Enterprise Distributed Application Service (EDAS)

User Guide

User Guide

Resources

Overview

The resources on the EDAS platform include Elastic Compute Service (ECS), Server Load Balancer(SLB), and Virtual Private Cloud (VPC). You can view and use these resources in the EDAS console. The EDAS console allows you to manage the use of these resources on the application layer, but does not perform actions such as resource purchasing or releasing. In addition, you can group EDAS resources and assign permissions to the groups, thereby controlling resource usage permissions with primary accounts and sub-accounts.

ECS

View ECS instances

After you install EDAS Agent on your ECS instances and synchronize them to EDAS, you can view and manage your ECS instances in the EDAS console.

Log on to the EDAS console.

In the left-side navigation pane, select **Resources** > **ECS** to go to the **ECS** page.

Select a specific region to see the ECS instances with EDAS Agent installed within this region.

To view the instances in a specific namespace within a region, select the namespace under the region.

ECS information/status descriptions:

- **Instance ID/Name**: The instance ID that is automatically generated by the system and the instance name that you specified. Click the instance ID/name to go to the ECS Console.
- **Network Type**: ECS instances can be located in a VPC or classic network.
- **IP Address**:
  - For ECS instances in a classic network, this field shows the intranet IP address and Internet IP address of the instances.
  - For ECS instances in a VPC, this field shows the private IP addresses and EIPs (if any).
- **Specifications**: The instance' s CPU and memory details.
- **Namespace**: The namespace where the ECS instance is located. Region information is displayed if there are no namespaces.
- **Cluster**: The cluster which the instance belongs to.
- **Deployed Application**: The applications deployed on this instance. Click an application name to go to the Application Details page.
- **Agent Status**:
  - If EDAS Agent has been installed successfully on the instance, its status is Online or Docker Online.
  - If EDAS Agent has not been installed, the status is Unknown.

Click the buttons in the upper-right corner of the ECS page to perform the following actions for the ECS instance:

Button	Description
Import ECS	Installs EDAS Agent on a purchased ECS instance and then synchronizes it to EDAS.
Create Instances	Takes you to the ECS purchase page on the Alibaba Cloud website, where you can purchase and create new ECS instances. For more information, see the Create an instance.

Import ECS instances

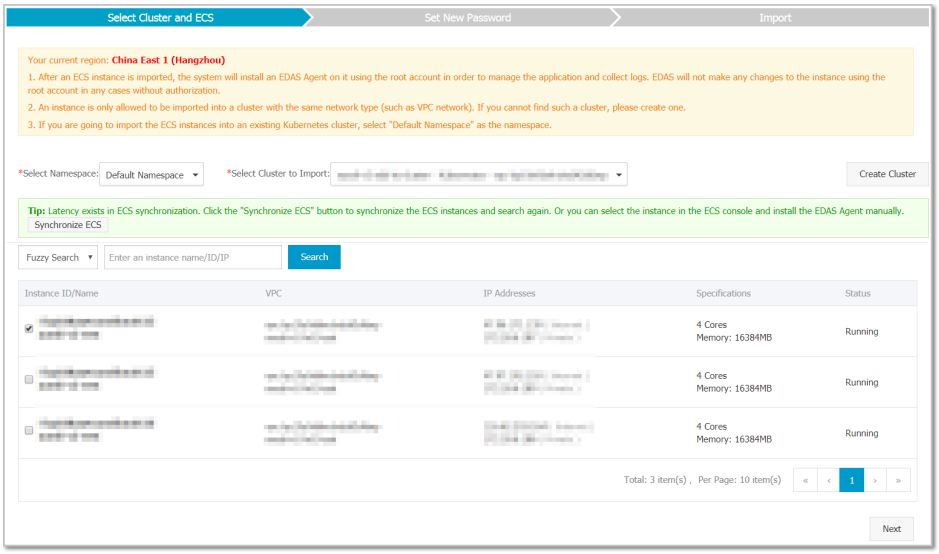
If you have purchased an ECS instance without EDAS Agent installed, you have to import the ECS instance, which automatically installs EDAS Agent on the ECS instance and then synchronizes the ECS instance to the EDAS console.

In the left-side navigation pane of the EDAS console, select **Resources** > **ECS**.

On the **ECS** page, click **Import ECS** in the upper-right corner.

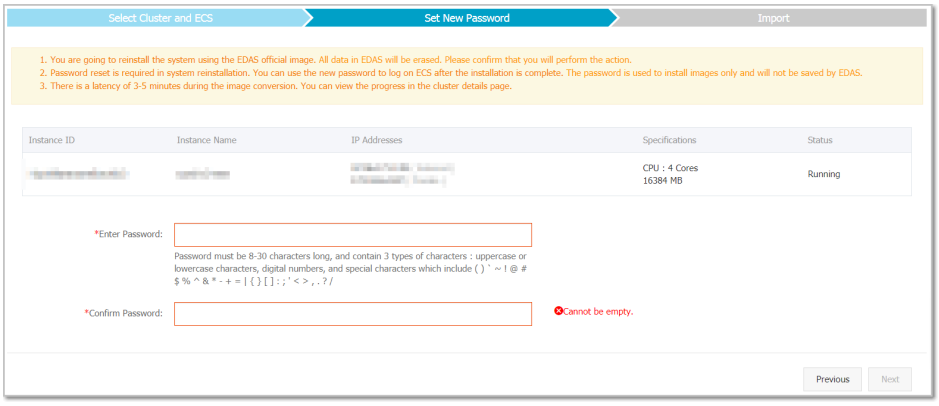
In the **Select Cluster and ECS** step, select the namespace and cluster that you want to import your ECS to, and then select the ECS instance and click **Next**.

- After you select a cluster, the system automatically displays the ECS instances available for this cluster.
- If you wish to create a new cluster, click **Create Cluster**.



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

After you confirm the ECS import, the system clears all data on the ECS instance and uses the EDAS official image to reinstall the operating system. **Make sure you remeber your new password**.



On the Import ECS page, 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. Then, following the prompts, click **Back** to go to the cluster details page under **Cluster Management**. In the **Cluster Deployment Information** area, you can view the import status and progress.

When the status of the ECS instance changes from **Converting** to **Online**, then the ECS instance is imported successfully.

Initialize EDAS Agent

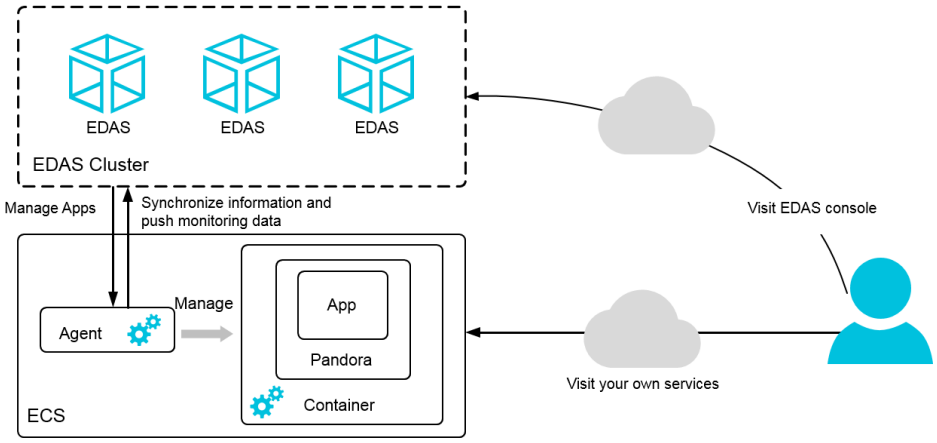
EDAS Agent overview

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 these application-based management functions, EDAS Agent is also responsible for communication between the EDAS console and your applications. Here is a simple example. EDAS Agent obtains and submits the information about whether a service from an application is published on a given ECS instance correctly and promptly.

Note: The above functions are transparent to users. You only have to install the 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:

- Automatic installation by using EDAS base image when you buy ECS
- Automatic installation by importing ECS instance
- Manual installation by executing script

Before you begin

- JDK 8 is installed in 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 user.
- Currently, EDAS Agent can be installed and run on **64-bit CentOS 6.5/6.8/7.0/7.2** 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, to upgrade EDAS Agent, simply run the same script again.
- The script for installation is region specific. You must switch to the appropriate region before clicking **Install EDAS Agent**.

Automatic installation by using EDAS base image when you buy ECS

The simplest and easiest method for installing EDAS Agent is to use an EDAS base image when you

purchase an ECS instance.

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

In the **Choose the Operating System** section of the purchase page, select **Marketplace Image**. Then, click **Select from image market (including operating system)**.

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

Select **EDAS JAVA Environment (common ECS)** in the results. The latest version is selected by default. Then, click **Use** to select it as the EDAS base image.

Click **Buy Now** to purchase the ECS instance.

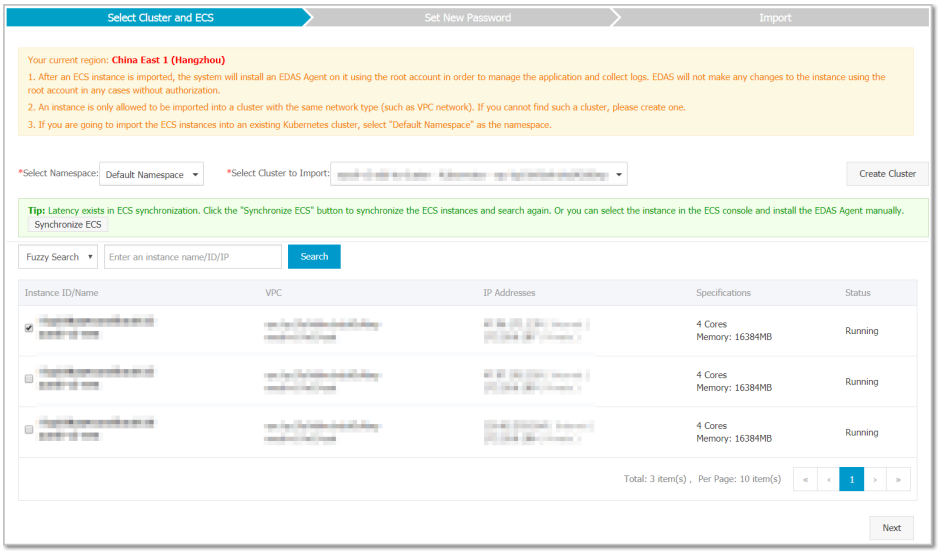
Automatic installation by importing ECS instance

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

In the left-side navigation pane of the EDAS console, select **Resources** > **ECS**. In the upper-right corner of the Instance List page, click **Import ECS**.

