ALIBABA CLOUD

Alibaba Cloud

Enterprise Distributed Application Service (EDAS)

Best Practices

Document Version: 20220712

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.ECS

1.1. Create ECS instances by using an instance launch template

Enterprise Distributed Application Service (EDAS) is seamlessly integrated with instance launch templates. In scenarios such as application creation, scale-out, and auto scaling, you can use an instance launch template to create resources for Elastic Compute Service (ECS) clusters in EDAS. This helps you create resources efficiently.

Introduction

An instance launch template persists ECS instance configurations and helps you create ECS instances. For more information about how to create an instance launch template, see Create a launch template. An instance launch template contains configurations that you can use to create instances. It can include all configurations except for passwords, such as key pairs, RAM roles, instance types, and network configurations. An instance launch template cannot be modified. However, you can create multiple versions of each template and set different parameters for each version. Instance configurations are updated with versions. For more information about how to create another version of the template, see Create a launch template version. Then, you can create an instance by using any version of the template.

In EDAS, when you create an instance by using an instance launch template or based on existing instance specifications, the instance you create is billed in pay-as-you-go mode. For more information, see Pay-as-you-go. After you release an instance, the billing rules vary with the recycling mode that you select when you create the instance:

- **Release Mode**: After an application is scaled in, EDAS automatically releases the instance that is no longer used. You need only to pay for the usage of the instance during its service period.
- **Shutdown and Reclaim Mode**: After an application is scaled in, the instance that is no longer used is stopped and is not billed for CPU and memory resources. However, disks such as system disks and data disks, elastic IP addresses (EIPs), and bandwidth are still billed. The public IP address will be recycled and reassigned upon startup, whereas the EIP is retained. You need only to pay a low fee for the storage to retain the instance.

When EDAS directs you to configure the logon credentials for an instance launch template in the ECS console, we recommend that you use an SSH key pair. For more information, see Overview. A key pair is much more secure than a common password because a key pair can prevent brute-force cracking. In addition, it is impossible for others to derive a private key from a public key. In EDAS, for access control between ECS instances or between an ECS instance and a cloud service, we recommend that you use a security group. For more information, see Overview. If you configure a security group when you create an ECS template, the ECS instances that you create by using the template in EDAS belong to the configured security group. Therefore, you can configure security group rules to control access permissions of the created ECS instance.

Prerequisites

An instance launch template is created. For more information, see Create a launch template.

> **Notice**
>
> - The created instance launch template and your application must be in the same virtual private cloud (VPC). Otherwise, you cannot select the created instance launch template.
> - You must specify a vSwitch for the instance launch template when you select a VPC. Otherwise, the instance launch template cannot be used in EDAS.

Limits

In EDAS, instance launch templates can be used to create instances only for ECS clusters, not for Kubernetes clusters.

Add an instance by using an instance launch template when you create an application

1.
2. In the left-side navigation pane, click **Applications**.
3. On the **Applications** page, select the region in which you want to create an application in the top navigation bar. In the upper part of the page, select the namespace that you want to use. Then, click **Create Application** in the upper-left corner of the page.
4. In the **Basic Information** step, set the **Cluster Type**, **Application Runtime Environment**, **Application Name**, and **Application Description** parameters, and then click **Next**. The application description is optional.
   - **Cluster Type**: Only an ECS cluster allows you to purchase an instance by using an instance launch template. Therefore, select **ECS Clusters**.
   - **Application Runtime Environment**: You can select **Java**, **Tomcat**, or **EDAS-Container (HSF)**. In this example, **EDAS-Container (HSF)** is selected.
5. In the **Configurations** step, set the **Instance Source** parameter to **Purchase Instance**, and then click **Next**.
   - **Environment**
     - If you do not have a VPC, namespace, or cluster, EDAS creates a default environment for you.
     - If you have created resources such as VPCs, namespaces, and clusters, a drop-down list is displayed for corresponding resources. You can select a resource from the drop-down list.
   - **Instance Source**: Select **Purchase Instance**, and then select **Purchase Based on Instance Launch Template** for **Purchase Method**.
     - From the **Select a launch template** drop-down list, select the template and template version for creating the instance. If no instance launch template is available, you can create a template in the ECS console. For more information, see Create a launch template.
     - Set the **Recycling Mode** parameter.
     - **Quantity**: Select the quantity of instances to be purchased, such as **1**.
     - **Terms of Service**: Select **Elastic Compute Service Terms of Service | Terms of Service for Images**.
6. In the **Advanced Settings** step, set the **Version** and **Application Health Check** parameters, and then click **Create Application**. The health check configuration is optional.

- **Version**: Enter a version in the format of yyyy-mm-dd hh:mm:ss. You can also enter other version IDs.
- **Application Health Check**: optional. You can set the URL for health check, which is used to check whether the application is healthy.

7. In the **Creation Completed** step, check information in the Basic Information, Configurations, and Advanced Settings sections, and then click **Create Application**.

Add an instance by using an instance launch template when you manually scale out an application

1.
2.
3. Click an application. On the application details page, click **Scale Out** in the upper-right corner. In the **Add Instance** dialog box, set the **Target Group** parameter in the **Scale-out Method** step.
4. Select **Purchase Based on Instance Launch Template** for the **Scale-Out Method** parameter.
5. Select the template and template version, set the **Recycling Mode** parameter, and then click **Next**.
   - **Use Bound Template**: You must bind an instance launch template to the instance group. For more information, see Bind an instance launch template to the group. Then, select the bound template for the scale-out.
   - **Use Specified Template**: If you have created multiple templates in the ECS console, you need to select a **specific template** and a **version**.
6. In the **Purchase Details** step, set the **Quantity** parameter, select **Elastic Compute Service Terms of Service Terms of Service for Images**, and then click **Next**.
7. In the **Confirm** step, check the quantity of ECS instances to be purchased and the information about the instance launch template. Verify all the information and click **Confirm**.
   In the upper part of the page, the **Automatic purchasing is triggered. Check the real-time information in the application change process** message appears.

Add an instance by using an instance launch template for auto scaling

You can add instances only for auto scaling of High-Speed Service Framework (HSF) applications in ECS clusters.

1.
2.
3. On the application details page, click **Auto Scaling** in the left-side navigation pane.
4. Turn on **Scale-out Rule**.
5. Set the scale-out rule parameters and click **Save**.
   i. **Trigger Metrics**: Set the thresholds of RT, Load, and CPU. When a threshold is exceeded, a scale-out is triggered.
   ii. **Trigger Conditions**:
      - **Any One of the Metrics**: A scale-out is triggered when the threshold of a metric is exceeded.
      - **All Metrics**: A scale-out is triggered only when the thresholds of all metrics are exceeded.

iii. **Last for More Than**: the duration when the metric continuously reaches the threshold, in minutes. Within the duration, if the average value of a metric every minute continuously reaches the set threshold, a scale-out is triggered. You can configure the duration based on the sensitivity of the cluster service capabilities.

iv. **Application Source**: Select **Elastic Resources**.

- **Creation Method**: Select **Purchase Based on Instance Launch Template**.
- **Launch Template**: Click **Select Template**. In the **Select a launch template** dialog box, select the template and template version, set the **Recycling Mode** parameter, and then click **OK**.
- **Terms of Service**: Select **Elastic Compute Service Terms of Service | Terms of Service for Images**.
- **Advanced Options**: Turn it on and set the **Network Type** and **Multi-zone Scaling Policy** parameters.
  - **Network Type**: Specify the network in which the application to be scaled out is located, which cannot be changed. If the network is a VPC, you must specify the vSwitch that is connected to the new instance. If you specify multiple vSwitches, EDAS will automatically allocate vSwitches based on the multi-zone scaling policy.
  - **Multi-zone Scaling Policy**: You can select **Priority Policy** or **Balanced Distribution Policy**.

v. **Number of Instances to Add for Each Scale-Out**: the number of instances that are automatically added in each scale-out. You can set this parameter based on the service capabilities of a single instance of the application.

vi. **Maximum Number of Instances in Group**: the maximum number of instances in a cluster. If the maximum number is reached, the scale-out stops. You can set this parameter based on the resource quota.

Verify the result

After you add an instance by using an instance launch template, you can view the number and status of the application instances on the Instance Information tab.

Additional information

- Add ECS instances to an application that is deployed in an ECS cluster: This topic describes how to manually scale out an application in three ways to balance the load of the application instances.
- Auto scaling: This topic describes how to dynamically adjust the number of application instances by using auto scaling to balance the load of the application instances.

1.2. Fix the JDK version during an ECS application change

This topic describes how to fix the Java Development Kit (JDK) version in the process of changing an ECS application. This way, new Elastic Compute Service (ECS) instances that are created for the application during a scale-out use the same JDK version as existing ECS instances for the application.

Context

By default, the latest version of OpenJDK is installed on ECS instances that are imported an ECS cluster in EDAS. However, existing ECS instances for an application may not use the latest version of OpenJDK. As a result, new ECS instances that are created for the application during a scale-out may use an JDK version different from that on existing ECS instances for the application. This causes exceptions during the execution of business code.

You can fix the JDK version during an ECS application change to prevent such exceptions. You can add a code snippet to be executed to the pre-launch script in mount scripts. In the code snippet, you can specify the download URL of JDK of the specified version. This way, the JDK version remains unchanged on ECS instances for the application during an application change, such as a scale-out or restart.

Procedure

1. Obtain the download URL of JDK of the specified version.
   i. Download the JDK package of the specified version, such as *oracle-jdk-8u202-linux-x64.tar.gz*, to your on-premises computer.
   ii. Upload the JDK package to an Object Storage Service (OSS) bucket in the same region as the ECS instances for the ECS application. For more information, see Upload objects.
   iii. Obtain the download URL of the JDK package and add -internal to the download URL. For more information about how to obtain the download URL, see Share objects.

   For example, the download URL of *oracle-jdk-8u202-linux-x64.tar.gz* is *http://doctest.oss-cn-hangzhou-internal.aliyuncs.com/tmp/oracle-jdk-8u202-linux-x64.tar.gz*.
2. Add the download URL of the JDK package to the code snippet.

