservice framework, service calls are affected if consumers cannot detect the exceptions on the application instances of a provider. This further affects the performance and even availability of the services provided by the consumers. The outlier ejection feature monitors the availability of application instances and dynamically adjusts the instances. This ensures successful service calls and improves the service stability and quality of service (QoS).

Context

A system includes Applications A, B, C, and D, where Application A calls Applications B, C, and D. If the instances of Application B, C, or D become abnormal and Application A does not identify the abnormal instances, a part of calls initiated by Application A fail. Application B has one abnormal instance, and Applications C and D each have two abnormal instances. If Applications B, C, and D have a large number of abnormal instances, the service performance and availability of Application A may be affected.

To ensure the service performance and availability of Application A, you can configure an outlier application removal policy. After the policy is configured, Enterprise Distributed Application Service (EDAS) can monitor the instance status of Applications B, C, and D, and dynamically add or remove instances to ensure successful service calls.

The following list describes the process of outlier instance removal:

1. EDAS detects whether Applications B, C, and D have abnormal instances. Then, EDAS determines whether to remove the abnormal instances from the applications based on the configured **Upper limit of instance removal ratio** parameter.
2. EDAS does not distribute the call requests of Application A to the removed instances.
3. EDAS detects whether the abnormal instances are recovered based on the configured **Recovery detection unit time** parameter.
4. The detection interval is proportional to the number of detection times and linearly increases by the value of the **Recovery detection unit time** parameter, which is 0.5 minutes by default. If the value of the **Maximum cumulative number of times not restored** parameter is reached, EDAS detects whether the abnormal instances are recovered at the maximum detection interval.
5. After the abnormal instances are recovered, they are added to the instance lists of the applications to continue processing call requests. The detection interval is reset to the value of the **Recovery detection unit time** parameter, such as 0.5 minutes.

> **Note**
>
> - If the provider has a large number of abnormal instances and the ratio of the abnormal instances exceeds the value of the Upper limit of instance removal ratio parameter, the number of actually removed instances equals the configured upper limit.
> - If the provider has only one instance available, this instance is not removed even if the error rate exceeds the configured limit.

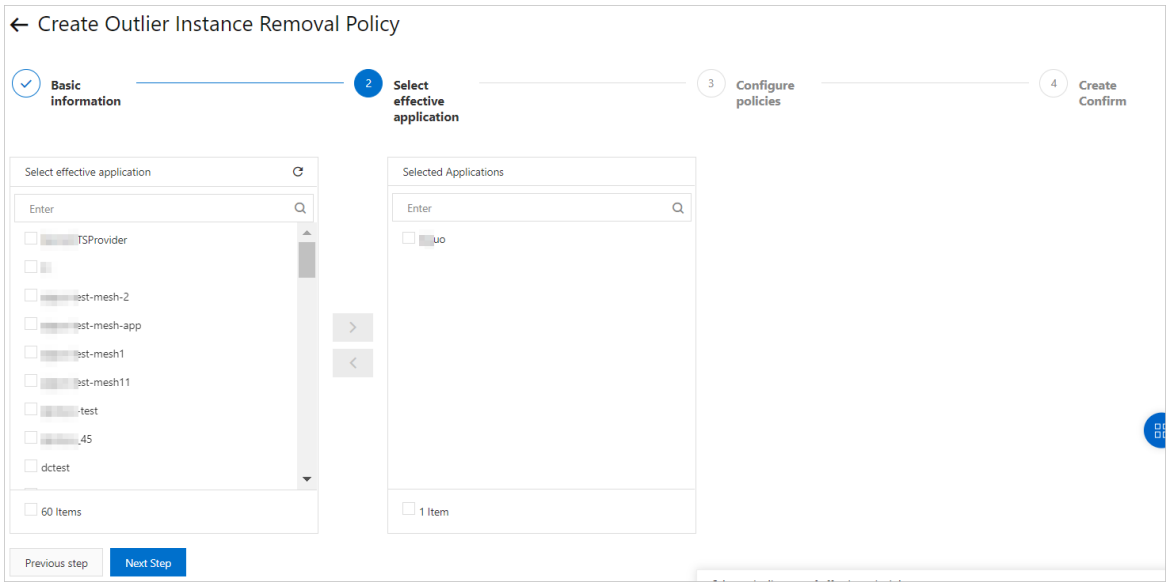
Create an outlier ejection policy

1.
2. In the left-side navigation pane, choose **Microservices Governance > Service Mesh**.
3. In the navigation tree of the **Service Mesh** page, click **Outlier Instance Removal**.
4. In the top navigation bar, select a region. On the **Outlier Instance Removal** page, select a microservice namespace and click **Create Outlier Instance Removal Policy**.
5. In the **Basic information** step of the **Create Outlier Instance Removal Policy** wizard, set the parameters and click **Next Step**.

