

Enterprise Distributed Application Service (EDAS)

User Guide

User Guide

Resources

ECS

Create ECS instances

If you do not have any ECS instances, follow the instructions in this document to create an ECS instance.

Prerequisites

Before creating an ECS instance, determine the network type of your ECS instance:

Alibaba Cloud stopped providing ECS instances in classic networks to new ECS instance users since 17:00 (UTC +8:00) on June 14, 2017. If you purchase ECS instances for the first time, you have to choose a VPC only. If you have already purchased ECS instances in classic networks, you can still select them.

To create an ECS instance in a VPC, you must select a VPC. The ECS instance is created in the selected VPC and cannot be moved to another VPC. Therefore, make sure you select the right VPC before creating the ECS instance. If you do not have any VPC, create a VPC first.

Create an instance

Log on to the EDAS console.

On the left-side navigation pane, choose **Resource Management > ECS**.

Select the region and namespace (optional) for the ECS instance to be created. Then click **Create Instance** in the upper-right corner.

On the ECS purchase page, select ECS type and configuration and pay for it. See [Create an ECS instance](#) for details.

NOTE: To use and manage an ECS instance on the EDAS console, you must install EDAS Agent on the ECS instance. When purchasing the ECS instance, you can select an EDAS basic image to automatically install EDAS Agent. The procedure is as follows:

In the **Image** section, select **Marketplace Images**. Then, click **Select from image market (including operating system)**.

Enter **EDAS** in the search box and click **Search**.

Select **EDAS JAVA Environment (common ECS instance)** in the results. The latest version is selected by default. Then click **Apply**.

Log on to the EDAS console. In the left-side navigation pane, choose **Resource Management > ECS**.

Select the region and namespace (optional) to view the **Agent Status** of the ECS instance you created. If **Agent Status** is **Online** or **Docker Online**, the ECS instance is in normal state.

NOTE: After the ECS instance is created, it is added to the default namespace and cluster.

Execution results

Click **Management Console** to go back to the **ECS Console**. It usually takes one to five minutes to create an instance. Click **Refresh**. The ECS instance is in **Running** state, indicating that the instance is created successfully.

Import ECS instances

If you have purchased an ECS instance without EDAS Agent installed, you need to import the ECS instance and synchronize it to the EDAS console.

On the left-side navigation bar of the EDAS console, choose **Resource Management > ECS**.

Click **Import ECS** in the upper right corner of the Instances page.

On the **Select Cluster and ECS** page, select **Namespace** and **Import Cluster**, select an ECS instance, and click **Next**.

- After a cluster is selected, the system detects and displays ECS instances available for the cluster.
- If no clusters are desired, click **Create Cluster**.

On the **Set New Password** page, enter a new logon password for the ECS instance and click **Next**.

After an ECS instance is imported, the system clears all data on the ECS instance, and you need to use the official EDAS image to reinstall the operating system. **Keep your new password in mind.**

In the Import ECS host dialog box, click **Import**.

EDAS installs EDAS Agent on the ECS instance and synchronizes the instance to the EDAS Console. The installation and synchronization take about five minutes. Based on the prompts, click **Back** to go to the Cluster Details page under **Cluster**. In the **Cluster Deployment Information** area, you can check the import status and progress.

When the ECS instance changes from **Converting** to **Online** status, the ECS instance has been imported successfully.

Install EDAS Agent

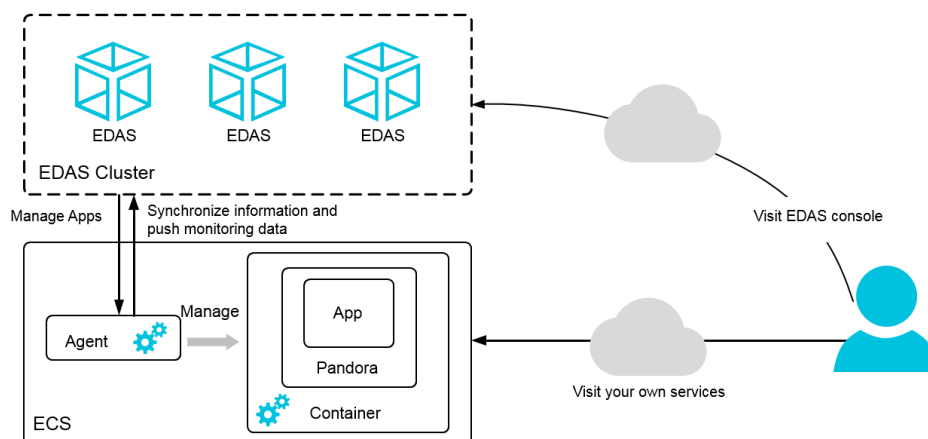
EDAS Agent overview

EDAS Agent is the Daemon program installed on ECS instances to implement communication between the EDAS cluster and the applications deployed on the corresponding ECS instances. EDAS Agent functions as follows:

- Application management: deploys, starts, and stops applications.
- Status reporting: reports application viability status, health check results, and Ali-Tomcat container status.
- Information retrieval: retrieves the monitoring information on ECS instances and containers.

In addition to the preceding application-based management, EDAS Agent is used for communication between the EDAS console and your applications. Here is a simple example. EDAS Agent obtains and submits the information whether a given ECS instance correctly and promptly publishes a service that an application publishes.

NOTE: The preceding functions are transparent to you. You only need to install EDAS Agent.



Install EDAS Agent

EDAS deploys applications (including first-time installation and later on expansion) on ECS instances that are installed with EDAS Agent only. The nodes in the EDAS billing system refer to the ECS instances which are installed with EDAS Agent and deployed with applications. To use EDAS, you must install EDAS Agent while or after you purchase an ECS instance.

You can install EDAS Agent in three ways:

- Use an EDAS base image
- Import an ECS instance
- Manually execute the script for installation

Notes:

- JDK 8 is installed on EDAS Agent by default in the three ways. To use JDK 7 or other versions, you can manually execute the script for installation.
- To manually execute the script for installation, you must log on to the ECS instance as a root. Log on to the ECS instance.
- Currently, EDAS Agent can be installed and run on **64-bit CentOS 6.5/6.8/7.0/7.2/7.3/7.4** or **64-bit Ali-Linux 5.7** only.
- This script can be run repeatedly. Running the script overwrites the existing version of EDAS Agent installed on the instance. Therefore, you upgrade EDAS Agent by running the same script.
- The script for installation is region specific. You must switch to the appropriate region before clicking **Install EDAS Agent**.
- Different installation methods or different images or clusters selected during installation result in different Agent states, which determines the type of applications that can be created on the ECS instance. Pay attention to instructions for different installation methods.

Use an EDAS base image to automatically install EDAS Agent when purchasing an ECS instance

The simplest method of installing EDAS Agent is to use an EDAS base image when purchasing an ECS instance.

NOTE: The disk is formatted if you use this method. To prevent the disk from being formatted, we recommend that you [use the command script to install EDAS Agent manually] (#CLI).

Log on to the EDAS console. On the left-side navigation bar, choose **Resource Management > ECS**.

In the upper-right corner of the ECS Instance List page, click **Create Instance** to go to the ECS purchase page.

On the purchase page, select **Image Market** in the **Image** column, and then click **Select from image market (including operating system)**.

Enter **EDAS** in the search box and click **Search**.

Select images from search results based on requirements of applications created. The latest version is selected by default (no earlier versions are recommended). Click **Use**.

To create a **Regular Application**, select **EDAS Java Environment (Regular ECS)**.

To create a **Docker application**, select **EDAS**.

Purchase an ECS instance based on prompts on the page.

Import an ECS instance to automatically install EDAS Agent

If you did not select any EDAS base image when purchasing an ECS instance, you can use click **Import ECS** in the EDAS Console to install EDAS Agent. The process is as follows:

NOTE: The disk is formatted if you use this method. To prevent the disk from being formatted, we recommend that you [use the command script to install EDAS Agent manually] (#CLI).

Log on to the EDAS console. On the left-side navigation bar, choose **Resource Management > ECS**.

Click **Import ECS** in the upper right corner of the Instances page.

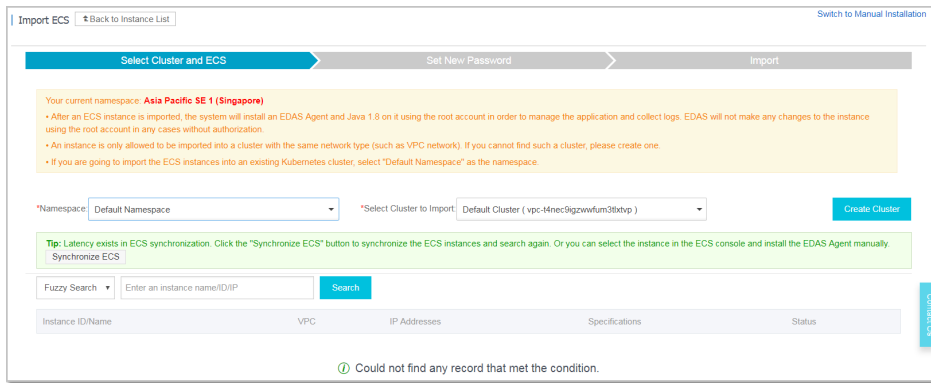
On the **Select Cluster and ECS** page, select **Namespace** and **Cluster**. From the instance list displayed, select ECS instances purchased but not provided with EDAS Agent, and click **Next**. To install EDAS Agent manually, you can select **Switch to Manual Installation** in the upper right corner to go to the **Install EDAS Agent on Single Instance Manually** page and [use the command script to install EDAS Agent manually] (#CLI).

To create a regular application, select ECS Cluster.

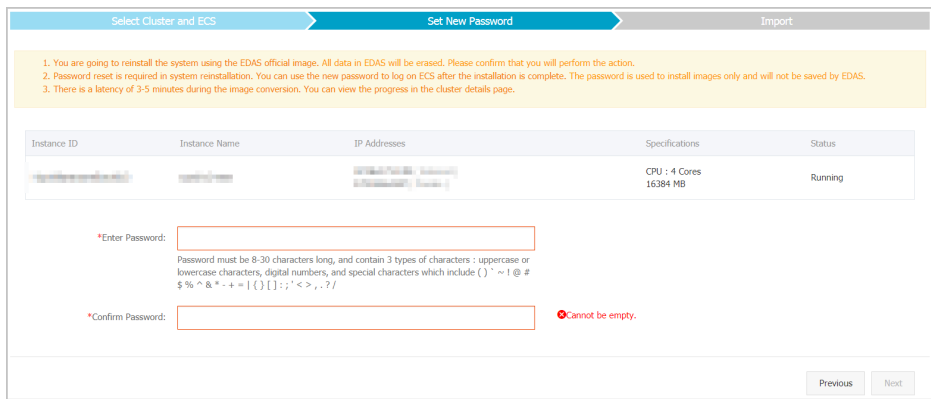
To create a Docker application, select Swarm Cluster.

To create a Kubernetes application, select Kubernetes Cluster.

If no clusters are desired, click **Create Cluster**.



On the **Set New Password** page, set a new password for logon to the ECS instance and click **Next**. In this way, all data in the ECS instance is erased. To reinstall the operating system, you need to reset the password. After image-based installation, you can enter the new password to log on to the ECS instance.



Click **Import** on the **Import ECS Instances** page.

Wait five minutes before the import process is finished. Based on prompts on the page, click **Back** to go to the **Cluster Information** page, and view the import status in the **Cluster Deployment Information** area. If the **health check** result is changed from **Converting** to **Online**, it indicates that the ECS instance is imported successfully with EDAS Agent installed.

Use the command script to install EDAS Agent manually

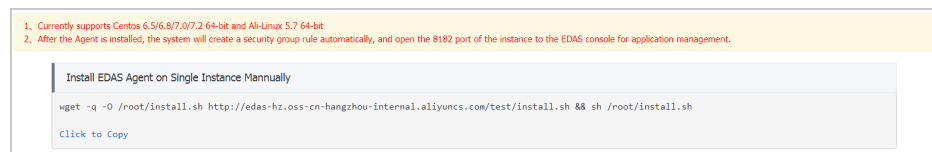
NOTE: This installation method is applicable only to regular applications.

Log on to the EDAS console. On the left-side navigation bar, choose **Resource Management** > **ECS**.

Select a region and namespace for the ECS instance in the upper left corner of the **Instances** page.

Click **Import ECS** in the upper right corner of the Instances page.

Click **Switch to Manual Installation** in the upper right corner of the **Import ECS** page. On the **Install EDAS Agent on Single Instance Manually** page, click **Click to Copy**.



NOTE: For image installation, click **Switch to Image Installation** in the upper right corner of the page and [import an ECS instance] (#ImportECS).

Log on to the ECS instance where you want to install EDAS Agent as a root user.

On the ECS instance, paste the copied command and execute it.

Result verification

After EDAS Agent is installed, choose **Resource Management** > **ECS** on the left-side navigation bar of the EDAS console. On the Instances page, select a region to view the **EDAS Agent Status**.

If EDAS Agent is installed successfully, the status is **Online** or **Docker Online** (Docker application or Kubernetes application).

If EDAS Agent installation failed, the status is **Unknown**.

Upgrade EDAS Agent

The procedure for upgrading EDAS Agent is similar to the procedure for installing EDAS Agent using the command script. For details, see [Use the command script to install EDAS Agent manually] (#CLI).

SLB

If you have purchased an Alibaba Cloud Server Load Balancer instance, EDAS synchronizes the Server Load Balancer instance to the EDAS Console, allowing you to manage the Server Load Balancer instances and to configure load balancing functions.

View Server Load Balancer instances

Log on to the EDAS console.

On the left-side navigation bar, choose **Resources** > **SLB**.

Select a region to view SLB instances in the region.

The following information is displayed for each Server Load Balancer instance:

- **Instance ID/Name:** The instance ID is automatically generated by the system and the instance name is user-defined. Click the instance ID to go to the Server Load Balancer Console.
- **Service Address:** Indicates the Internet or intranet IP address of the SLB instance.
- **Backend Servers:** Indicates ECS instances added in the SLB console, used to receive requests distributed by the SLB instance.
- **Status:** Indicates the status of the SLB server: running or stopped. Expired SLB instances are not displayed.

NOTE: To create an SLB instance, click **Create Instances** in the upper right corner of the page. Go to the SLB purchase page on the Alibaba Cloud website to purchase and create a new SLB instance. For more information, see the [Server Load Balancer documentation](#).

Configure Server Load Balancer

For load balancing configuration information, see [Create Server Load Balancer instances](#) and [Configure Server Load Balancer instances](#).

VPC

Alibaba Cloud provides two network types:

Classic networks

Cloud products on a classic network are all deployed in Alibaba Cloud's public infrastructure and planned and managed by Alibaba Cloud. These products are suitable for customers who have high ease-of-use requirements.

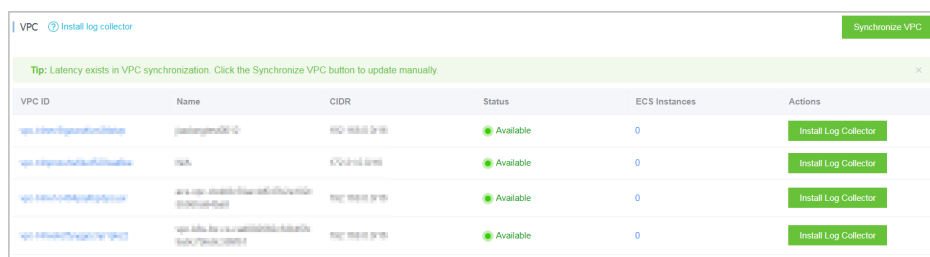
VPC networks

VPC networks are virtual private clouds that allow custom isolation settings. You can define the custom VPC topology and IP address. VPCs are suitable for customers with high cybersecurity requirements and network management capability.

After purchasing a VPC and synchronizing it to the EDAS Console, you can view its information in the EDAS Console.

Log on to the EDAS console.

On the left-side navigation bar, choose **Resources** > **VPC**.



The screenshot shows the EDAS console interface for VPC management. At the top right, there is a 'Synchronize VPC' button. Below it is a tip: 'Tip: Latency exists in VPC synchronization. Click the Synchronize VPC button to update manually'. The main content is a table with the following columns: VPC ID, Name, CIDR, Status, ECS Instances, and Actions. There are four rows of VPC instances, all with a status of 'Available' and 0 ECS instances. Each row has an 'Install Log Collector' button in the Actions column.

VPC ID	Name	CIDR	Status	ECS Instances	Actions
vpc-12345678901234567890	test-vpc-1	192.168.0.0/24	Available	0	Install Log Collector
vpc-23456789012345678901	test-vpc-2	192.168.0.0/24	Available	0	Install Log Collector
vpc-34567890123456789012	test-vpc-3	192.168.0.0/24	Available	0	Install Log Collector
vpc-45678901234567890123	test-vpc-4	192.168.0.0/24	Available	0	Install Log Collector

Select a region to view VPC instances in the region.

This page shows the following information:

- **VPC ID:** The VPC ID that is automatically generated by the system when the VPC is created. Click the VPC ID to go to the VPC Console.
- **Name:** It is specified by you when you create a VPC.
- **CIDR:** The VPC's CIDR Block you specified when the VPC is created.
- **Status:** The status of the instance: Running or Stopped. Expired VPCs are not displayed.
- **ECS instances:** The number of ECS instances created in this VPC network. Click the number to go to the ECS page, where you can see all the ECS instances in this VPC.

In a VPC network, ECS instances are isolated from EDAS instances. Therefore, you must install the log collector to collect the information on ECS instances. Click **Install Log Collector** in the Action column on the Instances page. For log collector installation instructions, see [Install log collector](#).

Resource groups

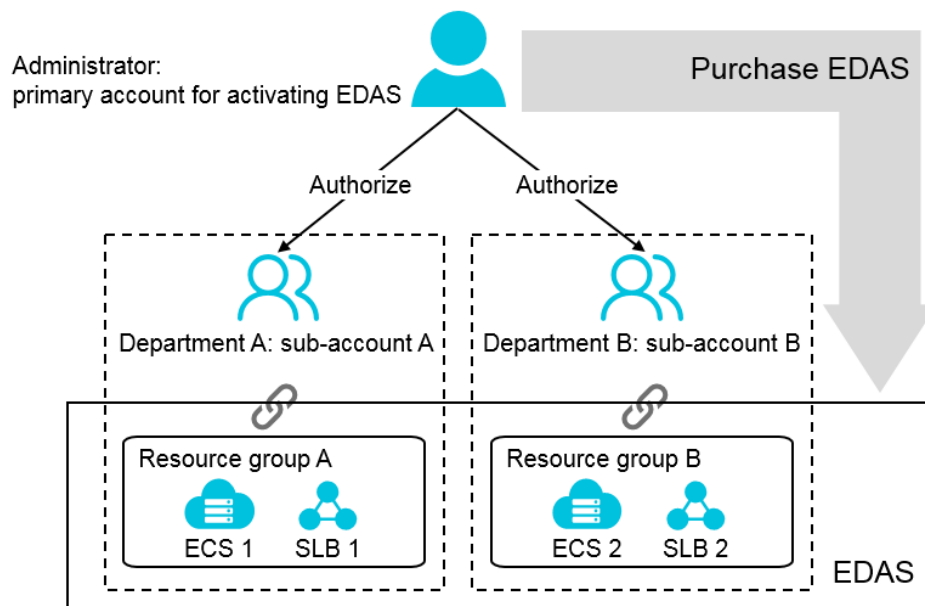
Overview

Resource groups are groups of EDAS resources, which can be ECS instances and SLB instances, but not VPCs. You can assign permissions to access resource groups to your sub-accounts, and each sub-account has the right to operate on all the resources in the specified group. You can assign permissions to access resource groups to your sub-accounts, and each sub-account has the right to operate on all the resources in the specified group. For more information on primary accounts and subaccounts, see [EDAS account system](#).

Typical use cases

- A company uses EDAS to create business applications. Department A is responsible for user-related applications and Department B for goods-related ones.
- The company registers for an EDAS account (the primary account) to activate EDAS and creates two subaccounts respectively for Departments A and B. Department A has a couple of ECS instances and Server Load Balancer instances for deploying user-related applications.
- The company creates a resource group in EDAS, adds Department A's ECS instances and Server Load Balancer instances to the resource group, and grants permissions for this resource group to Department A's subaccount.
- Department A can use its subaccount to operate on the resources in this resource group only. The resources for Department A and Department B do not conflict.

as shown below.



Resource group operations

Resource group operations are performed in the EDAS Console. Follow the procedure below to go to

the Resource Groups page.

Log on to the EDAS console.

On the left-side navigation bar, choose **Resources > Resource Groups**.

Select a region to view resource groups in the region as well as ECS instances and SLB instances in each group.

View resource groups

In the Instance List, you can view information about the resource groups, including resource group names, descriptions, ECS instances (intranet/Internet IP addresses), and Server Load Balancer instances.

Create a resource group

1. Click **Create Resource Group** in the upper right corner on the Instances page.
2. Enter the resource group name and description, and click **OK**.

Add an ECS instance to the resource group

1. In the Instance List, click **Bind ECS** in the **Actions** field of the resource group you want to add an ECS instance to.
2. In the Bind ECS dialog box, select the ECS instance to bind and click **OK**.

Add an SLB instance to the resource group

1. In the Instance List, click **Bind Server Load Balancer** in the **Actions** field of the resource group you want to add a Server Load Balancer instance to.
2. In the Bind Server Load Balancer dialog box, select the Server Load Balancer instance to bind and click **OK**.

Edit a resource group

1. On the Instances page, click **Edit** in the Action column for the resource group you want to edit.
2. In the Edit Resource Group dialog box, edit the resource group name and description, and then click **OK**.

Grant resource group permissions to a sub-account

1. Log on to the EDAS console using the primary account.
2. On the left-side menu bar of the console, choose **Account Management** > **Sub-Accounts**.
3. Click **Resource Group Permission** in the Action column for the account to which you want to grant permissions.
4. In the Resource Group Permission dialog box, select a resource group and click **OK**.

Delete a resource group

1. On the left-side menu bar of the EDAS console, choose **Resource Management** > **Resource Group**.
2. Click **Delete** in the Action column for the resource group you want to delete.
3. Click **OK** in the confirmation dialog box.

Clusters

Create ECS clusters

You can create an Elastic Compute Service (ECS) cluster for the application in two steps:

1. Create a Swarm cluster
2. Add hosts to the cluster

If necessary, you can Remove Cluster Host.

Create a Kubernetes cluster

Log on to the EDAS console.

On the left-side navigation bar, choose **Resource Management** > **Cluster** to go to the Clusters page.

Select **Region** and **Namespace** on the Clusters page.

Click **Create Cluster** in the upper right corner and enter information in the Create Cluster dialog box displayed.

Cluster field descriptions:

- **Cluster Name:** Enter the cluster name. The name contains no more than 64 characters, which can be letters, digits, underlines, or dots.
- **Type:** Select **ECS**.
- **Network Type:** classic network or VPC. Select a network type from the drop-down list as needed. **If you select VPC Network, ensure that you have created a Virtual Private Cloud (VPC).**
- **VPC Network:** Select VPC from the drop-down list.
- **Namespace:** It is displayed on the Clusters page and cannot be configured. If it is not selected, a region is displayed by default.

After configuration is finished, click **Create** to create the cluster.

After the cluster is created, a message **Cluster created successfully** is displayed in the upper right corner of the current page, and the new cluster is displayed in the cluster list.

NOTE: The created cluster is empty. You must **Add Cluster Host** so that applications can use the cluster.

Add a cluster host

Procedure

On the Clusters page, click the cluster name.

Click **Add Cluster Host** in the upper right corner of the Cluster Details page.

On the **Select Cluster and ECS** page, select an instance adding method and ECS instance from the instance list, and click **Next**.

- **Import ECS:** The namespace and imported cluster cannot be configured. An ECS instance is imported from the default namespace and cluster under the region.
- **From Existing Cluster:** Under the region, select a namespace and cluster from which an ECS instance is imported.