   Set the *JDK_DOWNLOAD_URL* variable in the following code snippet to the download URL of the JDK package:

```
JDK_DOWNLOAD_URL="http://doctest.oss-cn-hangzhou-internal.aliyuncs.com/tmp/oracle-jdk-8u202-linux-x64.tar.gz"
JDK_DOWNLOAD_TMP_FILE="/tmp/oracle-jdk-8u202.tar.gz"
JDK_HOME="/opt/edas/jdk"
JAVA_HOME="${JDK_HOME}/java"
if [ ! -f "${JAVA_HOME}/bin/java" ]; then
   rm -rf ${JAVA_HOME} && mkdir -p ${JDK_HOME}
   wget -q --dns-timeout=2 --connect-timeout=3 --read-timeout=30 ${JDK_DOWNLOAD_URL} -O ${JDK_DOWNLOAD_TMP_FILE}
   [ -f "${JDK_DOWNLOAD_TMP_FILE}" ] && tar zxf ${JDK_DOWNLOAD_TMP_FILE} -C ${JDK_HOME} && rm -f ${JDK_DOWNLOAD_TMP_FILE}
   [ -n "$(ls -ld ${JDK_HOME}/jdk* 2>/dev/null)" ] && mv ${JDK_HOME}/jdk* ${JAVA_HOME}
fi
chmod -R 755 ${JAVA_HOME}
```

3. Add the preceding code snippet to the pre-launch script of the ECS application.
   i. Log on to the .
   ii. In the left-side navigation pane, click **Applications**. In the top navigation bar, select the region in which the application resides. In the upper part of the **Applications** page, select the microservice namespace in which the ECS application resides. Then, find the ECS application and click the name of the ECS application.
   iii. In the **Application Settings** section of the **Basic Information** tab, click **Mount Script**.

iv. In the Pre-launch Script section of the **Mount Script** dialog box, turn off Ignore failed, add the code snippet that you prepared in Step 2, and then click **Modify**.

4. Restart the ECS application. Check whether the JDK version of the ECS application is the same as that specified.

> **Note** After the ECS application is restarted, you can also log on to an ECS instance for the application and check whether the JDK version is the same as that specified.

Additional information

- If you want to use the specified version of OpenJDK, add the command **yum install -y fontconfig** for installing the fontconfig library to the code snippet in the pre-launch script.
- If the ECS application needs to use JDK 11 or JDK of the specified version from another vendor, you can also configure a mount script to install JDK of the specified version to the */opt/edas/jdk/java* or */opt/ali/alijdk* directory. This is another method to fix the JDK version.

1.3. Configure an SSL certificate and enable secure HTTPS access

Secure Sockets Layer (SSL) encryption is the most common method for protecting data that is sent over the Internet. This topic describes how to bind an SSL certificate purchased from a trusted certification authority (CA) to an Enterprise Distributed Application Service (EDAS) application.

Purchase an SSL certificate

To configure SSL for an application, you must obtain an SSL certificate signed by a CA, a trusted third party that issued the certificate for this purpose. If you have no SSL certificate, you must purchase an SSL certificate from a company that sells SSL certificates.

- **Alibaba Cloud SSL Certificates Service**: If you want to use an Alibaba Cloud SSL certificate, you can purchase it from Alibaba Cloud SSL Certificates Service. For more information, see Purchase an SSL Certificates Service instance.
- **Third-party CA**: For information about how to obtain SSL certificates from a third-party CA, see the documentation that is provided by the CA.

Bind an SSL certificate to an application that is deployed with a WAR package

To bind an SSL certificate to an application that is deployed with a WAR package, package the certificate file in the WAR package and use the WAR package to deploy the application. Then, modify the `Connector` parameter in the `server.xml` file in Tomcat settings.

1. Package the certificate file into the WAR package and record the path of the certificate file. Example: *jks_path*.
2. Use the WAR package to deploy an application in the EDAS console. For more information, see Create and deploy an application in an ECS cluster.
3.
4.
5. On the **Basic Information** tab of the application details page, click **Edit** next to **Tomcat Context** in the **Application Settings** section.
6. In the **Application Settings** dialog box, click **Advanced Settings**. Modify the Connector parameter to the following configuration in *server.xml* and click **Configure Tomcat**.

   ```
   <Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true" scheme="https" secure="tru
   e" keystoreFile="../app/{app_ID}/{app_name}/{jks_path}" keystoreType="PKCS12" keystoreP
   ass="jks_password" clientAuth="false" SSLProtocol="TLS" connectionTimeout="15000" maxPa
   rameterCount="1000" maxThreads="400" maxHttpHeaderSize="16384" maxPostSize="209715200"
   acceptCount="200" useBodyEncodingForURI="true" URIEncoding="ISO-8859-1">
   ```

   > **Note** The advanced settings in the Application Settings dialog box of Tomcat Context are available only for applications that are deployed with WAR packages.

   Restart the application to apply the configurations.

Bind an SSL certificate to an application that is deployed with a JAR package

To bind an SSL certificate to an application that is deployed with a JAR package, modify the *application.properties* file to enable SSL configuration, package the certificate file in the JAR package, use the JAR package to deploy the application, and then change the application port of Tomcat to 8443 in the Application Settings dialog box.

1. Modify the `application.properties` file to enable SSL configuration. Sample configurations:

```
server.ssl.enabled=true
server.ssl.key-store=classpath:{jks}
server.ssl.key-store-password=jks_password
server.ssl.key-store-type=PKCS12
```

2. Store the certificate file in the `resources` path. The path is at the same file level as `application.properties`. Then, generate a JAR deployment package.
3. Deploy the application by using the JAR package. For more information, see Create and deploy an application in an ECS cluster.
4.
5.
6. On the **Basic Information** tab of the application details page, click **Edit** next to **Tomcat Context** in the **Application Settings** section.
7. In the **Application Settings** dialog box, set the **Application Port** parameter to 8443 and click **Configure Tomcat**.

   Restart the application to apply the configurations.

Bind an SSL certificate to an application that is deployed with an image

Both WAR and JAR Docker images can be used to deploy applications. If you want to bind an SSL certificate to an application that is deployed with an image, see the following content to perform relevant operations.

Create an image by using a WAR package

To bind an SSL certificate to an application that is deployed with an image created by using a WAR package, you must modify the configurations of Tomcat and package the certificate file in the Docker image.

1. Download the Ali-Tomcat package and decompress the downloaded package to a directory, such as *d:\work\tomcat*.
2. Modify the Connector parameter in the *server.xml* file. Sample configurations:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true" scheme="https" secure="true" keystoreFile="../app/{app_ID}/{app_name}/{jks_path}" keystoreType="PKCS12" keystorePass="jks_password">
```

3. Store the modified *server.xml* file and certificate file at the same file level as Dockerfile. Add the following settings to Dockerfile:

```
ADD server.xml ${CATALINA_HOME}/conf/ADD {jks} ${CATALINA_HOME}/conf/
```

4. Package the image and deploy the application.

Create an image by using a JAR package

To bind an SSL certificate to an application that is deployed with an image created by using a JAR package, modify the *application.properties* file to enable SSL configuration. Then, package the certificate file in the JAR package that is used to create the image, and change the application port in Dockerfile to enable SSL configuration.

1. Modify the configurations of the JAR package and generate a JAR package. For more information,

see Bind an SSL certificate to an application that is deployed with a JAR package.

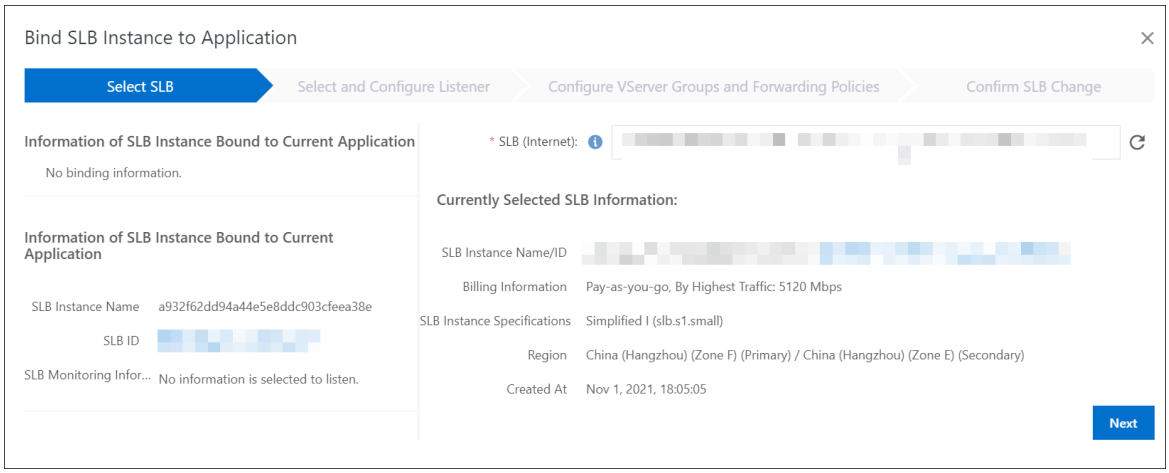
2. Set `server.port=8443` in `start.sh` of Dockerfile.
3. Package the image and deploy the application.

Bind an SLB instance to an EDAS application

Bind a public Server Load Balancer (SLB) instance to an application that is deployed in an Elastic Compute Service (ECS) cluster and set the listening protocol to HTTPS.

> **Notice** You must create an SLB instance in the SLB console in advance. For more information, see Create a CLB instance.

1.
2.
3. On the **Basic Information** tab of the application details page, click **Add** next to **SLB (Internet)** in the **Application Settings** section.
4. In the **Bind SLB Instance to Application** dialog box, bind an SLB instance to the application. For more information, see Configure a dedicated SLB instance for an application.



> **Notice** You must set the listener port to 443.

Verify the SSL connection

In the address bar of your browser, enter the IP address or domain name of the application and add the prefix `https://` to the IP address or domain name. If you can access the homepage, the SSL certificate is bound to the application.

Additional information

You can use an SLB instance to configure an SSL certificate for an application. For more information, see Add an HTTPS listener.

2.Application development

2.1. Create a development environment

You can create an on-premises or off-premises development environment as needed to develop and debug applications.

Solutions

Enterprise Distributed Application Service (EDAS) provides three solutions that you can use to create a development environment. The following table describes the three solutions.

Environment	Solution	Description
On-premises environment	Configure an on-premises lightweight configuration center to implement service registration and discovery, and locally develop and debug services.	The lightweight configuration center cannot match the performance level of a production environment, and may encounter performance problems when a large number of services are registered. Due to on-premises deployment, the configuration center cannot use EDAS features such as service governance, monitoring, or release. It is a completely on-premises environment.
Alibaba Cloud	Create an off-premises development environment in which developers can use the on- and off-premises interconnection plug-in to connect off-premises applications and develop and debug them.	All EDAS capabilities can be used. The cost is high when off-premises resources are used.
Hybrid cloud	Create a development environment in a hybrid cloud in which developers can perform on-premises development and debugging.	All EDAS capabilities can be used. A virtual private network (VPN) or an Express Connect connection is needed to connect the on-premises network to a virtual private cloud (VPC) of Alibaba Cloud. You must activate EDAS Professional Edition or Platinum Edition to use this solution.

Create an on-premises development environment

1. Configure an on-premises lightweight configuration center. For more information, see Start the lightweight configuration center.

2. Locally develop and debug applications.

Create a development environment on Alibaba Cloud

1. Activate EDAS. For more information, see Activate EDAS.
2. Create resources. For more information, see Overview of Kubernetes resource management.

   Microservice namespaces are used to isolate services and configurations. You can create microservice namespaces for development and test environments separately.
3. Deploy applications to the that corresponds to the development environment. For more information, see Overview and Overview.
4. Use the on- and off-premises interconnection plug-in to develop and debug applications.

Create a development environment in a hybrid cloud

> **Notice** Only EDAS Professional Edition or Platinum Edition supports a hybrid cloud.

1. Activate EDAS. For more information, see Activate EDAS.
2. Create resources. For more information, see Overview of Kubernetes resource management.
   - Microservice namespaces are used to isolate services and configurations. You can create microservice namespaces for development and test environments separately.
   - You need to create a hybrid cloud cluster, not an Alibaba Cloud cluster.
3. Deploy applications to the that corresponds to the development environment of the hybrid cloud. For more information, see Create a hybrid cloud ECS cluster.

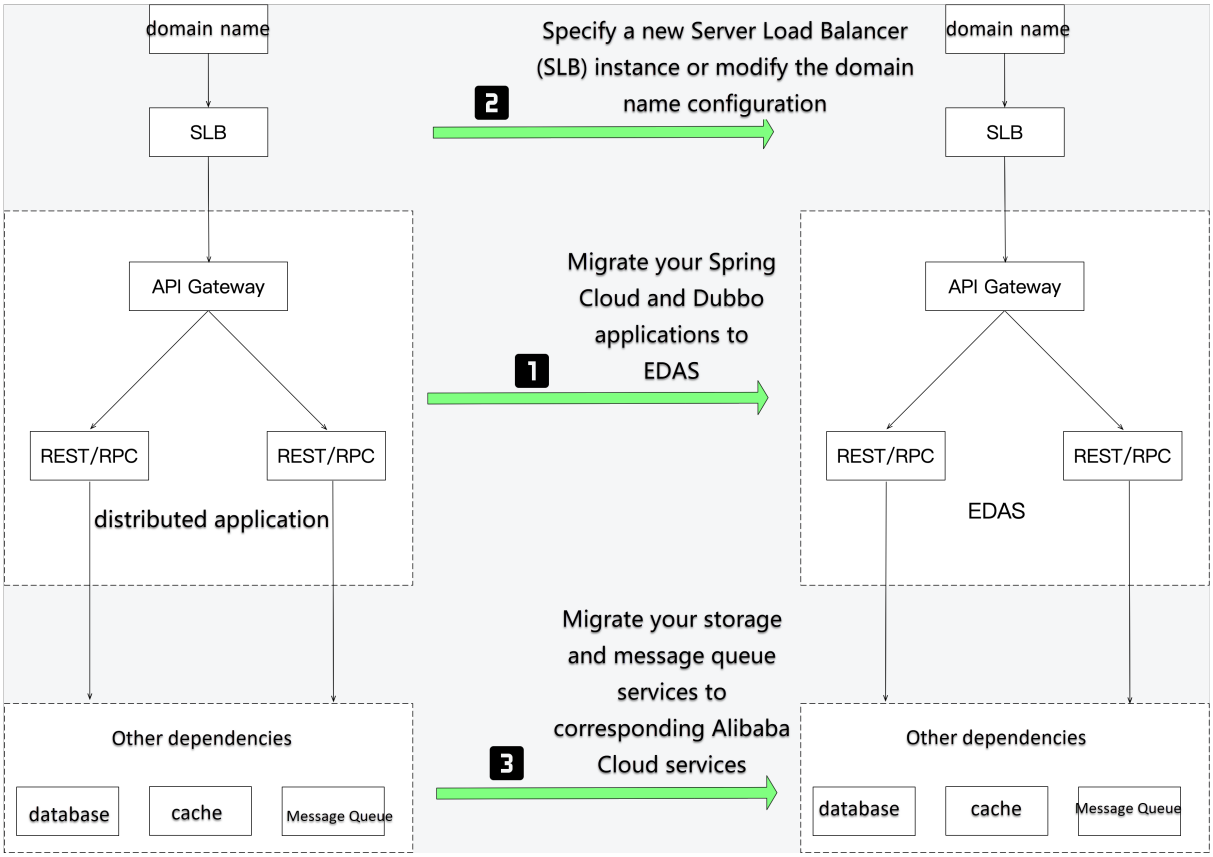
   > **Note** You must enable the required ports for both Alibaba Cloud Elastic Compute Service (ECS) instances and instances outside Alibaba Cloud.
4. Locally develop and debug applications.

3.Application migration

3.1. Seamlessly migrate Spring Cloud and Dubbo applications to EDAS

Enterprise Distributed Application Service (EDAS) allows you to migrate Spring Cloud applications of Edgware or later versions and Dubbo applications of version 2.5.3 or later to EDAS. You can migrate your Spring Cloud and Dubbo applications to EDAS without modifying the application code. After the migration, you can manage the lifecycle of the applications in EDAS. You can use microservice governance features such as integrated monitoring, management, and control, trace queries, and throttling and degradation for the applications. In addition, EDAS provides differentiating capabilities for microservice governance, such as canary release, outlier ejection, graceful shutdown, and service authentication.

Architecture of seamless migration



1. Required. Migrate your Spring Cloud and Dubbo applications to EDAS.
   - If you want to retain your existing self-managed registry after the migration, you need to only migrate the applications to EDAS. You do not need to migrate the configurations of the registry.
   - If you do not want to use the self-managed registry after the applications are migrated, you can use one of the following methods to migrate the configurations of the registry to EDAS. For more information about how to select a registry, see the Registry selection section of this topic.
     - Dual registration and dual subscription

- Registry switching

  Both methods allow you to migrate your applications without service discontinuity. This best practice shows you how to migrate applications to EDAS without changing your registry.

2. (Optional) Specify a new Server Load Balancer (SLB) instance or modify the domain name configuration.

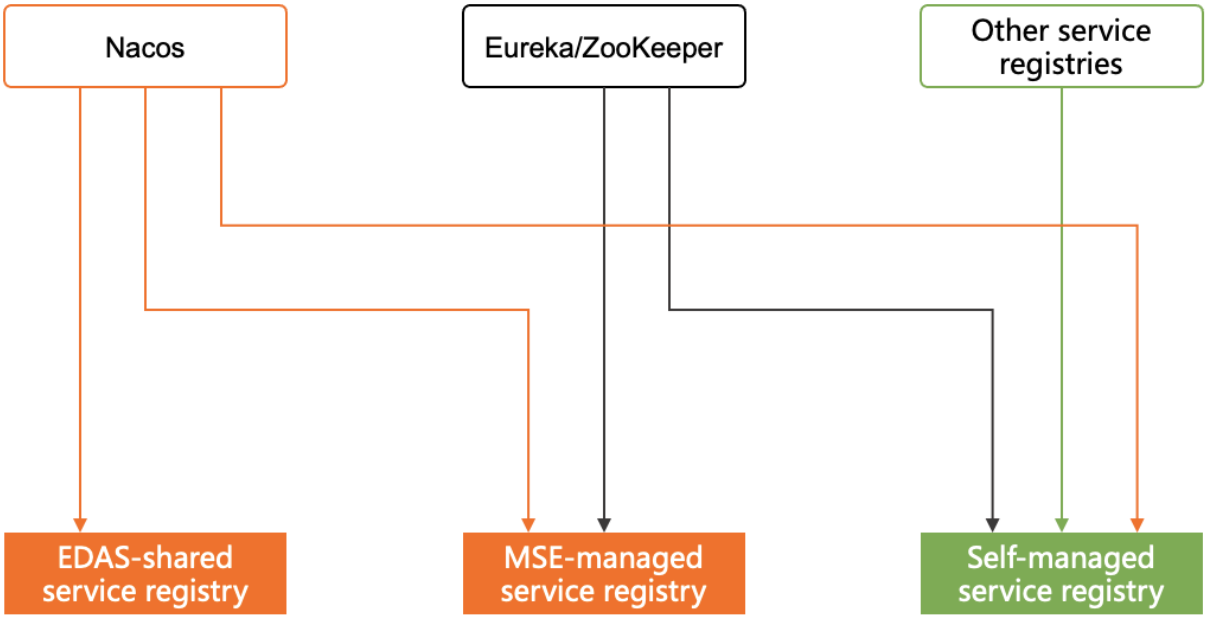
   After the applications are migrated, you need to specify a new SLB instance or modify the domain name configuration.

3. Optional. Migrate your storage and message queue services to corresponding Alibaba Cloud services.

   If the network connectivity to your storage and message queue services is normal after you migrate the applications, you do not need to migrate the storage and message queue services. If you want to use the storage and message queue services of Alibaba Cloud, such as ApsaraDB RDS and Message Queue for Apache RocketMQ, refer to the documentation about the corresponding Alibaba Cloud services after you migrate the applications.

Registry selection

Microservice applications use registries to implement service registration and discovery. When you develop an application, you can select a registry based on your actual needs.



You can use Nacos that is described in this topic as a registry to implement service registration and discovery for your application. You can also use other types of registries, such as Eureka, ZooKeeper, and Consul that are user-created or managed in Microservice Engine (MSE). After you deploy your application to EDAS, you can use the application management, microservice governance, and cloud native Platform as a Service (PaaS) features of EDAS regardless of the registry type.

- For more information about how to deploy an application to EDAS, see Overview and Overview.

Benefits

EDAS is compatible with mainstream open source remote procedure call (RPC) frameworks. EDAS provides the following capabilities for microservice-oriented applications that are built based on an RPC framework after the applications are migrated to EDAS:

- On top of cloud-native Kubernetes or Elastic Compute Service (ECS), EDAS provides an enhanced application hosting capability to implement open source microservice governance and lightweight O&M of applications in Kubernetes and ECS clusters from the application perspective.
  - From the application-centric perspective, EDAS manages Kubernetes-native workloads such as deployments and pods, and provides high-availability deployment of instances across zones.
  - EDAS provides phased release and canary release based on the traffic ratio and request parameters. EDAS monitors the entire change process and therefore makes your change records traceable.
  - EDAS integrates with mainstream DevOps systems to help enterprises implement continuous integration (CI) and continuous delivery (CD). This way, EDAS helps enterprises reduce costs and improve efficiency.
- On top of open source microservice systems, you can migrate the microservice-oriented applications that you build based on the Spring Cloud and Dubbo frameworks commercially available in the past five years to EDAS without the need to modify code. EDAS supports the following microservice governance capabilities for all the application frameworks:
  - Graceful shutdown and stress testing during application release
  - Service authentication, throttling and degradation, and outlier ejection during the application runtime
  - Service query and testing in application O&M
- Alibaba Cloud packages its idea of supporting observability, canary release, and rollback for application production security into services, and therefore allows you to immediately implement production security.
  - End-to-end monitoring: You can monitor the applications in multiple dimensions by using the application overview, release change records, and automatically generated release reports.
  - Canary release: Canary release is supported for applications based on the traffic ratio or request content policy configuration.
  - Rollback: One-click rollback is supported during the release process, and applications that are running can be rolled back to an earlier version.

Migration process

- Build a local development environment for microservices applications
  - Build a local development environment for OnlineShop
  - Create cloud resources
  - Configure the registry address in the configuration files of the microservices applications
  - Deploy the microservices applications on ECS instances
  - Associate an Internet-facing SLB instance with the Frontend application
  - Send local access traffic to the Frontend application
- Smoothly migrate microservice-oriented applications to EDAS
  - Create a microservice namespace
  - Create an ECS cluster
  - Deploy microservice-oriented applications in EDAS
  - Migrate access traffic to EDAS

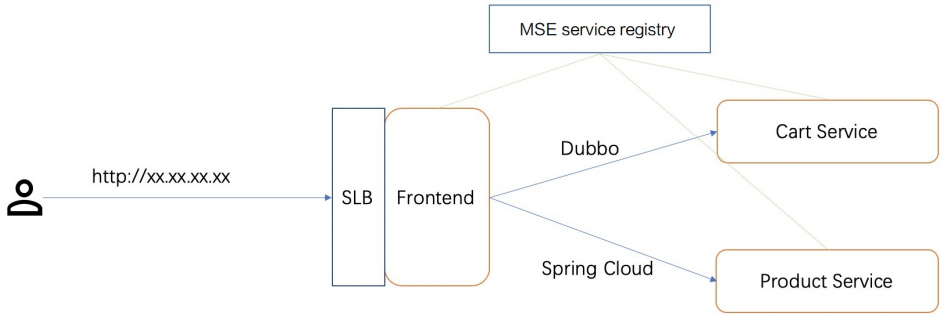
3.2. Build a local development environment for microservices applications

This topic uses the OnlineShop demo project as an example to describe how to build a local development environment for microservices applications.

Overview of OnlineShop

OnlineShop is a microservices demo project on GitHub. The project comprises Spring Cloud and Dubbo applications. It provides complete source code and well-built container images. Therefore, it can be used as a typical example in this topic.

OnlineShop provides three microservices: Frontend, Cart Service, and Product Service. Cart Service is a shopping cart microservice implemented by a Dubbo application. Product Service is a commodity microservice implemented by a Spring Cloud application. Frontend is a client microservice used to call the Dubbo and Spring Cloud applications. The following figure shows the architecture of OnlineShop.



As shown in the figure, the project provides only the service registration and discovery capabilities. It does not support production-facilitating microservices governance capabilities such as service queries, link monitoring, configuration management, and service authentication.

In this best practice, you can learn how to seamlessly migrate OnlineShop to Enterprise Distributed Application Service (EDAS) without service discontinuity and use the full-fledged microservices governance and application monitoring capabilities provided by EDAS.

Prerequisites

A Java environment is installed.

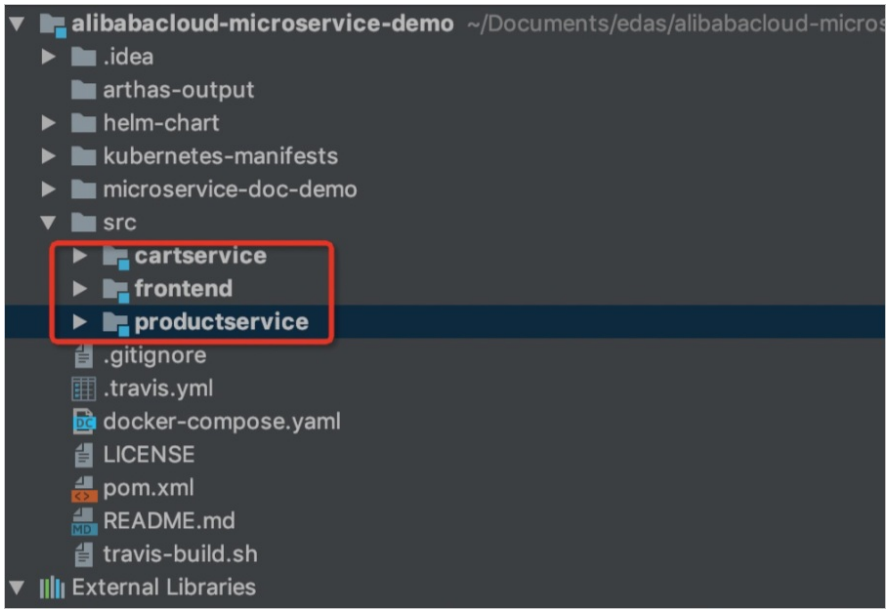
Build a local development environment for OnlineShop

1. Run the following command to download the source code of OnlineShop:

   ```
   git clone https://github.com/aliyun/alibabacloud-microservice-demo.git
   ```

2. Create a project and import the content of OnlineShop.

   The OnlineShop demo files are stored in the *src* directory.



3. Run the following command in the root directory of the project:

```
mvn package -DskipTests
```

If the compilation is successful, a command output similar to the following one is returned.

```
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for alibabacloud-microservice-demo 1.0.0-SNAPSHOT:
[INFO]
[INFO] cartservice ....................................... SUCCESS [  0.281 s]
[INFO] cartservice-interface ............................. SUCCESS [  1.492 s]
[INFO] cartservice-provider .............................. SUCCESS [  1.289 s]
[INFO] productservice .................................... SUCCESS [  0.012 s]
[INFO] productservice-api ................................ SUCCESS [  0.072 s]
[INFO] productservice-provider ........................... SUCCESS [  0.747 s]
[INFO] frontend .......................................... SUCCESS [ 45.674 s]
[INFO] alibabacloud-microservice-demo .................... SUCCESS [  0.012 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
```

Create cloud resources

1. Create a virtual private cloud (VPC). For more information, see Create an IPv4 VPC.
2. Purchase three Elastic Compute Service (ECS) instances that can be used in the created VPC. For more information, see Create and manage an ECS instance by using the ECS console (express version)Quick start.
3. Install Java Development Kit (JDK) and configure a Java environment on the ECS instances.
   i. Download JDK 1.8 or later and Maven 3.5 or later.
   ii. Log on to each of the ECS instances. Install JDK 1.8 or later and configure the **JAVA_HOME** environment variable.
   iii. Log on to each of the ECS instances. Install Maven 3.5 or later and configure the **MAVEN_HOME** environment variable.
4. Create a Microservices Engine (MSE) instance in the VPC and record the internal endpoint of the MSE instance. For more information, see Create a Nacos engine.

Configure the registry address in the configuration files of the microservices applications

In this example, the internal endpoint of the created MSE instance that serves as a registry is mse-****-nacos-ans.mse.aliyuncs.com:8848.

1. Open the *application.properties* file of each of the three microservices applications. Set the registry address to the internal endpoint of the MSE instance.
   i. Modify the registry address of the Cart Service application.

```
# Spring boot application
spring.application.name=cartservice
# Base packages to scan Dubbo Component: @org.apache.dubbo.config.annotation.Service
dubbo.scan.base-packages=com.alibabacloud.hipstershop.provider
# Dubbo Application
## The default value of dubbo.application.name is ${spring.application.name}
## dubbo.application.name=${spring.application.name}
# Dubbo Protocol
dubbo.protocol.name=dubbo
dubbo.protocol.port=12345
## Dubbo Registry
dubbo.registry.address=nacos://mse-[redacted]-nacos-ans.mse.aliyuncs.com:8848
dubbo.registry.check=false
dubbo.application.qos-enable=true
dubbo.application.qos-accept-foreign-ip=false
```

   ii. Modify the registry address of the Product Service application.

```
# Spring boot application
spring.application.name=productservice
# nacos discovery address
spring.cloud.nacos.discovery.server-addr=mse-[redacted]-nacos-ans.mse.aliyuncs.com:8848
# config server port
server.port=8082
```

   iii. Modify the registry address of the Frontend application.