   The following table describes the parameters in the **Basic information** step.

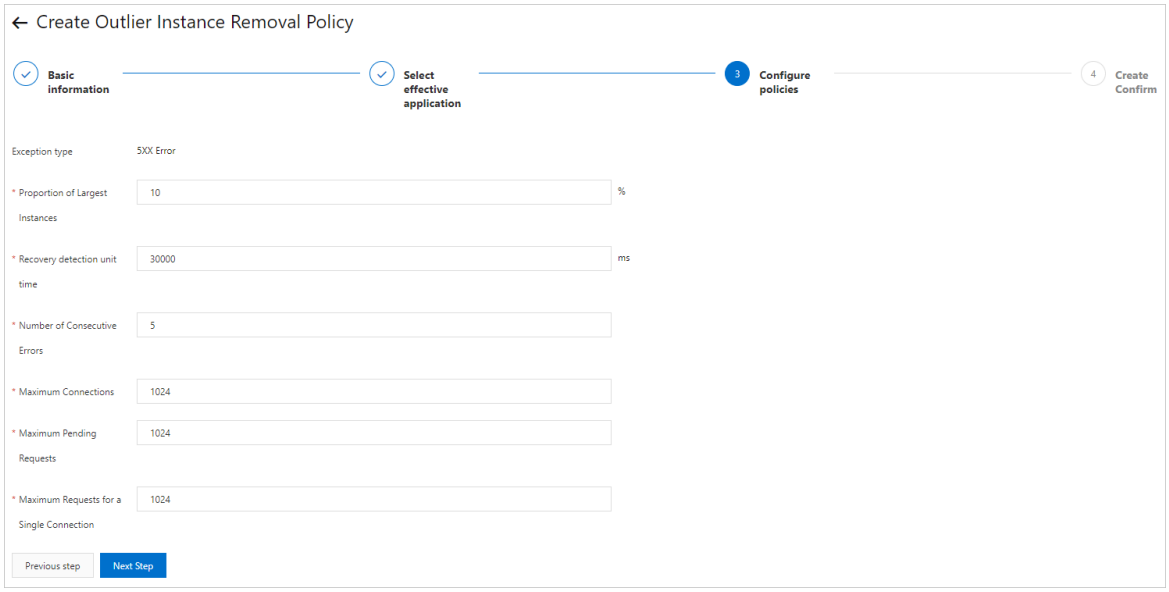
Parameter	Description
Microservice Namespace	Select a region and a microservice namespace.
Policy Name	Enter a name for the policy. The name can be up to 64 characters in length.
Framework	Select **Service Mesh**.

6. In the **Select effective application** step of the **Create Outlier Instance Removal Policy** wizard, select an application and click the **>** icon to add the application to the Selected Applications section. Then, click **Next Step**.



After the application is selected, the abnormal instances of all the applications that are called by this application can be removed. Call requests from this application are not distributed to the removed instances.

7. In the **Configure policies** step of the **Create Outlier Instance Removal Policy** wizard, set the parameters and click **Next Step**.