If no instances meet the necessary conditions, click **Create ECS Instance** in the upper-right corner of the page. This takes you to the ECS purchase page on the Alibaba Cloud official website, where you can purchase and create a new ECS instance. For details, see [Create ECS Instance](#).

On the **Set New Password** page, enter and confirm a new ECS logon password, and then click **Next**.

- An official EDAS image is used to reinstall the operating system. All data in the ECS instance is deleted during the reinstallation. Make sure that you want to import data and back up the data in the ECS instance.
- The reinstallation requires you to reset the password. After installing the image, use the new password to log on to the ECS instance. This password is only used to install images and is not stored in EDAS.

In the dialog box displayed, click **Import**.

The reinstallation process takes three to five minutes. You can check the import progress on the Cluster Details page.

Result verification

During the import, you can check the status and progress of ECS instance import in the Cluster Deployment Information area of the Cluster Details page.

When the ECS instance is imported, the health check result is **Converting**, and the progress is displayed in percentage form. The entire import process takes about five minutes.

After the conversion, go back to the Cluster Details page and check the conversion result in the Cluster Deployment Information area.

If conversion succeeded, the health check result is **Converting**. In this case, the node resources can be used to deploy applications.

If import failed because of an exception during conversion, the host node is in **Conversion failed** state.

Identify and troubleshoot the cause of failure and then click **Retry** next to the host node.

Remove a cluster host

Go to the Clusters page and click **Remove** in the Action column of the Cluster Deployment Information area.

Confirm the host information in the dialog box displayed and then click **Remove**.

Manage clusters

View cluster list

Log on to the EDAS console.

On the left-side navigation bar, choose **Resource** > **Clusters**.

On the **Clusters** page, select a region to view information on the clusters in this region.

NOTES:

- Two types of clusters are available: **ECS Cluster** and **Swarm Cluster**.
- There are two network types, **VPC** and **Classic Network**.

Cluster Name/ID	Network Type	Type	Nodes	Resource Amount	Remaining Resource Amount	Status	Namespace	Creation Time	Actions
ecs-cluster-xxxx-xxxx-xxxx-xxxx-xxxx	VPC	ECS Cluster (Alibaba Cloud)	0	CPU: 0 Core(s) Memory: 0 MB	CPU: 0 Core(s) Memory: 0 MB	Normal	xxxx-xxxx-xxxx-xxxx	2018-10-31 20:34:05	Delete
ecs-cluster-yyyy-yyyy-yyyy-yyyy-yyyy	VPC	ECS Cluster (Alibaba Cloud)	2	CPU: 4 Core(s) Memory: 16384 MB	CPU: 0 Core(s) Memory: 0 MB	Normal	xxxx-xxxx-xxxx-xxxx	2018-10-18 16:11:00	Delete

View cluster details and deployed hosts

On the Cluster List page, click the name of a cluster to go to the Cluster Details page.

Cluster Information		View Details	Settings	Refresh		
Cluster attribution : Alibaba Cloud	Cluster ID : xxxxxxxx-xxxx-xxxx-xxxx-xxxx					
Cluster Name : xxxxxxxx	Namespace : xxxxxxxx-xxxx-xxxx-xxxx					
Type : ECS Cluster	VPC ID : xxxxxxxx-xxxx-xxxx-xxxx					
Network Type : VPC Network	Virtual Switch : N/A					
Container Starting Segment : N/A	Status : Running					
Description : N/A						
Cluster Deployment Information		Transfer Cluster Host				
Instance ID/Name : Enter an instance name	Host type : Select	Search	Refresh			
Instance ID/Name	IPAddress	Specifications	Available Resources	Health Check	Application deployed	Actions
xxxx-xxxx-xxxx-xxxx-xxxx	47.94.204.203 (Internet) 192.168.10.165 (Private)	CPU:2Core(s) Memory:8192MB	CPU:Shared Memory:0MB		xxxx-xxxx-xxxx-xxxx	View Details
xxxx-xxxx-xxxx-xxxx-xxxx	47.94.204.8 (Internet) 192.168.10.166 (Private)	CPU:2Core(s) Memory:8192MB	CPU:Shared Memory:0MB		xxxx-xxxx-xxxx-xxxx	View Details

The Cluster Details page has two main sections: cluster information and cluster deployment information.

- The cluster information section displays basic cluster information.
- The cluster deployment information section displays a lists of hosts for the cluster. The host list shows the basic information, health check status, and deployed applications for each host.

Click the buttons to perform the corresponding cluster and host operations.

- Click **View Details** to view details about clusters and hosts.

- Click **Event** to view cluster and host events. Event information helps you locate the involving clusters and hosts.
- In the host list, click the name of an **Application deployed** to go to the Instance Information page associated with the Application Details page.

Transform a cluster host

In the Cluster Deployment Information area of the Cluster Details page, select an ECS instance and then click **Transfer Cluster Host** on the right.

On the Select Target Cluster page, select a namespace and target cluster, and click **Next**.

On the Set New Password page, set a new password for the ECS instance, and click **Next**.

In the "Add a host to the ECS cluster" dialog box, confirm the host transfer information, and click **OK**.

Namespaces

Overview

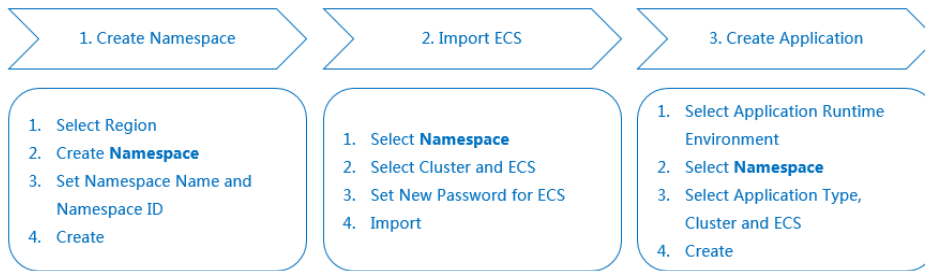
During use of Enterprise Distributed Application Service (EDAS), resource (ECS instance) isolation is usually required. EDAS allows you to allocate resources by account so that you can isolate resources using accounts. You can also isolate resources through Virtual Private Cloud (VPC) isolation. However, both of the preceding methods increase the workload in operation difficulty (account switching) and implementation cost (multiple VPCs needed) and cannot balance resource isolation with unified management. Resources cannot be isolated over classic networks.

To solve this problem, we provide EDAS with a new function: **Namespace**. On classic networks or VPCs in a region, one namespace corresponds to one environment (including one or more clusters), and different namespaces are logically isolated from each other by nature. Multiple namespaces can be created for one account. Namespaces help you completely isolate resources in different environments from each other, and you can use one account to manage them in unified mode.

Scenarios

A namespace is mainly used to isolate resources. The procedure is as follows:

1. Create a namespace
2. Import an ECS instance
3. Create an application



After the preceding actions are finished, resources are isolated.

In addition, you can add or transfer cluster hosts based on namespaces in a region. For details about actions, see [Create ECS clusters](#) and [Cluster management](#). With resources isolated, unified management can be implemented using one account.

Create a namespace

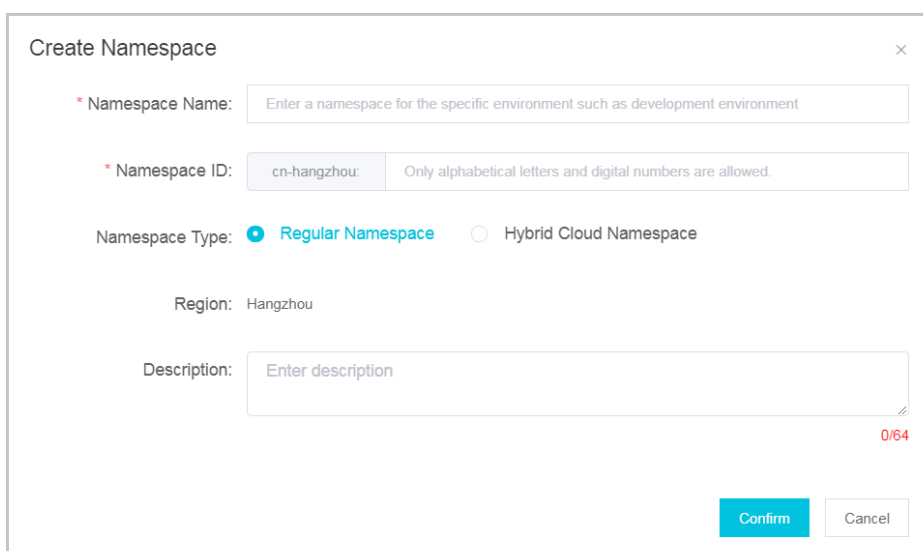
Log on to the EDAS console.

In the left-side navigation pane, Click **Namespaces**.

On the **Namespaces** page, select **Region**, and then click **Create Namespace** in the upper-right corner.

In the **Create Namespace** dialog box, set the **Namespace Name** (required), **Namespace ID** (required), **Namespace Type** and **Description**, and click **OK**.

NOTE: The prefix of Namespace ID is determined based on the specified Region and is not editable. Only the custom part is editable.



The image shows a 'Create Namespace' dialog box with the following fields and options:

- Namespace Name:** A text input field with a placeholder 'Enter a namespace for the specific environment such as development environment'.
- Namespace ID:** A text input field with 'cn-hangzhou' entered and a note 'Only alphabetical letters and digital numbers are allowed.'
- Namespace Type:** Two radio buttons: 'Regular Namespace' (selected) and 'Hybrid Cloud Namespace'.
- Region:** A dropdown menu showing 'Hangzhou'.
- Description:** A text area with a placeholder 'Enter description' and a character count '0/64'.
- Buttons:** 'Confirm' and 'Cancel' buttons at the bottom right.

Edit a namespace

Click **Edit** in the **Action** column on the **Namespaces** page.

In the **Edit Namespace** dialog box, edit the **Namespace Name** and Description, and then click **OK**.

NOTE: The Namespace ID cannot be changed.

Delete a namespace

The following conditions must be met before you delete a namespace:

- No ECS instances exist under the namespace.
- No clusters exist under the namespace.

Click **Delete** in the **Action** column on the **Namespaces** page.

In the dialog box displayed, confirm the namespace name and then click **Delete**.

Application management

Applications in ECS Cluster

Create general applications

A common application is an application deployed directly on an ECS instance. Only one common application can be deployed on each ECS instance. To publish a common application is to install an EDAS container on your ECS instance and then use a WAR/JAR package to deploy the application in the EDAS container.

Prerequisite

Before creating an application on EDAS, you need to create resources.

Create a common application

Note: The ECS instance for publishing applications needs to be imported (install EDAS Agent and synchronize data to EDAS). If not, import an ECS instance first.

Log on to EDAS console.

Click **Applications** in the left-side navigation pane to go to the application list page.

Click **Create Application** in the upper-right corner of the Applications page.

In the **Application Information** page, enter application information and click **Next Step: Application Configurations**.

- **Namespace:** Select **Region** and **Namespace** from the drop-down list. If you do not select an option, the **Default** namespace is automatically selected.
- **Cluster:** Select an **ECS cluster** from the drop-down list.
- **Type:** The application type is determined by the cluster where the application is deployed. If you select an ECS cluster, the application type is **Java application** (that is, common application).
- **Name:** Enter the application name.
- **Runtime Environment:** Select the EDAS container version from the drop-down list. The latest version is selected by default.
- **Java Environment:** Select **JDK 7** or **JDK 8** from the drop-down list.
- **Application Description:** Enter basic information about the application. The description contains no more than 100 characters.

On the **Application Configuration** page, select **Deployment Method** and click **New** to add the instance as prompted. Click **Create**.

Deployment Method: You can select **WAR** or **JAR**.

Select Instances: Click **New**. In the Select the single-machine dialog box, select an ECS instance on the left, click > to add the instance to the area on the right, and click **OK**.

- If you click **Create Blank Application** without selecting any instance, you can create a blank application. Then you can click **Scale Out**, **Add Instance**, or **Deploy Application** to release the application.
- If you select an instance, click **Create** to create a blank application containing instances. Then you can click **Deploy Application** to release the application.

Deploy Now: This option is available only after you select an instance.

- If you select **Deploy Now**, you can deploy the application when you create it. For more information about parameter setting, see **Deploy the application while creating it**.

- If you do not select **Deploy Now**, you can click **Deploy Application** on the **Application Details** page after you create the application to deploy the application.

In the **Application Creation Complete** tab, click **Application Details** to view the basic information and instance information about the application.

Deploy the application while creating it

WAR package deployment and **JAR package deployment** are similar in the configuration procedure. The following describes **WAR package deployment** as an example.

File Upload Method: You can select **Upload WAR Package** or **WAR Location**.

Upload WAR Package: Click **Select** and select a WAR package to upload.

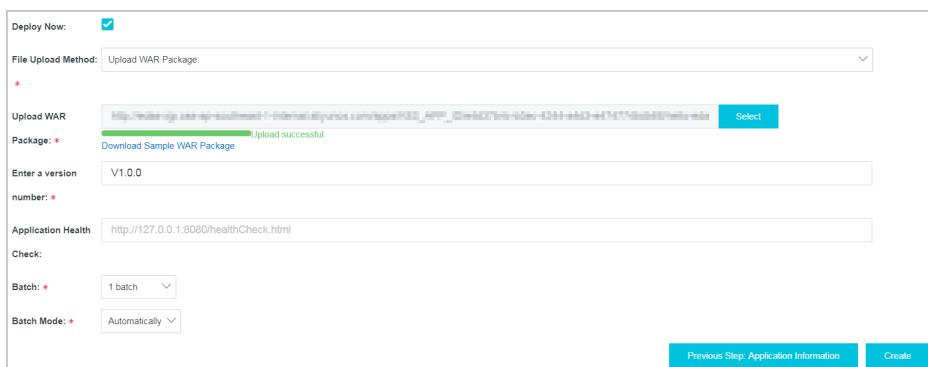
If you select **WAR Location**, enter the WAR package address in the configuration bar.

Enter a version number: Set the version (for example, 1.1.0). We recommend that you do not use the timestamp as a version number.

Application Health Check (optional): Configure the URL, port, and configuration file for health check. The system checks health of the application after the container is started or when the container is running to verify whether the application is normal, and executes service routing based on the health check result. The default URL is `http://127.0.0.1:8080/healthCheck.html`.

Batch: If you select more than two batches, you need to set the batch time.

Batch Mode: Select **Automatic**.



The screenshot shows a configuration form for deploying an application. The form includes the following fields and options:

- Deploy Now:** A checked checkbox.
- File Upload Method:** A dropdown menu set to "Upload WAR Package".
- Upload WAR Package:** A text input field containing a URL, with a "Select" button to the right. Below it, a green progress bar indicates "Upload successful".
- Enter a version number:** A text input field containing "V1.0.0".
- Application Health Check:** A text input field containing "http://127.0.0.1:8080/healthCheck.html".
- Batch:** A dropdown menu set to "1 batch".
- Batch Mode:** A dropdown menu set to "Automatically".

At the bottom right of the form, there are two buttons: "Previous Step: Application Information" and "Create".

Configure SLB (optional)

Server Load Balancer distributes application access traffic to ECS instances based on a forwarding policy. This enhances the service capabilities and availability of applications. If your application is deployed on multiple ECS instances, we suggest configuring Server Load Balancer.

Today, most applications are deployed on VPCs. To open your application on the Internet, you must use Server Load Balancer (Internet).

Go to the Basic Information tab of the Application Details page. In the Application Settings area, click **Add of Server Load Balancer (Internet)** or **Server Load Balancer (Intranet)**.

In the **Bind Server Load Balancer to Application** dialog box, set the Server Load Balancer parameters and then click **Configure Server Load Balancer**.

- **SLB:** Select the intranet or Internet Server Load Balancer address from the drop-down menu.
- **Use the virtual server group:** A virtual server group is a group of frontend ECS instances used to process requests distributed by Server Load Balancer. Listeners can be associated with different virtual server groups to monitor request forwarding. If you select **Use Virtual Server Group**, you must configure the Virtual Server Group parameter.
 - **Virtual Server Group:** Select an existing virtual server group from the drop-down menu. If no virtual server group is available, click **Create Virtual Server Group** from the drop-down menu.
 - **Virtual Server Group Name:** Enter the name for the new virtual server group if you select **Create Virtual Server Group**. The system creates a virtual server group with the specified name.

Listener: Server Load Balancer listeners regulate how requests are forwarded to backend servers. At least one listener must be created for each Server Load Balancer instance. You can select a listening port from the Listener drop-down menu. If no listener has been created, click **Create Listener**.

- **SLB Frontend Protocol:** The default setting is TCP, which cannot be configured manually.
- **SLB Frontend Port:** Enter the frontend port of the Server Load Balancer instance.

Application Port: The default setting is 8080 and cannot be changed.

Note: Do not delete the listener in the SLB console; otherwise, the application cannot be accessed normally.

Result verification

Check the status of the ECS instance in the **Instance Information** tab of the Application Details page. If the status/time is **Normal**, then the application has been published successfully.

To open your application on the Internet, you must configure Server Load Balancer (Internet). Go to the **Basic Information** tab on the Application Details page. In the **Application Settings** area, copy the **Server Load Balancer (Internet) IP Address and Port**, for example, *118.31.159.169:81*. Then, paste it in your browser and press Enter. If the Welcome page of the application is displayed, this also indicates that the application has been successfully released.

Config general applications

This topic describes how to set parameters for common applications, including **JVM parameters**, Tomcat, Server Load Balancer (SLB), health check, and **basic information**.

Go to the settings page

1. Log on to the EDAS console.

Click **Applications** in the left-side navigation pane, and click an application on the Applications page to go to the **Basic Information** page.

Click **Settings** in the **Application Settings** area.

Set JVM parameters

JVM parameters are the container parameters that must be configured when an application is started. Correct setting of JVM parameters helps reduce the overhead of garbage collection and thus shorten the server response time and improve throughput. If container parameters are not set, JVM parameters are allocated by default.

In the Application Settings dialog box, click the **JVM** tab.

Click **Memory**, **Application**, **GC**, **Tools**, and **Custom** to set parameters. Click **Save** to save the settings.

Note: After you set JVM parameters, restart the application manually through EDAS for the settings to take effect.

Configure Tomcat

You can configure the following parameters for the Tomcat container in the EDAS console: port number, application access path, and maximum number of threads in the connection pool.

In the Application Settings dialog box, click the **Tomcat** tab.

Set Tomcat parameters, and click **Configure Tomcat**.

Description of Tomcat parameters:

Configuration	NOTE
Application Port	The port number range is (1024, 65535). The admin authority is needed for container configuration and the root authority is required to operate ports with numbers less than 1024. Therefore, enter a port number greater than 1024. The default value is 8080. Therefore, enter a port number greater than 1024. If this parameter is not set, the default value is 8080.
Tomcat Context	Select an application access path. <ul style="list-style-type: none"> - If you select Package Name, you do not need to set the Custom Path parameter. The application access path is the name of the WAR package. - If you select Root, you do not need to set the Custom Path parameter. The application access path is /. - If you select Custom, set the Custom Path parameter. If the "Custom Path" parameter is not set, the default application access path is the same as the name of the WAR package.

Maximum Threads	Set the number of connections in the connection pool. It corresponds to the maxThreads parameter. The default value is 400. We recommend that this parameter be set under professional guidance.
Tomcat Encoding	Select an encoding format of Tomcat. The options include UTF-8, ISO-8859-1, GBK, and GB2312. Default format is ISO-8859-1.

Configure an SLB instance

If you have bought Alibaba Cloud Server Load Balancer, EDAS synchronizes your Server Load Balancer instance to the EDAS console and provides the relevant configuration function. For more SLB information, see [What is Server Load Balancer](#).

Server Load Balancer instances are classified into Internet- and intranet-based instances by IP address, which share the same configuration method.

The following describes how to configure an Internet-based Server Load Balancer instance.

In the Application Settings dialog box, click the **SLB** tab.

Set SLB instance parameters, and click **Configure SLB**.

Description of SLB parameters:

SLB (intranet): intranet SLB instance. After you select **Use the virtual server group**, click **New virtual server group**, and set **Virtual Server group name**.

Listener (intranet): Set the port, or click **Create new listener** and set **SLB frontend port number**.

SLB (Internet): Internet SLB instance. After you select **Use the virtual server group**, click **New virtual server group**, and set **Virtual Server group name**.

Listener (Internet): Set the port, or click **Create new listener** and set **SLB frontend port number**.

Health check

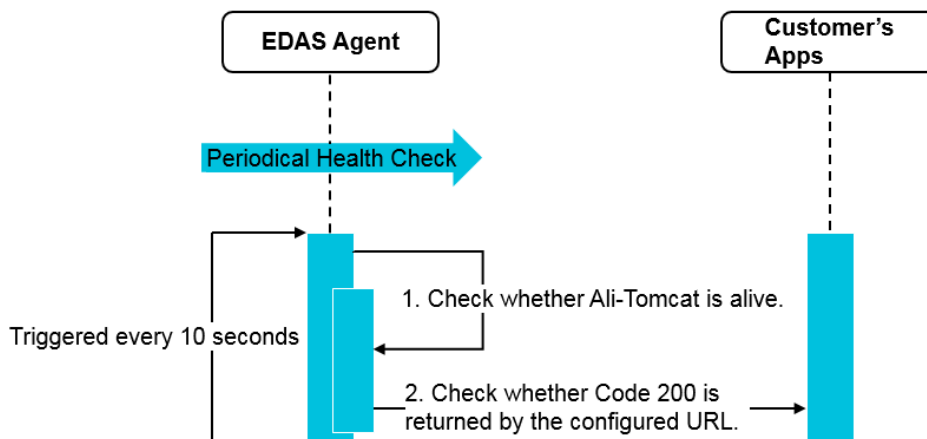
EDAS provides the health check function used to check whether deployed applications run

properly. To use this function, configure a health check URL.

Introduction to health check

Health check is the process where EDAS Agent periodically checks and reports the status of containers and applications and then sends the check results to the console. Health check helps you understand the overall service running status in a cluster environment and facilitates auditing and troubleshooting. You can configure a health check URL in the EDAS console to check whether deployed applications run properly.

The following figure shows how EDAS Agent performs health check for applications.



Health check is triggered every 10 seconds. The steps marked by 1 and 2 are described as follows.

EDAS Agent checks whether the Ali-Tomcat process for running your application is alive.

If the process is alive, the agent proceeds to Step 2.

If the process is not alive, health check ends and the check result is "Fail".

EDAS Agent checks whether Code 200 is returned by the configured URL.

If no URL is configured, health check does not start; if a URL is configured, EDAS Agent checks whether HTTP 200 is returned by the configured URL.

For description of status in Steps 1 and 2, see the following section.

View the health check status

go to the Application Details page .

View the **Running Status** field of the **Instance Information** list in the lower part of the page.

Description of real-time status:

Container Exits: Displayed when the agent detects that the Ali-Tomcat process is not alive in Step 1.

Application Exception: displayed when any other code than 200 is returned by the configured URL in Step 2.

Normal: displayed if no exception occurs in Step 1 and Step 2.

If EDAS Agent detects no configured URL in Step 2, the **Normal** state is still displayed, followed by an exclamation mark. When you place the pointer over it, the "Configure a health check URL to check the application status accurately" message appears."

Agent Abnormal: Displayed if the agent does not report status information to the EDAS server in 30 seconds.

Configure health check

If health check is not configured, containers with agent versions later than EDAS Agent 2.8.0 automatically allocate the health check path `http://127.0.0.1:8080/healthCheck.html`. Click **Modify** next to the health check URL in the **Basic Information** tab of the Application Details page.

Click the **Health Check** tab in the Application Settings dialog box, set the health check URL, and click **Save**.