```
spring.application.name=frontend
spring.cloud.nacos.discovery.server-addr=mse-[redacted]-nacos-ans.mse.aliyuncs.com:8848
dubbo.registry.address=nacos://mse-[redacted]-nacos-ans.mse.aliyuncs.com:8848
dubbo.consumer.check=false
feign.httpclient.enabled=true
feign.hystrix.enabled=false
#management.server.port=8081
#spring.cloud.sentinel.eager=true
#spring.cloud.sentinel.transport.dashboard=127.0.0.1:8081
```

2. Run the following command to compile the project:

```
mvn clean install
```

Deploy the microservices applications on ECS instances

1. Run the following command to upload the *cartservice-provider-1.0.0-SNAPSHOT.jar* package to the */tmp* directory of the ECS instance on which the Cart Service application is to be deployed:

```
scp src/cartservice/cartservice-provider/target/cartservice-provider-1.0.0-SNAPSHOT.jar root@XX.XX.XX.XX:/tmp
```

```
➜ alibabacloud-microservice-demo git:(master) ✗ scp src/cartservice/cartservice-provider/target/cartservice-provider-1.0.0-SNAPSHOT.jar root@[redacted]:/tmp
root@[redacted]'s password:
cartservice-provider-1.0.0-SNAPSHOT.jar                                                    100%
```

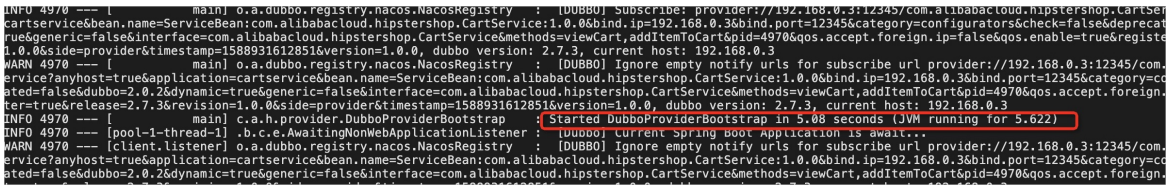
2. Use SSH to log on to the ECS instance on which the Cart Service application is deployed.

> **Notice** You must add a security group rule that allows inbound SSH access to port 22 on the ECS instance. For more information, see Create a security group.

3. Move the /tmp/cartservice-provider-1.0.0-SNAPSHOT.jar package to the */root* directory.
4. Run the following command to start the Cart Service application:

```
nohup java -jar cartservice-provider-1.0.0-SNAPSHOT.jar &
```

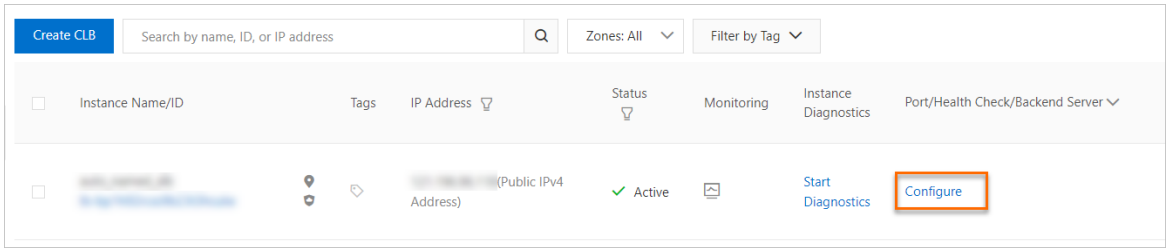
5. Check the *nohup.out* log file. If the message framed in red in the following figure is displayed, the Cart Service application is started.



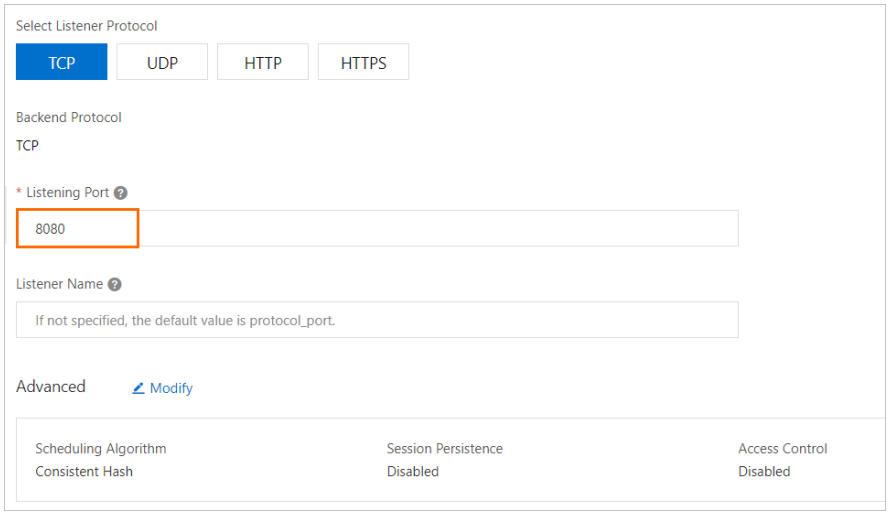
6. Repeat the preceding steps to deploy the Product Service and Frontend applications on the other two ECS instances.
7. In the address bar of your browser, enter the endpoint of the Frontend application in the format of `http://{IP address of the Frontend application}:8080/` and press the ENTER key. If the home page of the Frontend application appears, the access is successful.

Associate an Internet-facing SLB instance with the Frontend application

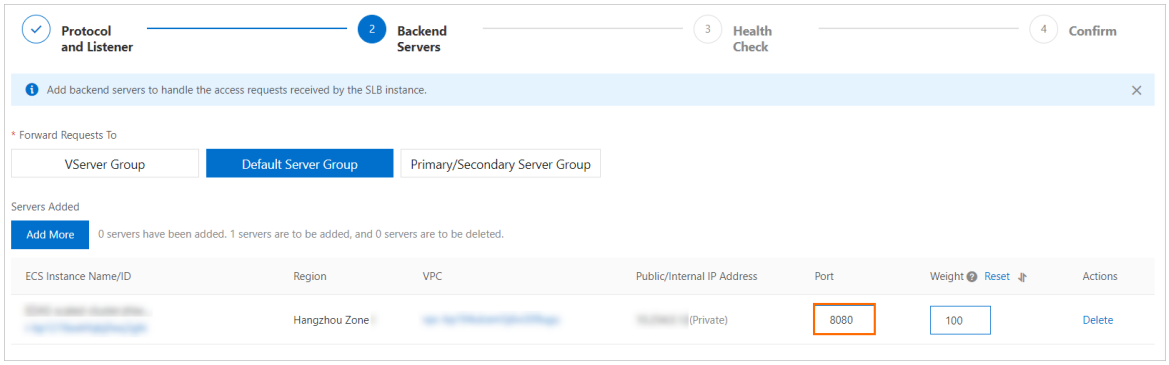
1. Log on to the Server Load Balancer (SLB) console.
2. Purchase an SLB instance in the zone in which the ECS instance for deploying the Frontend application resides. For more information, see Create a CLB instance.
3. Find the created SLB instance on the Instances page and click **Configure**.



4. Set the Select Listener Protocol parameter to **TCP** and the Listening Port parameter to *8080*.



5. Configure the ECS instance on which the Frontend application is deployed as the backend server. Set the **port number** to *8080* and configure listeners for the SLB instance as prompted.



6. In the address bar of your browser, enter the public endpoint of the SLB instance in the format of `http://{IP address of the SLB instance}:8080/` and press the ENTER key. If the home page of the Frontend application appears, the SLB instance is associated with the Frontend application.

Send local access traffic to the Frontend application

Run the following command to keep sending local access traffic to the Frontend application:

```
while :
do
        result=`curl $1 -s`
        if [[ "$result" == *"500"* ]]; then
                echo `date +%F-%T` $result
        else
                echo `date +%F-%T` "success"
        fi
        sleep 0.1
done
```

A command output similar to the following one is returned.