In the **Select Cluster and ECS** step, select the namespace and cluster that you want to import your ECS to, and then select the ECS instance and click **Next**.

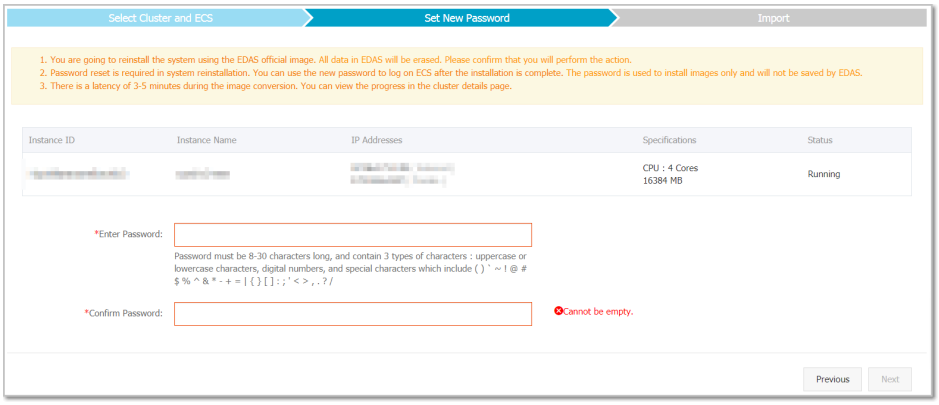
- After you select a cluster, the system automatically displays the ECS instances available for this cluster.
- If you wish to create a new cluster, click **Create Cluster**.



Note: You can also select **Switch to Manual Installation** in the upper-right corner of the page and manually install EDAS Agent by executing the script.

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

After you confirm the ECS import, the system clears all data on the ECS instance and uses the EDAS official image to reinstall the operating system. **Make sure you remeber your new password**.



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

It takes about five minutes to import an ECS instance. Then, following the prompts, click **Back** to go to the cluster details page under **Cluster Management**. In the **Cluster Deployment Information** area, you can view the import status and progress.

When the status of the ECS instance changes from **Converting** to **Online**, the ECS instance is imported successfully.

Manual installation by executing script

Log on to the EDAS console and choose **Resources** > **ECS** in the left-side navigation pane.

In the upper-left corner of the page, select the region where the ECS instance is, for example, "China East 1".

Note: Make sure you select the correct region, otherwise EDAS Agent installation may fail.

On the ECS page, select the appropriate ECS instance and click **Import ECS** in the upper-right corner.

On the **Import ECS** page, click **Switch to Manual Installation** in the upper-right corner, then click **Copy** to copy the script.

Note: You can also click **Switch to Image Installation** In the upper-right corner of the page to install EDAS Agent by importing an ECS instance.

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.

Verify result

After installing EDAS Agent, select **Resources** > **ECS** in the left-side navigation pane of the EDAS console. On the Instance List page, select the appropriate region to view the **Agent Status** for instances in the region.

If EDAS Agent is installed successfully, its status is **Online** or **Docker Online**.

If EDAS Agent installation fails, its status is **Exception**.

Upgrade EDAS Agent

The process for upgrading EDAS Agent is the same as that for installing it. Refer to Manual

installation by executing script section for the detailed steps.

The script automatically reinstalls and restarts the EDAS Agent.

SLB

EDAS synchronizes the Server Load Balancer(SLB) instance that you purchased to the EDAS console, allowing you to manage the SLB instances and configure load balancing functions.

View SLB instances

Log on to the EDAS console.

In the left-side navigation pane, select **Resources** > **SLB**.

Select a region to view SLB instances in this region.

This page shows the following information:

- **Instance ID/Name**: The instance ID that is automatically generated by the system and the instance name that you specified. Click the instance ID to go to the SLB Console.
- **IP address**: The IP address of the SLB instance.
- **Backend server**: An ECS instance added in the SLB console, used to receive the requests distributed by the SLB instance.
- **Status**: The status of the SLB instance, including running or stopped. Expired SLB instances are not displayed.

Note: If you want to create SLB instance, click **Create Instance** in the upper-right corner to go to the SLB purchase page. For more information, see the SLB documentation.

Configure Server Load Balancer

For load balancing configuration information, see Create SLB instances and Configure SLB instances.

VPC

View VPC instances

Alibaba Cloud provides two network types:

Classic network

Cloud products in 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 network

VPC networks are virtual private clouds that allow custom isolation settings. You can define the network topology and IP addresses. 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 menu bar, select **Resources** > **VPC**.

Select a region to view the VPC instance information in this region.

- **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**: The VPC name you specified when creating the 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 **Actions** field on the Instance List page to install the log collector. For detailed instructions, see Install log collector.

Install log collector

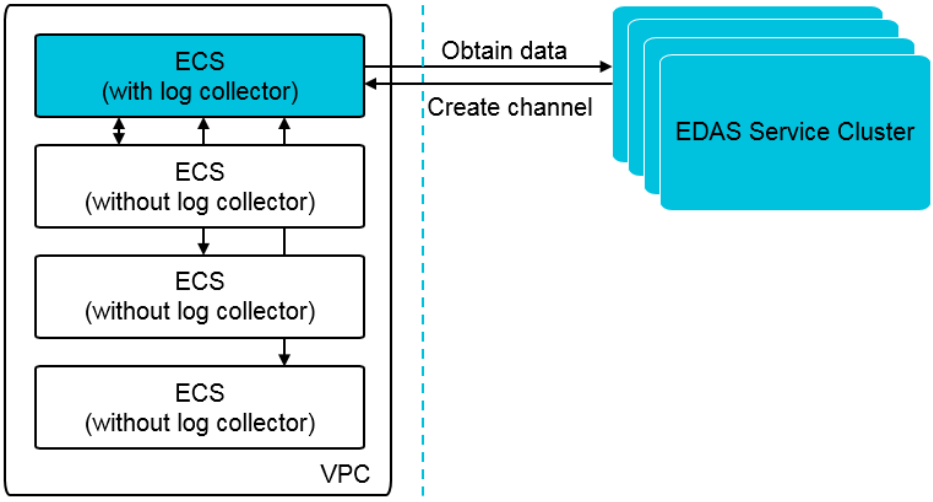
Log collector overview

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

Alibaba Cloud' s network 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 isolated from the servers. EDAS provides a special utility for VPCs: log collector.

The log collector has two components: Server and Client. SProxy is the log collector client installed on user machines, as shown below:



Install log collector

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

Log on to the EDAS console.

On the left-side menu bar, select **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 installation script.

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

After the installation is complete, manually run the netstat -ant|grep 8000 command.

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

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

Resource groups

Resource group overview

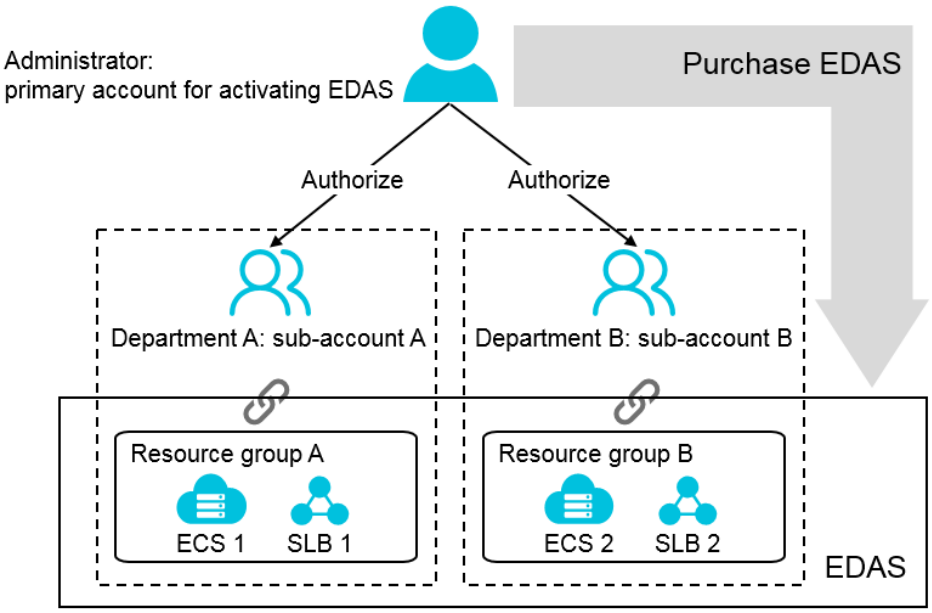
Resource groups are groups of EDAS resources, which can be ECS instances and SLB instances, but not VPCs. You can assign permissions to access resource groups to your sub-accounts, and each sub-account has the right to operate on all the resources in the specified group. For more information on primary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

Typical use case

- 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 sub-accounts respectively for Departments A and B. Department A has a couple of ECS instances and SLB instances for deploying user-related applications.
- The company creates a resource group in EDAS, adds Department A's ECS instances and SLB instances to the resource group, and grants permissions for this resource group to the

sub-account of Department A.
- Department A can use its sub-account to operate on the resources in this resource group only.

As shown in the following figure.



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**.

In the left-side navigation pane, choose **Resources** > **Resource Group**.

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

In the displayed list, you can view information about the resource groups, including resource group names, descriptions, ECS instances (intranet/Internet IP addresses), and SLB instances.

Create a resource group

1. On the **Resource Groups** page, click **Create Resource Group** in the upper-right corner.
2. Enter resource group name and description and click **OK**.

Add an ECS instance to the resource group

1. On the **Resource Groups** page, click **Bind ECS** in the **Actions** field of the resource group you want to add an ECS instance to.
2. On the **Bind ECS** dialog box, select the ECS instance and click **OK**.

Add a Sever Load Balancer instance to the resource group

1. On the **Resource Groups** page, click **Bind SLB** in the **Actions** field of the resource group you want to add a SLB instance to.
2. On the **Bind SLB** dialog box, select the SLB instance and click **OK**.

Edit a resource group

1. On the **Resource Groups** page, click **Edit** in the **Actions** field of the resource group you want to edit.
2. On the **Edit Resource Group** dialog box, edit the resource group name and description and click **OK**.

Grant resource group permissions to a sub-account

1. Log on to the EDAS console using the primary account.
2. In the EDAS console, select **Accounts** > **Sub-Accounts** in the left-side navigation pane.
3. Click **Authorize Resource Group** in the **Actions** field of the user you want to grant resource group permissions.
4. In the Authorize Resource Group dialog box, select the appropriate resource group and click **OK**.