If you have set related container parameters, configure a health check URL in the format of `http://127.0.0.1:[custom port number]/[configured path]/healthCheck.html` based on the container configuration. "`healthCheck.html`" is the default path. Replace it with a custom path as needed. Ensure that the health check URL is accessible by applications and can return HTTP Code 200 to Code 500.

[Example] Assume that the WAR package name is "`order.war`". You can configure such a health check URL `http://127.0.0.1:8080/order/healthCheck.html` if no other container parameters are configured; or you can configure the health check URL as `http://127.0.0.1:8081/healthcheck.html` if the container path is configured as the root path, the port number is set to "`8081`", and the WAR package contains the "`healthcheck.html`" file for the purpose of health status marking.

Set basic information

In the Application Settings dialog box, click the **Basic Information** tab, set **Application Name** and **Application Description**, and click **Modify**.

Deploy general applications

Common applications support **WAR package deployment** and **JAR package deployment**. If you have deployed an application when creating a common application, you can upgrade the application as follows.

Deploy the application

Log on to the EDAS console.

Click **Application Management** in the left-side navigation pane, and click an application on the Applications page to go to the **Basic Information** page.

Click **Deploy Application**, set deployment parameters as prompted, and click **Publish**. If you upgrade the application, the optional type of the file upload mode is determined by the type of the first deployment. (The WAR package and JAR package are almost the same in terms of deployment and parameter configuration. The following describes how to deploy the WAR package.)

Deploy Application
✕

***File Uploading Method:**

***Upload War Package:**

***Version:**

Description:

***Group:**

***Batch :**

***Batch Mode:**

[Download Sample Project](#)

[Use Timestamp as Version Number](#)

Container Version ▼

Configuration	NOTE
File Uploading Method	Select Upload WAR Package , WAR URL , or Use Previous Version . (Different pages appear when you select WAR package address and <i>Use Previous Version</i> . Set parameters as prompted.)
Upload WAR Package	Click Select File , and select a WAP package .
Version	Set the version (for example, 1.1.0). We do not recommend using a timestamp as the version.
Description	The description can be blank. Enter the purpose of deploying this application. The length is a maximum of 128 characters.
Group	Select a group.
Batch	Set the batch. If you select more than two batches, you need to set Batch Time.
Batch Mode	Select Automatically.
Java env	Select JRE 8 or JRE 7.

The file uploading progress is shown at the top of the dialog box. After it reaches 100%, you are redirected to the **Change Details** page, where you can view the deployment process and operation logs. After deployment, the status changes to **Execution Successful**.

3. Configure SLB (Optional)

SLB distributes application access traffic to ECS instances based on a forwarding policy. This enhances the service capabilities and availability of applications. Today, most applications are deployed on VPCs. To open your application on the Internet, you must use Server Load Balancer (Internet).

Go to the Basic Information tab of the Application Details page. In the Application Settings area, click **Add of Server Load Balancer (Internet)** or **Server Load Balancer (Intranet)**.

In the **Bind Server Load Balancer to Application** dialog box, set the Server Load Balancer parameters and then click **Configure Server Load Balancer**.

Bind SLB To Application

After SLB Port Monitoring is enabled, the system will add port monitors to newly-added SLB ports. Do not delete the monitor on the SLB console. Otherwise, application access will be affected.

SLB(Internet) : 161.117.107.235 (info)

Use the virtual server group

Virtual Server group (Internet) : New virtual server group

Virtual Server group name : test

Listener (Internet) :

Create new listener :

SLB Frontend Protocol: TCP

SLB Frontend Port: Please enter the SLB frontend port number.

Application Port: 8080

Save Cancel

- **Server Load Balancer:** Select the intranet or Internet address of Server Load Balancer as needed from the drop-down menu.

- **Use Virtual Server Group:** A virtual server group is a group of frontend ECS instances used to process requests distributed by Server Load Balancer. Listeners can be associated with different virtual server groups to monitor request forwarding. If you select **Use Virtual Server Group**, you must configure the Virtual Server Group parameter.
 - **Virtual Server Group:** Select an existing virtual server group from the drop-down menu. If no virtual server group is available, click **Create Virtual Server Group** from the drop-down menu.
 - **Virtual Server Group Name:** Enter the name for the new virtual server group if you select **Create Virtual Server Group**. The system creates a virtual server group with the specified name.

Listener: Server Load Balancer listeners regulate how requests are forwarded to backend servers. At least one listener must be created for each Server Load Balancer instance. You can select a listening port from the Listener drop-down menu. If no listener has been created, click **Create Listener**.

- **Server Load Balancer Frontend Protocol:** The default setting is TCP, which cannot be configured manually.
- **Server Load Balancer Frontend Port:** Enter the frontend port of the Server Load Balancer instance.

Application Port: The default setting is 8080 and cannot be changed.

Note: Do not delete the listener in the SLB console; otherwise, the application cannot be accessed normally.

Deployment result verification

Check the status of the ECS instance in the **Instance Information** tab of the Application Details page. If the status/time is **Normal**, then the application has been published successfully.

To open your application on the Internet, you must configure Server Load Balancer (Internet). Go to the **Basic Information** tab on the Application Details page. In the **Application Settings** area, copy the **Server Load Balancer (Internet) IP Address and Port**, for example, *118.31.159.169:81*. Then, paste it in your browser and press Enter. If the Welcome page of the application is displayed, this also indicates that the application has been successfully released.

Stop and start applications

After an application is released, you can stop and start the application on the Enterprise Distributed Application Service (EDAS) console.

Stop the application

Log on to the EDAS console.

Click **Applications** in the left-side navigation pane, and click an application on the Applications page to go to the **Basic Information** page.

When you click **Stop**, the application is stopped.

Start the application

Common applications and Docker applications are released upon deployment. You do not need to start them. Therefore, you only need to restart it after it is stopped.

Log on to the EDAS console.

Click **Applications** in the left-side navigation pane, and click an application on the Applications page to go to the **Basic Information** page.

When you click **Start**, the application is restarted.

Scale out and scale in applications

- When the Elastic Compute Service (ECS) instance on which the applications are deployed is overloaded, you can scale out the application manually.
- When fewer resources are required, you can stop and delete an ECS instance to scale in the application.

Scale out an application

When the ECS instance on which the applications are deployed is overloaded, you can scale out the application manually.

Log on to the EDAS console.

Click **Applications** in the left-side navigation pane, and click an application on the Applications page to go to the **Basic Information** page.

Click **Scale Out** in the upper right corner of the Application Details page.

In the **Scale Out** dialog box, choose **Target Group > ECS**, click **Confirm**.

The running status of an ECS instance added by scaling up the application depends on the running status of the application.

If the application is running when being scaled out, the ECS instance automatically deploys, starts, and runs the application.

If the application stops when being scaled out, the ECS instance automatically deploys the application but does not start or run the application.

Scale in the application

Log on to the EDAS console.

Click **Applications** in the left-side navigation pane, and click an application on the Applications page to go to the **Basic Information** page.

On the Application Details page, click the **Instance Information** tab.

Deprecate the instance on the **Instance Deployment Information** tab page.

If an ECS instance is running, choose **Stop > Delete**.

If an ECS instance stops, click **Deprecate** to delete the instance from the application.

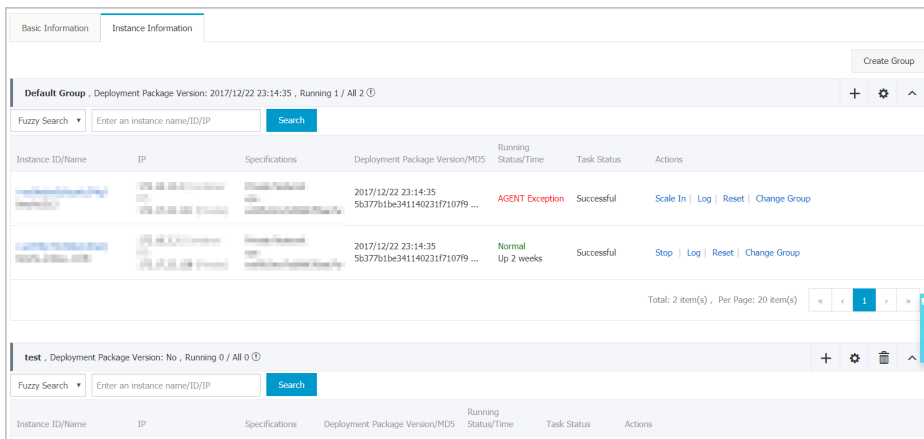
Manage instance groups

Overview

The instance group puts all instances (ECS) for an application in a group so that you can deploy different application package versions to instances in different groups.

Example: There are 10 instances in the “itemcenter” application, divided into two groups: “Default Group” and “Beta Group”. The default group has six instances and the Beta group has four instances. Now there are two groups of instances in the application to which you can deploy different versions of the application package.

For an overview of groups in an application, see:



NOTES:

1. When an application is created in EDAS, a new group “Default Group” will be created by default for the application, and cannot be deleted.
2. If multi-version deployment is not required, the default group is adequate so that you do not need to create other groups.

User guide

The instance group is a feature of Enterprise Distributed Application Service (EDAS) designed to manage instances in an application by group. By grouping instances for the application, you can operate and maintain it through A/B test and gated release. You can quickly improve the O&M efficiency through application lifecycle management as well as resource monitoring and alarm by group.

The group instance management mainly includes the following tasks:

View groups

Create groups

Add instances to a group

Delete groups

View groups

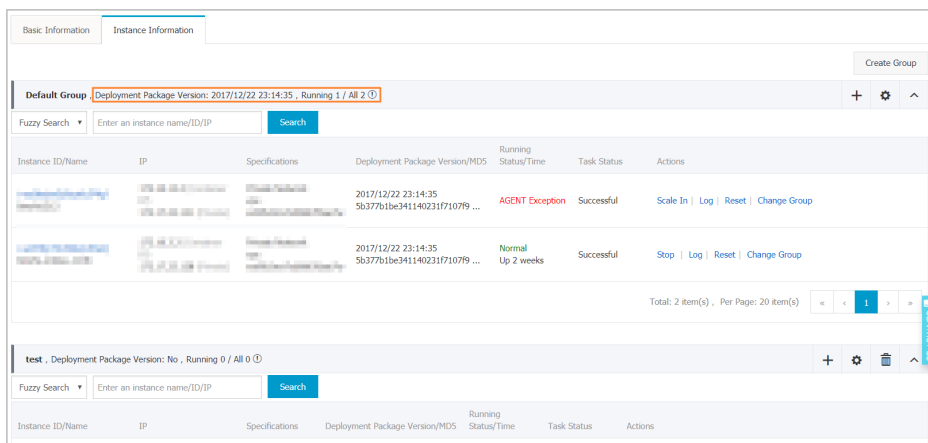
Log on to the EDAS console.

Click **Applications** on the left-side menu bar to go to the Application List page.

In the application list, find the application for which you want to view instance groups. Click the application name to go to the application details page.

On the Application Details page, click the **Instance Information** tab.

Check the instance group information and application package versions across different groups in the application.



The screenshot shows the 'Instance Information' tab in the EDAS console. It displays two instance groups. The first group, 'Default Group', has a deployment package version of '2017/12/22 23:14:35' and is in a 'Running' state with '1' instance. The second group, 'test', has a deployment package version of 'No' and is in a 'Running' state with '0' instances.

Instance ID/Name	IP	Specifications	Deployment Package Version/MDS	Running Status/Time	Task Status	Actions
...	2017/12/22 23:14:35 5a377b1be341140231f71079 ...	AGENT Exception	Successful	Scale In Log Reset Change Group
...	2017/12/22 23:14:35 5a377b1be341140231f71079 ...	Normal Up 2 weeks	Successful	Stop Log Reset Change Group

Total: 2 item(s) , Per Page: 20 item(s)

Create a group

Gated release is often used for the launch of a new application version, so that the new version can be test run without affecting the traffic in the production environment. In this case, you need to

create a new group for the application.

Log on to the EDAS console.

Click **Applications** on the left-side menu bar to go to the Application List page.

In the application list, find the application for which you need to create a group. Click the application name to go to the application details page.

On the Application Details page, click the **Instance Information** tab and click **Create Group** in the upper right corner.

Enter the **Group Name** in the Create Group dialog box, and then click **Create**.

After a group is created, a message **Group created successfully** is displayed in the upper right corner.

Notes about groups:

- A default group is created automatically when an application is created.
- A single package version is available for a single group, and the version information is displayed right after the group name (*Package version: 2017/1/20 15:36:12* as shown in the figure).
- No package version is available for a new group. The package version for a group is always the version of the package last deployed to the group.
- The instance deployment information for an application is displayed by instance group.

Add instances to a group

After a group is created, you can add instances to the new group through **Scale Out** and **Change Group**. The procedure is as follows:

Add instances to a group through **Scale Out**

Click **Scale Out** in the upper right corner of the Application Details page.

Select a **Target Group** and an **Elastic Compute Service (ECS) instance**.

Click **Scale Out**.

Notes about adding instances to a group by scaling up the application

- If no package is ever deployed to a group, no package will be deployed to any

instances added to the group.

- If a package was deployed to a group, the package last deployed to the group will be deployed to the instances added to the group.
- The package version deployed to the group is shown right after the group name.

Add instances to a group through **Change Group**

In the **Instance Information** tab of the Application Details page, select the instance which you want to change the group for, and click **Change Group** to the right of the list.

In the displayed dialog box, select the target group.

If the application package version for the instance is different from that of the target group, choose to use the version for the target group or keep the existing version for the instance.

Click **Change**.

Notes about changing instance groups

- If no package version is available for the target group and a package is already deployed to the instance whose group you want to change, the package version for the instance will be used as the package version for the group.
- Select **Redeploy current instance for target group** to redeploy the deployment package on the instance with that in the group.
- If you select **Change the group only, will not re-deploy**, the deployment status of the instance remains unchanged.
- If the package version for the instance is different from that of the group to which the instance belongs, a prompt is displayed.

Delete Group

A group with no instances in it can be deleted. The delete operation cannot be undone. Please use caution. The procedure is as follows:

On the Application Details page, click the Instance Information tab, and click the **Delete Group** button.

In the dialog box displayed, click the **Delete** button.

Roll back applications

After you upgrade your application, you can roll back to its earlier version if you detect problems in the application.

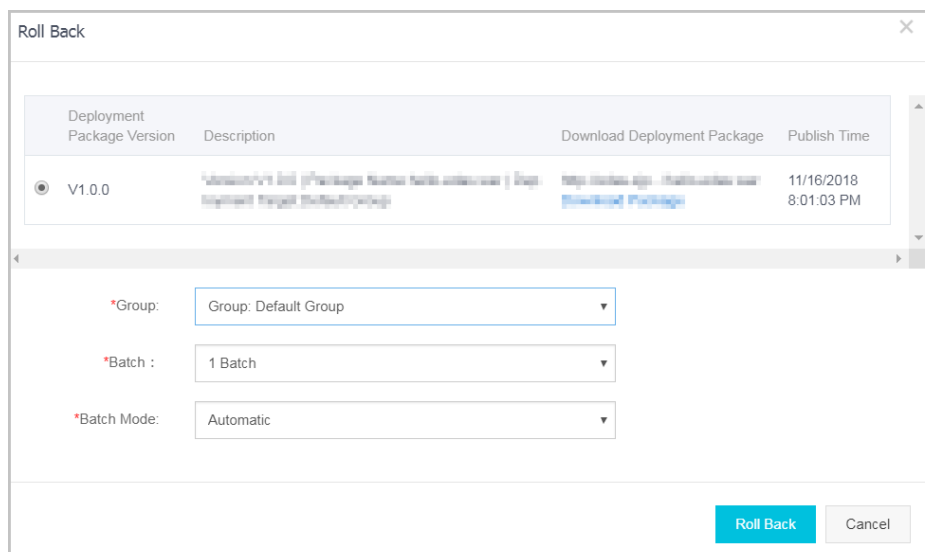
Roll back an application

Log on to the EDAS console.

Click **Applications** in the left-side navigation pane, and click an application on the Applications page to go to the **Basic Information** page.

Click **Roll Back** in the upper-right corner of the Application Details page.

In the **Roll Back** dialog box, select **Deployment Package Version** and **Group**, set **Batch** and **Batch Mode**, and click **Roll Back**.



Deployment Package Version	Description	Download Deployment Package	Publish Time
<input checked="" type="radio"/> V1.0.0	Version V1.0.0 (Package Name: test-application Dep. ...)	Download Package	11/16/2018 8:01:03 PM

*Group:

*Batch :

*Batch Mode:

Delete applications

After an application is deleted, all information related to the application is deleted, all instances under the application are released, and all application WAR packages and container files in the ECS instance are deleted.

Note: Before deleting an application, ensure that logs, WAR packages, and configurations on all instances in the application are backed up.

Delete an application

When an application is deleted, all instances in the application are deleted.

Log on to the EDAS console.

Click **Applications** in the left-side navigation pane, and click an application on the Applications page to go to the **Basic Information** page.

Click **Delete** on the Application Details page.

Delete the application as promoted. After the application is deleted, the **Disablement triggered successfully** message appears in the upper-right corner of the page.

Delete an instance

When an instance is deleted, the WAR packages and container files in this instance are automatically deleted.

Log on to the EDAS console.

Click **Applications** in the left-side navigation pane, and click an application on the Applications page to go to the **Basic Information** page.

Click **Instance Information**. On the Instance Information tab, click **Delete** in the **Action** column.

Delete the application as promoted. After the application is deleted, the **Disablement triggered successfully** message appears in the upper-right corner of the page.

Application in Container Service Kubernetes Cluster

Overview

Kubernetes is a popular orchestration technology for open-source containers. Applications published with Kubernetes poses unique management advantages. For more information, see [Kubernetes documentation](#).

The Container Service Kubernetes clusters provided by Alibaba Cloud have passed CNCF standardized tests and can operate stably while integrated with other Alibaba Cloud products, such as Server Load Balancer (SLB) and Network Attached Storage (NAS). After creating a Kubernetes cluster in Container Service and importing it to Enterprise Distributed Application Service (EDAS), you can deploy applications in the cluster from EDAS. The complete procedure is as follows:

Create a Kubernetes cluster on the Container Service console.

Create a Kubernetes application image based on Container Service.

Publish a Kubernetes application based on Container Service.

After publishing the application, you can scale it up or down as needed.

Create Container Service Kubernetes cluster

Alibaba Cloud provides Kubernetes for Container Service. You can log on to the Container Service console to create a Kubernetes cluster and import it to the EDAS console.

Prerequisites

Make sure the account you use to create the Kubernetes cluster in Container Service has EDAS operation permissions. We recommend that you use the same account for Container Service and

EDAS. If you use different accounts, make sure they belong to the same primary account and have been authorized.

Procedure

1. Log on to the Container Service console to create a cluster

Log on to the Container Service console by using the account with EDAS operation permissions and create a Kubernetes cluster. For more information, see [Create a Kubernetes cluster](#).

You can create a Kubernetes cluster or a Kubernetes hosting cluster in Container Service. Select either type as needed.

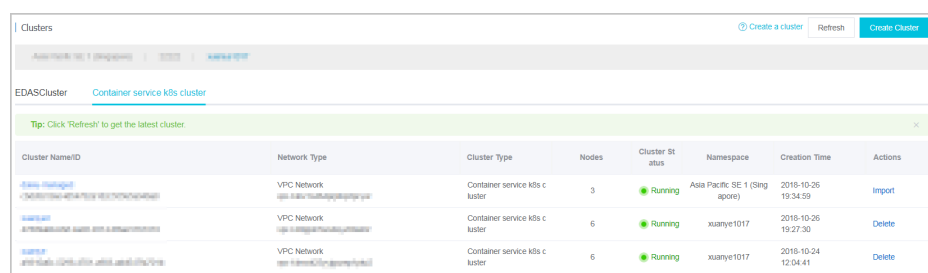
- Kubernetes: Three of the instances that you buy and add must be used as master nodes. Applications cannot be deployed on these three instances. You can only deploy applications on the other instances (workers).
- Kubernetes hosting: All the instances that you buy and add are workers and can be used for application deployment.

2. Synchronize the Kubernetes cluster in Container Service to the EDAS console

Log on to the EDAS console.

On the left-side menu bar, choose **Resources** > **Clusters**.

On the **Clusters** page, click **Container Service K8s Cluster**. Then, click **Import** in the Actions column next to the Kubernetes cluster in the cluster list.



Cluster Name/ID	Network Type	Cluster Type	Nodes	Cluster Status	Namespace	Creation Time	Actions
edas-xxxxxx	VPC Network	Container service K8s cluster	3	Running	Asia Pacific SE 1 (Singapore)	2018-10-26 19:34:59	Import
edas-xxxxxx	VPC Network	Container service K8s cluster	6	Running	xuanyue1017	2018-10-26 19:27:30	Delete
edas-xxxxxx	VPC Network	Container service K8s cluster	6	Running	xuanyue1017	2018-10-24 12:04:41	Delete

Result verification

After the Kubernetes cluster in Container Service is upgraded on the EDAS console, the button in the Actions column changes to **Delete** and the cluster status is **Running** in the Cluster Status column on

the **Container Service K8s Cluster** tab.

Create a image for Container Service K8s application

With EDAS, you can create a custom image and publish a remote procedure call (RPC) application (based on the high-speed service framework [HSF]) to a Kubernetes cluster in Container Service.

Specifications and constraints

Observe the following specifications and constraints when creating a custom image by using a Dockerfile:

Tenant and encryption information

The tenant and encryption information is used for user authentication and credential encryption of EDAS applications.

Resources

Resource type	Resource name	Description
Secret	edas-certs	Encryption dictionary, which stores EDAS tenant information

Environment variables

Environment variable key	Type	Description
tenantId	String	EDAS tenant ID
accessKey	String	Access Key ID for authentication
secretKey	String	Access Key Secret for authentication

Local files

Path	Type	Description
/home/admin/.spas_key/defa	File	EDAS tenant authentication

ult		information, which includes the preceding Env information
-----	--	---

Service information

The service information includes the EDAS domain and port to be connected to during runtime.

Resources

Resource type	Resource name	Description
ConfigMap	edas-envs	EDAS service information

Environment variables

Environment variable key	Type	Description
EDAS_ADDRESS_SERVER_DOMAIN	String	Service domain or IP address of the configuration center
EDAS_ADDRESS_SERVER_PORT	String	Service port of the configuration center
EDAS_CONFIGSERVER_CLIENT_PORT	String	CS service port

Environment variables during application runtime (mandatory)

The following environment variables (Env) are provided during EDAS deployment to ensure the proper running of applications. For this reason, do not overwrite the current configuration.

Environment variables

Environment variable key	Type	Description
POD_IP	String	POD IP
EDAS_APP_ID	String	EDAS application ID
EDAS_ECC_ID	String	EDAS ECC ID
EDAS_PROJECT_NAME	String	Same as EDAS_APP_ID, which is used to parse the call trace
EDAS_JM_CONTAINER_ID	String	Same as EDAS_ECC_ID, which is used to parse the call trace
EDAS_CATALINA_OPTS	String	CATALINA_OPTS parameter, which is required during middleware operation
CATALINA_OPTS	String	Same as

		EDAS_CATALINA_OPTS, which is the default Tomcat startup parameter
--	--	---

Create a custom image

This document explains how to create a custom image. To create a custom image, complete the following steps:

1. Define a standard Dockerfile.
2. Customize settings in the Dockerfile.

Define a standard Dockerfile

A standard Dockerfile defines a runtime environment for EDAS applications, including the methods for downloading, installing, and starting JDK, Tomcat, and WAR or JAR packages.

By modifying the Dockerfile, you can replace the JDK version, modify the Tomcat configuration, change the runtime environment, and make other changes.

The following example explains how to define an EDAS application.

Note: The example will be occasionally updated to incorporate the latest EDAS features.