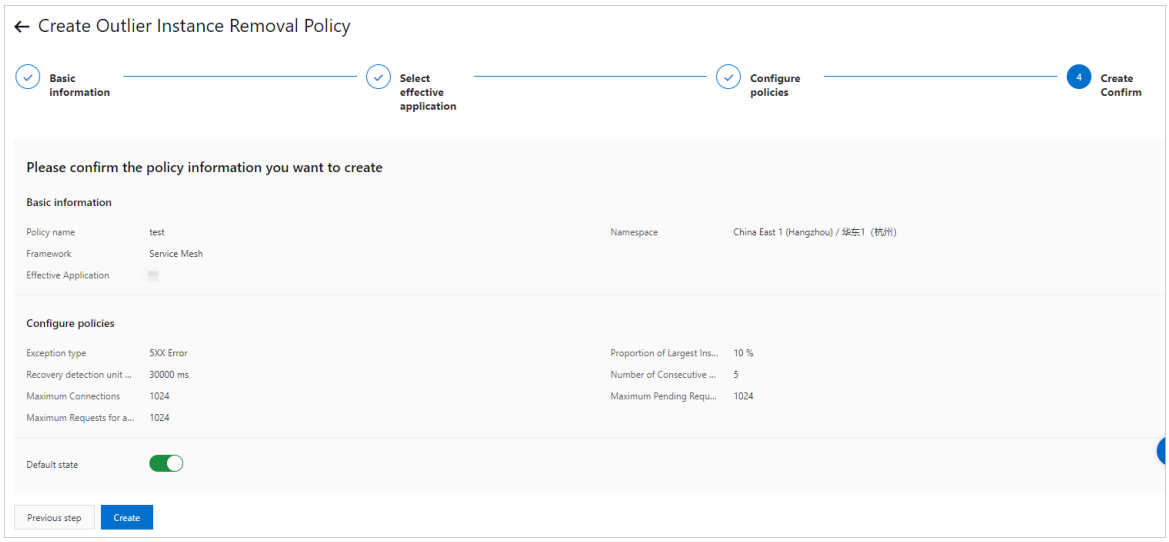


The following table describes the parameters in the **Configure policies** step.

Parameter	Description
Exception type	Default value: **5XX Error**. You cannot change the value of this parameter.
Proportion of Largest Instances	Enter the upper limit for the proportion of abnormal instances that can be removed. If the limit is reached, no more abnormal instances are removed. For example, an application has six instances in total. If you set this parameter to 60%, the maximum number of instances that can be removed is 3.6, which is rounded down to the nearest integer 3. The number is calculated by using the following formula: 6 × 60% = 3.6. If the calculated result is less than 1, no instances are removed.
Recovery detection unit time	Specify a unit interval to detect whether abnormal instances are recovered, in milliseconds. After abnormal instances are removed, EDAS linearly increases the detection interval by the specified unit interval. The default unit interval is 30,000 ms, which is equal to 0.5 minute.
Number of Consecutive Errors	Specify the threshold of the number of consecutive errors during requests. If the threshold is reached on an instance, the instance is removed.

Parameter	Description
Maximum Connections	Specify the maximum number of connections that are supported by a service. Default value: 1024.
Maximum Pending Requests	Specify the maximum number of pending requests that are supported by a service. Default value: 1024.
Maximum Requests for a Single Connection	Specify the maximum number of requests that are supported by a connection to a service. Default value: 1024.

8. In the **Create Confirm** step of the **Create Outlier Instance Removal Policy** wizard, verify the settings and click **Create**.



Verify the result

The outlier ejection feature is enabled after you configure and create an outlier ejection policy. You can go to the details page of the application for which you have configured outlier ejection to view the application monitoring information. For example, you can check whether call requests are still forwarded to abnormal instances and whether the error rate per minute for application calls is higher than the value of the **Error Rate Threshold** parameter in a topology. This way, you can check whether the outlier ejection policy takes effect.