```
➜  ~ sh curlservice.sh http://[redacted]:8080/
2020-05-11-10:24:42 success
2020-05-11-10:24:42 success
2020-05-11-10:24:42 success
2020-05-11-10:24:42 success
2020-05-11-10:24:42 success
2020-05-11-10:24:43 success
2020-05-11-10:24:43 success
2020-05-11-10:24:43 success
2020-05-11-10:24:43 success
2020-05-11-10:24:43 success
2020-05-11-10:24:43 success
2020-05-11-10:24:44 success
2020-05-11-10:24:44 success
```

What's next

Smoothly migrate microservice-oriented applications to EDAS

3.3. Smoothly migrate microservice-oriented applications to EDAS

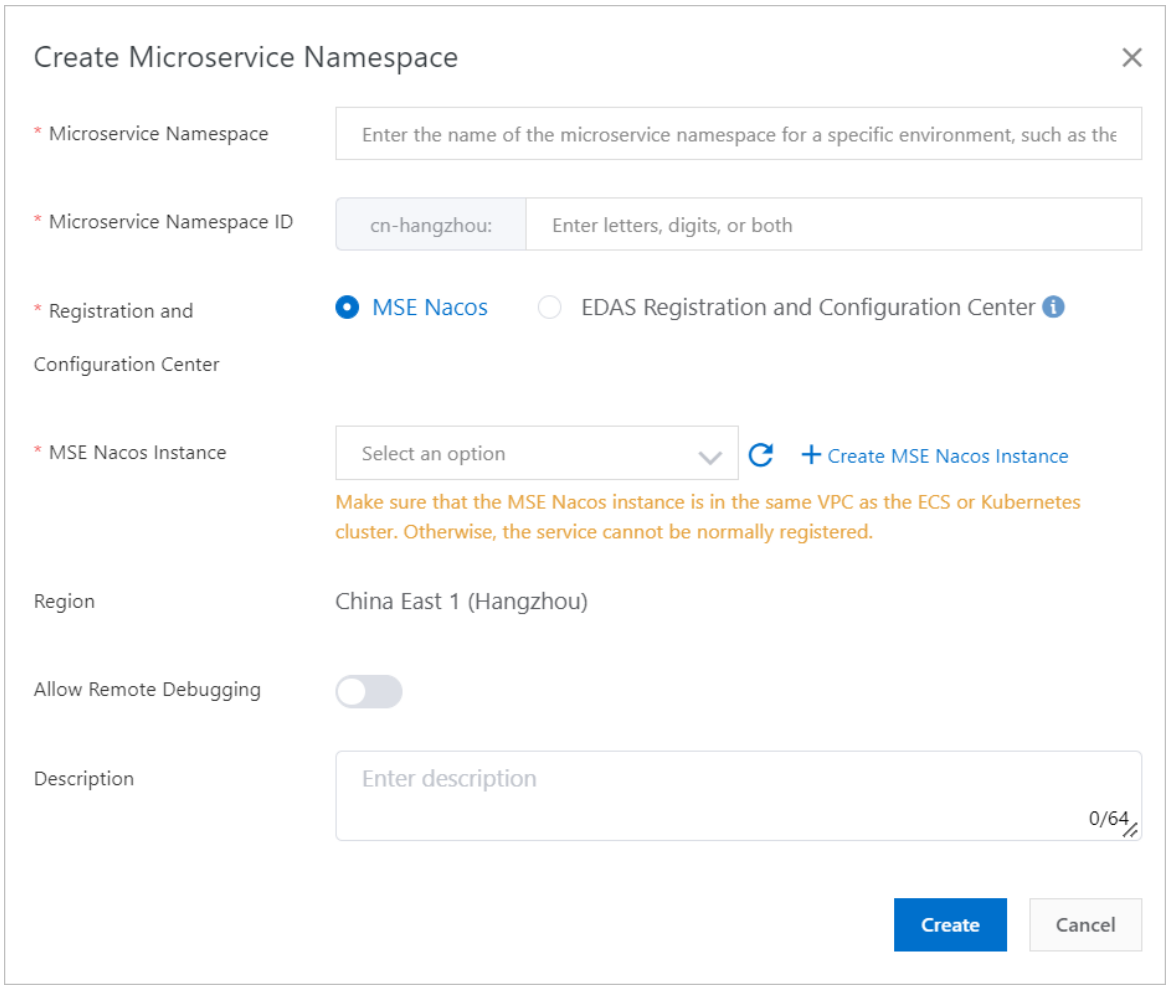
This topic describes how to smoothly migrate microservice-oriented applications that are running on your Elastic Compute Service (ECS) instances to Enterprise Distributed Application Service (EDAS).

Prerequisites

The microservice-oriented applications of the OnlineShop demo project are deployed on your ECS instances. For more information, see Build a local development environment for microservices applications. You can refer to this topic to migrate other microservice-oriented applications that are running on your ECS instances to EDAS.

Create a microservices namespace

1.
2. In the left-side navigation pane, choose **Resource Management > Microservice Namespaces**.
3. In the upper-right corner of the **Microservice Namespaces** page, click **Create Microservice Namespace**.
4. In the **Create Microservice Namespace** dialog box, set the parameters and click **Create**. The following table describes the parameters.



Parameter	Description
Microservice Namespace	Enter a name for the microservices namespace that you want to create.

Parameter	Description
Microservice Namespace ID	Enter a string to specify the ID of the microservices namespace. The ID can contain only letters and digits.
Registration and Configuration Center	○ MSE Nacos: the Microservice Engine (MSE) Nacos engine that you purchased. The Nacos engine can be seamlessly integrated with EDAS for service registration and configuration management. ○ EDAS Registration and Configuration Center: a free service registration and configuration center provided by EDAS. If your application requires high performance and stability, we recommend that you use MSE Nacos as the service registration and configuration center.
MSE Nacos Instance	The Nacos instance created by MSE. For more information, see Create a Nacos engine.
Region	The region to which the microservices namespace belongs. You cannot change the parameter value.
Allow Remote Debugging	To enable communication between the cloud and on-premises applications, you can turn on **Allow Remote Debugging** in the Edit Microservice Namespace dialog box for the microservices namespace in which your application is deployed.
Description	The description of the microservices namespace.

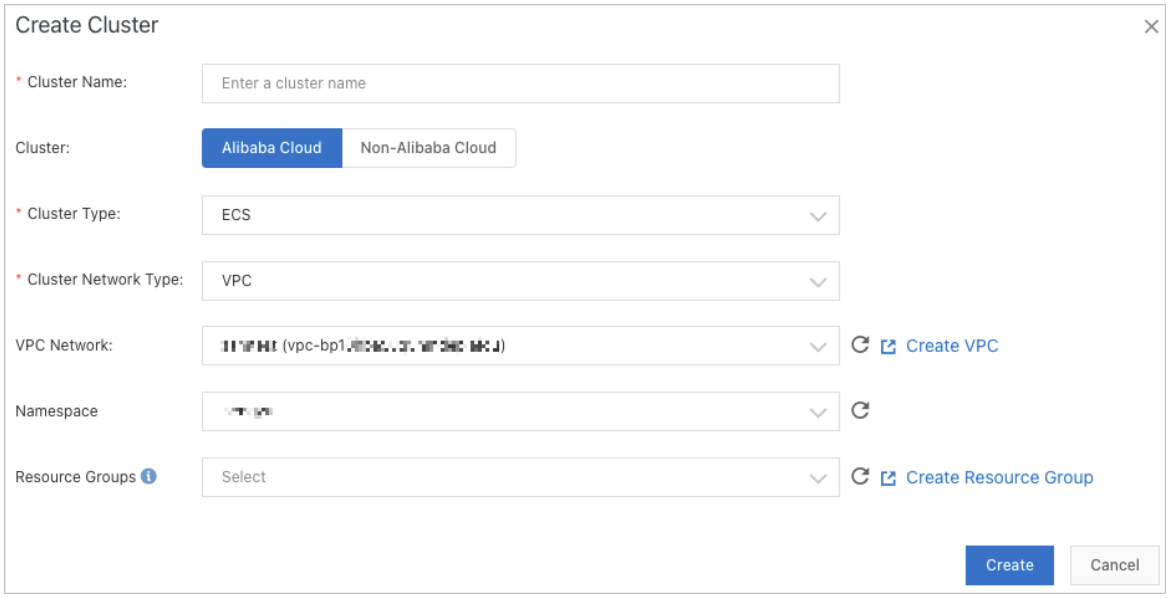
Procedure

1.
2. In the left-side navigation pane, choose **Resource Management > ECS Clusters**.
3. On the **ECS Cluster** page, select a region and a microservice namespace. Then, click **Create Cluster**.

   You can select a microservice namespace on this page or in the **Create Cluster** dialog box.

   ○ If resource or service isolation is required, select the microservice namespace that you create.

   ○ If resource or service isolation is not required, select **Default** from the Microservice Namespace drop-down list.
4. In the **Create Cluster** dialog box, set the parameters for the cluster, and click **Create**.



Parameter	Description
Cluster Name	The name of the cluster. The name can be up to 64 characters in length and can contain letters, digits, underscores (_), and periods (.).
Cluster	○ **Alibaba Cloud**: specifies an Alibaba Cloud ECS cluster. ○ **Non-Alibaba Cloud**: specifies a hybrid cloud ECS cluster. In a hybrid cloud ECS cluster, the instances provided by Alibaba Cloud ECS, self-managed data centers, and other cloud service providers are interconnected over Express Connect circuits. For more information, see Create a hybrid cloud ECS cluster.
Cluster Type	This parameter can be set only to **ECS**.
Cluster Network Type	Valid values: **Classic Network** and **VPC**. **Notice** If you set the Cluster parameter to **Non-Alibaba Cloud**, you can set this parameter only to **VPC**.
VPC Network	After you select **VPC**, you must select a created VPC.
Microservice Namespaces	The microservice namespace to which the cluster belongs. By default, the namespace selected on the **ECS Clusters** page is displayed. If no microservice namespace is selected, *Default* is displayed in the Microservice Namespaces drop-down list. Set this parameter as needed.

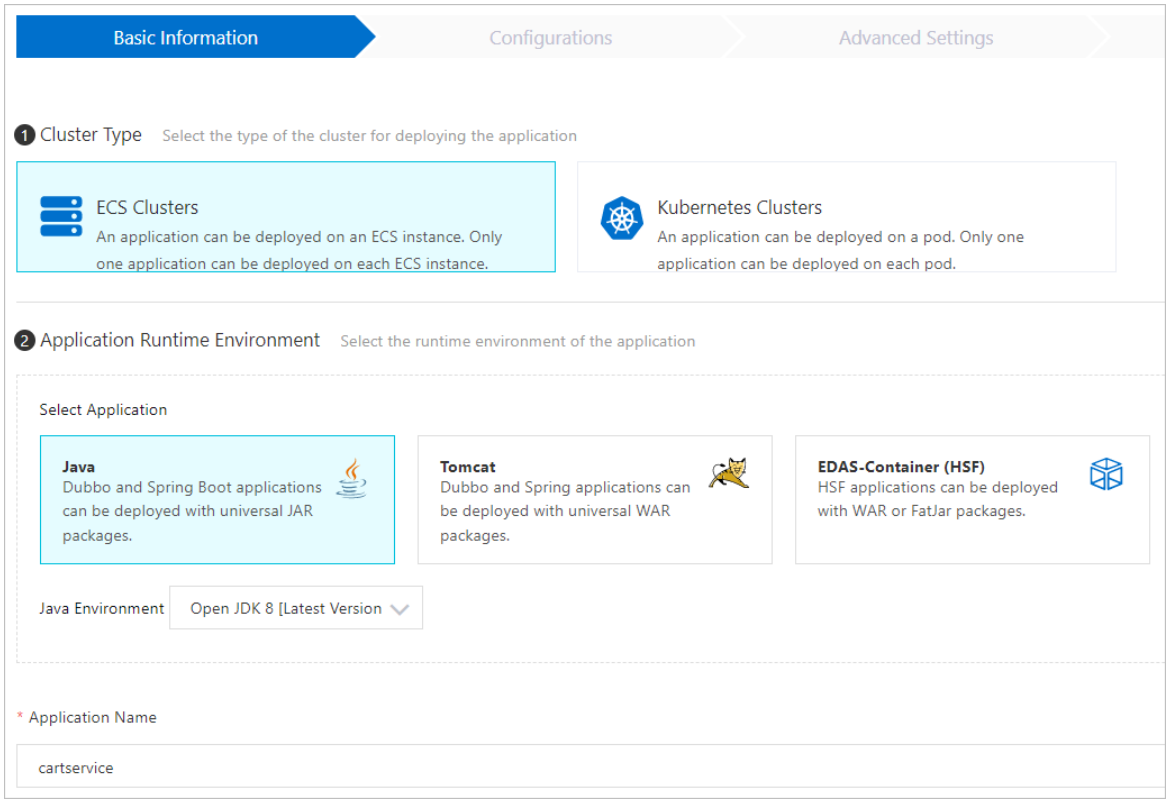
Parameter	Description
Resource Groups	The resource group to which the cluster belongs. The resource group is created by the current Alibaba Cloud account in the Resource Management console. This group is not an EDAS resource group. If no resource groups are available, click **Create Resource Group** to go to the Resource Management console and create a resource group. For more information, see Create a resource group.

After a cluster is created, the **Created** message appears on the top of the page, and the cluster appears in the cluster list.

Deploy microservice-oriented applications in EDAS

Deploy the Cart Service, Product Service, and Frontend applications in EDAS by performing the following steps. The following procedure describes how to create a service provider. You can follow a similar procedure to create a service consumer.

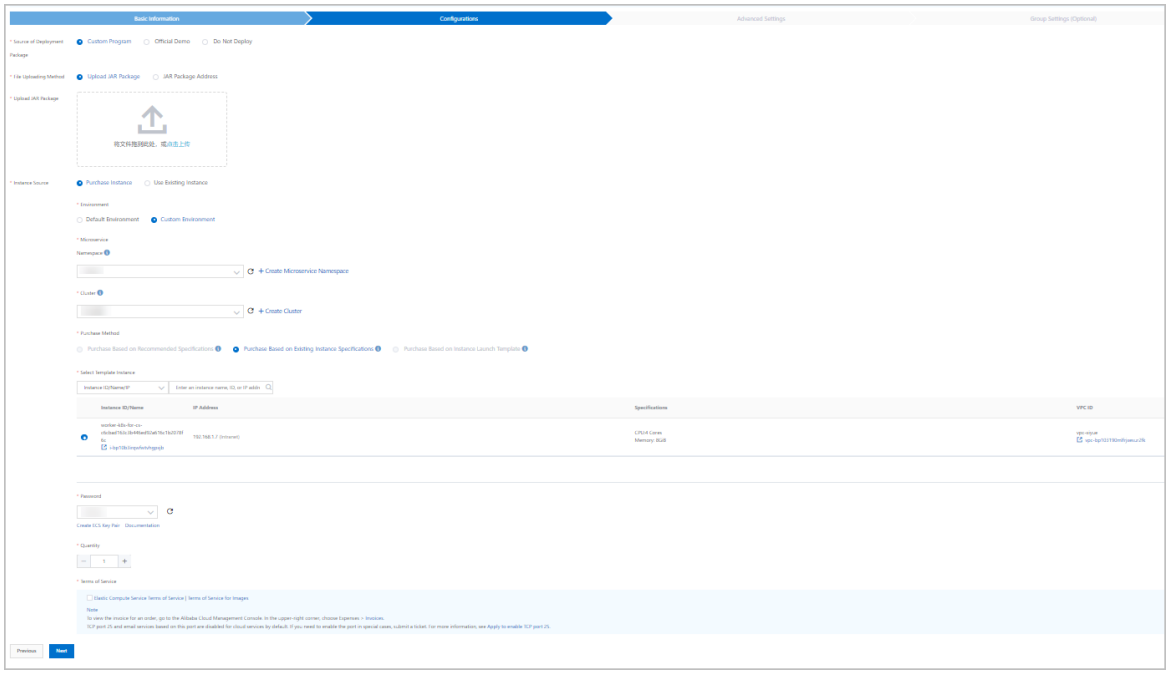
1.
2.
3. In the **Basic Information** step of the **Create Application** wizard, configure the basic information of the Cart Service application, and then click **Next**.



Parameter	Description
Cluster Type	In this example, **ECS Clusters** is selected.

Parameter	Description
Application Runtime Environment	In this example, **Java** is selected for the Select Application parameter, and **Open JDK 8** is selected from the Java Environment drop-down list.
Application Name	Enter a custom name for the application.
Application Description	Optional. Enter a description for the application.