Sample Dockerfile using Tomcat and a WAR package

```
FROM centos:7
Maintainer: EDAS R&D team <edas-dev@list.alibaba-inc.com>

# Install and package the required software
RUN yum -y install wget unzip

# Prepare JDK and Tomcat system variables
ENV JAVA_HOME /usr/java/latest
ENV CATALINA_HOME /home/admin/taobao-tomcat
ENV PATH $PATH:$JAVA_HOME/bin:$CATALINA_HOME/bin

# Set the EDAS-Container version
ENV EDAS_CONTAINER_VERSION V3.5.0
LABEL pandora V3.5.0

# Download and install JDK 8
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/agent/prod/files/jdk-8u191-linux-x64.rpm -O
/tmp/jdk-8u191-linux-x64.rpm && \
yum -y install /tmp/jdk-8u191-linux-x64.rpm && \
rm -rf /tmp/jdk-8u191-linux-x64.rpm

# Download and install Ali-Tomcat 7.0.85 to the /home/admin/taobao-tomcat directory
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/edas-container/7.0.85/taobao-tomcat-
production-7.0.85.tar.gz -O /tmp/taobao-tomcat.tar.gz && \
```

```

mkdir -p ${CATALINA_HOME} && \
tar -xvf /tmp/taobao-tomcat.tar.gz -C ${CATALINA_HOME} && \
mv ${CATALINA_HOME}/taobao-tomcat-production-7.0.59.3/* ${CATALINA_HOME}/ && \
rm -rf /tmp/taobao-tomcat.tar.gz ${CATALINA_HOME}/taobao-tomcat-production-7.0.59.3 && \
chmod +x ${CATALINA_HOME}/bin/*sh

# Download and install EDAS Container based on environment variables
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/edas-
plugins/edas.sar.${EDAS_CONTAINER_VERSION}/taobao-hsf.tgz -O /tmp/taobao-hsf.tgz && \
tar -xvf /tmp/taobao-hsf.tgz -C ${CATALINA_HOME}/deploy/ && \
rm -rf /tmp/taobao-hsf.tgz

# Download and deploy the EDAS demo WAR package
RUN wget http://edas.oss-cn-hangzhou.aliyuncs.com/demo/hello-edas.war -O /tmp/ROOT.war && \
unzip /tmp/ROOT.war -d ${CATALINA_HOME}/deploy/ROOT/ && \
rm -rf /tmp/ROOT.war

# Set the Tomcat installation directory as the container startup directory, start Tomcat in run mode, and
output the catalina log on the standard command line interface (CLI).
WORKDIR ${CATALINA_HOME}
CMD ["catalina.sh", "run"]

```

Sample Dockerfile using a JAR package

```

FROM centos:7
Maintainer: EDAS R&D team <edas-dev@list.alibaba-inc.com>

# Install and package the required software
RUN yum -y install wget unzip

# Prepare JDK and Tomcat system variables
ENV JAVA_HOME /usr/java/latest
ENV CATALINA_HOME /home/admin/taobao-tomcat
ENV PATH $PATH:$JAVA_HOME/bin

# Set the EDAS-Container version
ENV EDAS_CONTAINER_VERSION V3.5.0
LABEL pandora V3.5.0

# Download and install JDK 8
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/agent/prod/files/jdk-8u191-linux-x64.rpm -O
/tmp/jdk-8u191-linux-x64.rpm && \
yum -y install /tmp/jdk-8u191-linux-x64.rpm && \
rm -rf /tmp/jdk-8u191-linux-x64.rpm

# Download and install EDAS Container to the /home/admin/taobao-tomcat directory based on
environment variables
RUN mkdir -p ${CATALINA_HOME}/deploy/
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/edas-
plugins/edas.sar.${EDAS_CONTAINER_VERSION}/taobao-hsf.tgz -O /tmp/taobao-hsf.tgz && \
tar -xvf /tmp/taobao-hsf.tgz -C ${CATALINA_HOME}/deploy/ && \
rm -rf /tmp/taobao-hsf.tgz

```



```
# Download and deploy the EDAS demo JAR package
RUN mkdir -p /home/admin/app/ && wget http://edas.oss-cn-hangzhou.aliyuncs.com/demoapp/fatjar-test-case-provider-0.0.1-SNAPSHOT.jar -O /home/admin/app/provider.jar

# Include the startup command in the startup script start.sh
RUN echo '$JAVA_HOME/bin/java -jar $CATALINA_OPTS -Djava.security.egd=file:/dev/./urandom -Dcatalina.logs=$CATALINA_HOME/logs -Dpandora.location=$CATALINA_HOME/deploy/taobao-hsf.sar "/home/admin/app/provider.jar" --server.context-path=/ --server.port=8080 --server.tomcat.uri-encoding=ISO-8859-1 --server.tomcat.max-threads=400' > /home/admin/start.sh && chmod +x /home/admin/start.sh

WORKDIR $CATALINA_HOME
CMD ["/bin/bash", "/home/admin/start.sh"]
```

Customize settings in the Dockerfile

The following explains how to customize settings in the standard Dockerfile prepared previously.

Upgrade JDK

Change the download and installation methods in the standard Dockerfile. The following uses JDK 8 as an example.

```
# Download and install JDK 8
RUN wget http://edas-hz.oss-cn-hangzhou.aliyuncs.com/agent/prod/files/jdk-8u191-linux-x64.rpm -O /tmp/jdk-8u191-linux-x64.rpm && \
yum -y install /tmp/jdk-8u191-linux-x64.rpm && \
rm -rf /tmp/jdk-8u191-linux-x64.rpm
```

Upgrade EDAS Container

When using a WAR package and Tomcat, upgrade EDAS Container to use new middleware features or fix known bugs. The upgrade procedure is as follows:

1. Log on to EDAS console, and click **Applications** in the left-side navigation pane to go to the application list page. Retrieve the latest EDAS-Container version (3.X.X). Click **Create Application** in the upper-right corner of the Applications page. In the **Application Information** page, Retrieve the latest EDAS-Container version.
2. Replace the version number in the Dockerfile, such as 3.5.1.
3. Re-create and publish an application image.

```
# Prepare ENV
ENV EDAS_CONTAINER_VERSION V3.5.1
```

Add the EDAS runtime environment to Tomcat startup parameters

For the environment variables, see [Environment variables during application runtime](#). EDAS provides the JVM environment variable 'EDAS_CATALINA_OPTS' , which contains the minimum parameters required during runtime. Tomcat provides the custom JVM parameter configuration option 'JAVA_OPTS' for setting Xmx, Xms, and other parameters.

```
# Set the JVM parameters of the EDAS application
ENV CATALINA_OPTS ${EDAS_CATALINA_OPTS}
# Set the JVM parameters
ENV JAVA_OPTS="\
-Xmx3550m \
-Xms3550m \
-Xmn2g \
-Xss128k"
```

Publish Container Service Kubernetes applications

Currently, EDAS only allows you to publish Container Service Kubernetes applications by using [images](#).

Prerequisites

Refer to [Create a Kubernetes application image based on Container Service](#) to create an image and upload it to the image repository.

Create an application

Log on to the EDAS console.

Click **Applications** in the left-side navigation pane to go to the Application page.

On the Applications page, select **Region** and click **Create Application** in the upper-right corner.

On the **Basic Information** tab, set the basic information about the application. Then click **Next Step: Application Configuration**.

Field description:

- **Namespace:** Select **Region** and **Namespace** from the drop-down list. You can select **Default** for **Namespace**.
- **Cluster:** Select a **Kubernetes (from Container Service)** cluster from the drop-down list.
- **Type:** The application type is determined by the cluster where the application is deployed. If you select a Kubernetes cluster from Container Service, the application type is Kubernetes application. This parameter cannot be set manually.
- **Name:** Enter the application name.
- **Description:** Enter the basic information about the application.

in the **Application Configuration** page, configure an image.

Select **Image** as the **Deployment Method**.

Select an image in **My Image**.

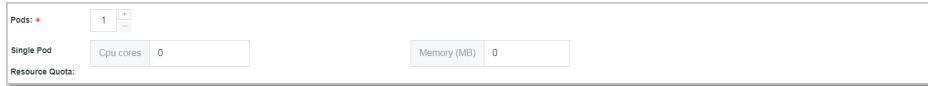
NOTES:

If you are publishing a Kubernetes application from Container Service for the first time, no images will be available in the image market. - Refer to **Create a Kubernetes application image based on Container Service** to create an image, and then upload the local image.

You cannot pull images across regions.

Set pods.

Pods are the smallest units for deploying an application. An application can contain multiple pods. In a Server Load Balancer instance, a request is randomly allocated to a pod for processing.



Set Pods.

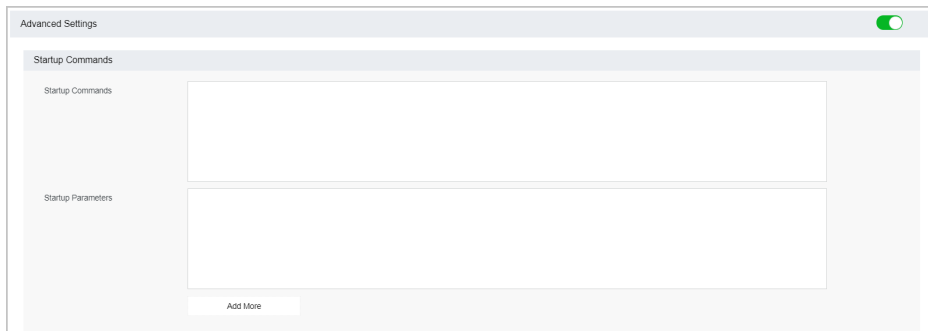
When pods fail to run or encounter faults, they can automatically restart or quickly migrate to ensure a high availability for applications. For stateful applications using persistent storage, instance data can be saved. For redeployed stateless applications, instance data is not saved.

Set the Single Pod Resource Quota.

No quota is set by default. Therefore, both the values of **CPU Cores** and **Memory** of a single pod are 0. To set the quota, enter a number.

Set the startup command and parameters.

Note: We recommend that you do not modify the custom startup command and parameters if you are not familiar with the **CMD** and **ENTRYPOINT** of the original Dockerfile. An incorrect custom command could lead to an application creation failure.



Startup Commands: Enter only the content within [""]. For example, enter `/usr/sbin/sshd -D` for the command `CMD ["/usr/sbin/sshd", "-D"]`.

Startup Parameters: Enter one parameter per line. For example, for the command `args:["-c"; "while sleep 2"; "do echo date"; "done"]`, which contains four parameters, enter those parameters in four lines.

Set environment variables.

When creating the application, inject the entered environment variables in the container to be generated. This saves you from having to repeatedly add common environment variables.

The screenshot shows a configuration window titled "Environment Variables". It contains a table with two columns: "Name" and "Value". Below the table is an "Add More" button. There is a red minus sign in the top right corner of the table area.

If you are using a MySQL image, refer to the following environment variables:

- `MYSQL_ROOT_PASSWORD` (required) allows you to set a root password for MySQL and is required.
- `MYSQL_USER` and `MYSQL_PASSWORD` (optional) allows you to add an account besides the root account and set a password.
- `MYSQL_DATABASE` (optional) allows you to set the database to be created when the container is generated.

If you are using another type of image, configure the environment variables as needed.

Set persistent storage.

In the Container Service Kubernetes cluster, the native Volume object corresponds to non-persistent physical storage, whose lifecycle is the same as the Kubernetes pods, which are a transient storage object. **Network Attached Storage (NAS)** is a persistent storage service that can permanently store instance data and ensure data integrity after application upgrade or migration.

Note: Before configuring persistent storage, ensure that the NAS service is active for your EDAS account. Because the NAS billing method is Pay-As-You-Go, ensure that your account has a sufficient balance or adopts post payment.

The screenshot shows a configuration window titled "Persistent Storage". It has a toggle switch in the top right corner. Below the title, there are two dropdown menus: "Storage Type: NAS" and "Storage Service Type: SSD (Perform...)". Under "Select NAS", there are two radio buttons: "Buy a new NAS" (selected) and "Use an existing NAS". A yellow banner with "NAS Pricing" and a close button is visible. Below that, there are two input fields for "Mount Directory": "NAS Mount Directory" (containing "/") and "Local Mount Directory" (containing "/mnt"). An "Add More" button is at the bottom.

Description of persistent storage parameters:

- **Storage Type:** This parameter defaults to NAS and cannot be set manually.
- **Storage Service Type:** Currently, this parameter only supports SSD (Performance-Type) and cannot be set manually.

Select NAS:

Buy a new NAS: Select a NAS mount directory and a local mount directory.

A single region supports up to 10 NAS instances. Once you have 10, you cannot create any more. If you must create more instances, open a ticket.

Use an existing NAS: Select an existing NAS instance.

You can create up to two mount points. Only compliant NAS instances are displayed in the drop-down list.

Set host machine storage.

You can map part of the file system of the host machine to the container as needed. Before using this feature, read this document to determine if this is the correct solution.

Description of file types:

Parameter	value	Description
Default	Null string	Mount directly without checking the type.
(New) File directory	DirectoryOrCreate	File directory; a new directory is created if it does not exist.
File Directory	Directory	File directory; container startup fails if it does not exist.
(New) File	FileOrCreate	File; a new file is created if it does not exist.
File	File	File; container startup fails if it does not exist.
Socket	Socket	Standard Unix Socket file; container startup fails if it does not exist.
CharDevice	CharDevice	Character device file; container startup fails if it does not exist.
BlockDevice	BlockDevice	Block device file; container startup fails if it does not exist.

Note: You do not need to concern yourself with the Value column in this step. However, the Value column may be used by APIs after the application is created.

Set the lifecycle management script of the application (**for stateful applications**).

Kubernetes applications take either of the following states:

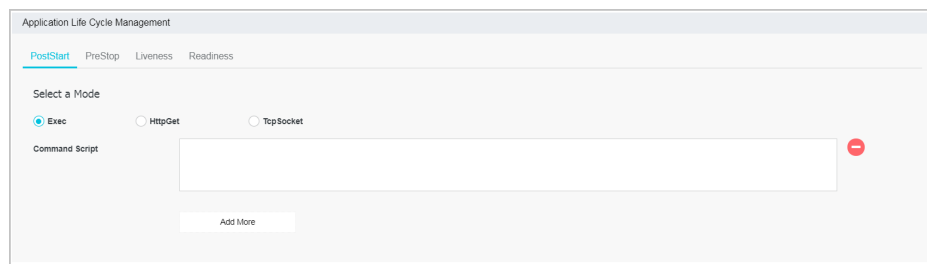
Stateless: A stateless application supports multi-replica deployment. When a stateless application is redeployed, instance data is not saved. A stateless application can be:

- A web application that does not retain instance data during upgrade or migration.
- An application that can be scaled up horizontally to address changing service volumes.

Stateful: A stateful application stores data that requires persistent storage and retains instance data during upgrade or migration. A stateless application can be:

- An application that frequently operates on containers through SSH.
- An application that is used for persistent data storage (such as the database application MySQL) or that supports inter-cluster election and service discovery, such as ZooKeeper and etcd.

You can set lifecycle management for a stateful application as needed.



Description of the lifecycle management script:

PreStop: a container hook, which is triggered before a container is deleted. The corresponding hook handler must be completed before the container deletion request is sent to Docker daemon. Docker daemon sends an SGTERN semaphore to itself to delete the container, regardless of the hook handler execution result. For more information, see [Container Lifecycle Hooks](#).

Liveness: a container status probe, which monitors the health status of applications. If an application is unhealthy, the container is deleted and created again. For more information, see [Pod Lifecycle](#).

Readiness: a container status probe, which monitors whether applications are started successfully and are running properly. If an application is abnormal, the

container status is updated. For more information, see [Pod Lifecycle](#).

Poststart: a container hook, which is triggered immediately after a container is created to notify the container of its creation. The hook does not transfer any parameters to the corresponding hook handler. If the corresponding hook handler fails to run, the container is killed and the system determines whether to restart the container according to the container's restart policy. For more information, see [Container Lifecycle Hooks](#).

After completing the settings, click **Next Step: Application Access Settings**.

(Optional) On the **Application Access Settings** page, enable **SLB Settings** and set **Intranet SLB Port**, **Internet SLB Port**, **Container Port**, and **Network Protocol**.

SLB corresponds to TCP/UDP settings. You can configure multiple port mappings for multi-port listening.

- **Intranet SLB:** This option ensures that all the nodes in a VPC can access the application.
- **Internet SLB:** After you enable this option, the system buys an internet SLB instance for the application to ensure that the application is accessible from the Internet.

Note: SLB is billed on a Pay-As-You-Go basis, and you can view information about the bought SLB instance on the SLB console.

Description of SLB parameters:

- **SLB Port:** This parameter indicates the frontend port of the intranet or internet SLB instance, which is used to access the application. For example, Nginx uses port 80 by default.
- **Container Port:** This parameter indicates a port that listens to processes. It is generally defined by the program. For example, the web service uses port 80/8080 by default, while the MySQL service uses port 3306 by default. The container port can be the same as the port used by the SLB instance.
- **Network Protocol:** You can select TCP or UDP.

After completing the settings, click **Create**.

Result verification

Creating an application can take up to several minutes. During the creation process, you can view the application publishing order to track the creation progress. After the application has been created, return to the Application Details page and check the basic information of the application on the Application Information tab.

Scaling Container Service Kubernetes Applications

Compared with common applications and Docker applications, Kubernetes applications feature much greater scalability due to the advantages of Kubernetes in container orchestration.

Log on to the EDAS console and click **Applications** on the left-side menu bar.

On the **Applications** page, click a Container Service Kubernetes application.

Click **Application Scaling** in the upper-right corner of the Application Details page.

In the Application Scaling dialog box, set the number of pods to be added or removed and click **Confirm**.

If the number of pods is 0, the system physically deletes all pods under the application and retains only basic creation information of the application.

After configuration, scale-up or scale-down of the Kubernetes application is automatically completed in the cluster without manual operations.

Application monitoring

Application monitoring overview

Application monitoring accurately reflects the real-time traffic and history information of an application, allowing you to monitor application health and quickly discover and locate problems.

Terminology

TraceId: Corresponds to a request. It is globally unique and transmitted between systems.

IP Addresses: Indicates the IP address (in hexadecimal format) of the ECS instance that creates the TraceId.

Creation Time: Indicates the time for link creation.

Order: It is used for link sampling.

Flag Bit: (Optional) It is used for debugging and marking.

Process ID: (Optional) It is used for single-host multi-process applications.

RpcId: Calls and flags the log track order and nesting relationship. It is transmitted between systems.

Service Dimension: Service data is monitored in application and service dimensions. Data in the application dimension is aggregated by application, while data in the service dimension is aggregated by custom service. For example, you have an application A that provides services a, b, and c.

Drill Down: Views metrics of upstream (downstream) applications associated with the target metric.

Types of metric data

Tabs of different data types are available on the Service Monitoring page, allowing pertinent monitoring.

- **Provided RPC Service:** Displays the RPC services (including the HSF and other custom services) provided by an application as the server.
- **RPC Call Source:** Displays records of the following applications calling the RPC service

provided by the current application.

- **RPC Call Dependency:** Displays records of the current application calling RPC services (including HSF and other custom services) provided by other applications.

Types of monitoring reports

- **Mix of Graph and Table (Default):** Displays data in “table + graph” form, including monitoring target, time, QPS, response time, server response time, errors, and results. By default, the graph shows data for the last hour, and the table shows data for the last five minutes.
- **Multi-graph:** Displays data in graphs, including monitoring target, time, QPS, response time, errors, and results. By default, the graphs show data for the last hour with the latest data separately listed.
- **Table:** Displays data in a table, including the monitoring target, QPS, response time, errors, and results. Data for the last minute is displayed.

Metric description

- **Error/s:** Records the rate of RPC errors per minute, which is the total number of errors within the minute divided by 60.
- **Result/s:** Records the returned result in the format of “Result: QPS” , where “Result” indicates the RPC result. The HTTP result is consistent with the HTTP ErrorCode.

Install log collector

Log collector overview

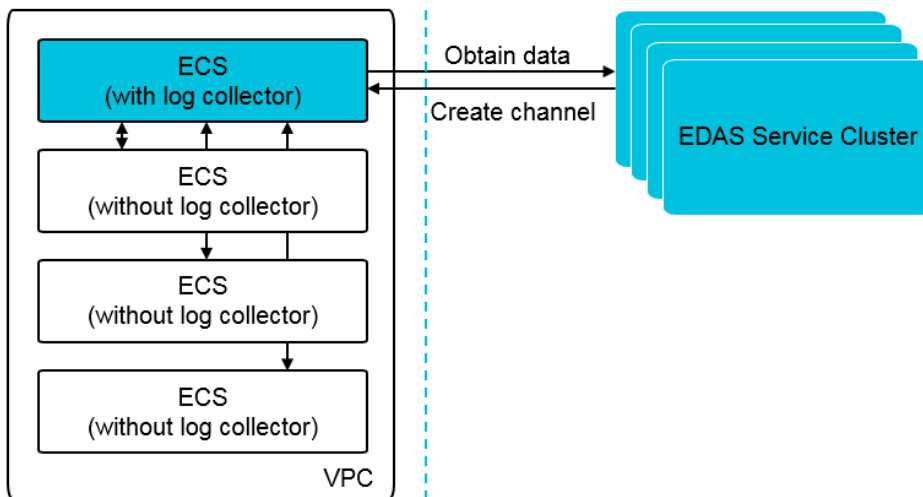
Install a log collector before using the application monitoring function of EDAS.

EDAS provides a suite of functions, where a lot of data is pulled from local machines. This requires that the server can be connected to the relevant machines.

Alibaba Cloud’s network environment consists of classic networks and VPCs.

- In a classic network, if the firewall and security groups have no port (8182) restrictions, the server can be connected directly.
- In a VPC, the machines are intrinsically isolated from the server. EDAS provides a special utility for VPCs: log collector.

The log collector has two editions: Server and Client. SProxy is a log collector client installed on the instance.as shown below:



Install a log collector

To implement the solution shown above, you must first install the log collector on an ECS instance (ECS A in this example) on the VPC. The installation procedure is as follows:

Log on to the EDAS console.

On the left-side navigation bar, choose **Resources > VPC**.

Switch to the region that contains ECS A and, in the VPC list for this region, find ECS A's VPC ID. Then, click **Install Log Collector** in the **Actions** field.

In the dialog box, copy the script for installation.

Log on to the ECS instance where you want to install EDAS Agent as a **root**. Log on to the instance A, paste the copied script for installation, and press **Enter**.

After the installation is complete, manually run the `netstat -ant|grep 8002` Command.

If a connection is established, this indicates that the log collector was installed successfully.

If no connection is established, this indicates a problem with the installation. In this case, open a ticket to consult with us.

Dashboard

Based on different groups, the monitoring dashboard displays the overall metrics related to provided services, service consumption, and infrastructure monitoring using charts.

Service provided: Displays information about RPC services and HTTP services provided by the application.

Service consumption: Displays the database access metrics.

Infrastructure monitoring: Displays the metrics about CPU, load, memory, disk, and network.

Follow these steps to view the monitoring dashboard.

Log on to the EDAS console.

Click **Applications** on the left-side menu bar.

In the application list, click the application name which you want to view information about.

On the application details page, select **Application Monitoring** > **Dashboard** from the left-side menu bar.

The page shows information about the service provided, service consumption and infrastructure monitoring.

Hover the mouse over a point on an abscissa of a monitoring chart to view the information and status data at a specific time point.

Click a project name, "RPC Service" for example, at the top of a monitoring chart to switch to the service monitoring page and view details. For details about the monitoring parameters, see [Application monitoring overview](#).

Infrastructure monitoring

EDAS collects data from the ECS instances that run applications and provides the single-instance and cluster views of the CPU, memory, load, network, and disk metrics based on the analysis results. Data in all monitoring views is collected and processed from the application point of view.

Note: A latency exists from data collection to analysis. Therefore, EDAS cannot provide exactly real-time monitoring views. The current latency is about two minutes.

Perform the following steps to view cluster or single-instance statistics:

Log on to the EDAS console, click **Applications**, and click an application in the application list.

On the application details page, select **Application Monitoring > Infrastructure Monitoring** on the left-side menu bar to go to the basic monitoring page.