What to do next

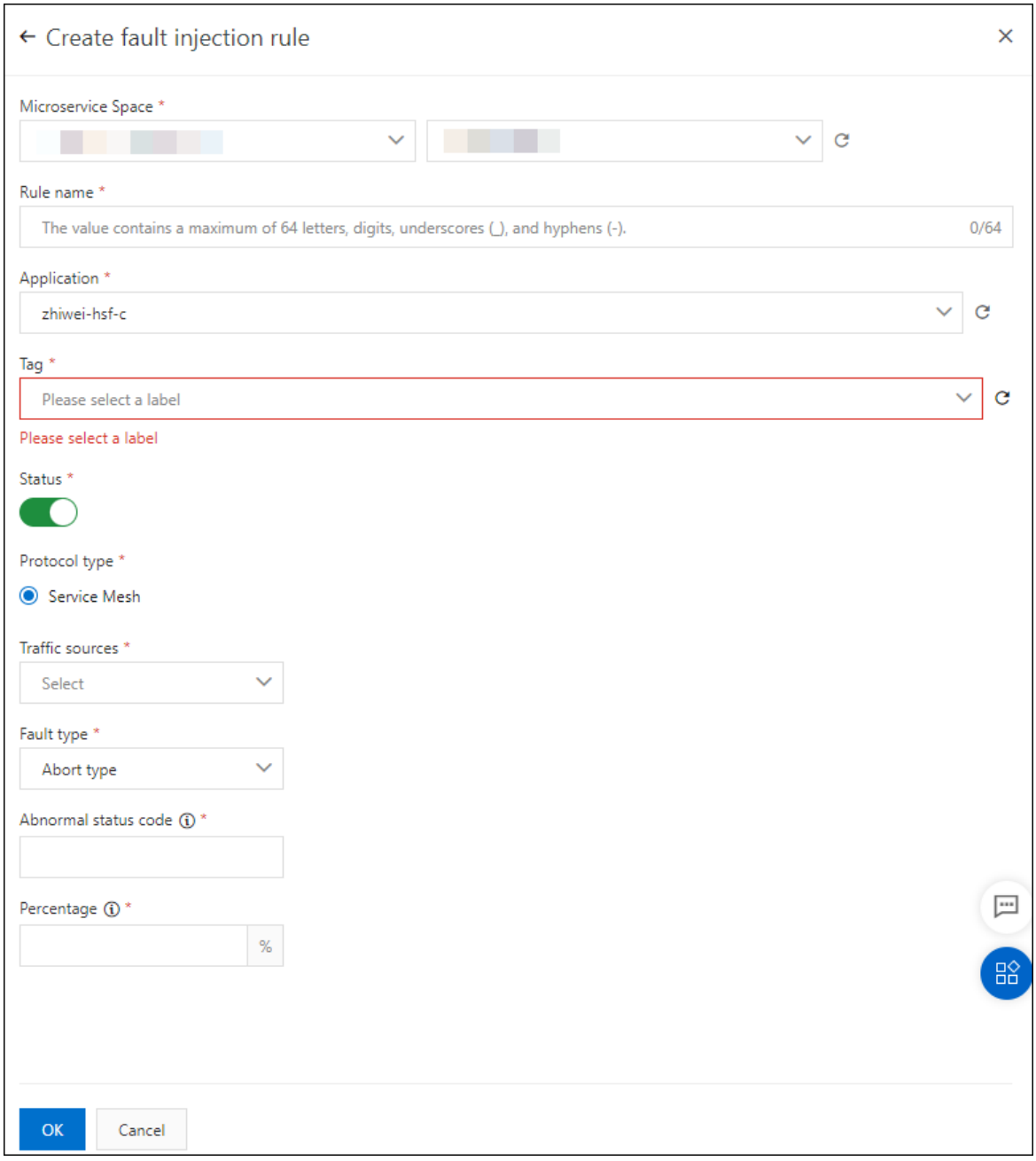
On the **Outlier Instance Removal** page, you can click **Edit** or **Delete** in the Operation column to manage the policies.

4.5. Create a fault injection rule for a multi-language application

Fault injection is a method to imitate the scenario in which a specific fault occurs. You can inject a fault into an application to test how the consumer applications process this fault. If a consumer application cannot properly process the fault, you can optimize the consumer application to improve its stability. This topic describes how to create a fault injection rule for a multi-language application.

Create a fault injection rule

1.
2. In the left-side navigation pane, choose **Microservices Governance > Service Mesh**.
3. In the left-side navigation tree of the **Service Mesh** page, click **Fault Injection**.
4. Select a region in the top navigation bar. Select a microservice namespace from the drop-down list next to **Fault Injection** and click **Create rule**.
5. In the **Create fault injection rule** panel, set the parameters and click **OK**.



The following table describes the parameters.

Parameter	Description
Microservice Space	The region and microservice namespace where the application resides.
Rule name	The name of the fault injection rule, such as fault-example.
Application	The application for which you want to inject a fault.
Tag	The tag that implements tag-based routing.
Status	Specifies whether to enable the fault injection rule. ○ On: enables the rule after it is created. ○ Off: disables the rule after it is created. To enable the rule, find the rule on the Fault Injection page and click **Open** in the **Operation** column.
Protocol type	The type of the application framework. Default value: **Service Mesh**.
Traffic sources	The one or more consumer applications that send requests. You can select All or specify specific applications. **Note** The fault can be injected based on the percentage that you set only if the consumer application that sends a request is specified for this parameter.
Fault type	The type of the fault that you want to inject. Valid values: **Abort type** and **Delay type**.
Abnormal status code	The HTTP status code to return for a requestif the fault is injected into a consumer application. Valid values: 200 to 599. This parameter is available only if you set the Fault type parameter to **Abort type**.
Fixed delay time	The latency to send a request if the fault is injected into a consumer application. Unit: milliseconds. This parameter is available only if you set the Fault type parameter to **Delay type**.
Percentage	The possibility that the fault is injected into a consumer application.