Delete a resource group

1. In the EDAS console, select **Resources** > **Resource Groups** in the left-side navigation pane.
2. Click **Delete** in the **Actions** field of the resource group to delete.
3. Click **OK** in the confirmation dialog box.

Clusters

Create a cluster

It takes two steps to create a Swarm cluster for applications:

1. Create a cluster
2. Add ECS instances to the cluster

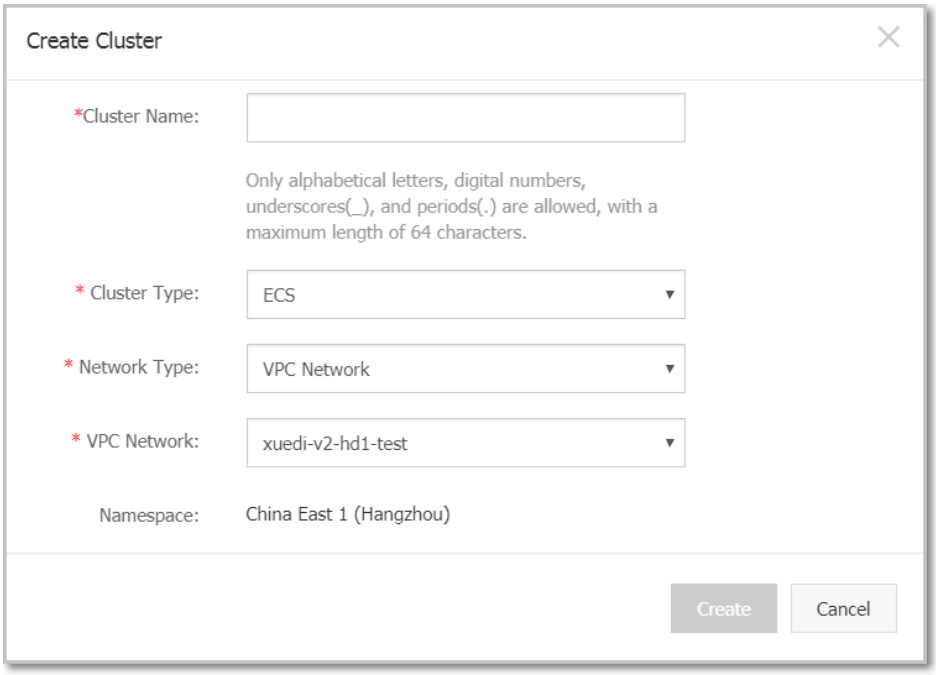
You can remove ECS instances from the cluster as needed.

Create a cluster

Log on to the EDAS console.

In the left-side navigation pane, select **Resources** > **Clusters** to go to the **Clusters** page.

In the upper-right corner, click **Create Cluster** and then set the fields in the **Create Cluster** dialog box.



Cluster field descriptions:

- **Cluster Name**: Up to 64 characters are allowed and can contain alphabetical letters, digital numbers, underscores, and periods.
- **Cluster type**: Select **ECS**.
- **Network type**: Classic network or VPC. Select the network type as needed. If you select VPC, you must select the specific VPC from the **VPC Network** drop-down menu.

- **VPC Network**: Select a VPC from the drop-down menu.
- **Namespace**: The namespace where the cluster is located. Region is displayed if no namespace is specified. Cannot be selected.

After setting the fields, click **Create** to create the cluster.

After the cluster is created, it 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

Adding a Swarm cluster host may involve converting a Docker host. The Docker host conversion uses a Docker image to reinstall the operating system of the ECS instance on which the Docker host runs. For more information, see Docker image deployment.

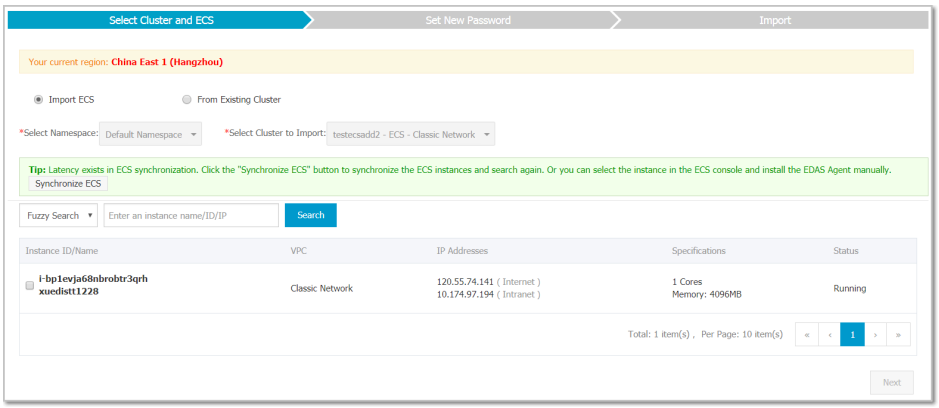
Procedure

On the cluster details page, click **Add Cluster Host** in the upper-right corner.

In the **Select Cluster and ECS** page, select the ECS instance and then click **Next**.

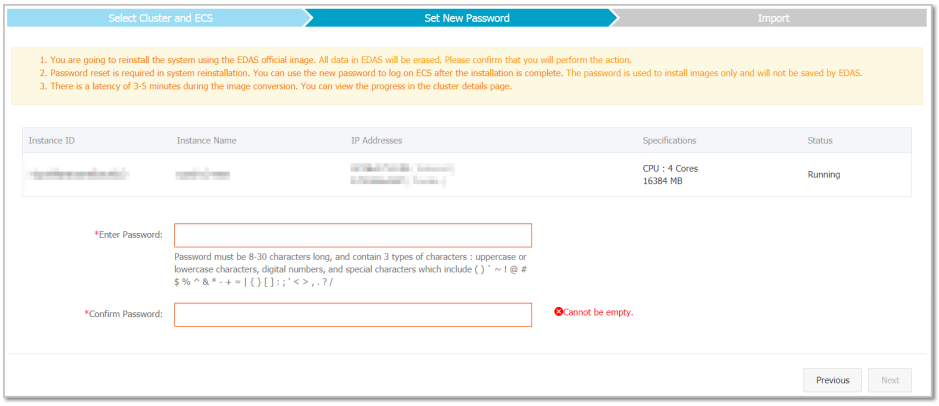
- **Import ECS**: Namespace and Cluster cannot be configured if you select this option. You can only import ECS instances in the specified namespace and cluster.
- **From Existing Cluster**: Select the namespace and cluster, and then choose the ECS instances to be imported.

If no instances are available, 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 more information, see the Create an instance.



On the **Set New Password** page, enter and confirm a new logon password for this Docker host. Then, click **Next**.

- An EDAS Docker image will be used to reinstall the operating system on the ECS instance. All data in the ECS instance is erased during the reinstallation. Make sure you still want to perform the conversion.
- 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 that appears, confirm that you would like to convert the ECS instance and click **Import**.

After the conversion starts, the Docker host status is shown in the cluster deployment information list.

- When the conversion starts, the Docker host status is **Converting**. The entire conversion process takes about five minutes. You can perform other operations in the EDAS Console during this period.
- There are two possible conversion results: **Conversion Failed** and **Online**.
- The **Offline** status indicates that EDAS cannot detect the Docker host’ s heartbeat.

Result verification

After the conversion, go back to the cluster details page and check the conversion result in the cluster deployment information area.

- If the conversion was successful, the health check shows **Online**. The node resources are ready for application deployment.
- If the conversion failed as the result of an exception, the host node is in the **Conversion Failed** status. In this case, identify and troubleshoot the cause of the failure and then click **Retry** next to the host node.

Remove a cluster host

Go to the cluster details page and find the host to remove in the list in the cluster deployment information area. Then, click **Remove** in the **Actions** field.

In the dialog box that appears, confirm the information of the host to be removed and then click **Remove**.

Note: Docker conversion is skipped if you re-add a Docker host that was previously removed from a cluster.

Manage clusters

View cluster list

Log on to the EDAS console.

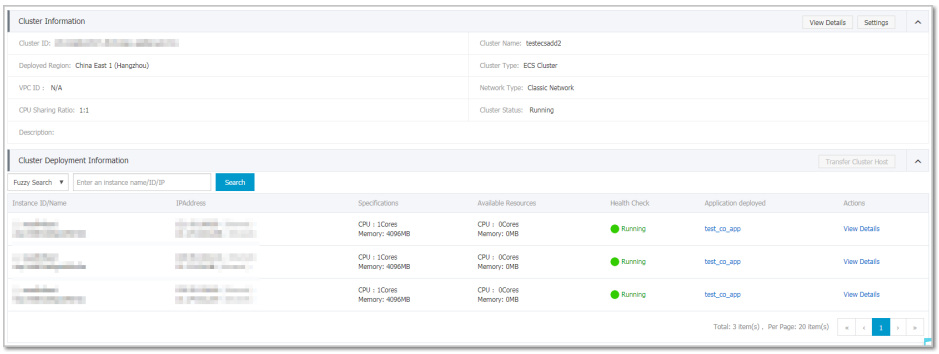
In the left-side navigation pane, select **Resources** > **Clusters**.

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

There are two network types, **VPC** and **Classic Network**.

View cluster details and deployed hosts

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



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 in 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 detailed cluster and host information.
- 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 in the **Deployed Application** field to go to the instance deployment information page associated with the details page for this application.

Transfer cluster host

In the Cluster Deployment Information area of cluster details page, select ECS instances and click **Transfer Cluster Host** on the left side.

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

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

In the **add a host to the ecs cluster** dialog, check the transfer information, then click **OK**.

Namespaces

Introduction

Resource (ECS) isolation is often required in the process of using EDAS. EDAS achieves resource isolation by allocating resources to different accounts. At the same time, resource isolation can also be achieved by using the isolation characteristics of a VPC network. However, both of these approaches will cause unnecessary burden to users in terms of operational difficulty (account switching) and implementation costs (need to purchase multiple VPCs), and cannot balance resource

isolation with unified account management. Furthermore, network isolation for resources is impossible in classic networks.