On the basic monitoring page, group data in the latest half an hour is monitored by default.

You can select a time range to monitor group data for a different time range, or select the **Single Instance Data** tab to see single-instance data.

Select a type of monitored data.

Data to be monitored includes group data and single-instance data.

The following metrics of the two data types are monitored from different dimensions:

CPU data: Indicates the CPU usage, which is the sum of the user usage and system usage. The group data graph displays the average value of the usage of all ECS instances in the application group.

Memory data: Indicates the total size and actual usage of the physical memory. The group data graph displays the total size and total usage of all ECS instances in the application group.

Load data: Indicates the “1 min load” field in the system load. The group data graph displays the average value of “1 min load” of all ECS instances in the application group.

Network speed data: Indicates the write/read speed of the network card. If an ECS instance contains multiple network cards, the data indicates the sum of write/read speeds of all network cards whose names start with “eth” . The group data graph displays the average value of all ECS instances in the application group.

Disk data: Indicates the total size and actual usage of all disks attached to the ECS instance. The group data graph displays the total size and total usage of all ECS instances in the application group.

Disk read/write speed: Indicates the sum of the read/write speeds of all disks attached to the ECS instance. The group data graph displays the average value of the data of all ECS instances in the application group.

Disk read/write numbers: Indicates the sum of the read/write per second of all disks attached to the ECS instance. The group data graph displays the average value of the data of all ECS instances in the application group.

Set the time range.

You can set the time range to "Half an Hour" , "Six Hours" , "One Day" , or "One Week" .

Half an Hour: Collects monitored data in last half an hour by default when you log on to the infrastructure monitoring page. In this statistical period, data is collected every minute, which is the finest query granularity by EDAS.

Six Hours: Collects monitored data in last six hours. In this statistical period, data is collected every five minutes.

One Day: Collects monitored data in last 24 hours. In this statistical period, data is collected every 15 minutes.

One Week: Collects monitored data in last seven days. In this statistical period, data is collected every one hour, which is the longest statistical cycle provided by EDAS.

Note:

Start Time and **End Time** on the page indicate the time span of the current view. When you set one of them, the other one is automatically updated. For example, if you select "30 minutes" and set "End Time" to "2016-05-20 12:00:00" , "Start Time" automatically changes to "2016-05-20 11:30:00" .

Monitor data is automatically updated based on the selected interval.

(Optional) Click "Zoom In" under a metric to view the enlarged graph of the metric, and adjust the time range in the enlarged graph.

Service monitoring

By collecting and analyzing tracked logs of the various middleware services in the network calls, you can obtain the call traces of a specific request across systems. This helps sort out application request entrances, service call initiators and dependencies, and helps you to analyze system call bottlenecks, estimate capacity, and quickly locate exceptions.

Monitor a service

Log on to the EDAS console.

Click **Applications** in the left-side navigation pane.

Click the name of the application in the application list.

On the application page, select **Application Monitoring** > **Service Monitoring** from the left-side navigation pane.

The service monitoring page contains the following tabs:

- **RPC Call Overview:** Displays the call records of the RPC service provided by the current application.
- **RPC Call Source:** Displays the applications that call the RPC service provided by the current application.
- **RPC Call Dependency:** Displays the applications whose services are called by the current application.

(Optional) Set the monitoring conditions, and click **Update** to refresh monitor data.

Latest: Displays data at the current time by default. Select a period from the drop-down list.

Sort by: Sorts data by QPS by default. Select an option from the drop-down menu to sort data by the elapsed time or errors/s (average QPS errors per minute).

Results: 10 records are displayed by default. Select the number of results to be displayed from the drop-down menu. Options are 1, 5, 30, 50, 100, and unlimited.

Display: Results are displayed in blocks by default. You can also set the display mode to chart or table.

View monitor data.

For details about the metrics, see [Application monitoring overview](#).

Click a metric of a column in the monitoring graph. The custom query page is displayed.

In the Metrics area, select metrics to view data of different groups.

View traces

When monitoring a service, you can monitor the service call trace between the application and other applications. You can also view detailed call traces.

In the monitoring graph, click **View Trace** next to a calling or called service to go to the **Trace Query** page.

On the trace query page, you can view the call trace between the application and the calling/called service.

For details about how to query traces, see [Trace query](#).

Monitor a drilled-down application

On the service monitoring page, besides querying call traces related to an application, you can drill down to view monitor data about interdependent applications.

On the **RPC Call Overview**, **RPC Call Source**, or **RPC Call Dependency** tab, click **Source Application**, **Called Service**, or **Calling Service** next to **Drill Down** at the top of the monitoring graph. The monitoring page of the drilled down application is displayed.

Monitor data of the drilled down application.

The method for monitoring data of a drilled-down application is the same as that for monitoring the current application.

Log directories

The EDAS console allows you to view application runtime logs without having to log on to the server. When an exception occurs, you can check logs to troubleshoot the problem.

Follow these steps to view a runtime log:

Log on to the EDAS console. Click **Applications** in the left-side navigation pane.

On the **Applications** page, click the name of the application you want to check to go to the application details page.

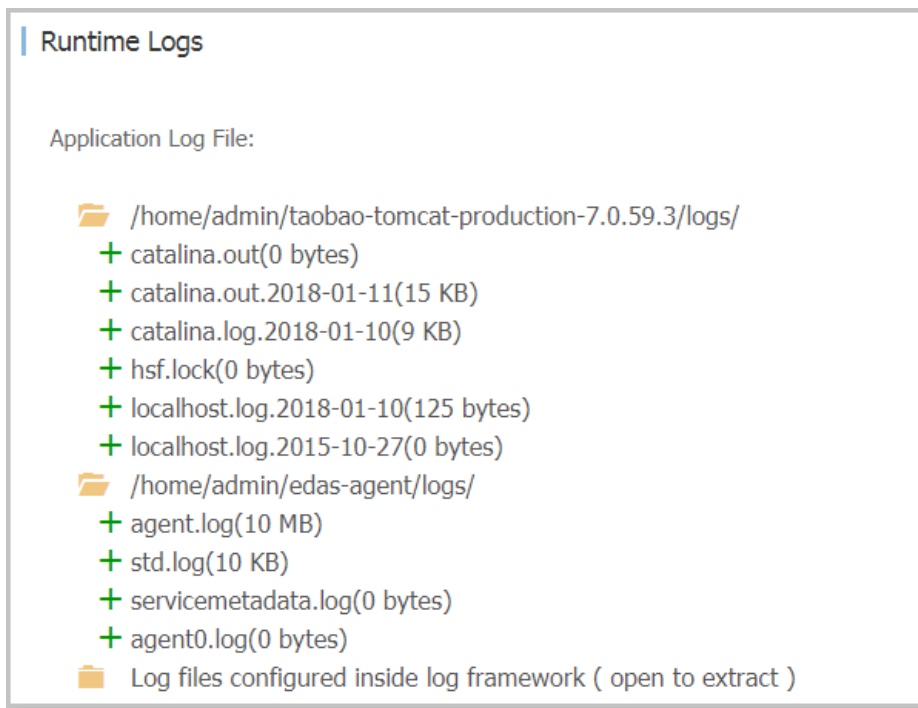
In the left-side navigation pane, select **Logs > Log Directories**. Alternatively, you can also go to the **Instance Information** tab of the application details page, and click **Logs** in the **Actions** column.

On the **Log Directories** page, 3 log directories are displayed by default:

Tomcat container logs directory: The specific paths for Tomcat container logs depend on its actual version.

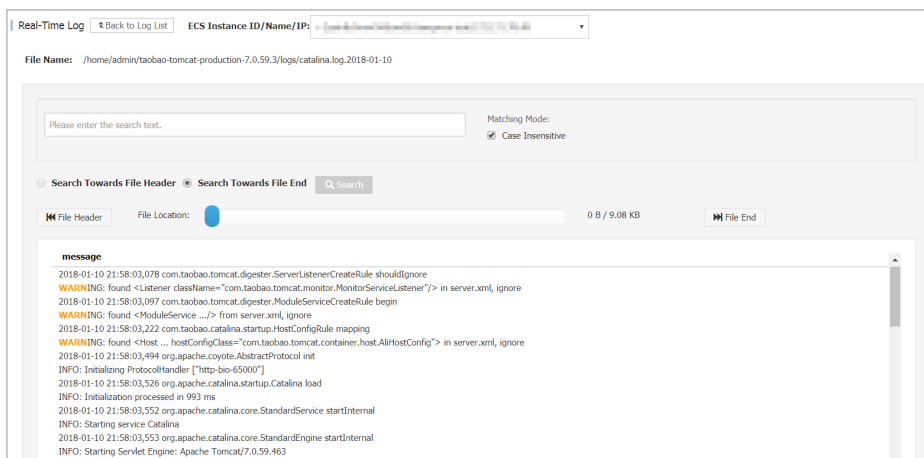
EDAS Agent logs directory.

Log files for log framework configurations.



Note: Only readable files are displayed in the file directory. No folders are displayed.

Double-click a log file to view the details of the log.



At the top of the page, select an instance ID or name in the drop-down list next to **ECS Instance ID/Name/IP Address** to view the details.

At the bottom of the page, click **Enable Real-Time Additions** to keep loading the latest additions to the log file (similar to the tailf command).

In addition to checking the logs in the default path, you can also add log paths to favorites for later viewing, or remove a path from your favorites.

Bookmark a directory

On the **Log Directories** page, click **Bookmark Log Directory** to add a log directory.

Note: This path must be under the /home/admin directory, and must contain wordings "log" or "logs" in the complete path. The file must end with a slash "/" to indicate that it is a folder.

Remove from bookmark

On the **Log Directories** page, click to select a folder name. Then click **Remove Directory from Bookmark** at the top right corner of the page. The path will no longer be displayed on the page. This operation does not delete or change any files on the server.

Alarm and notification

EDAS provides the alarm function to notify users of online problems when resource usage exceeds the limit. Based on policies configured by users and data collected in the background, EDAS checks whether the resource usage limit is exceeded. If the limit is exceeded, a text message or an email is sent to specified contacts.

Note: Currently, EDAS only provides SMS and email notification and does not support custom notification.

Configure alarm policies

Follow these steps:

Log on to the EDAS console, select **Applications** in the left-side navigation pane, and click the name of the application from the application list.

Select **Alarm and Notification > Alarm Rules** in the left-side navigation pane and click **Create Rule** in the upper-right corner.

Enter relevant information in the **Create Rule** page.

*Rule Name: Rule names must only contain numbers, letters and underscores.

*Monitoring Target:

Monitoring Metrics	Compare	Threshold	Actions
CPU Usage	>	%	

+ Add Monitor Items

*Trigger Conditions: Any One o

*Statistical Cycle: 5 Minutes

*Retries Before Alarm: 1

OK Cancel and Return to Rule List

Field description:

Rule Name: Name of the alarm rule, which can contain numbers, letters, and underscores (_). Use an understandable name.

Monitoring Target: Create comparison policies based on metrics (basic monitoring, HTTP, HSF, and application container) and the configured threshold. You can add more than one target as needed.

Trigger Conditions: Select **Any One of the Indicators** or **All Indicators**.

- **Any One of the Indicators:** An alarm is triggered when any of the indicators of the monitored object meets the alarm rules.
- **All Indicators:** An alarm is triggered when all of the indicators of the monitored object meet the alarm rules.

Statistical Cycle: It can be set to 1 minute, 5 minutes, 15 minutes, 30 minutes, or 1 hour. A false alarm might be generated when the system encounters transient jitter, for example, when CPU usage is high during service startup but recovers to the normal range within 2 minutes. To avoid false alarms, you can select a statistical period to allow alarm trigger only when **alarm rules are continuously satisfied** within this period. For example, if you select the 5-minute statistical period for the metric "CPU usage above 30%", then EDAS determines an exception occurs when the CPU usage of the system exceeds 30% for 5 **consecutive** minutes.

- **Retries Before Alarm:** The number of consecutive statistical cycles when alarm policies are satisfied that are required to trigger an alarm. Optional values include 1, 3, and 5.

Click **OK**.

Alarm rules take effect once created. To disable an a rule, select it from the rule list and click

“Delete” . The rule is obsolete immediately.

Add alarm contacts

Follow these steps:

1. Log on to the EDAS console, click **Applications** on the left-side menu bar, and click the name of the application in the application list.
2. Select **Notification Alert > Alarm Contacts** on the left-side menu bar and click **Add Alarm Contacts** in the upper-right corner.
3. Select the contact from the contact list and click **OK**.

Note:

- **Alarm Contact Source:** You can configure to send alarms to the contacts that have a primary and sub-account relationship with the current account. Details are as follows:
 - Other Alibaba Cloud accounts that are bound as sub-accounts to the primary account
 - RAM sub-accounts with EDAS logon history
- **Contact Info (Email Address and Mobile Number):** By default, emails and contacts are obtained from Alibaba Cloud. For privacy protection, the contact information is only available after logon. If you want to receive notifications using a mobile number other than the one registered at Alibaba Cloud, make modifications on **Personal Information** page.

Add employees as alarm contacts

To add an employee that has never used EDAS as an alarm contact, following these steps. Assume that the current EDAS primary account is master@alibabacloud.com and the account to be added is employee@company.com:

Add a RAM sub-account:

Log on to the Alibaba Cloud console with the account master@alibabacloud.com and select **Products & Services > RAM** to go to the RAM Console.

Click **Users** in the left-side navigation pane to go to the RAM sub-account page, click **Create User** in the upper-right corner, and fill in employee information to create a sub-account (assume that the employee name is “employee”).

Log on to EDAS with the sub-account and modify information.

Log on to Alibaba Cloud with the employee RAM sub-account by clicking the link

provided in RAM, and select **EDAS** to go to the EDAS console.

Select **Accounts > Personal Information** in the left-side navigation pane and enter your mobile number and email address.

After relevant information is modified, follow the steps in the **Add alarm contacts** section to add the employee to the alarm contact list.

View alarm records

After an alert is generated, the system sends the alert to contacts while recording the alert.

1. Log on to the EDAS console, click **Applications** in the left-side navigation pane, and select the expected application from the application list.
2. Select **Alarm and Notification > Alarm Record** on the left-side menu bar.

Alarm records from the past 10 days are displayed. After an alarm is cleared, a notification is generated and sent to contacts by means of text message and email.

Configuration push

Configuration push in EDAS includes global configuration push and intra-application configuration push based on different permission control. Global configuration push is targeted at all configurations of a specific user, whereas intra-application configuration push is targeted at the configurations of a specific application.

This topic describes intra-application configuration push.

Configuration in EDAS consists of Group, Data ID, and Content.

Group: Name of a group, which is created in **Service Group** and used to isolate services in a namespace. for example, a package in Java. The maximum length allowed is 128 characters.

Data ID: Configuration name, for example, a class name in Java. The configuration name contains a maximum of 256 characters.

A piece of configuration is identified by group and Data ID collectively and corresponds to a value. The names of group and Data ID can contain four types of special characters: period (.), colon (:), hyphen (-), and underscore (_).

Content: Configuration value, which may contain a maximum of 1,024 characters.

You can add, modify, and delete configurations in real time, and apply configurations dynamically without the need to modify code, republish services, or restart services.

Note: The **group** element of configuration push is created in a service group. If no service is created, the configuration page displays the system configuration that is automatically generated. You can ignore this configuration.

Configuration push

Log on to the EDAS console.

Select **Applications** in the left-side navigation pane.

Click an application name on the **Applications** page.

Click **Configurations** in the left-side navigation pane of the **Application Details** page.

On this page, you can create, view, update, and delete configuration push within the selected application.

Listen for configuration changes

After you create or update configurations in the EDAS console, you can enable configuration listening in code to keep updated with configuration changes.

Introduce the following dependency to code:

```
<dependency>
<groupId>com.alibaba.edas.configcenter</groupId>
<artifactId>configcenter-client</artifactId>
<version>1.0.2</version>
</dependency>
```

Sample code:

```
import java.io.IOException;
import com.alibaba.edas.configcenter.config.ConfigChangeListener;
import com.alibaba.edas.configcenter.config.ConfigService;
public class ConfigCenter {
// Properties/Switch
private static String config = "";
```



```
//Add a configuration listener during initialization
private static void initConfig() {
    ConfigService.addListener("YourDataId", "YourGroup",
    new ConfigChangeListener() {
    public void receiveConfigInfo(String configInfo) {
    try {
    //Obtain the new configuration after configuration is updated
    config = configInfo;
    System.out.println(configInfo);
    } catch (Exception e) {
    e.printStackTrace();
    }
    }
    });
}

public static void main(String[] args) throws IOException {
    // This class is equivalent to the init method if Spring is used.
    initConfig();
    // Prevent the main thread from exiting during the test. If the main thread exits, the daemon thread for
    configuration subscription also exits.
    while (true) {
    try {
    Thread.sleep(1000);
    } catch (InterruptedException e) {
    }
    }
}

// Use the GET API to expose the configuration value for external use
public static String getConfig() {
    return config;
}
}
```

Auto Scaling

To ensure the service quality and availability of a distributed cluster, an important O&M capability is to detect the status of each server in the cluster and to scale in or out in real time based on the system load.

EDAS provides the auto scaling function to automatically scale in or out a cluster based on the CPU, RT, and Load of the servers in the cluster.

Metric description:

- **CPU:** The CPU utilization of the server in percentage. If multiple servers exist in the application, the average value of all servers is used.

- **RT:** The time for the system to respond to a request, in ms.
- **Load:** The system load, which is a positive integer.

All these metrics are entered in positive integers without floating point numbers. If multiple servers exist in the application, the average values of all servers are used for all the preceding metrics.

Auto scaling includes automatic scale-in and scale-out, for which the rules can be configured separately.

Automatic scale-out

Log on to the EDAS console.

Select **Applications** in the left-side navigation pane, and click the name of the application.

Select **Auto Scaling** > **Scaling Rules** from the left-side menu bar of the application details page.

Select **Scale Out Rule** to enable scale-out rule configuration.

Configure the scale-out rules.

Trigger Indicators: Set CPU, RT, and Load.

Trigger Conditions:

- **Any One of the Indicators:** Automatic scale-out is implemented if any of the set indicators is triggered.
- **All Indicators:** Automatic scale-out is implemented only when all of the set indicators are triggered.

Duration: The period in which the indicator is continuously triggered by minute. If the average value in a minute continuously reaches the set threshold in this period, automatic scale-out is implemented. You can configure this parameter based on the sensitivity of the cluster service capabilities.

Number of Instances for Each Scale-Out: The number of servers automatically added after each scale-out operation is triggered. You can configure this parameter based on the service capabilities of a single server of the application.

Maximum Number of Instances: When the number of servers in the cluster reaches

this threshold, scale-out is not implemented. You can configure this parameter based on your resource quota.

Automatic scale-in

The method for configuring rules of **Automatic scale in** is the same as that for configuring rules of **Automatic scale out**. For details about the indicator meanings and setting methods, see **Automatic scale out**.

Note: When the scale-in and scale-out rules are configured simultaneously, the indicator values of the scale-in rules cannot be larger than those of the scale-out rules. Otherwise, an error message is displayed when you click **Save**.

View auto scaling results

After the auto scaling rules are set, if automatic scale-out or scale-in is implemented, use any of the following methods to view the auto scaling results:

On the application details page, select **Instance Information** tab to check whether the number of instances is increased or reduced.

On the application details page, select **Auto Scaling > Scaling history** from the left-side navigation pane to view the scale-out and scale-in history records.

Rate limiting and degradation

Rete limiting and degradation overview

The throttling and downgrade feature of EDAS solves slow system responses or crashes caused by high pressure of the backend core service. This feature is generally used in high-traffic scenarios, for example, flash sale, shopping spree, great promotion, and anti-empty box scams.

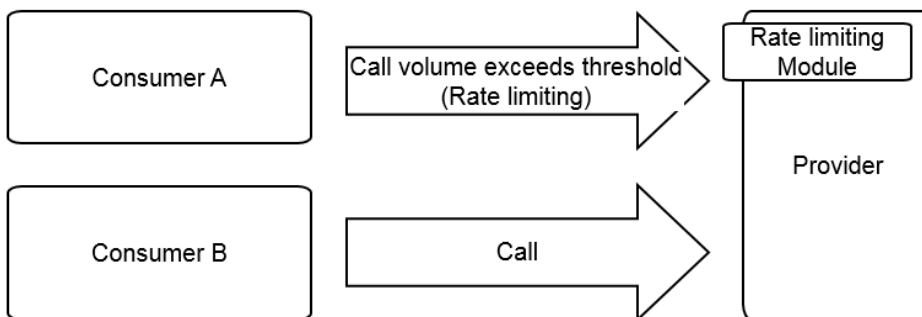
Throttling

This feature is used to control the traffic threshold or adjust the ratio. When a frontend website

encounters high-traffic access, the traffic is controlled to prevent damage to the backend core system and service unavailability. By adjusting the traffic threshold, the maximum traffic volume of the system is controlled to ensure secure and stable system running.

Basic principle

After a throttling module code is configured for a service provider and a throttling policy is configured on EDAS, the service provider has the throttling function. When a service consumer calls the service provider, all access requests are calculated by the throttling module. If the call volume of the service consumer exceeds the preset threshold in a specific period, the throttling policy is triggered.

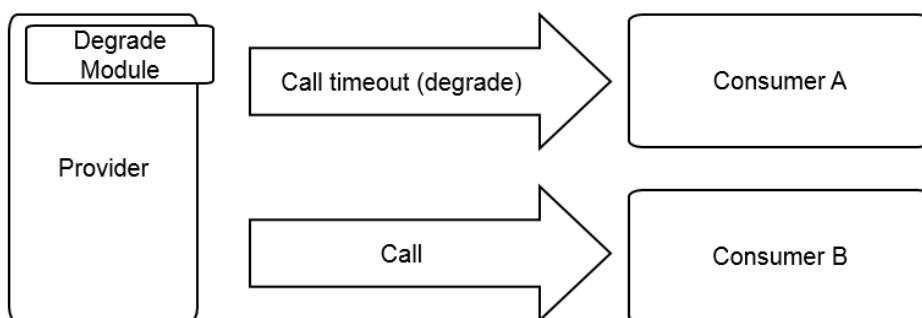


Downgrade

Downgrade is to lower the called priority of non-core service providers that are timed out to ensure the availability of core service consumers.

Basic principle

After a downgrade module code is configured for a service consumer and a downgrade policy is configured on EDAS, the service consumer has the downgrade function. When the service consumer calls a service provider, if the response time of the service provider exceeds the preset threshold, the degradation policy is triggered.



Rate limiting

Each application provides many services. EDAS allows you to configure rate limiting and throttling rules for the services, ensuring service stability and rejecting traffic that exceeds the service capabilities.

EDAS configures the rate limiting and throttling rules based on the QPS and threads to ensure the system's best operation stability during traffic peaks.

HSF rate limiting: When the traffic at a traffic peak exceeds the upper limit of the threshold defined in the throttling rules, the `BlockException` error occurs for some consumers. Based on the set threshold, the same number of services as set in the threshold are successfully called within 1s.

HTTP rate limiting: When a traffic peak occurs, some consumers are redirected to an error page. During actual access, the Taobao homepage is displayed. Based on the value set in the threshold, some requests are handled successfully.

Note: The throttling rules apply only to service providers and cannot be configured for service consumers. Before configuration, make sure whether the application serves as the service provider.

Add a rate limiting rule

Add the rate limiting rule code.

Log on to the EDAS console, select **Applications** in the left-side navigation pane to go to the application list page, and select a deployed application of the service provider to go to the application details page.

Select **Rate Limiting and Degradation > Rate Limiting Rules** from the left-side navigation pane of the Application Details page.