After the fault injection rule is created and enabled, check whether the rule takes effect.

Related operations

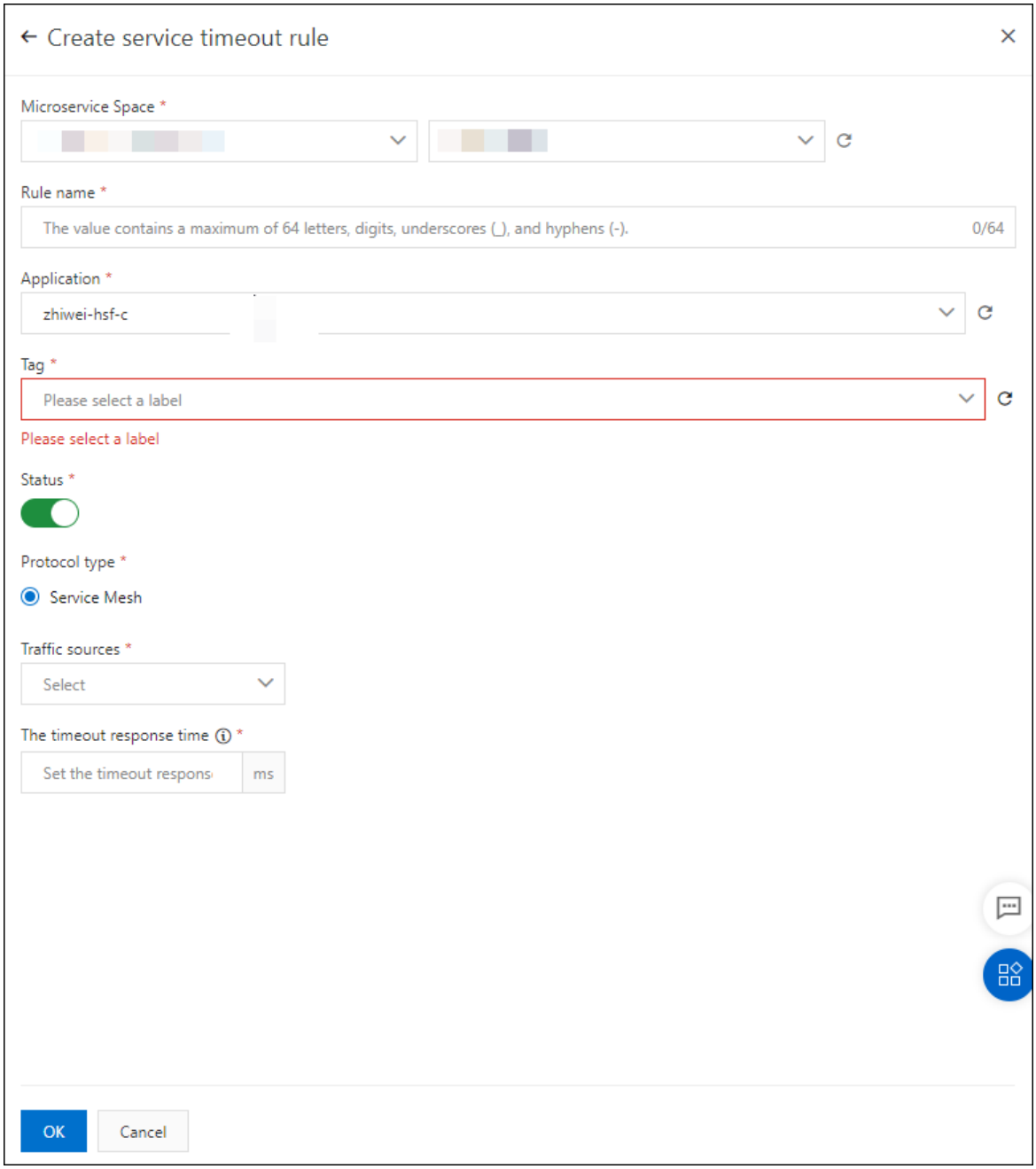
After you create a fault injection rule, you can click **Edit**, **Close**, or **Open** in the Operation column to manage the rule. If the fault injection rule is no longer required, you can click **Delete** to delete the rule.