4. In the **Configurations** step of the **Create Application** wizard, specify a deployment package and an ECS instance for the application, and then click **Next**.



Parameter	Description
Source of Deployment Package	In this example, **Custom Program** is selected.
File Uploading Method	In this example, **Upload JAR Package** is selected.
Upload JAR Package	Click **Select File** and upload your JAR deployment package.
Instance Source	In this example, **Purchase Instance** is selected.
Environment	In this example, **Custom Environment** is selected.
Microservice Namespace	Select a namespace that you create. Namespaces can be used to isolate resources and services. If no namespace is created, the **Default** namespace is selected.

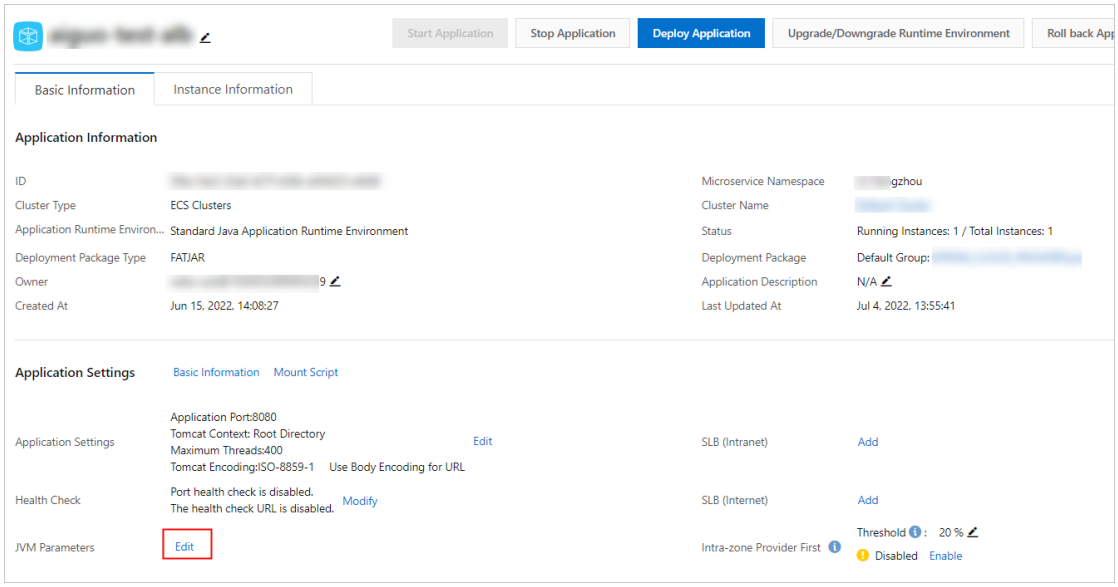
Parameter	Description
Cluster	Select a cluster that you create. A cluster is a collection of cloud resources that are required to run applications.
Purchase Method	In this example, **Purchase Based on Recommended Specifications** is selected. **Note** ◦ If you have added ECS instances to your cluster, you can select **Purchase Based on Existing Instance Specifications**. ◦ If you have created a launch template in the ECS console, you can select **Purchase Based on Instance Launch Template**.
Select Specifications	In this example, **Low-spec Instance** is selected.
Quantity	Set the value to 1 in the spinner box.
Logon Password	Enter a password that is used to log on to the purchased ECS instance. Record and keep the password properly. If you forget the password, you can reset the password in the ECS console. The new password takes effect after the instance is restarted.
Terms of Service	Select **Elastic Compute Service Terms of Service** **Terms of Service for Images**.

5. In the **Advanced Settings** step of the Create Application wizard, configure the parameters that are described in the following table, and then click **Create Application**.

Parameter	Description
Version	By default, EDAS uses the current timestamp as the version number of the application, in the format of `yyyymmdd:hhmmss`. You can also customize the version number.
Application Health Check	Optional. Specify a URL to perform health checks on the application.

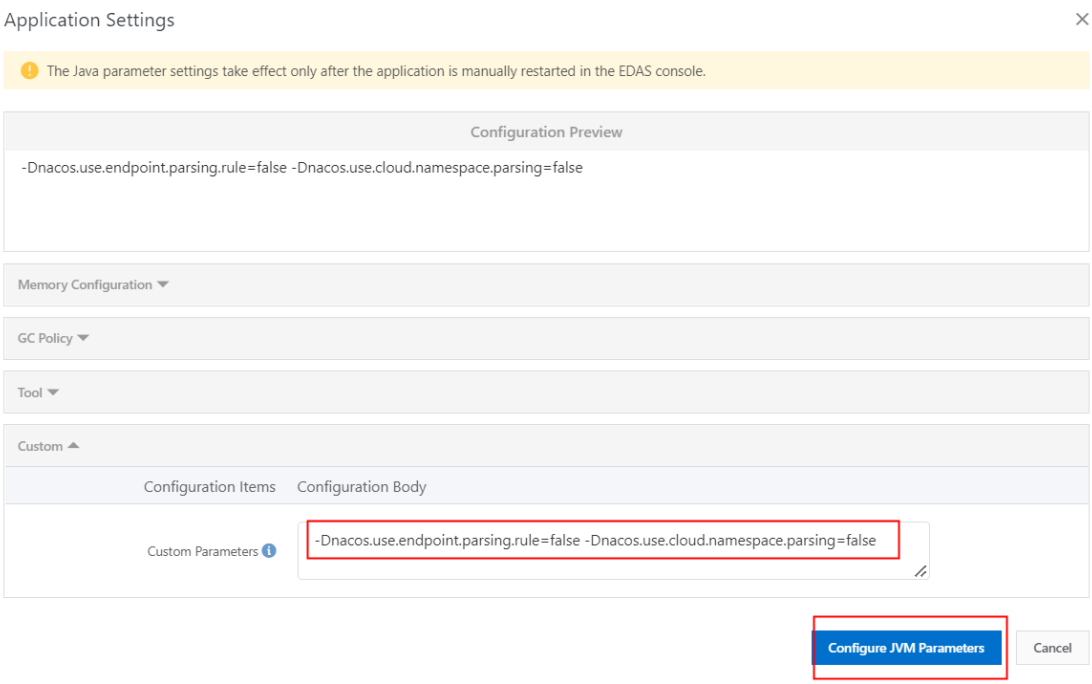
6. In the **Creation Completed** step of the Create Application wizard, check the basic information, configurations, and advanced settings of the application, and then click **Create Application**.
7. Configure Java virtual machine (JVM) parameters for the application.

i. On the **Basic Information** tab of the application, click **Edit** to the right of **JVM Parameters** in the Application Settings section.

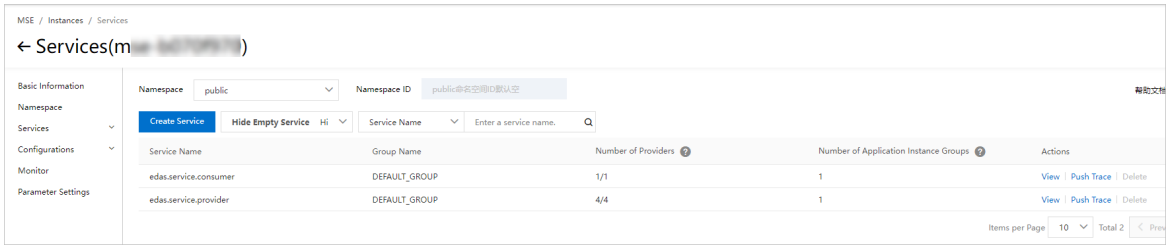


ii. In the **Application Settings** dialog box, show the **Custom** section, enter `-Dnacos.use.endpoint.parsing.rule=false -Dnacos.use.cloud.namespace.parsing=false` in the Custom Parameters field, and then click **Configure JVM Parameters**.

> **Notice**
>
> - If you use a self-managed Nacos registry, you must add the preceding JVM parameters.
> - If you use a registry that is provided by EDAS or a self-managed non-Nacos registry, you do not need to add the preceding JVM parameters.



8. Click the **Instance Information** tab. In the **Actions** column of the instance, click **Restart**. Then, restart the application as prompted.
   Log on to the MSE console. On the **Instances** page, click the ID of the instance to go to the **Basic Information** page. In the left-side navigation pane, click **Services**. On the Services page, view the service information. You can see that the number of providers for the Cart Service application changes to 2 in the Number of Providers column.

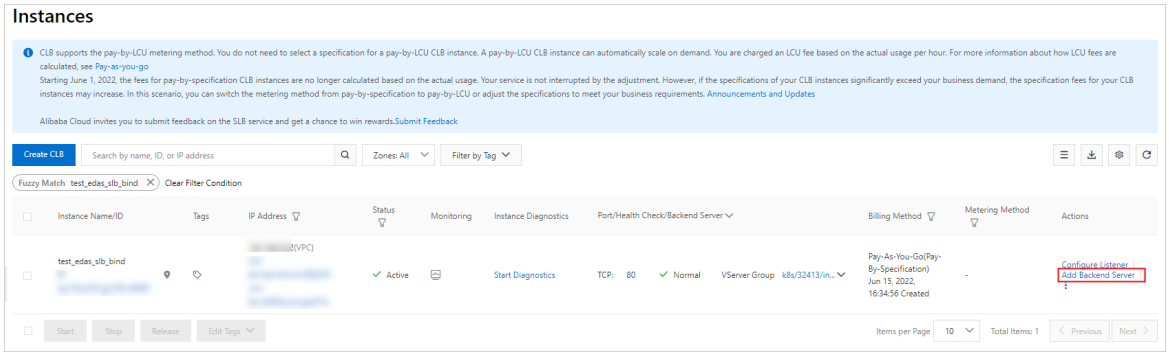


9. Repeat Steps 1 to 8 to deploy the Product Service and Frontend applications.
   After the three applications are deployed, you can enter `http://{public IP address of the Frontend application}:8080` in the address bar of the browser to access the Frontend application.

Migrate access traffic to EDAS

Till now, the OnlineShop project is deployed in both EDAS and ECS. All the access traffic is routed to the ECS instance on which you deploy the Frontend application in ECS. No access traffic is routed to the Frontend application in EDAS. To migrate the access traffic to EDAS, you need to change the default backend server in the Server Load Balancer (SLB) console.

1. Log on to the SLB console.
2. Find the SLB instance that is associated with the Frontend application and click **Add Backend Server** in the **Actions** column. In the **Available Servers** panel, select the ECS instance on which you deploy the Frontend application in EDAS.



3. Modify the weights of the two backend servers.

4. Check whether the access traffic is routed as expected.

5. Remove the backend server for the Frontend application in ECS so that all the access traffic is routed to the Frontend application in EDAS.

6. Check whether the access traffic is routed as expected.
7. Stop the OnlineShop project that you deploy in ECS and delete the related ECS instances if all the

access traffic is routed to EDAS.

3.4. Microservice governance capabilities

After you migrate self-managed open source Spring Cloud or Dubbo applications to Enterprise Distributed Application Service (EDAS), you can manage the lifecycle of the applications. You can enable microservice governance features such as integrated monitoring, management, and control, trace queries, throttling and degradation, canary release, outlier ejection, graceful shutdown, and service authentication for the applications.

Scenario-based application lifecycle management

EDAS allows you to manage the lifecycle of applications deployed in Elastic Compute Service (ECS) clusters or Container Service for Kubernetes (ACK) clusters. For example, you can deploy, roll back, stop, restart, reset, and manually scale in or scale out applications. You can also configure auto scaling policies for applications based on your business requirements in different scenarios.