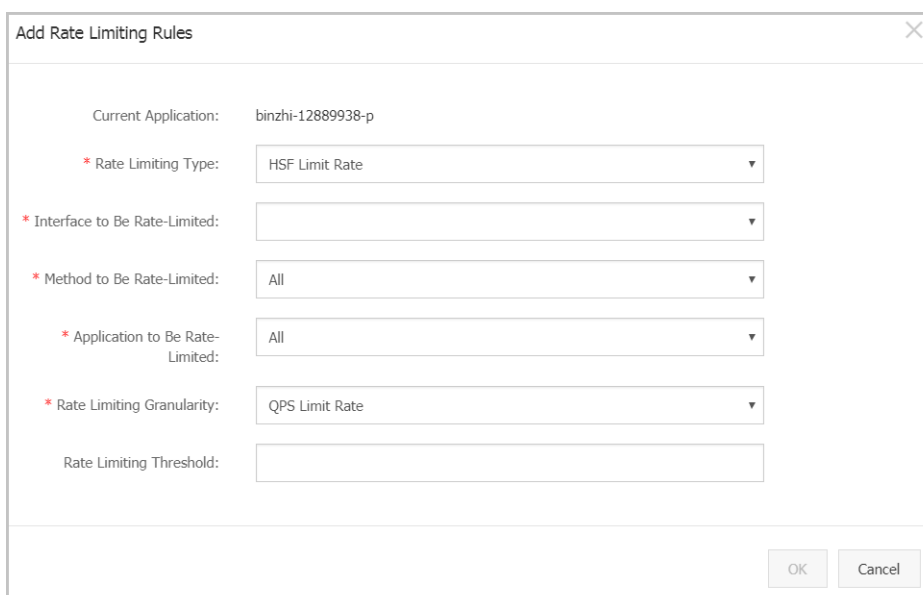
Click **Application Configuration Guide** in the upper right corner of the page.

Follow the steps in the application configuration guide to add the rate limiting rule code.

Compile and publish the application. For details, see [Publish an application](#).

Click **Add Rate Limiting Rules** in the upper-right corner of the rate limiting rules page.

In the displayed **Add Rate Limiting Rules** dialog box, set the throttling rule parameters.



The screenshot shows a dialog box titled "Add Rate Limiting Rules" with a close button (X) in the top right corner. The dialog contains the following fields and controls:

- Current Application: binzhi-12889938-p
- * Rate Limiting Type: HSF Limit Rate (dropdown menu)
- * Interface to Be Rate-Limited: (empty dropdown menu)
- * Method to Be Rate-Limited: All (dropdown menu)
- * Application to Be Rate-Limited: All (dropdown menu)
- * Rate Limiting Granularity: QPS Limit Rate (dropdown menu)
- Rate Limiting Threshold: (empty text input field)
- Buttons: OK and Cancel

Description of the throttling rule parameters:

- **Rate Limiting Type:** This parameter can be set to "HSF Rate Limiting" or "HTTP Rate Limiting". Select a specific throttling type based on the access type of the application.
- **Interface:** All interfaces of the application are listed. Select the interface to be throttled as required.
- **Method:** All methods in the interface are automatically loaded based on the selected interface. You can select whether to throttle a specific method or all methods.
- **Application:** All applications in the list, except the current application, are loaded because each application may access the current application. Select the application to be throttled as required.
- **Granularity:** This parameter can be set to "QPS" or "Thread". QPS throttling is used to control the number of requests per second, while thread throttling is used to control the number of threads. Generally, a large thread quantity leads to a large QPS value. However, the QPS value of a thread is usually greater than 1 because a thread continuously sends requests and a request response time lasts for only tens of milliseconds.
- **Throttling Threshold:** Throttling is triggered if this threshold is reached.

After completing the above settings, click **OK**.

Note: After you configure the rate limiting rules, you can click **Configure HTTP Redirections during Rate Limiting** to configure redirection URL. When a service request satisfies the rate limiting rule and is throttled, the user will be redirected to the page configured here.

Manage throttling rules

On the **Rate Limiting Rules** page, you can **Edit**, **Stop**, **Start** or **Delete** a rule.

Service degradation

Each application calls many external services. Service degradation can be configured to pinpoint and block poor services. This feature ensures the stable operation of your application and prevents the functionality of your application from being compromised by dependency on poor services.

EDAS allows you to configure degradation rules based on the response time, preventing your application from depending on poor services during traffic peaks. A consumer that triggers a degradation rule does not initiate an actual remote call in the specified time window. Instead, it throws `DegradeException`. After the time window ends, the original remote service call is restored.

Note: The degradation rules apply only to **service consumers** and cannot be configured for service providers. Before configuration, make sure that the application serves as a service consumer.

Add a degradation rule

Add the degradation rule code.

Log on to the **EDAS Console**, click **Applications** in the left-side navigation pane to go to the application list page, and select a deployed application of the service provider to go to the application details page.

Select **Rate Limiting and Degradation > Degradation Rules** from the left-side menu bar of the application details page.

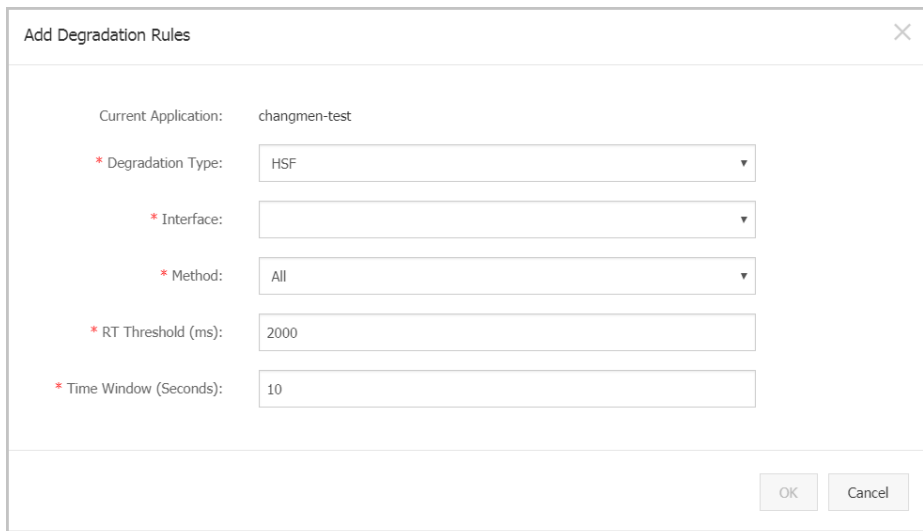
Click **Application Configuration Guide** in the upper-right corner of the degradation rule page.

Add the degradation rule code by following the steps in the application configuration guide.

Compile and publish the application. For details, see **Publish an application**.

Click **Add Degradation Rules** in the upper right corner of the degradation rule page.

In the displayed **Add Degradation Rule** dialog box, set the degradation rule parameters.



The screenshot shows a dialog box titled "Add Degradation Rules" with a close button (X) in the top right corner. The dialog contains the following fields and values:

- Current Application: changmen-test
- * Degradation Type: HSF (dropdown menu)
- * Interface: (empty dropdown menu)
- * Method: All (dropdown menu)
- * RT Threshold (ms): 2000 (text input)
- * Time Window (Seconds): 10 (text input)

At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

Description of the degradation rule parameters:

- **Degradation Type:** This parameter can be set to "HSF" or "HTTP". Select a specific degradation type based on the access type of the application.
- **Interface:** All interfaces that the consumer is consuming are listed. Select the interface to be degraded as required.
- **Method:** All methods are automatically loaded based on the selected interface. You can select whether to degrade all methods or a specific method as required.
- **RT Threshold:** The threshold of the service response time that triggers degradation, in ms. If this threshold is exceeded, the selected interface or method is degraded.
- **Time Window:** The duration for which the rule lasts after degradation is triggered.

After the above settings, click **OK**.

Manage degradation rules

On the **Degradation Rules** page, you can **Edit**, **Stop**, **Start** or **Delete** a rule.

Application diagnostics

Common operations

EDAS provides a container monitoring function – application diagnosis that collects data to help you detect problems (such as memory and class conflict) of the applications that are deployed and run inside the Tomcat container. EDAS provides refined statistic summaries for application containers, which collects statistics of the instance where the application runs from a range of dimensions, including JVM heap memory, non-heap memory, class loader, thread, and Tomcat connector. Similar to basic monitoring, container monitoring (application diagnosis) displays application-specific single-host data.

The differences are as follows:

- The monitored object of basic monitoring is ECS instance, whereas that of container monitoring is application container.
- Application diagnosis supports query of diagnostic information in single-host mode rather than cluster mode.
- Basic monitoring has latency, whereas container monitoring is near real-time because statistical computing is not performed on collected data (except memory monitoring data).

To view container details, follow these steps:

Log on to the EDAS console, click **Applications** in the left-side navigation pane, and click the name of the application in the application list.

Select **Application Diagnosis** in the left-side navigation pane of the **Application Details** page.

Select an ECS instance from the **ECS Instance (Instance ID/Name/IP)** drop-down list.

Click tabs to view the monitoring details of the container.

The application diagnosis page has the following tabs:

Memory: Memory is monitored on a per-instance basis. EDAS provides statistics on the heap memory and non-heap memory of the JVM process of the Tomcat container where the application is located. The **Memory** tab appears by default after you go to the container diagnosis page. The statistical periods include 30 minutes, 6 hours, 1 day, and 1 week.

Class Loading: Provides real-time information about JAR package loading. When a JAR package of the application has version conflict, you can use this function to

easily locate the path to which the JAR package is loaded, which lowers troubleshooting costs.

Thread: Displays the basic information of all threads of the current JVM process, including the thread ID, status, and name. The statistical fields are native information of JVM.

Connector: The Tomcat connector is indicated by `<Connector />` in the XML configuration of Tomcat. The configuration of each `<Connector />` can be considered as a line of pulled information. This view displays the running status of the corresponding connector for the last 10 minutes.

Each connector has a certain number of threads (which forms a thread pool) to process incoming requests. When concurrency or throughput bottlenecks occur, statistics of the processing status of the connector's thread pool needs to be collected. For example, an HTTP connector has the following XML configuration:

```
<Connector port="8080" protocol="HTTP/1.1" maxThreads="250" .... />
```

Click **Thread Pool Info** in the "Action" field next to the connector. Relevant details are displayed.

The preceding figure shows that the application is almost load-free. If the value of "Busy Thread Count" is close to that of "Thread Pool Max. Value", the system encounters serious concurrency issue. To solve the problem, scale up the application or optimize the service code.

Object Memory Distribution: Select **System Class**, **Java Primitive Object Class**, **Class Loading Related**, then statistics about number of objects, occupied disk space, and memory usages will be displayed in pie-charts and tables.

- **Method Tracking:** For details about method tracking, see [Method tracking](#).
- **Hot Thread:** Provides thread snapshots and statistical analysis of service calls.
 - **Thread Snapshot:** Similar to the `jstack` command, this command collects stack frames of all threads from the target machines. It identifies the idle threads after obtaining the thread stack, such as HSF, Tomcat, GC. In order to avoid excessive overhead, only 30 threads are retrieved among the rest of threads.
 - **Service Call Statistics:** Performs statistic analysis of the method calls in the application during a period of time, and shows the method calls and call relations (or call stack). Final results are displayed in tree diagrams and graphs. It highlights your business methods automatically, allowing you to locate the sources of the busiest method calls. This call request will last for about 10 seconds before it returns the results.

Method tracking

Overview

EDAS method tracing helps you quickly troubleshoot problems of running applications. Typical use cases include:

- When you find that it takes a long time to run a service logic, and you want to identify the part of code that causes the time consumption.
- Applications and services are all running smoothly for most of the time. However, a user reports the problem that service response is extremely slow when a specific parameter is passed. In this case, you may need a mechanism to check the code execution related to a specific parameter in a method.
- When a method with complex service logic is executed, you want to have a clear view of the logic and time sequence of service calls in details.

EDAS method tracing is designed to meet the preceding requirements without interfering with code or stopping applications.

EDAS method tracing adopts the JVM bytecode enhancement technique when recording the time consumption and sequence during the entire call process of the selected method, enabling you to check the execution sequence while execution is in progress.

Restrictions

If the following restrictions affect your services or troubleshooting, open a ticket to consult with us so that we can improve some restrictions or configure a whitelist.

In principle, only tracing of service-type classes is supported. Therefore, packages are filtered by name before tracing starts.

Sampled output is adopted to prevent excessive logs due to large amount of method calls. The default policy is to output logs on 10 calls per second for the method.

When you exit and then log on to the EDAS console again, or refresh the screen, historical trace records are lost and previously pulled tracing information is no longer retained.

Automatic stop policy: If method tracing is in inactive state for 10 minutes, EDAS

automatically detaches the tracing module and restores the method to the initial state (state prior to enhancement).

Parameter printing: Currently, EDAS only supports printing of basic Java types (string, char, int, float, double, short, and boolean).

If the selected string is too long, EDAS truncates the string to output the first 100 characters.

If the JVM instance restarts during the tracing process, you must stop tracing and then restart it.

Currently, a maximum of 10,000 trace logs can be output. To output more logs, restart the tracing function.

The current version does not support Docker-based applications.

Environment check

Because the method tracing function adopts the JVM bytecode enhancement technique, to ensure normal running of applications, this function is disabled when the environment check fails.

Before the method tracing function starts, EDAS automatically checks that:

1. Ali-Tomcat is in **Running** state.
2. CPU usage is lower than 60%.
3. Available system memory is more than 100 MB.
4. The available space of JVM PermGen or Metaspace is more than 20 MB.

If the environment check fails, we recommend that you clear the alarms and then click **Retry**.

Usage instructions

Log on to the EDAS console.

Click **Applications** in the left-side navigation pane and click the name of the application to be checked in the application list.

Click **Application Diagnosis** in the left-side navigation pane of the Application Details page.

Click the **Method Tracing** tab on the application diagnosis page.

Note:

If the **Method Tracing** tab does not appear, follow these steps:

Check that you are using Google Chrome as the web browser, and refresh the page. (We performed tests in Google Chrome only.)

If you log on with a sub-account, check that the sub-account has required permissions.

To check permissions, choose **Applications > Application Diagnosis > Method Tracing and Tool Authorization**.

Before the permission check starts, EDAS performs an environment check on the ECS instance where the application is located. When the environment check dialog box appears, select the checkbox to confirm the precondition and click **Confirm** to start the environment check.

The method tracing page appears after the environment check is successful.

Set tracing parameters.

Note: **Class Name** and **Method Name** are required. Set the two parameters to the class and method you want to trace.

The configuration items are described as follows:

Class Name: Required. Enter a full path name starting with the package path, for example, com.test.alibaba.demo.HelloWorldServlet. EDAS does not support tracing of classes whose names start with the following package paths:

- java.*
- javax.*
- com.google.*
- com.alibaba.*
- com.aliyun.*
- com.taobao.*
- org.apache.*
- org.dom4j.*
- org.springframework.*
- redis.clients.*

After a complete package path is entered, EDAS checks whether the class exists on the selected ECS instance.

- If the entered class exists, it is displayed in the drop-down list. Select the class to continue.
- If the entered class does not exist, the message "Class does not exist" is displayed in the drop-down list.

Method Name: Required. After a class is selected, the system automatically searches for all methods under the class and displays the method list below the textbox.

The icon on the left of each method indicates the modifier of the method.

- public: A green lock
- protected: A yellow key
- private: A red lock
- package: A blue block
- abstract: No icon

Select the method to be traced from the drop-down list and continue.

Exception Tracing Only: The execution of a method either returns a response normally or ends execution due to an exception. If you select "Exception Tracing Only" , the tracing results are printed and output only when the method is ended due to an exception.

Print Returned Values: If you select this option, the returned values of the method are printed on the result page. null is output if the return type of the method is void.

After a method is selected, the **Start Tracing** button is available in blue.

Click **Start Tracing** to trace the method. Whenever the method is called, the call information is displayed in the result area.

Note: After method tracing starts, EDAS periodically checks whether the tracing is in active state. If method tracing is in inactive state for 10 minutes, EDAS automatically stops the tracing and restores the method to the initial state.

Check the method call information.

After method tracing starts, EDAS displays the generated call logs in the EDAS console.

On the left of the display area, each record represents the log that is generated each time the method is called.

44-62/150 at the bottom of the table indicates that the browser returns 150 records in total and currently lists the trace records in rows 44 to 62.

The prompt "Press key H to show help information" is displayed at the bottom of the table. Press "H" on the keyboard to display the usage instructions on shortcut keys.

- H: Displays the help document.
- Ctrl+G: Displays the latest data in real time. As the call times increase, it is impossible to render and display all records. Similar to the tail function, Ctrl+G is used to display the latest data once retrieved.
- G: Jumps to a specific record. Search for the trace record in a specific row and select it to display details.
- Ctrl+C or ESC: Ends the command that is being executed.
- Ctrl+H: Goes to the next page to display the next 10 trace records.
- Ctrl+I: Returns to the previous page to display the preceding 10 trace records.
- J or ↓: Selects the next trace record.
- K or ↑: Selects the previous trace record.
- Enter or Double-click: Zooms in or restores the selected trace record.

The right side of the display area displays the details or basic information of the selected record. You can double-click or press **Enter** in the details section to zoom in, or press **ESC** to restore to initial display.

- **Tracing details:** Shows the time consumption and execution sequence for each call. **The time in blue** is the total time consumed by calling the method. **The time in red** indicates that the time of the specific call is more than 30% of the total time consumed.
- **Output details:** Shows the exceptions, return values, and input parameters (which are selected using the **More** option) during execution.
- **Method stack details:** Shows the stack information before the traced method is called.

Stop tracing.

After method tracing starts, the **Start Tracing** button changes to **Stop Tracing**. After you click **Stop Tracing**, EDAS restores the traced method to the initial state (state prior to enhancement) and records tracing information in the instance dimension. Next time you go to the method tracing page, the last tracing information is displayed.

If you modify tracing items (for example, method name) after tracing stops and then click

Start Tracing, the modified information is submitted and tracing starts based on the latest submitted information.

Commons pool monitoring

When Commons Pool2 (2.0) (for example, Jedis and Commons DBCP2 connection pool on a Redis client) is used by an application or application library, the EDAS Commons Pool monitoring component monitors the pool configuration and usage. The monitoring data is recorded every 10s.

Start Commons Pool monitoring

Log on to the EDAS console, click **Applications** on the left-side menu bar, and click an application in the application list.

Click **Application Diagnosis** in the left-side navigation pane on the Application Details page.

On the application diagnosis page, click **ECS Instance (Instance ID/Name/IP)** from the drop-down list, and select an instance.

Click the **CommonsPool** tab.

Click **Start Monitoring**.

After monitoring starts, EDAS automatically tracks the existing or newly created pools. The page shows the usage and configuration of each pool object. The data on the page is refreshed every 10s by default. Tracking automatically stops when the pool is closed.

NOTE: If the application does not have the Commons Pool2 class library or the pool has not been loaded, the **Start Monitoring** button is unavailable.

Click **Stop Monitoring**.

Monitoring information description

Pool monitoring information

The monitoring information of the pool consists of two parts:

- Pool usage: presented in a graph. It includes the number of active objects, the number of idle objects, the number of blocked threads, the maximum number of objects, and the maximum number of idle objects.
- Pool configuration: presented in a table. It includes various configuration values of the pool. Each configuration item is displayed in English. For the specific meanings, see `GenericObjectPoolConfig`.

Commons Pool usage field description:

Field	English	Description
Maximum number	Max Total	The maximum number of objects in the pool, including busy and idle objects
Maximum number of objects	Max Idle	Maximum number of available objects in the pool
Number of active objects	Num Active	Number of active objects obtained from the pool. If this number frequently exceeds the maximum number of idle objects, you need to increase the maximum number of idle objects.
Number of idle objects	Num Idle	Number of available objects in the pool
Number of blocked threads	Num Waiters	Number of blocked threads. If this value is greater than 0, you need to increase the maximum number of threads.

Druid database connection pool monitoring

When the data connection pool of the application uses a Druid database, the EDAS Druid database connection pool monitoring agent will monitor the data connection pool and SQL execution. The monitored data is recorded once every 10s and reset.

Procedure

Take the following steps to operate and use the Druid database connection pool:

Log on to the EDAS console, click **Applications** on the left-side menu bar, and click an application in the application list.

Click **Application Diagnosis** in the left-side navigation pane on the Application Details page.

Click the **ECS Instance (Instance ID/Name/IP)** drop-down list on the Application Diagnosis page, and select an EDAS instance.

Click the **Druid Database Connection Pool Monitoring** tab.

Click **Start Monitoring**.

The page displays information about the data connection pool and SQL execution. The page is refreshed once every 10s by default.

Note: When you click "Start Monitoring", if StatFilter provided by the Druid data connection pool is not configured for the application, EDAS automatically adds StatFilter to the application. This may slightly affect performance. Therefore, we strongly recommend that you manually add StatFilter to your application.

Click **Close** to exit monitoring.

Monitoring information description

Monitoring information of the data connection pool

The monitoring information of the data connection pool consists of the following:

- Monitoring indexes of the data connection pool, including the database type, Driver class, size of the initial connection pool, and maximum number of connections.
- Information about the data connection pool during running, including the size of available connections, peak size of available connections, and number of active connections.

The following table describes the monitoring indexes of the Druid database connection pool.

Field	English	Description
DB Type	DB Type	Type of the database to which the data source is connected, for example, MySQL
Driver Class	Driver Class	Class of the data driver
User Name	User Name	User of the connection pool
Init Size	Init Size	Initial size of the connection

		pool
Max Active	Max Active	Maximum number of connections in the connection pool
Pool Size	Pool Size	Number of available connections in the connection pool
Maximum Pool Size	Maximum Pool Size	Maximum number of available connections in the connection pool
Active Count	Active Count	Number of active connections in the connection pool

SQL execution information

The SQL execution information consists of **information about SQL statements executed in the current 10s** and **information about SQL statements of which the maximum execution time exceeds 100 ms**. The monitoring indexes of these two parts are the same and are described in the following table.

Field (Chinese)	Field (English)	Description
SQL	SQL	Executed SQL statement
Executed Count	Executed Count	Number of executions of this SQL statement
Total Executed Time	Total Executed Time	Total execution time of this SQL statement
Maximum Executed Time	Maximum Executed Time	Maximum execution time of this SQL statement
Maximum Returned Rows	Maximum Returned Rows	Maximum number of returned rows of this SQL statement
Monitor Time	Monitor Time	Recording time of this SQL record

Hot thread

Overview

Thread hotspot has two functions:

Obtain the thread snapshot Similar to the `jstack` command, it can be used to obtain stack frames of all the current threads. After the thread stacks are obtained, it will filter out identified idle threads, for example, HSF, Tomcat, and GC threads. To avoid overhead, the data of only 30 threads out of the remaining threads are returned.

Analyze the call statistics

It can collect statistics and analyze the method call in the application within a certain period of time and display the call method and call relationship (call stack). The final result is displayed in two views (tree graph and flame graph). Besides, your service method is automatically highlighted so that you can quickly locate the call source of the service method with large time consumption. The call will last for about 10s and then the result is returned.

Usage instructions

Log on to the EDAS console, click **Application Management** on the left-side menu bar, and click the name of the application to be diagnosed on the **Application Management** page.

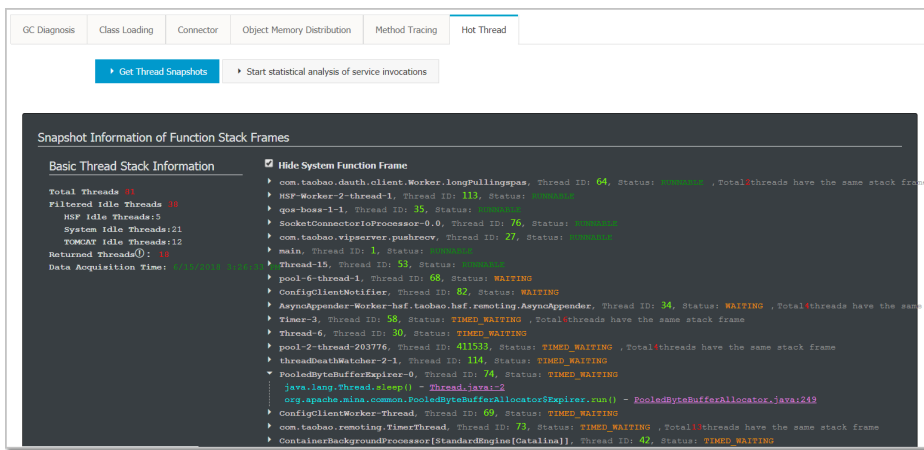
Click **Application Diagnosis** on the left-side menu bar on the Application Details page.