4.6. Create a service timeout rule for a multi-language application

Microservice Governance in Enterprise Distributed Application Service (EDAS) allows you to create a service timeout rule for a multi-language application without the need to modify the business code. After you create a service timeout rule for an application, if the application fails to process a request within the specified timeout period, a timeout error is returned. This prevents the consumer applications from waiting a large amount of time.

Create a service timeout rule

1.
2. In the left-side navigation pane, choose **Microservices Governance > Service Mesh**.
3. In the left-side navigation tree of the **Service Mesh** page, click **Service Timeout Period Setting**.
4. Select a region in the top navigation bar. Select a microservice namespace from the **Microservice Namespace** drop-down list and click **Create rule**.
5. In the **Create service timeout rule** panel, set the parameters and click **OK**.



The following table describes the parameters.

Parameter	Description
Microservice Space	The region and microservice namespace where the application resides.
Rule name	The name of the service timeout rule, such as timeout-example.
Application	The application for which you want to create the service timeout rule.
Tag	The tag that implements tag-based routing.

Parameter	Description
Status	Specifies whether to enable the service timeout rule. ○ On: enables the rule after it is created. ○ Off: disables the rule after it is created. To enable the rule, find the rule on the Service Timeout Period Setting page and click **Open** in the **Operation** column.
Protocol type	The type of the application framework. Default value: **Service Mesh**.
Traffic sources	The one or more consumer applications that send requests. You can select All or specify specific applications. **Note** A timeout error can be returned based on the rule only if the consumer application that sends the request is specified for this parameter.
The timeout response time	The timeout period of requests. If the application fails to process a request within the specified period, a timeout error is returned. Unit: milliseconds.

After the service timeout rule is created and enabled, check whether the rule takes effect.

Related operations

After you create a service timeout rule, you can click **Edit**, **Close**, or **Open** in the Operation column to manage the rule. If the service timeout rule is no longer required, you can click **Delete** to delete the rule.

4.7. Create a service retry rule for a multi-language application

After you create a service retry rule for a multi-language application, if the application is temporarily inaccessible or encounters an occasional error, a request is retired to ensure that the expected results can be returned. This improves the robustness of the system. This topic describes how to create a service retry rule for a multi-language application.