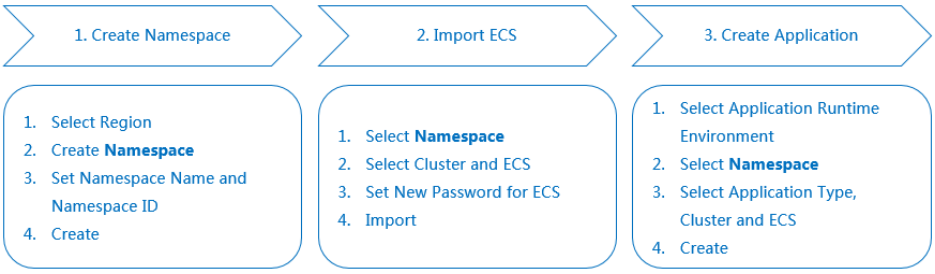
Namespace is designed to solve this problem in EDAS. In a classic network or VPC within a region, a namespace corresponds to an environment (consisting of one or more clusters), with natural logical isolation between different namespaces. At the same time, a single account can create multiple namespaces. Namespaces help users to completely isolate resources across multiple environments and can be managed uniformly with one account.

Scenarios

The main scenario for namespace is resource isolation. The workflow is as follows:

The detailed steps are described as follows:

1. Create Namespace
2. Import ECS
3. Create Application



After these actions are complete, resource isolation is achieved.

Furthermore, you can add or move hosts in the namespaces within the same region. For details, please refer to Create Cluster and Cluster Management. In addition to resource isolation, this also meets the requirements of single-account management.

Create a Namespace

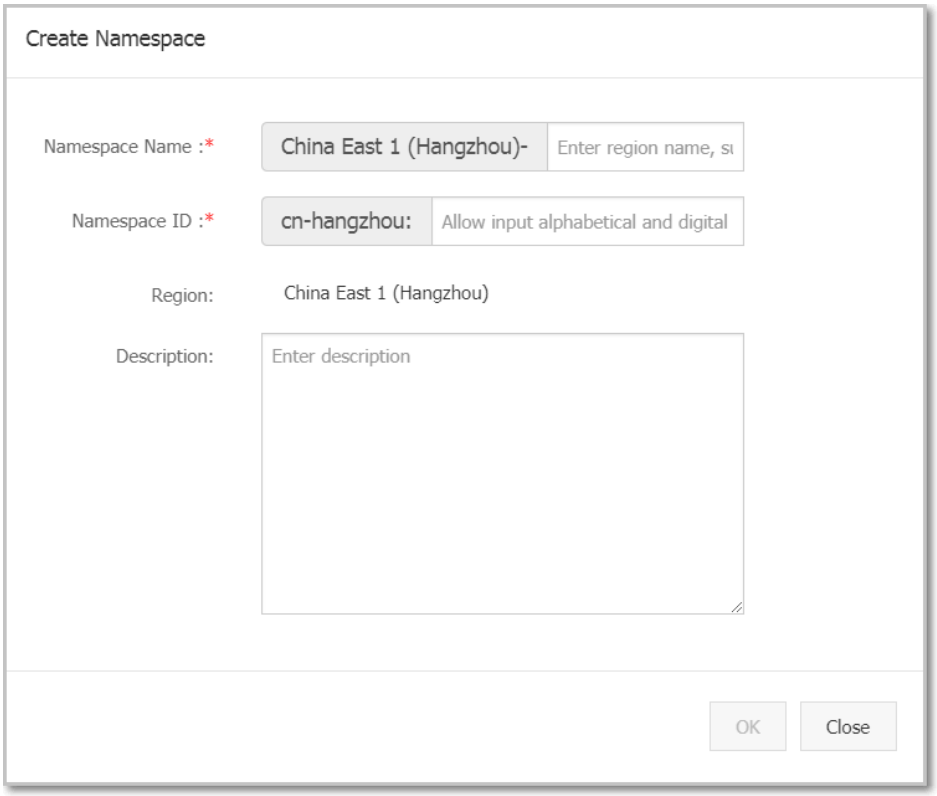
Login to the EDAS console.

From the left-side navigation pane select **Resources** > **Namespaces**.

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

In the **Create Namespace** dialog box, set the **Name**, **Namespace ID** and the description, then click **OK**.

Note: The Namespace ID prefix is automatically determined based on the selected region and cannot be edited. Only customizable sections can be set.



Edit Namespace

To edit the name of description of a namespace, go to the **Namespaces** page, and select **Edit** in the **Actions** column to the right of the namespace you want to edit.

Note: The Namespace ID cannot be edited.

Delete Namespace

Make sure the following conditions are met before you delete a namespace:

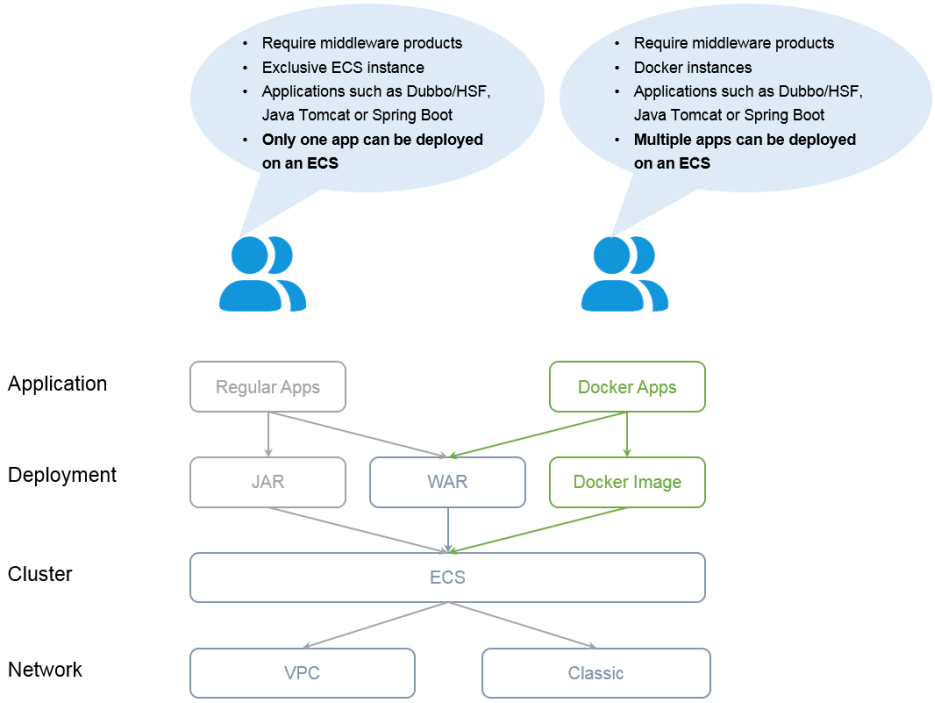
- **There are no ECS instances in the namespace.**
- **There are no clusters in the namespace**.

Application management

Application overview

EDAS applications are mainly divided into two types: regular applications and Docker applications.

- A common application is deployed using a JAR/WAR package.
- A Docker application is deployed in a Docker container.



Scenarios

- Regular applications: Apply to traditional deployment scenarios.
- Docker applications: Apply to scenarios in which customized container running environment is needed (for example, a decryption package, local certificate, and Java version) and higher user resource utilization is desired.

Life cycle management overview

Applications are the basic units in EDAS management. EDAS provides a complete life cycle

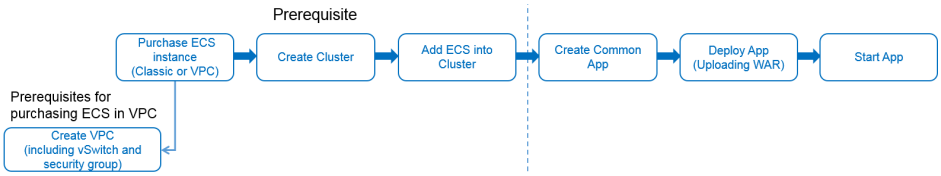
management process for applications, including application creation, deployment, startup/stop, and deletion.

Publish an application

Application publishing includes application creation, deployment, startup, and stop.

Application creation and deployment are usually performed consecutively. After creating an application, you will need to deploy it. The ways for creating and deploying regular and Docker applications are different.

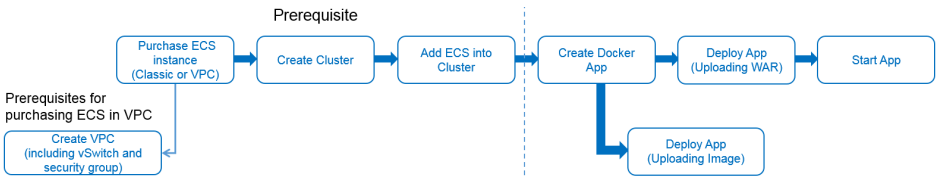
Publish a regular application: Create an EDAS container on your ECS instance and deploy the WAR package of the application to the container. The process is as follows:



Publish a Docker application: You can use either of the following ways to publish a Docker application:

- Upload a WAR package: Create Docker instances on the specified ECS instance in the specified Swarm cluster, create an EDAS container in each Docker instance, and deploy the WAR package of the application in the EDAS container of each Docker instance.
- Use images: Locally prepare a Docker image using the Docker and EDAS basic images (including the EDAS container) and the WAR package of the application, and upload it to the image repository.

The process for publishing a Docker application is as follows:



Manage an application

Management of regular and Docker applications is different. Main operations include:

- Roll back an application: After the rollback version is specified, applications can be automatically rolled back by group and by batch.
- Application scale in or out : You must add an ECS instance to scale up the application, or stop and deactivate the ECS instance to scale down the application.
- Delete an application: You must stop the application, put the ECS instance out of service and then delete the application.
- Manage instances：Manage the instances where the application is deployed.

Start/Stop an application

After a regular or Docker application is deployed, you must start it before using the application.

Set an application

Settings may include intranet and Internet load balancing, container configurations, JVM parameters, health check and basic information, and the health check URL.

Note:

- You can deploy, scale up, roll back, reset, and configure an application when the application is running or stopped.
- After the parameters of the Tomcat container and JVM are configured and saved, the related configuration files are modified. The changes take effect only after you restart the application.

Application life cycle management

Publish applications

Regular applications and Docker applications

- A regular application is deployed directly on an ECS instance.
- A Docker application is deployed through a Docker container. When a Docker application is deployed, multiple Docker container instances are created on the target ECS instance and the Docker application runs in a Docker container.

An ECS instance can be deployed with only one regular application but can be deployed with multiple applications running in different Docker containers.

You can decide to choose regular applications or Docker applications depending on your ECS resources/costs available and your specific needs for maintenance modes.

Note: Before publishing an application on an ECS instance, you must import the ECS instance (install the EDAS Agent and synchronize the ECS instance to EDAS). For details, see Import an ECS instance.