Click **Thread Hotspot** on the Application Diagnosis page.

On the Thread Hotspot page, click **Obtain Thread Snapshot** or **Start to Analyze Call Statistics**.

Obtain thread snapshot

After you click **Obtain Thread Snapshot**, the service attempts to group and identify the threads in the result, locate the service threads and service stack frames and then expand them.

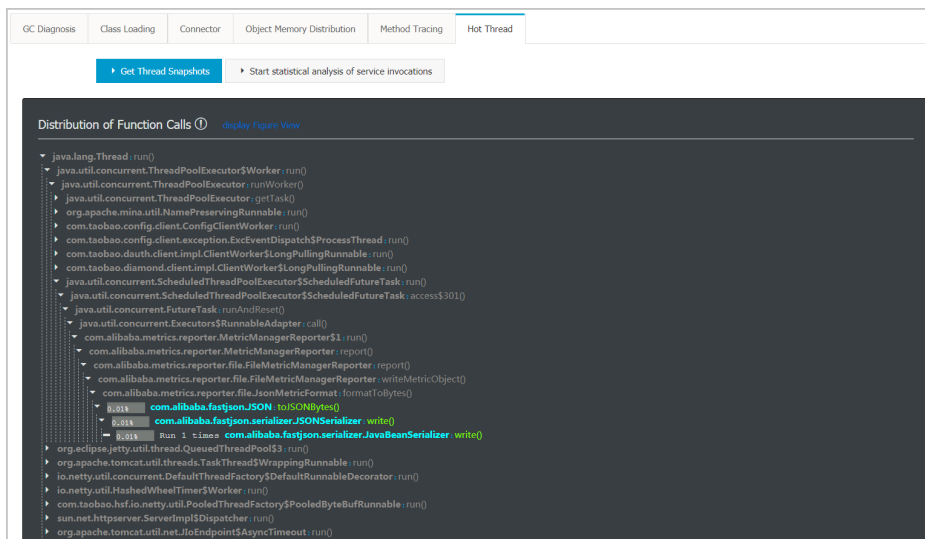


Start to analyze call statistics

After you click **Start to Analyze Call Statistics**, the service performs call analysis. The call analysis result is displayed in either of the following two ways:

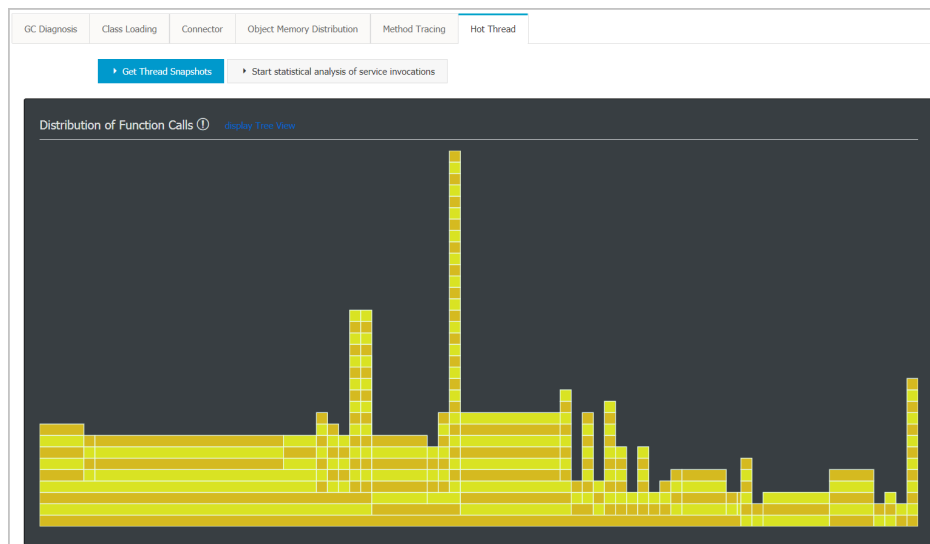
Tree graph (default)

The service attempts to locate and expand the service logic stack frames.



Flame graph

The call statistics and call relationship are displayed in a flame graph. You can click a specific stack frame (method) to view the call statistics of a call path.



Container version management

An EDAS container consists of AliTomcat, Pandora, and custom Pandora plug-ins. Besides supporting the existing core functions of Apache Tomcat, EDAS provides class isolation mechanism, QoS service, and Tomcat Monitor. Besides, highly customized plug-ins are added to EDAS containers to implement complex and advanced functions, such as container monitoring, service monitoring, and distributed tracing. Applications deployed using EDAS must run in EDAS containers.

AliTomcat

AliTomcat is developed by Alibaba middleware team based on Apache Tomcat, with a series of performance optimization, bug fixes, and new features. AliTomcat is widely deployed and used in Alibaba Group, which is greatly improved compared with the community version in terms of performance, security, and stability.

Pandora and Pandora plug-ins

Pandora is a lightweight isolation container, which is taobao-hsf.sar. It is used to isolate dependence between web applications and middleware products and between middleware products so that they do not affect each other. Plug-ins implementing service discovery, configuration push, tracing, and other functions are integrated in Pandora. By using these plug-ins, you can monitor, process, track, analyze, maintain, and manage services of EDAS applications in all dimensions.

Container version

You must select the container version when creating an application in EDAS. EDAS containers are

maintained and published by the EDAS development team. You can view the publishing history and description of each version of a container by selecting **Applications > Container Versions** or referring to **Container Version Notes**. Generally, a container of a higher version is superior to a container of a lower version in stability and function variety.

Publishing of an EDAS container does not affect deployed applications. After a new container is published, you can immediately upgrade to the new version.

Upgrade or downgrade a container

In the EDAS Console, click **Applications** in the left-side navigation pane to go to the Application List page.

Click the name of the application to be operated to go to the application details page.

Click **Container Version** in the left-side navigation pane to go to the container version page.

Click **Upgrade to This Version** or **Degrade to This Version** next to the container version to be upgraded or downgraded. The container version can be upgraded or downgraded by one click.

Container Version			
Version	Description	Release Date	Actions
3.3.6	Fixed the displayed "Unknown" error in EagleEye trace which causes the application topology unable to be display. Support template Support fatjar Support restful	2017-12-20	Upgrade to This Version
3.3.5	HSF V2.2 supports ACM. Support template Support fatjar Support restful	2017-11-30	Upgrade to This Version
3.3.4	use for test, please upgrade as soon as possible. Support fatjar	2017-11-16	This version is not available.
3.3.3	HSF V2.2 release & improve Pandora OOS command. Support template Support fatjar Support restful	2017-10-18	✓
3.3.2	HSF V2.2 Release Support template Support fatjar Support restful	2017-09-22	Apply to Use This Version
3.3.1	Tomcat supports HTTP service monitoring in Docker applications. Support template	2017-07-13	Degrade to This Version

Global configuration

Configuration push services can be classified into global configuration push and intra-application configuration push in EDAS.

- The **Global Configuration Push** service pushes configurations to all applications under a given username.
- The **(Intra-Application) Configuration Push** service only pushes configurations within a given

application.

This article describes the global configuration push service. For information about the intra-application configuration push service, see **(Intra-application) configuration push**.

Configuration in EDAS is a trio that contains Group, DataId, and Content. The three elements are defined as follows:

- **Group**: Name of a group, which is created in **Service Group** and used to isolate services in a namespace. for example, a package in Java. The maximum length allowed is 128 characters.

DataId: Configuration name, for example, a class name in Java. The maximum length allowed is 256 characters.

A piece of configuration is identified by Group and DataId collectively and corresponds to a value. The symbols that are allowed in the names of Group and DataId are period (.), colon (:), hyphen (-), and underscore (_).

Content: Configuration value. The maximum length allowed is 1,024 characters.

You can add, modify, and delete configurations in real time, and apply configurations dynamically without the need to modify code, republish services, or restart services.

Note: If you have not created any services, the configuration list displays a piece of configuration that is automatically generated by the system, which you can ignore.

Create a global configuration

Log on to the EDAS console.

Choose **Service Market** > **Service Groups** on the left-side menu bar.

In the container upgrade prompt, select **Upgrade Later**.

Click **Create Service Group** in the upper-right corner of the page. In the **Create HSF Service Group** dialog box, enter the **Service Group Name** and click **Create**.

Click **Global Configuration** on the left-side menu bar.

In the **Configuration List** page, select the region and then select the service group you just created. Then, click **Create Configuration** in the upper-right corner of the page.

In the **Create Configuration** dialog box, enter the **DataId** and **Content** and then click **OK**.

Note: The group has already been selected on the **Configuration List** page. It is not editable in this dialog box.

View configuration list

Click **Global Configuration** on the left-side menu bar of the EDAS Console.

In the **Configuration List** page, select the region in which to view configurations.

View the global configuration list for this region.

By default, this page shows all the configuration information for the first group. From the group drop-down menu, you can select the group of which you want to view the configurations.

View global configuration details

On the **Configuration List** page, click the **View** button in the **Actions** column of the desired configuration.

The dialog box that appears shows the Group, DataId, and Content for the selected configuration.

Update global configuration

On the **Configuration List** page, click the **Update** button in the **Actions** column of the configuration to update.

The dialog box that appears allows you to modify the content of the configuration.

After completing the modification, click **OK** to update the configuration.

Delete global configuration

You can delete any global configuration that will no longer be used.

Note: A piece of configuration can no longer be used once deleted. Please proceed with caution.

On the **Configuration List** page, click the **Delete** button in the **Actions** column of the configuration to delete.

In the **Delete Configuration** dialog box, confirm the information and click **Delete**.

Digital operations

Overview

Data-driven operations is an important set of functions within EDAS. The most important data-driven operations function is distributed link analysis.

Distributed link analysis analyzes every service distributed system call, all sent and received troubleshooting messages of interest, and all database access to help you precisely identify system bottlenecks and risks.

Data-driven operations provides these functions:

View application topologies

The topology map intuitively presents the calling between applications and relevant performance data.

Query trace links

By setting query conditions, you can accurately find businesses with poor performance or exceptions.

View trace link details

Based on the trace query results, you can view detailed information for slow or malfunctioning businesses and reorganize their dependencies. This information allows you to identify frequent failures, performance bottlenecks, strong dependencies, and other problems. You can also evaluate business capacities based on trace ratios and peak QPS.

Application topology

The application topology function is used to view the real-time (last second) call topology between applications in the system.

Log on to the EDAS console, and choose **Digital Operations** > **Application Typology** on the left-side navigation bar.

View the application topology.

The application topology shows the real-time (last second) call topology between all applications under the current account.

Hover your cursor over an application to view the call topology for this application.

Click on an application to view its call topology and traffic data.

Traffic data refers to the current application's QPS, including:

Source traffic: The QPS for calls from other applications to this application.

Call traffic: The QPS for calls from this application to other applications.

Trace query

By using the trace query function, you can view the status of the invocation trace in the system, especially for tasks that are slow or have encountered an error.

Log on to the EDAS console and choose **Digital Operations** > **Trace Query** in the left-side navigation pane.

Click **Show Advanced Options** in the upper-right corner of the **Trace Query** page to display more query conditions.

Specify the query conditions and click **Query**.

The screenshot shows the 'Trace Query' interface with the following fields and options:

- Time Range:** A date selector set to '2018-01-15 09:50:09' and a dropdown for 'To 10 Minut'.
- Application Name:** A dropdown menu with the placeholder text 'Enter the application name to ... please inp...'. There is an 'Error' checkbox and a 'Query' button to the right.
- Client IP:** A text input field with the placeholder 'Intranet IP Of Client'.
- Server IP:** A text input field with the placeholder 'Intranet IP Of Server'.
- Entrance IP:** A text input field with the placeholder 'Internet IP or Intranet IP'.
- Service Name:** A text input field with the placeholder 'Service Name or URL'.
- Results:** A text input field with the value '1~1000'.
- Call Type:** A dropdown menu with the value 'Any'.
- Elapsed Time >:** A text input field with the value 'MS'.
- Request >:** A text input field with the value 'byte'.
- Response >:** A text input field with the value 'byte'.
- Response Code:** A text input field with the value 'value'.

The descriptions for the parameters of the invocation traces (advanced query conditions) are as follows:

Time range: Click the time selector, set the query start time, and then select the end time. The options for the end time are "This Second" , "To 1 Minutes Later" , and "To 10 Minutes Later" . Therefore, the latest time periods are: last second, last one minute, and last ten minutes.

Application name: Select an application from the drop-down list. You can also enter a keyword to search for an application. Manual input of an application name is not supported.

Call type: Select the call type to query from the drop-down list. Options are HTTP, HSF provider, HSF consumer, MySQL, Redis cache, message sending, and message receiving.

Set the threshold values for time elapsed, request, or response for querying slow tasks in the system.

Select the **Error** check box in the upper-right corner to query the error cases only.

Specify other parameters as needed.

In the query result, click on a slow or erroneous task to view trace details.

For the procedure to view the trace details, see [Call trace details](#).

Trace details

The trace details function enables you to query by TraceId the details of a specific service invocation trace in a selected region.

The trace details page displays the trace of the RPC service calls, not including local method calls.

The trace details function is used mainly for tracking the consumed time and occurred exceptions at each point of the distributed service calls. Local methods are not the core content of the calls, so it is recommended that you use logs to track the consumed time and occurred exceptions for local methods. For example, the trace details page will not display the local trace of methodA() calling localMethodB() and localMethodC(). Therefore, it could happen that the elapsed time on a parent node is longer than the total elapsed time on all subnodes.

You can log on to the EDAS console and choose **Digital Operations > Trace Details** in the left-side navigation pane to view the details of a service invocation trace. However, a more typical scenario is to view the trace details of the slow or erroneous services. The following example demonstrates how to view the trace details entering from **Trace Query** on the left-side menu bar.

In the trace query result, find the HSF method, DB request, or other RPC service call that consumes the longest time.

For DB, Redis, MQ, or other simple calls, find out the reason why accesses to these nodes are slow and check whether they are caused by slow SQL or network congestion.

For HSF methods, further analyze the reason why the method consumes so much time.

Confirm the time consumed by a local method.

Hover the cursor over the time bar on the method row, and in the displayed page, view the elapsed time for the client to send the request, the elapsed time for the server to process the request, and elapsed time for the client to receive the response.

If it takes a long time for the server to process the request, analyze the tasks. Otherwise, conduct the analysis using the method that is used for analyzing call timeout.

Check whether the total time consumed on subnodes is close to that consumed on the method.

If the time difference is small, it indicates that most of the time is consumed on network calls. In this case, reduce network calls as many as possible to shorten the time consumed on each method.

The preceding figure shows that the same method is cyclically called. Instead, it could be just called once in batch.

If the time difference is large, for example, the time consumed on the parent node is 607 ms while the total time consumed on the subnodes does not reach 100 ms. Then it indicates most of the time is consumed on the task logic of the server itself, rather than the RPC service call.

Locate the time-consuming call.

By looking at the time bars to first locate the call before which much time is consumed. The time is purely consumed by the local logic, for which further troubleshooting is required.

After locating the time-consuming logic, review the codes or add logs to the codes to locate the errors.

If it is found that the codes do not consume so much time, perform the following step.

Check whether GC occurred at that time. Therefore, the gc.log file is important.

Locate the timeout error.

An timeout error occurs. Perform the following steps to evaluate the time.

The time is divided into three parts:

Client sends request (0 ms): indicates the time duration from the client sends the request to the server receives it. This process includes serialization, network transmission, and deserialization. If this process takes a long time, consider if a consumer GC should be triggered. It will take a long time if the object for serialization or deserialization is large, the network is under great transmission pressure, or the provider GC occurs.

Requests processed on server (10,077 ms): indicates the time duration from the server receives the request to the server returns the response to the client. The time is taken only by the server to process the request, not including other operations.

Client receives response (3,002 ms): indicates the time duration from the server sends the response to the client receives it. As the timeout time of 3s is set, the server directly returns timeout after 3s, but the server is still processing the request.

If this process consumes much time, perform troubleshooting using the same method that is used for the client.

Redis tracing

Function overview

After Redis tracing support is added, whenever applications access and perform operations on Redis, the process is recorded in EagleEye trace logs and EDAS collects, analyzes the statistics of the logs. Then information about Redis calls is displayed on the tracing and call analysis page of the EDAS platform.

Supported scope

Due to the wide range of Redis database variants and the usability of Spring Data, Redis trace support is only available for Spring Data Redis of 1.7.4.RELEASE. If you use any other database (for example, Jedis) than Spring Data Redis, you cannot view relevant information on the EagleEye trace interface (which is accessible from **Digital Operations** > **Trace Details** on the left-side menu bar of the EDAS console).

Note: If you use Spring Data Redis later than 1.7.4.RELEASE and the version does not support the provided functions, open a ticket to consult with us.

Usage instructions

For applications on the EDAS platform, Redis trace support replaces Spring Data Redis and is used in the same way as Spring Data Redis. For usage instructions on Spring Data Redis, see the user guide. At the code level, EDAS is compatible with Spring Data Redis 1.7.4-RELEASE. To enable Redis tracing support, follow these steps:

Open the {user.home}/.m2/settings.xml file to configure the local Maven repository.

```
<profile>
<id>edas.oss.repo</id>
<repositories>
<repository>
<id>edas-oss-central</id>
<name>taobao mirror central</name>
<url>http://edas-public.oss-cn-hangzhou.aliyuncs.com/repository</url>
```

```

<snapshots>
<enabled>true</enabled>
</snapshots>
<releases>
<enabled>true</enabled>
</releases>
</repository>
</repositories>
<pluginRepositories>
<pluginRepository>
<id>edas-oss-plugin-central</id>
<url>http://edas-public.oss-cn-hangzhou.aliyuncs.com/repository</url>
<snapshots>
<enabled>true</enabled>
</snapshots>
<releases>
<enabled>true</enabled>
</releases>
</pluginRepository>
</pluginRepositories>
</profile>
</profiles>

```

Activate the corresponding profile:

```

<activeProfiles>
<activeProfile>edas.oss.repo</activeProfile>
</activeProfiles>

```

Add dependency to the pom.xml file in the Maven project.

```

<dependency>
<groupId>com.alibaba.middleware</groupId>
<artifactId>spring-data-redis</artifactId>
<version>1.7.4.RELEASE</version>
</dependency>

```

Redis command support

The following tables list the Redis commands supported by Spring Data Redis and the support for EagleEye trace logs.

Key-type operations

Data structure/Object	Operation	Spring Data Redis method	EDAS support for EagleEye tracing (Y/N)	Remarks
Key	DEL	RedisOperation	Y	

		s.delete		
	DUMP	RedisOperation s.dump	Y	
	EXISTS	RedisOperation s.hasKey	Y	
	EXPIRE	RedisOperation s.expire	Y	
	EXPIREAT	RedisOperation s.expireAt	Y	
	KEYS	RedisOperation s.keys	Y	
	MIGRATE	N		
	MOVE	RedisOperation s.move	Y	
	OBJECT	N		
	PERSIST	RedisOperation s.persist	Y	
	PEXPIRE	RedisOperation s.expire	Y	
	PEXPIREAT	RedisOperation s.expireAt	Y	
	PTTL	RedisOperation s.getExpire	Y	
	RANDOMKEY	RedisOperation s.randomKey	Y	
	RENAME	RedisOperation s.rename	Y	key: oldKey : \${oldKey};newKey:\${newKey}
	RENAMENX	RedisOperation s.renameIfAbsent	Y	
	RESTORE	RedisOperation s.restore	Y	
	SORT	RedisKeyComm ands.sort	Y	key: query:\${SortQuery}
	TTL	RedisOperation s.getExpire	Y	
	TYPE	RedisOperation s.type	Y	
	SCAN	RedisKeyComm ands.scan	N	

String-type operations

Data structure/Object	Operation	Spring Data Redis method	EDAS support for EagleEye tracing (Y/N)	Remarks
String	APPEND	ValueOperations.append	Y	
	BITCOUNT	N		
	BITOP	N		
	BITFIELD	N		
	DECR	ValueOperations.increment	Y	
	DECRBY	ValueOperations.increment	Y	
	GET	ValueOperations.get	Y	
	GETBIT	ValueOperations.getBit	Y	
	GETRANGE	ValueOperations.get	Y	
	GETSET	ValueOperations.getAndSet	Y	
	INCR	ValueOperations.increment	Y	
	INCRBY	ValueOperations.increment	Y	
	INCRBYFLOAT	ValueOperations.increment	Y	
	MGET	ValueOperations.multiGet	Y	
	MSET	ValueOperations.multiSet	Y	
	MSETNX	ValueOperations.multiSetIfAbsent	Y	
	PSETEX	ValueOperations.set	Y	
	SET	ValueOperations.set	Y	
	SETBIT	ValueOperations.setBit	Y	
	SETEX	ValueOperations	Y	

		s.set		
	SETNX	ValueOperation s.setIfAbsent	Y	
	SETRANGE	ValueOperation s.set	Y	
	STRLEN	ValueOperation s.size	Y	

Hash-type operations

Data structure/Object	Operation	Spring Data Redis method	EDAS support for EagleEye tracing (Y/N)	Remarks
Hash	HDEL	HashOperation s.delete	Y	
	HEXISTS	HashOperation s.hasKey	Y	
	HGET	HashOperation s.get	Y	
	HGETALL	HashOperation s.entries	Y	
	HINCRBY	HashOperation s.increment	Y	
	HINCRBYFLOAT	HashOperation s.increment	Y	
	HKEYS	HashOperation s.keys	Y	
	HLEN	HashOperation s.size	Y	
	HMGET	HashOperation s.multiGet	Y	
	HMSET	HashOperation s.putAll	Y	
	HSET	HashOperation s.put	Y	
	HSETNX	HashOperation s.putIfAbsent	Y	
	HVALS	HashOperation s.values	Y	
	HSCAN	HashOperation s.scan	Y	
	HSTRLEN	N		

List-type operations

Data structure/Object	Operation	Spring Data Redis method	EDAS support for EagleEye tracing (Y/N)	Remarks
List	BLPOP	ListOperations.leftPop	Y	
	BRPOP	ListOperations.rightPop	Y	
	BRPOPLPUSH	ListOperations.rightPopAndLeftPush	Y	key: sourceKey:\${sourceKey};destKey:\${destKey}
	LINDEX	ListOperations.index	Y	
	LINSERT	ListOperations.leftPush	Y	
	LLEN	ListOperations.size	Y	
	LPOP	ListOperations.leftPop	Y	
	LPUSH	ListOperations.leftPush	Y	
	LPUSHX	ListOperations.leftPushIfPresent	Y	
	LRANGE	ListOperations.range	Y	
	LREM	ListOperations.remove	Y	
	LSET	ListOperations.set	Y	
	LTRIM	ListOperations.trim	Y	
	RPOP	ListOperations.rightPop	Y	
	RPOPLPUSH	ListOperations.rightPopAndLeftPush	Y	key: sourceKey:\${sourceKey};destKey:\${destKey}
	RPUSH	ListOperations.rightPush	Y	
	RPUSHX	ListOperations.rightPushIfPresent	Y	

		nt		
--	--	----	--	--

Set-type operations

Data structure/Object	Operation	Spring Data Redis method	EDAS support for EagleEye tracing (Y/N)	Remarks
Set	SADD	SetOperations.add	Y	
	SCARD	SetOperations.size	Y	
	SDIFF	SetOperations.difference	Y	
	SDIFFSTORE	SetOperations.differenceAndStore	Y	
	SINTER	SetOperations.intersect	Y	
	SINTERSTORE	SetOperations.intersectAndStore	Y	
	SISMEMBER	SetOperations.isMember	Y	
	SMEMBERS	SetOperations.members	Y	
	SMOVE	SetOperations.move	Y	
	SPOP	SetOperations.pop	Y	
	SRANDMEMBER	SetOperations.randomMember randomMembers distinctRandomMembers	Y	
	SREM	SetOperations.remove	Y	
	SUNION	SetOperations.union	Y	
	SUNIONSTORE	SetOperations.unionAndStore	Y	
	SSCAN	SetOperations.scan	Y	