- For more information about how to manage the lifecycle of applications in ACK clusters, see Manage the application lifecycle.
- For more information about how to manage the lifecycle of applications in ECS clusters, see Manage lifecycle for applications deployed in ECS clusters.

Graceful release and production security

The graceful release feature allows you to release applications in the daytime regardless of traffic volumes. The feature ensures that no traffic is interrupted. EDAS supports canary release, observation, and rollback of applications. This helps you release applications in the daytime when traffic is heavy.

EDAS supports graceful shutdown. Applications are shut down in the prestop phase before the applications are stopped. You can directly notify application users of the shutdown. You do not need to send notifications by using the registration center. This reduces the amount of time consumed by a shutdown from minutes to almost zero minute and ensures that the shutdown does not affect your business. By default, the graceful shutdown feature is enabled.

How graceful shutdown works

- **Canary release**

  Canary release is supported for applications based on the traffic ratio or request content policy configuration. For more information, see Use the EDAS console to implement canary releases of applications in Kubernetes clusters.

- **Observation**

  End-to-end monitoring in multiple dimensions is implemented based on application overview, release change records, and automatic generation of release reports. For more information, see Service and API monitoring.

- **Rollback**

One-click rollback is supported during the release process, and applications that have been run can be rolled back to an earlier version. For more information, see Manage the application lifecycle.

Instance-related metrics

Integrated monitoring, management, and control

EDAS supports application monitoring and allows you to view the key health metrics of an application. This helps you identify issues with high efficiency. The health metrics include overall metrics such as total requests and average response time, metrics related to the services that are provided by the application, metrics related to dependent services of the application, and system metrics such as CPU utilization and memory usage.

You can view the instance-related metrics such as Java Virtual Machine (JVM) metrics, host metrics, and memory snapshots. For more information, see Instance details.

You can create alerts to define alert rules for specific monitored objects. When an alert rule is triggered, the system sends an alert notification to the specified alert group based on the specified notification method. This reminds the alert contacts to take necessary actions to solve the problem. For more information, see Create an alert rule for application monitoring and manage alert notifications.

EDAS allows you to view event information, alert information, diagnostic reports, and microservice governance information of Kubernetes-native applications. This helps you understand the application status and troubleshoot errors with high efficiency. For more information, see Event center.

Alerts

Event center

Throttling and degradation

EDAS supports throttling and degradation for Spring Cloud applications, Dubbo applications, and High-speed Service Framework (HSF) applications by using Application High Availability Service. EDAS allows you to view the throttling and degradation details and dynamic change rules in real time. This ensures the availability of your applications. For more information, see Overview of throttling and degradation.

Multiple enabling methods

- You can enable the throttling and degradation feature by using multiple methods. If you enable this feature by mounting the Java agent, you do not need to modify the application code. For more information, see .
- If you enable this feature by using other methods, you must add the corresponding Sentinel dependency to the pom.xml file.

> **Notice** We recommend that you do not enable the throttling and degradation feature provided by EDAS and Hystrix degradation at the same time. Otherwise, the degradation results may not meet your expectations. If you have enabled Hystrix degradation, disable Hystrix degradation before you enable the throttling and degradation feature provided by EDAS.

Fault tolerance

Fault tolerance may be required in the following scenarios:

- The code logic of an application is invalid. After you perform a canary release for the application, the thread pool is full, and the client fails to send a request.
- When an application is running, the disks of some instances may be full or the host may compete for resources. As a result, the loads of the instances are heavy and the request of the client times out.

EDAS allows you to remove outliers in multiple ways to ensure the stability of your business.

- Outlier ejection on the client:
  - Removes outliers in real time.
  - Determines whether to remove outliers based on the specified lower limit of the error rate and upper limit for the proportion of outliers that can be removed, and detects whether the outliers are recovered based on the specified recovery detection unit time.
  - Allows you to configure flexible ejection policies by adjusting the lower limit of the error rate based on specific scenarios.
- Global outlier ejection: integrates monitoring, management, and control in multiple dimensions.
- Solution after ejection: performs auto scaling to replace outliers.

For more information, see the following topics:

- Ensure the availability of Spring Cloud applications by using outlier instance removal
- Ensure the availability of Dubbo applications by using outlier ejection

Security assurance

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. The service authentication feature provided by EDAS ensures the security of your applications and interfaces.

For more information, see the following topics:

- Implement access control on Spring Cloud applications by using service authentication
- Implement access control of Dubbo applications through service authentication