To publish an application, follow these steps:

Create an application

Deploy the application

Configure load balancing

Create an application

Two types of applications can be created:

Create a regular application

Create a Docker application

Create a regular application

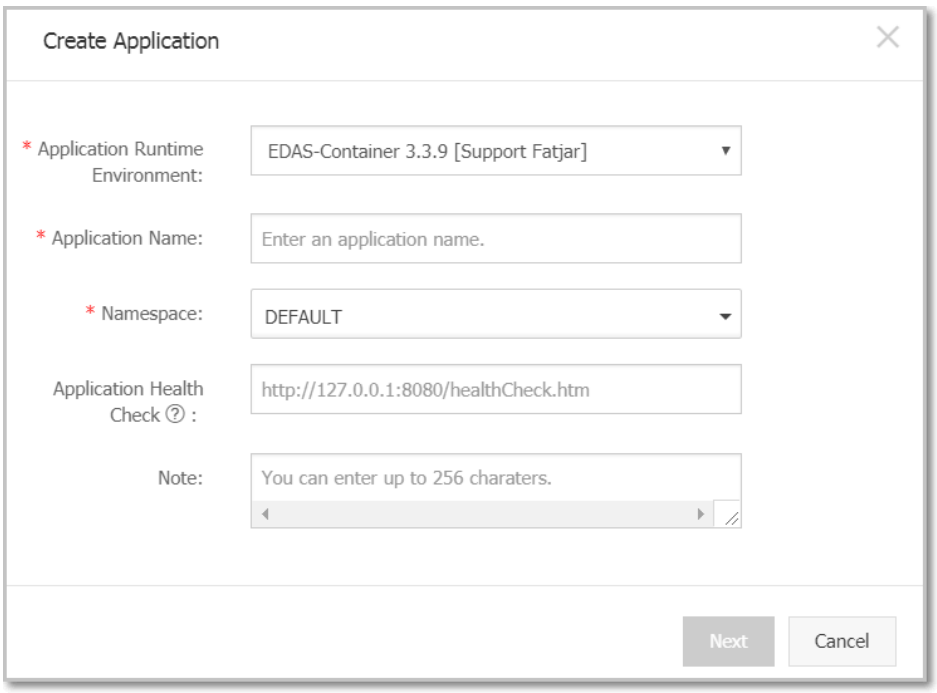
Note: Before you create an application, ensure that EDAS Agent has been successfully installed on the ECS instance.

Log on to the EDAS console.

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

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

In the **Create Application** dialog box, enter application information and click **Next**.



Field description:

Application Runtime Environment: Version of the Ali-Tomcat container in which the application runs. By default, the latest version is used.

Application Name: Name of the application, which must be unique under the corresponding primary account.

Namespace: The namespace where the application is located. After selected, the application can only be deployed on an ECS instance in this namespace.

Application Health Check: Optional.

If you configure a health check URL, EDAS periodically accesses the URL regularly and determines the survival status of the application according to the response code.

If no URL is specified, EDAS does not perform health check. This has no impact on the normal running of the application.

Note: Description of the application.

Select **Application Type** and **Cluster** and specific **ECS instances**.

Application Type: Select **Regular Application**.

Cluster and **ECS instances**: Select cluster and ECS instances where to create the application.

Network type deponds on the selected cluster.

After you complete the setting, click **Create**.

Create a Docker application

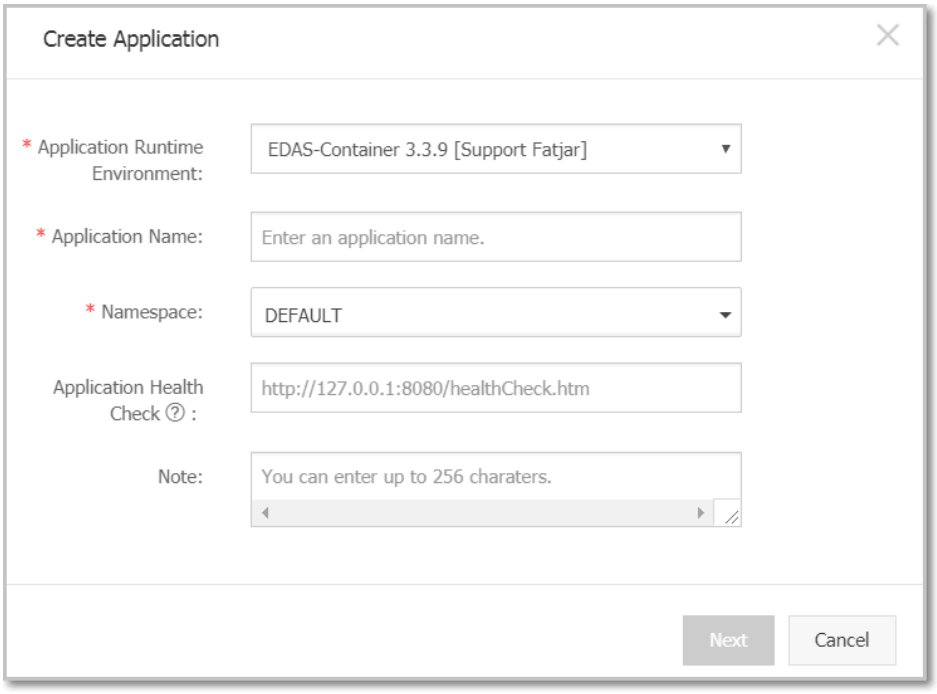
Before you create a Docker application, designate a cluster and ensure that the cluster has an ECS instance that has been converted to a Docker host.

Log on to the **EDAS console**.

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

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

In the **Create Application** dialog box, enter application information and click **Next**.



Description of fields:

Application Runtime Environment: Version of the Ali-Tomcat container in which the application runs. By default, the latest version is used.

Application Name: Name of the application, which must be unique under the corresponding primary account.

Namespace: The namespace where the application is located. After selected, the application can only be deployed on an ECS instance in this namespace.

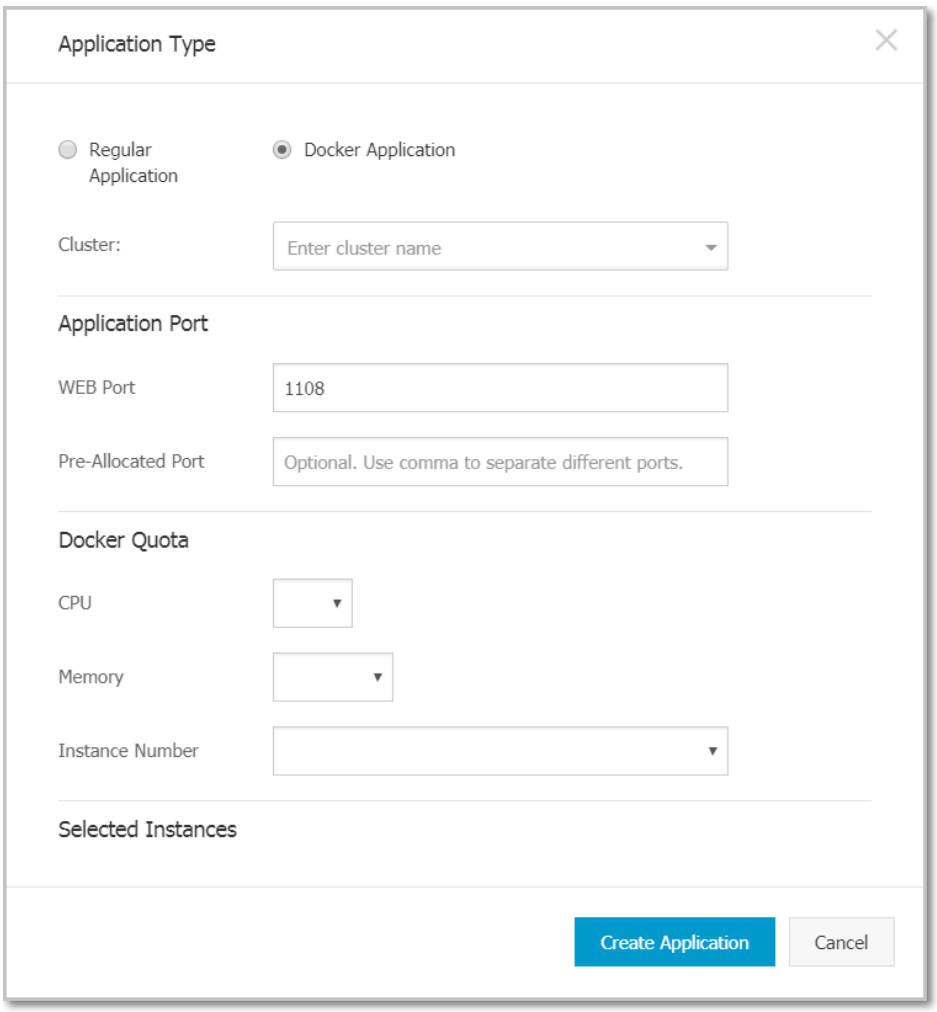
Application Health Check: Optional.

If you configure a health check URL, EDAS periodically accesses the URL regularly and determines the survival status of the application according to the response code.

If no URL is specified, EDAS does not perform health check. This has no impact on the normal operation of the application.

Note: Description of the application.

Select **Docker Application** as the application type and set relevant parameters.



Field description:

- **Cluster**: Cluster where the application is deployed. The cluster cannot be changed once the application is successfully created. Application scaling is completed in the cluster. **An application cannot be deployed in more than one cluster.**
- **Web Port**: A Docker application adopts the host network mode. Designate a web port of Tomcat when you create a Docker application. By default, the Create Application dialog box displays an available port that is automatically allocated.
- **Pre-allocated Port**: When your application uses any other port than the Tomcat Web port, you can designate a reserved port to avoid port conflict.
- **Docker Quota**: To isolate resources, you can declare the resources required to run the Docker application. The declarable resources include CPU and memory. The selected resources cannot exceed the available resources of the available Docker hosts in the cluster.
- **Selected Instances**: After you set the application quota and number of instances, the system automatically allocates the ECS hosts used to deploy the application instances.

After you complete the setting, click **Create**.

After the application is successfully created, the message **Creation successful** is displayed in the upper-right corner.