SortedSet-type operations

Data structure/Object	Operation	Spring Data Redis method	EDAS support for EagleEye tracing (Y/N)	Remarks
SortedSet	ZADD	ZSetOperations.add	Y	
	ZCARD	ZSetOperations.size/zCard	Y	
	ZCOUNT	ZSetOperations.count	Y	
	ZINCRBY	ZSetOperations.incrementScore	Y	
	ZRANGE	ZSetOperations.range rangeWithScores	Y	
	ZRANGEBYSCORE	ZSetOperations.rangeByScore rangeByScoreWithScores	Y	
	ZRANK	ZSetOperations.rank	Y	
	ZREM	ZSetOperations.remove	Y	
	ZREMRANGEBYRANK	ZSetOperations.removeRange	Y	
	ZREMRANGEBYSCORE	ZSetOperations.removeRangeByScore	Y	
	ZREVRANGE	ZSetOperations.reverseRange reverseRangeWithScores	Y	
	ZREVRANGEBYSCORE	ZSetOperations.reverseRangeByScore reverseRangeByScoreWithScores	Y	
	ZREVRANK	ZSetOperations.reverseRank	Y	
	ZSCORE	ZSetOperations.score	Y	
	ZUNIONSTORE	ZSetOperations	Y	

		.unionAndStore		
	ZINTERSTORE	ZSetOperations.intersectAndStore	Y	
	ZSCAN	ZSetOperations.scan	Y	
	ZRANGEBYLEX	ZSetOperations.rangeByLex	Y	
	ZLEXCOUNT	N		
	ZREMRANGEBYLEX	N		

HyperLogLog operations

Data structure/Object	Operation	Spring Data Redis method	EDAS support for EagleEye tracing (Y/N)	Remarks
HyperLogLog	PFADD	HyperLogLogOperations.add	Y	
	PFCOUNT	HyperLogLogOperations.size	Y	
	PFMERGE	HyperLogLogOperations.union	Y	key: dest:\${destination}

Pub/Sub (publish/subscribe) operations

Data structure/Object	Operation	Spring Data Redis method	EDAS support for EagleEye tracing (Y/N)	Remarks
Pub/Sub	PSUBSCRIBE	N		
	PUBLISH	RedisOperations.convertAndSend	Y	key: msg:\${msg}
	PUBSUB	RedisMessageListenerContainer. .setMessageListeners .addMessageListener	N	
	PUNSUBSCRIBE	N		
	UNSUBSCRIBE	N		

Transaction operations

Data structure/Object	Operation	Spring Data Redis method	EDAS support for EagleEye tracing (Y/N)	Remarks
Transaction	DISCARD	RedisOperations.discard	Y	
	EXEC	RedisOperations.exec	Y	key: execRaw
	MULTI	RedisOperations.multi	Y	
	UNWATCH	RedisOperations.unwatch	Y	
	WATCH	RedisOperations.watch	Y	

Script operations

Data structure/Object	Operation	Spring Data Redis method	EDAS support for EagleEye tracing (Y/N)	Remarks
Script	EVAL	ScriptExecutor.execute	Y	key: Null
	EVALSHA	ScriptExecutor.execute	Y	key: Null
	SCRIPT EXISTS	RedisScriptingCommands.scriptExists	N	
	SCRIPT FLUSH	RedisScriptingCommands.scriptFlush	N	
	SCRIPT KILL	RedisScriptingCommands.scriptKill	N	
	SCRIPT LOAD	RedisScriptingCommands.scriptLoad	N	

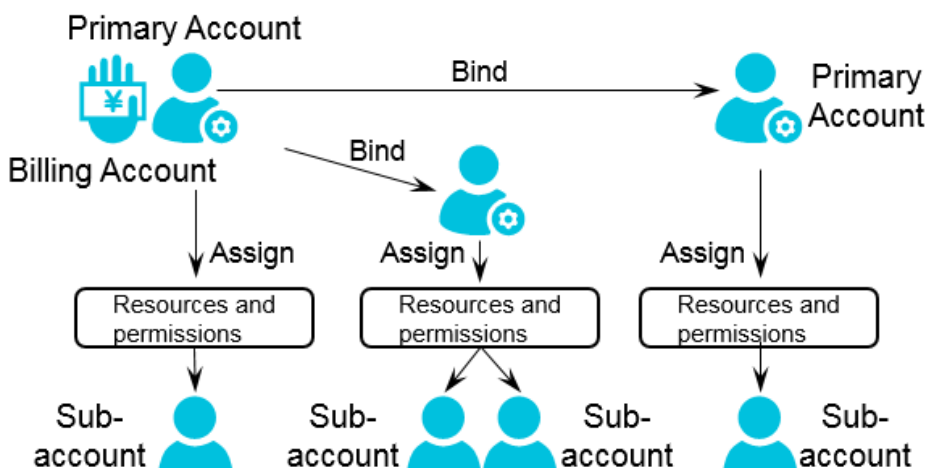
Account management

Account system introduction

EDAS provides a comprehensive primary and sub-account management system to help you achieve enterprise-level account management and improve enterprise information security. A primary account can assign permissions and resources to multiple sub-accounts on demand in accordance with the minimum permission principle, which lowers the risks of enterprise information security and reduces the load of the primary account.

Before adopting the account system of Alibaba Cloud Resource Access Management (RAM), EDAS is developed with a strict primary and sub-account system to implement separation between users and permissions. After being upgraded on July 2016, EDAS also supports the RAM primary and subaccount system.

The following figure shows the EDAS account system.



The billing account is a primary account used to buy EDAS service. If multiple departments of an enterprise need to use EDAS, a user can create a billing account to buy the EDAS product and then binds it with multiple primary accounts to give other primary account users access to EDAS. This helps customers maximize their benefits.

Note: A billing account can be bound with a maximum of five primary accounts.

Billing account and primary account:

All billing accounts are primary accounts, but not all primary accounts are billing accounts.

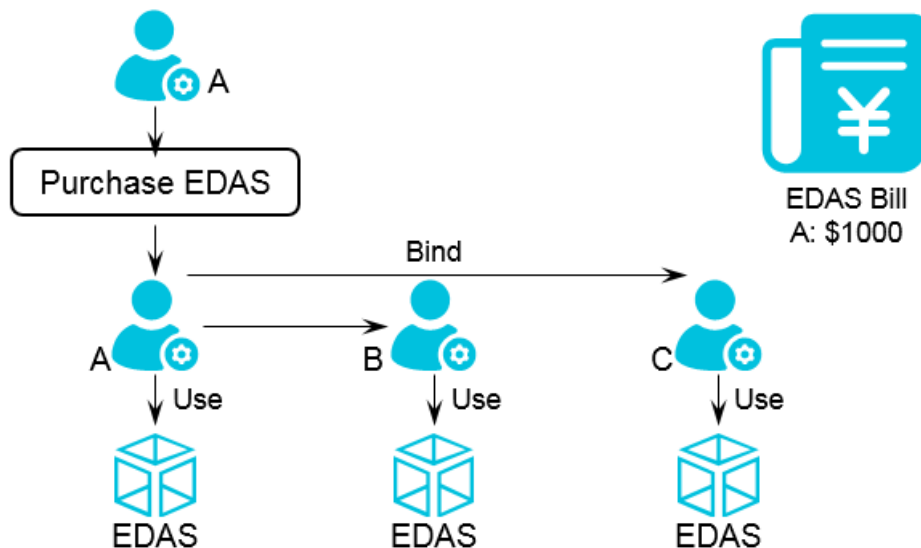
Each primary account is an independent account that owns all resources bought with the account and has full permissions on EDAS except that it cannot bind other primary accounts.

A billing account and a primary account are two independent accounts of Alibaba Cloud. The payment binding relationship between billing account and primary account is only effective for the purpose of EDAS purchase. The billing account cannot be used to buy any other resources than EDAS on behalf of the primary account. A primary account should still buy resources such as ECS and SLB by itself even if it is bound to a billing account for EDAS purchase. (For details about specific resources, see [Resource management](#).)

The following describes three use cases of the EDAS account system.

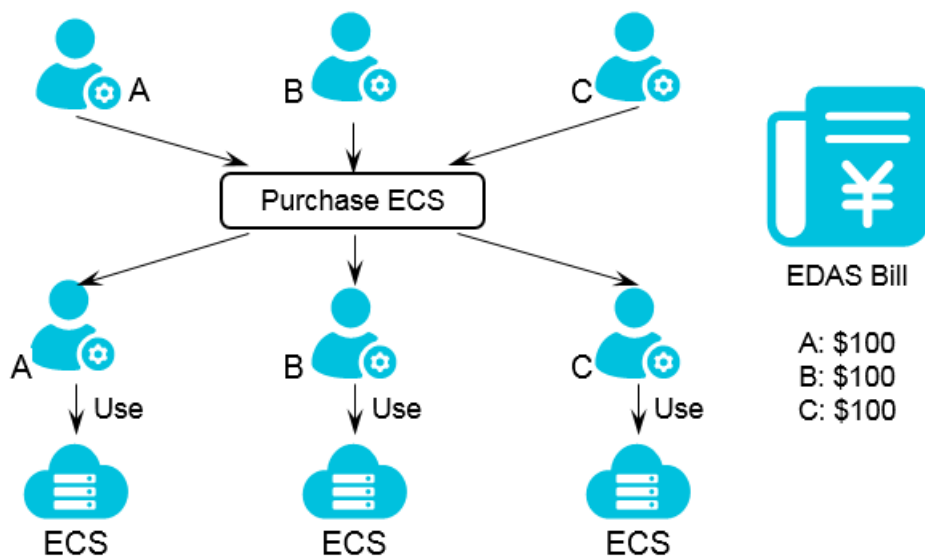
Scenario 1

A company uses Account A to buy EDAS. Account A is a billing account and also a primary account. The company binds this billing account with the primary accounts (Account B and Account C) of two departments to enable the departments to access EDAS without purchasing EDAS again. See the following figure.



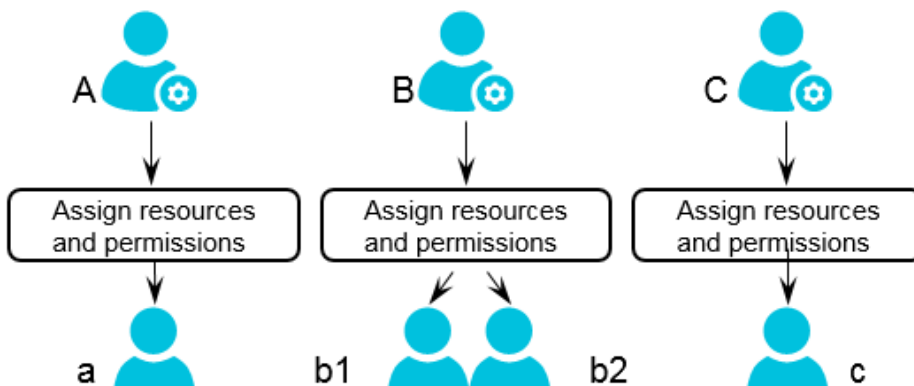
Scenario 2

If users of Account B and Account C require the full functions of EDAS, for example, to create or run applications, the two accounts rather than Account A must be used to buy resources such as ECS, as shown in the following figure.



Scenario 3

After resources are prepared, sub-accounts are created under the three primary accounts and used to allocate and manage permissions and resources. Sub-account a is created under Account A and assigned all ECS resources and permissions. Two roles, application administrator and operation administrator, are created under Account B and allocated to Sub-account b1 and Sub-account b2, respectively. A role for application query is created under Account C and allocated to Sub-account c.



Primary accounts

In EDAS, a primary account owns all resources under the account and has full operation permissions on EDAS. The primary account used to buy the EDAS product is also the billing account. After a company buys the EDAS service, it can bind the billing account with other primary accounts of the company to give the primary account users access to EDAS without secondary purchase. This helps customers lower resource costs.

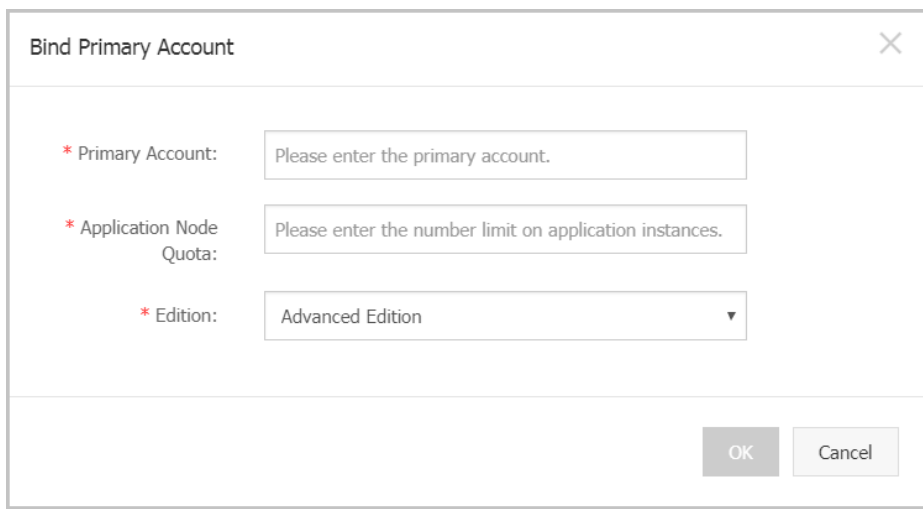
Bind a primary account

To bind a primary account, follow these steps:

In the EDAS console, select **Accounts** > **Primary Accounts** in the left-side navigation pane.

Click **Bind Primary Account** in the upper-right corner.

Enter a primary account name, set the maximum number of applications allowed for this account, select a product edition, and click **OK**.



The screenshot shows a dialog box titled "Bind Primary Account" with a close button (X) in the top right corner. The dialog contains three required fields, each marked with a red asterisk:

- * Primary Account:** A text input field with the placeholder text "Please enter the primary account."
- * Application Node Quota:** A text input field with the placeholder text "Please enter the number limit on application instances."
- * Edition:** A dropdown menu currently showing "Advanced Edition" with a downward arrow.

At the bottom right of the dialog, there are two buttons: "OK" and "Cancel".

Primary Account: It must be a valid Alibaba Cloud account that has never been used to buy the EDAS service.

Application Quota: Maximum number of applications that can be owned by the primary account and its sub-accounts. When the billing account allocates a quota to each primary account, the sum of the quotas of all primary accounts bound to it cannot be greater than the total application quota of the billing account.

Product Edition: The product edition that the billing account allocates to each bound primary account must be the same as that of the billing account.

Note:

- A billing account can be bound with a maximum of five primary accounts.
- Open a ticket to unbind primary accounts from the billing account, .

For bound primary accounts:

- The number of actual applications you have should not exceed your application instance quota.

The **actual number of applications** of the **billing account** plus the sum of the application **quotas of other primary accounts** bound to the billing account should not exceed the application quota of the billing account.

Primary Accounts Bind Primary Account			
Instances with Applications Deployed: 125			
Primary Account	Application Node Quota	Instances with Applications Deployed	Product Series
edas_test1@aliyun-test.com (Billing Account)	1000	125	Professional Edition
ddf101616	1	0	Professional Edition

Note: One billing account can bind 5 primary accounts. If you need to unbind an account, please open a ticket.

Role management

A primary account can define different operation permissions for its sub-accounts by creating different roles.

In the EDAS console, select **Roles** in the left-side navigation pane.

Click **Create Role** in the upper-right corner.

Enter a role name, select the permissions from the left-side field and add to the right, and click **OK**.

Create Role
✕

***Role:**

Optional Permissions

- Overview
- View
- Resources
 - ECS
 - View ECS
 - Create ECS
 - Synchronize ECS
 - Install Agent
 - Restart ECS
 - Migrate ECS
 - SLB
 - View SLB
 - Create SLB
 - VPC

total 9 | Permission

Add >>

<< Delete

Selected Permissions

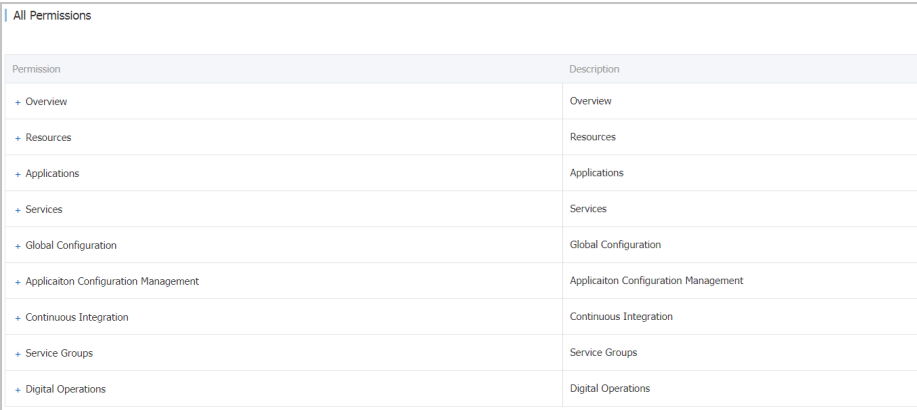
Selected 0 | Permission

You can view, manage Permissions or Delete roles on the **Roles** page.

View all permissions

You can view all permissions of the EDAS system in the console.

In the EDAS console, select **Accounts** > **All Permissions** in the left-side navigation pane, and unfold the lists to display specific information.



Permission	Description
+ Overview	Overview
+ Resources	Resources
+ Applications	Applications
+ Services	Services
+ Global Configuration	Global Configuration
+ Application Configuration Management	Application Configuration Management
+ Continuous Integration	Continuous Integration
+ Service Groups	Service Groups
+ Digital Operations	Digital Operations

Sub-accounts

EDAS supports the account system of Alibaba Cloud Resource Access Management (RAM). You can create RAM sub-accounts under your primary account to avoid sharing your account key with other users. You can also assign minimum permissions to sub-accounts as needed to separate responsibilities and conduct efficient enterprise management. This topic introduces the following subjects:

- Introduction to RAM sub-accounts
- Create a RAM sub-account
- Use a RAM sub-account for EDAS logon
- Authorize a RAM account
- Unbind a RAM account

Introduction to RAM sub-accounts

When you use your primary account in EDAS, you can allocate different roles and resources to the sub-accounts under the primary account so as to complete different types of jobs with different user identities, such as application administrator (with the permissions to create, start, query, and delete applications) and operation administrator (with the permissions to view resources, check application

monitoring, and manage alarm policies, throttling and degradation rules). The primary and sub-account permission mode is similar to the classification of system user and common user in Linux. A system user can grant and revoke permissions to/from common users.

RAM sub-accounts:

- A RAM account is created by a primary account in the RAM system. Validity check is not required for the RAM account, but the account name must be unique under the primary account.
- A dedicated logon portal is available for RAM accounts. For details about the logon portal, see relevant description in the RAM console.

Create a RAM sub-account

Follow these steps:

In the EDAS console, choose **Accounts** > **Sub-Account** in the left-side navigation pane.

Click **Bind Sub-Account** in the upper-right corner to go to the RAM console. After you create a RAM user in the RAM console, by default, the RAM user is also a sub-account under your primary account in EDAS.

Click **Users** in the RAM console.

Click **Create User** in the upper-right corner. In the **Create User** dialog box that appears, enter your logon name and other information, and click **OK**. The user management page shows a new username, indicating that a RAM user is successfully created.

Note: The logon name must be unique under the primary account.

Click the **logon name/displayed name** link of the RAM user to go to the user information page.

Click **Enable Console Logon** in **Web Console Logon Management**. The password setting dialog box appears.

1. Enter a **new password**, and select **Require to reset the password upon next logon** as needed.

After the preceding steps are complete, a RAM user with the console logon permission is successfully created.

Use a RAM sub-account for EDAS logon

Following these steps:

Click the **RAM User Logon Link** on the **Dashboard** page of the RAM console.

Note: The RAM user logon link varies depending on different primary accounts.

Enter the sub-account name and password on the RAM user logon page, and click **Logon** to go to the RAM console.

Note:

Enterprise Alias: Already exists in the logon link of the sub-account.

Sub-account Name: Logon name that the primary account sets when creating the RAM user.

Sub-account Password: Password that the primary account sets when enabling console logon for the RAM account. If **Require to reset the password upon next logon** is selected, the RAM account is required to reset the password after initial logon to the console. The new password is used for future logons.

Click **Products & Services** in the top navigation bar of the RAM console, and click **Enterprise Distributed Application Service (EDAS)** under the Middleware category to go to the EDAS console.

Authorize a RAM account

Two authorization methods are available:

- RAM authorization
- EDAS authorization

The two authorization methods are mutually exclusive. After you authorize a RAM account in RAM, you cannot authorize the same account in EDAS. You must revoke RAM authorization in the RAM console before you can perform EDAS authorization in the EDAS console. To authorize a RAM account in EDAS, ensure that the account is not authorized in RAM.

RAM authorization is performed at the EDAS service level, indicating that a RAM-authorized account has full permissions on EDAS. RAM authorization and revocation must be performed in the RAM console.

The procedure of RAM authorization is as follows:

In the RAM console, click **Users** in the left-side navigation pane, select the user to be authorized, and click **Authorize** in the “Action” field on the right.

Enter **EDAS** in the search box in the left part of the dialog box, select **AliyunEDASFullAccess** and add this option to **Selected Authorization Policy Name** on the right, and click **OK** to grant full EDAS permissions to the account.

After authorization is complete, use the primary account to log on to EDAS and select **Accounts > Sub-Accounts** on the left-side menu bar. Then the page lists the permissions, resources, and applications granted to the RAM account. The authorization function is disabled for the RAM account in the EDAS console.

To revoke RAM authorization, follow these steps:

In the RAM console, click **Users** in the left-side navigation pane, select the user, and click **Authorize** in the **Actions** field.

Move the **AliyunEDASFullAccess** option in the right-side field to the left and click **OK**.

After authorization is revoked, use the primary account to log on to EDAS and select **Accounts > Sub-Accounts** in the left-side navigation pane. Then the page shows that all resources and permissions of the RAM account are revoked. The authorization function is enabled for the RAM account on EDAS.

RAM users that are not authorized for EDAS can manage roles and resource groups, authorize applications, and perform other operations in EDAS, but cannot perform the unbind operation.

Supported operations:

Manage roles

A primary account can assign a role to a sub-account to grant the role-specific permissions to this sub-account. The procedure of role management is as follows:

In the EDAS console, choose **Accounts > Sub-Accounts** on the left-side menu bar. Find the sub-account to be authorized and select **Manage Roles** in the “Actions” field on the right.

Select roles on the left side and click **>** to add the roles to the right side, then click **OK**.

Select **Accounts > Roles** on the left-side menu bar. The role name is displayed in the **Roles**

page.

Authorize an application

A primary account can assign an application to a sub-account to grant the application ownership to this sub-account. The procedure of application authorization is the same as that of role management.

Note: Application authorization only grants the application ownership to the sub-account. To grant application operation permissions (to start or delete the application, for example) to the sub-account, you need to assign a role to the sub-account. Therefore, application authorization is typically followed by role authorization.

Authorize a resource group

A primary account can assign a resource group to a sub-account to give the sub-account access to resources in the resource group. For details about the concept of resource group, see [Resource group](#). The procedure of resource group authorization is the same as that of role management.

Unbind a RAM account

Follow these steps:

Log on to the RAM console.

Click **Users** in the left-side navigation pane, find the account to be unbound, and click **Delete** in the "Actions" field on the right.