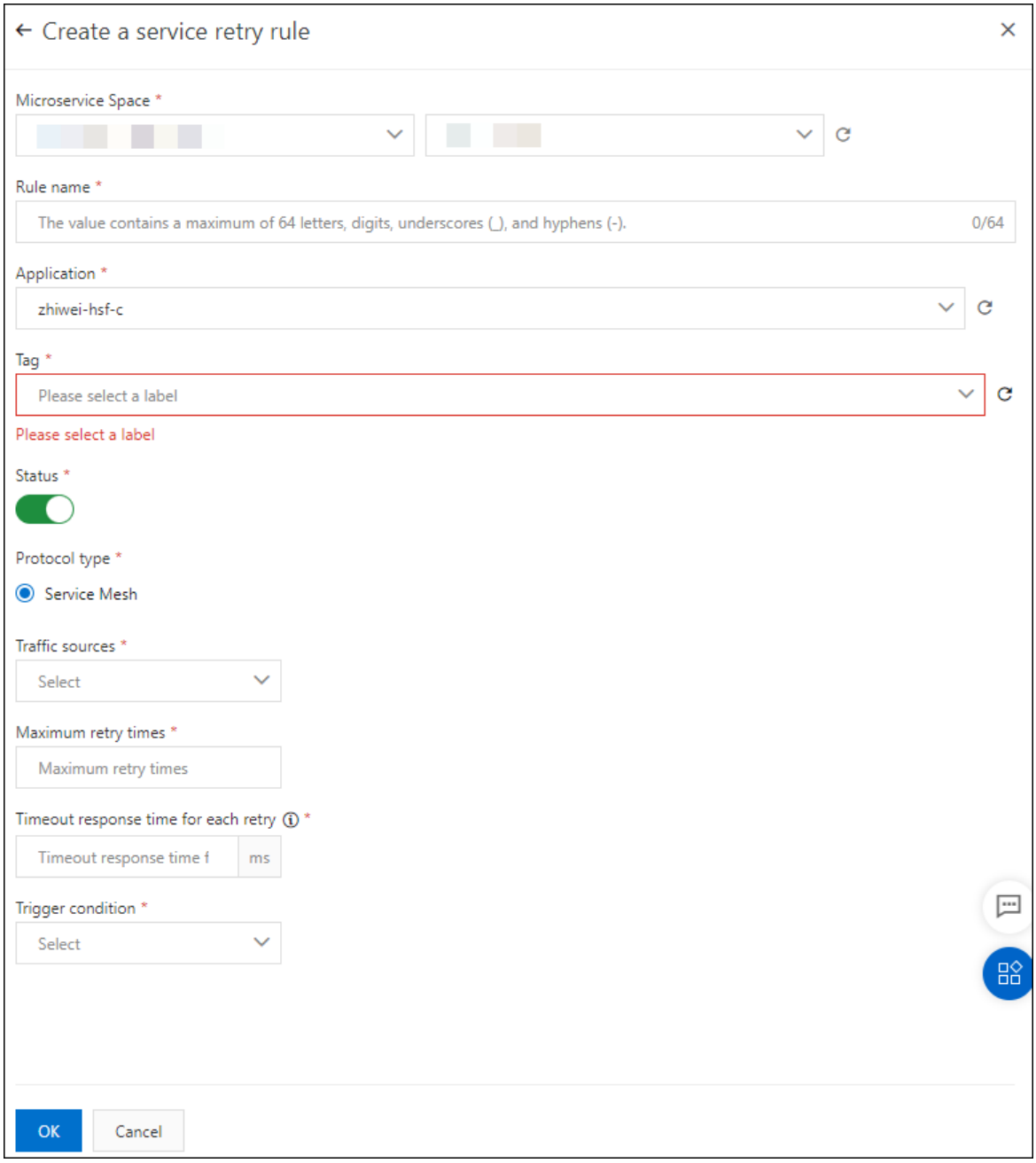
Precautions

If you create both a service timeout rule and a service retry rule for an application, the timeout period that you specify for the former rule may affect the number of retries allowed.

For example, you set the timeout period to 1,000 ms and the maximum number of retries to five for errors with a code of 5xx for an application. If the application takes 300 ms to process a request, a timeout error occurs after the application attempts to process the third retry although a maximum of five retries are allowed. The following figure shows the detailed information:

Create a service retry rule

1.
2. In the left-side navigation pane, choose **Microservices Governance > Service Mesh**.
3. In the left-side navigation tree of the **Service Mesh** page, click **Service Retry Setting**.
4. Select a region in the top navigation bar. Select a microservice namespace from the drop-down list next to **Retry** and click **Create rule**.
5. In the **Create a service retry rule** panel, set the parameters and click **OK**.



The following table describes the parameters.

Parameter	Description
Microservice Space	The region and microservice namespace where the application resides.
Rule name	The name of the service retry rule, such as retry-example.
Application	The application for which you want to create the service retry rule.
Tag	The tag that implements tag-based routing.
Status	Specifies whether to enable the service retry rule. ○ On: enables the rule after it is created. ○ Off: disables the rule after it is created. To enable the rule, find the rule on the Retry page and click **Open** in the **Operation** column.
Protocol type	The type of the application framework. Default value: **Service Mesh**.
Traffic sources	The one or more consumer applications that send requests. You can select All or specify specific applications. **Note** A request can be resent only if the consumer application that sends the request is specified for this parameter.
Maximum retry times	The maximum number of retries allowed.
Timeout response time for each retry	The timeout period of retries. If the application fails to process the retry within the specified period, a timeout error is returned. Unit: milliseconds. **Note** If more retries are allowed, the request can be resent.

Parameter	Description
Trigger condition	The condition to trigger retries. Valid values: ○ 5xx: resends the request if an HTTP status code from 500 to 599 is returned. The value 5xx represents the following four trigger conditions: ■ gateway-error: resends the request if the 502, 503, or 504 HTTP status code is returned. ■ reset: resends the request if no results are returned. ■ connect-failure: resends the request if the network connection fails. For example, retires are triggered if the connection times out. ■ reused-stream: resends the request if the stream is refused. ○ retriable-4xx: resends the request if the 409 HTTP status code is returned.

After the service retry rule is created and enabled, check whether the rule takes effect.

Related operations

After you create a service retry rule, you can click **Edit**, **Close**, or **Open** in the Operation column to manage the rule. If the service retry rule is no longer required, you can click **Delete** to delete the rule.