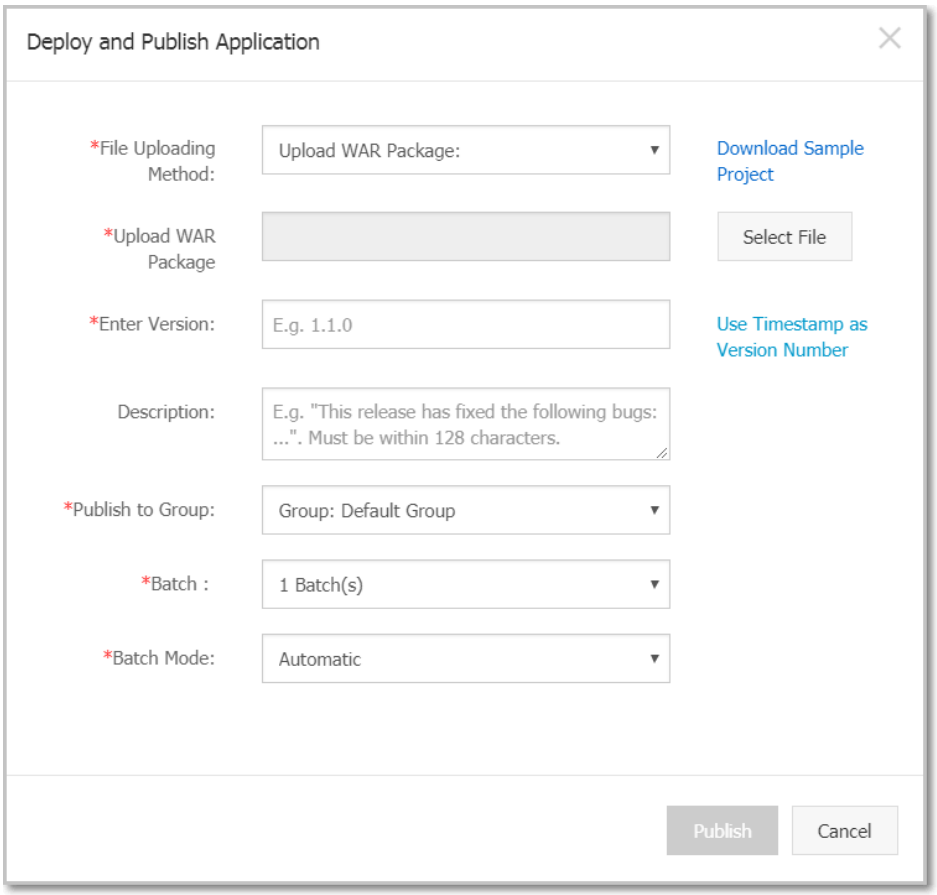
Deploy an application

Upload the WAR package to deploy the created application. You can use the compiled sample project. The procedure of application deployment is as follows:

In the application list, click the name of the created application.

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

Set deployment parameters in the application publication order dialog box.



Description of application deployment parameters:

File Upload Mode:

Upload WAR Package: If you select this option, click **Download Sample**

Project on the right to download the compiled WAR package. Click **Select File** next to **Upload WAR package** to open the local folder and select the WAR package to be deployed, or the downloaded sample project.

It takes time to upload the WAR package. Wait until the upload is complete.

WAR URL: If you select this option, enter the accessible URL of the WAR package in the field of WAR Location, for example, http://edas-public.oss-cn-hangzhou.aliyuncs.com/install_package/edas-app-demo/app-latest.war.

Use Previous Version: If you select this option, select the expected version from the "Historical Version" drop-down list.

Specify Version: This options is applicable if you select **Upload WAR Package** or **WAR Package Address**.

An application version identifies the version of the deployment package used to publish the application and helps you locate a specific publication during rollback.

Note: You can add a version or text description when deploying an application. We do not recommend using **a timestamp as version number**.

Historical Version: Applicable if you select **Use Previous Version**. Select the expected version from the **Previous Version** drop-down list.

Target Publication Group: Group where the application to be published is located.

Batch: Select the number of deployment batches from the drop-down list.

Batch Mode: Automatic.

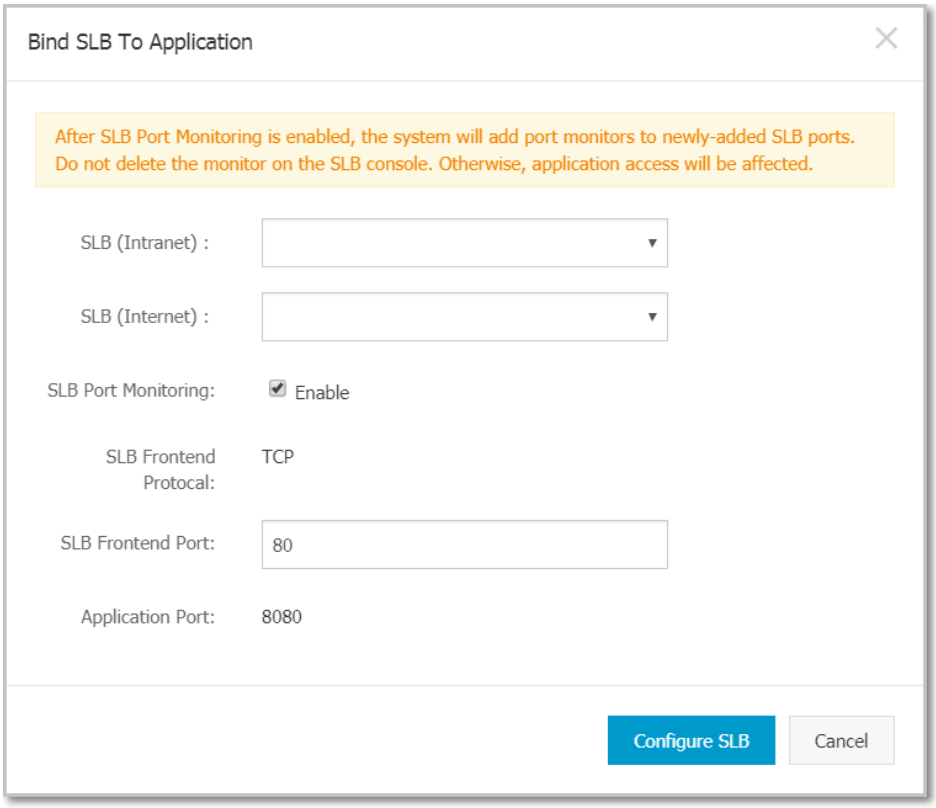
After you complete the setting, click **Publish**.

The file upload progress is shown at the top of the dialog box. After it reaches 100%, you are redirected to the Change Order page, where the application is deployed. After deployment, the status changes to **Success**.

Configure Load Balancing

In the left-side navigation pane, select **Basic Information** to return to the Application Details page. In the application settings area, click **Add** to the right of SLB (Internet) or SLB (Intranet).

In the **Bind SLB to Application** dialog box, select the Intranet or Internet address of your public or private SLB instance from drop-down menu.



If monitoring the Server Load Balancer port is required, select the **SLB Port Monitoring** checkbox, and set the **SLB Frontend Port**. Then, click **Configure SLB**.

Note:

- Do not delete the listener in the SLB console; otherwise, the application cannot be accessed normally.
- The front-end protocol and application port of SLB have been set and cannot be modified here.

After port listening is enabled, a port listener is then added to SLB.

Verify result

After publishing the application, copy the SLB IP address and port number, paste it in your browser's address bar, and press **Enter**. The application's welcome page is displayed.

Manage applications

After an application is published, you can view the application information, perform application scale in or scale out, or delete the application in the EDAS console.

View information about an application

This section describes how to view information about an application. For details about how to deploy an instance, see "Instance management". For details about how to set an application, see "Application settings".

Log on to the EDAS console.

Click **Applications** in the left-side navigation pane, and click an application in the application list to go to the application details page.

On the **Basic Information** tab of the application details page, view the application information and settings.

You can view the following information for common and Docker applications respectively.

- **ID**: This ID is automatically generated by the system when the application is created.
- **Region**: The region in which the application is located. It is set when the application is created.
- **Status**: The number of ECS instances of the application and the number of running ECS instances.
- **Application Type**: The type of the application.
- **Application package**: The package used to deploy the application, which can be downloaded.
- **System Checkup Score**：The score for checking the current applicaiton. Click **View** to show the datails of check items including status, capacity and high availability.

Application scale in/out

You can manually scale out for an application when the instance load of the application is too high. You can also stop and put an instance out of service to scale down an application when the application does not require too many resources.

Scale out

On the application details page, click **Scale Out** in the upper right corner.

On the **Scale Out** page, select the target group and then the specific ECS instance, and click **Scale Out**.

Note: The status of the added ECS instance depends on that of the application.

- If the application is running, it will be deployed and run on the added ECS instance automatically.
- If the application is not running when the ECS instance is added, then the application is deployed on the instance but will not be started.

Scale in

On the application details page, select **Instance Information** tab.

Based on the status of the ECS instance, perform one of the following steps to remove it from the application.

If an ECS instance is running, click **Stop** to stop the instance, and then click **Scale in** in the **Actions** column.

If an ECS instance is already stopped, click **Scale in** to delete the instance from the application.

Delete an application

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. Therefore, **before deleting an application, make sure that logs, WAR packages, and configurations on all instances under the application are backed up**.

On the application details page, click the **Stop** button on the upper-right corner.

Select the **Instance Information** tab. On the instance deployment information tab, click **Scale in** in the **Actions** column of the application.

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

Click the **Delete** button in the upper-right corner of the application details page.

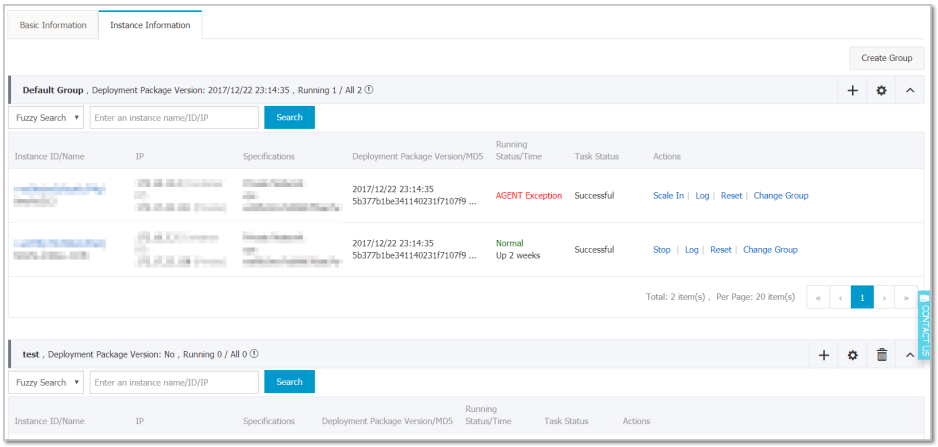
Manage instance groups

Overview

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

For example: There are a total of 10 instances in the application called "itemcenter", and they are divided into two groups: "Default Group" and "Beta Group". The default group has 6 instances and the Beta group has 4. Now there are two groups of instances in the application to which you can deploy different versions of the application package.

The overview of groups in an application is shown as follows:



Note:

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 there is no needs to deploy multiple application package versions, the default group usually works for you and there is no need to create additional groups.

Usage instructions

The instance group is a feature in EDAS designed to manage instances in an application by group. This enables you to carry out A/B testing and gray releases. You can improve your maintenance efficiency by performing application lifecycle management, and resource monitoring and alarming by groups.

Instance management by group mainly includes the following tasks:

View groups

Create groups

Add instances to a group

Deploy an application to a group

Delete groups

View groups

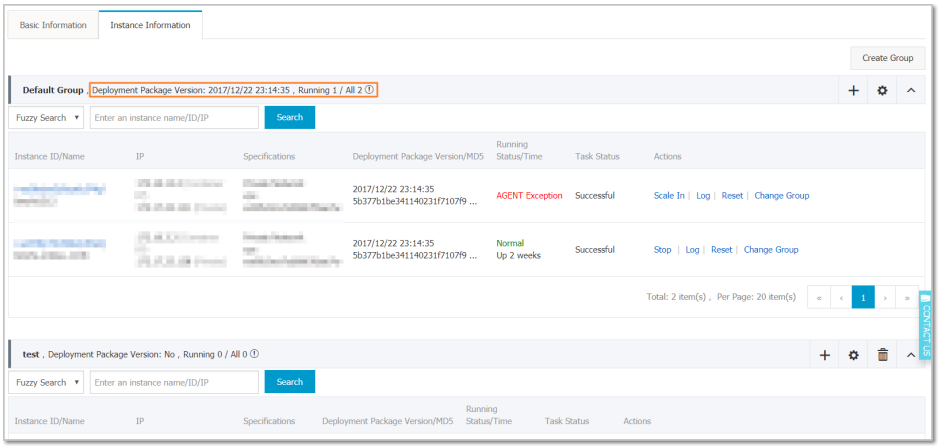
Log on to the EDAS console.

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



Create groups

Gray release is often used for the launch of a new application version, so that the new version can be tested without affecting the production environment. In this case, you need to create a new group for the application.

Log on to the EDAS console.

Click **Applications** in the left-side navigation pane 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 a name for the **Group Name** in the **Create Group** dialog and click **Create**.

Note:

- A default group is created automatically when an application is created.
- Each group corresponds to a single package version, and the version information is displayed right after the group name (*Package version: 2017/1/20 15:36:12* as shown in the figure).
- Newly created group does not relate to any package version. The package version for a group is always the version of the package last deployed to the group.
- The instance information for an application is displayed by instance groups.

Add instances to a group

After a group is created, you can add instances to the new group in two ways: **Sacle Out** and **Change**

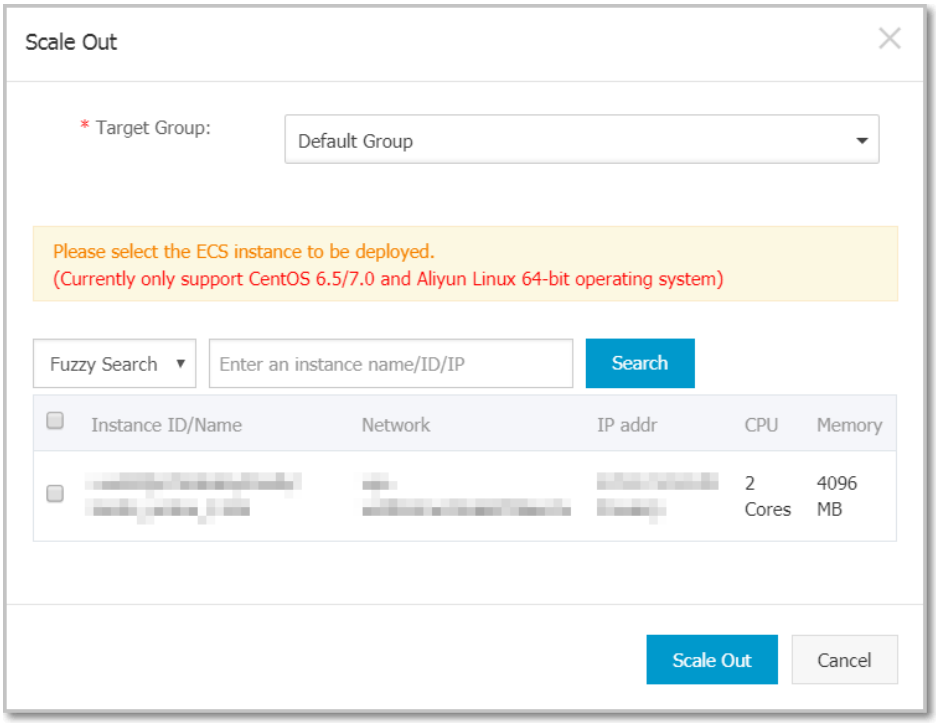
Group.

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

Click **Scale Out** at the top right corner of the application details page.

Select a group and then the ECS instance to add.

Click **Scale Out** to complete the process.



Note:

- 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 using **Change Group**

On the **Instance Information** tab of the application details page, select the instance whose group you want to change and click **Change Group** in the **Actions** column.

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** to complete the operation.

Note:

- 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.
- If you select **Redeploy application on the current instance in the target group**, the package in the group will be used to redeploy application on current instance.
- If you select **Change group without redeployment**, no changes will be made to the deployment status of the instance.
- If the package version for the instance is different from that of the group to which the instance belongs, a prompt will appear.

Deploy an application to a group

You can deploy an application to instances in a certain group.

After specifying a group, the application will only be deployed to the instances in the specified group without affecting instances in other groups.

Click **Deploy Application** at the upper right corner of the application details page.

Set the application deployment parameters. For details, see Deploy an application.

Select the group to deploy the application to.

Click **Publish** to complete the process.

Delete groups

A group with no instances in it can be deleted. The delete operation cannot be undone. Please use caution. Supported operations:

On the **Instance Information** tab of the application details page, click the **Delete Group** icon.

In the popped-up dialog box, click the **Delete** button to complete the operation.

Application settings

This document describes how to set the parameters of applications.

Log on to the EDAS console, click **Applications** in the left-side navigation pane, and click the application name to go to the **Application Details** page.

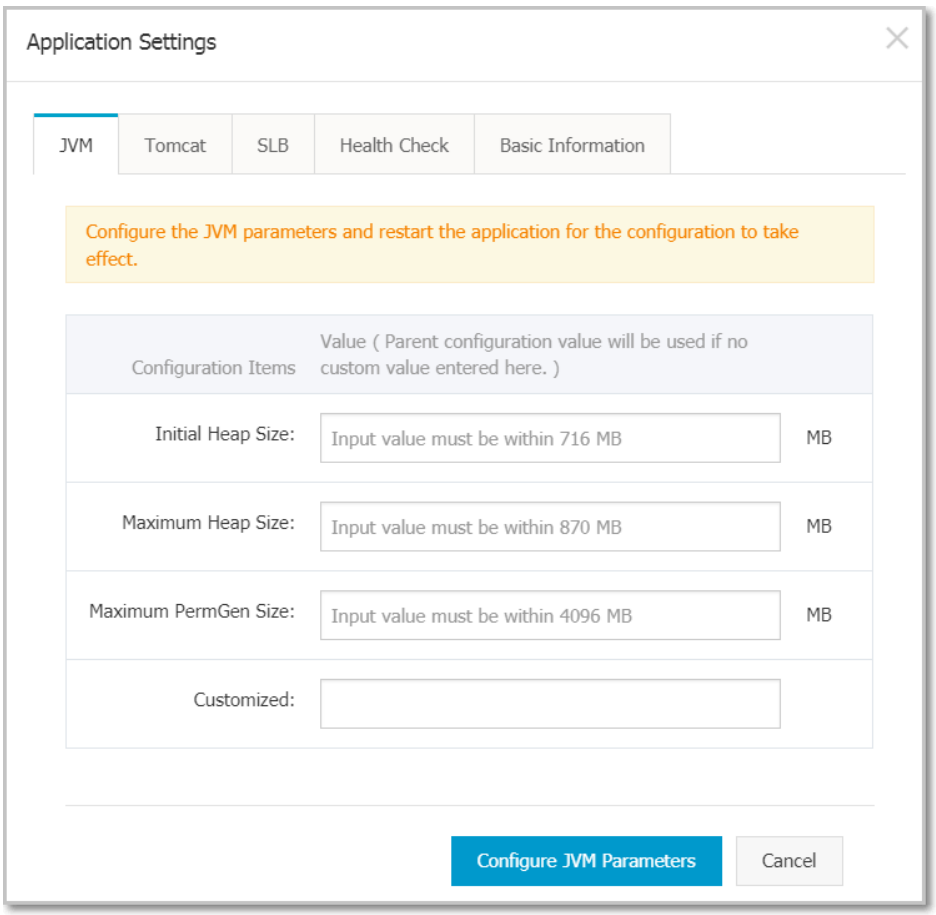
On the **Application Details** page, select the **Basic Information** tab and go to the **Application Setting** section. click **Settings** on the right.

Set JVM parameters

JVM parameters are used to configure the container parameters 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.

Note: The JVM parameter settings are written to the bin/setenv.sh file in the container directory. To apply the settings, restart the application.

In the **Application Settings** dialog box that appears, click the **JVM** tab and set JVM parameters. Then click **Configure JVM Parameters**.



Description of JVM parameters:

- **Initial Heap Size**: Corresponds to the -Xms parameter.
- **Maximum Heap Size**: Corresponds to the -Xmx parameter.
- **Maximum permanent generation size**: Corresponds to the -XX:MaxPermSize parameter.
- **Custom**: Corresponds to the -D parameter and is used to set the system attribute when a Java program is started.

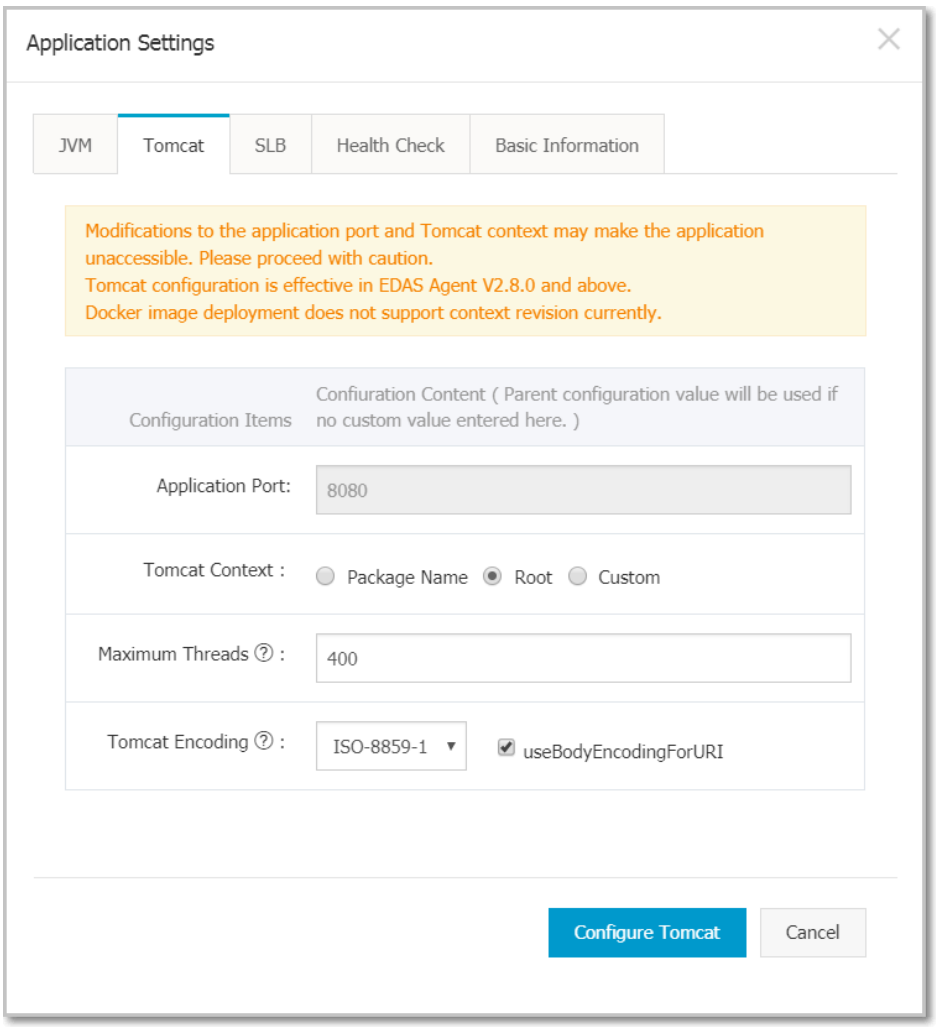
Configure Tomcat

You can configure settings such as the port number, thread pool size, and encoding type the Tomcat container in the EDAS console.

Note:

- After you set Tomcat container parameters, restart the container to apply the parameter settings.
- Tomcat container configuration is supported by EDAS Agent 2.8.0 and later.

In the **Application Settings** dialog box, click the **Tomcat** tab and set Tomcat parameters. Then click **Configure Tomcat**.



Description of Tomcat parameters:

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.

Tomcat Context: Application access path.

- If you select **Package Name**, you do not need to set the custom path, and the application access path is the same as the name of the WAR package.
- If you select **Root**, you do not need to set the custom path, and the application access path is /.
- If you select **Custom**, fill in the path in the customized path field. If not set, the default application access path is the same as the name of the WAR package.

Maximum Threads: Number of connections in the connection pool. It corresponds to the maxThreads parameter. The default value is 400. This parameter has significant implication on performance. We recommend that this parameter be set under professional guidance.

Tomcat Encoding: Code format of Tomcat including UTF-8, ISO-8859-1, GBK and GB2312. Default format is ISO-8859-1.

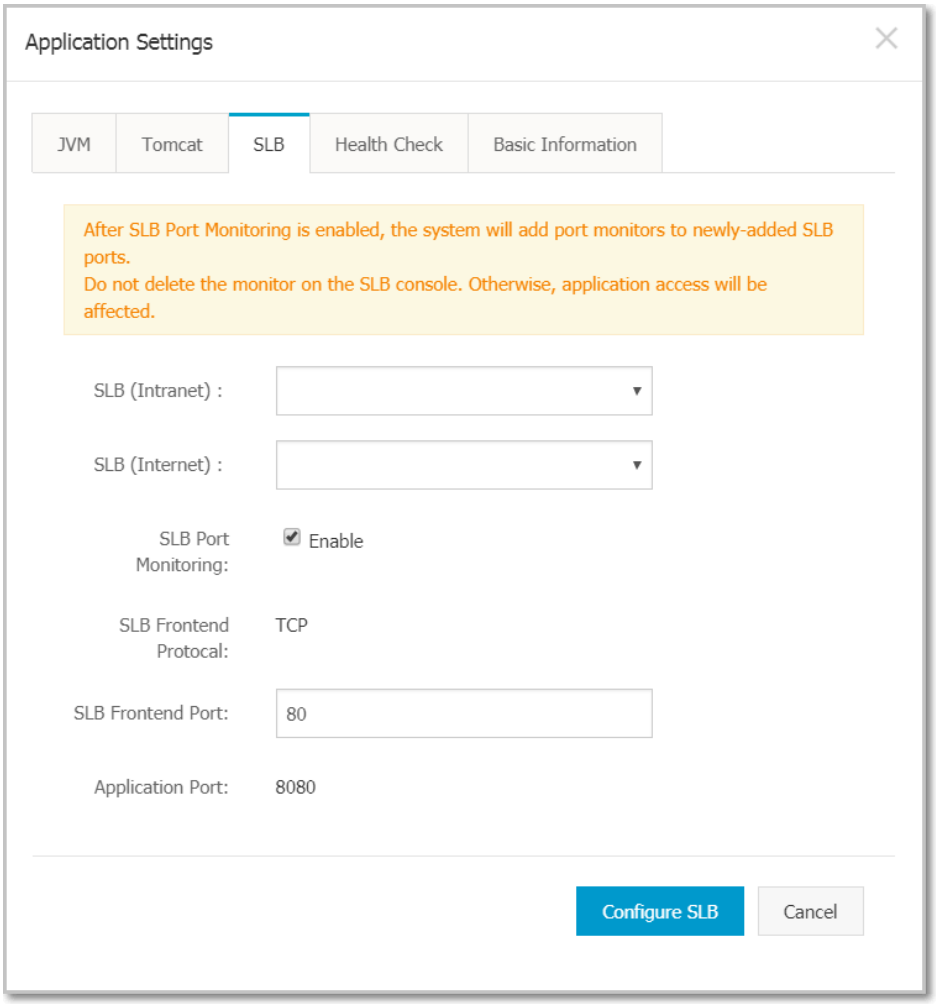
Configure SLB

If you have purchased Alibaba Cloud SLB, EDAS synchronizes your SLB instance to the EDAS console and provides the relevant configuration function. For details about SLB, see the **SLB document**.

SLB instances are classified into Internet- and intranet-based instances, which share the same configuration method.

The following describes how to configure an Internet-based SLB instance.

In the **Application Setting** dialog box that appears, click the **SLB** tab, select the SLB instance from the "SLB (Intranet)" or "SLB (Internet)" drop-down list, and click **Configure SLB**.



Description of SLB parameters:

- **SLB (Intranet)**: SLB instance located in Alibaba Cloud.
- **SLB (Internet)**: SLB instance located in Internet.

- **SLB Port Monitoring**: Enable or disable SLB port monitoring. After you enable the SLB port monitoring, you need to set the frontend port.

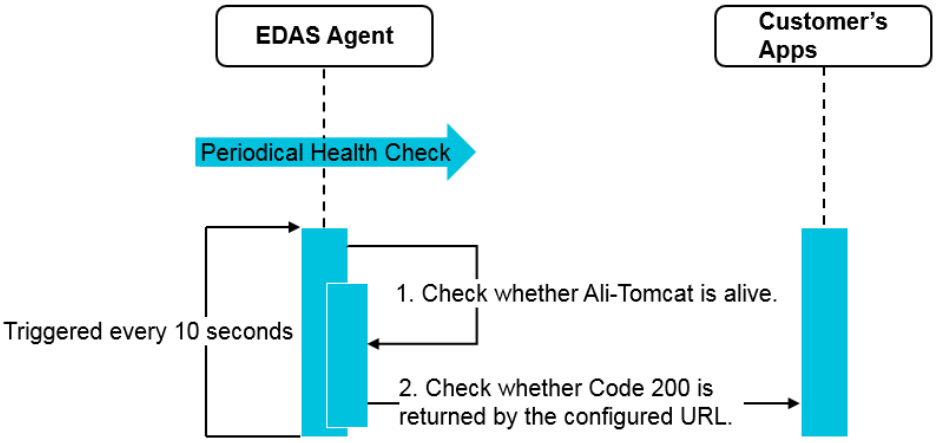
Configure health check

EDAS provides the health check function to check whether deployed applications are running properly.

Overview

Health check is the process whereby the 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 assists you with audit 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.

The agent checks whether the Ali-Tomcat process used to run the user 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 fails.

The agent checks whether Code 200 is returned by the configured URL.

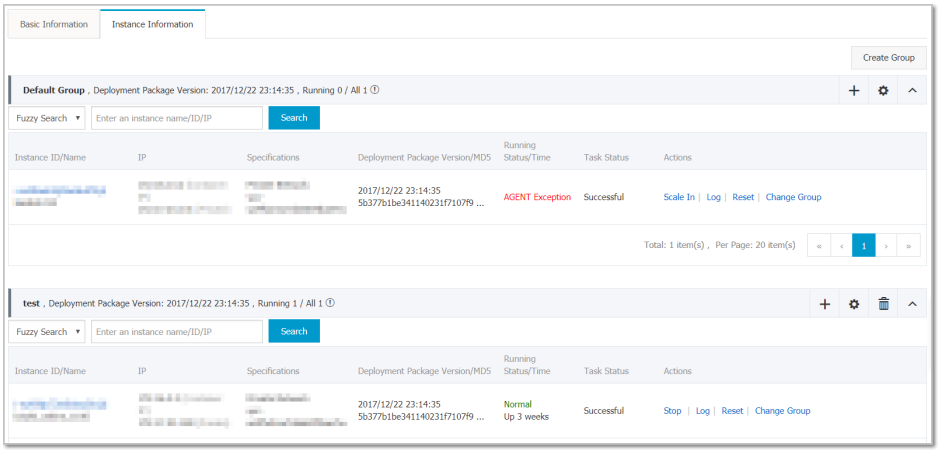
If no URL is configured, this check is not performed; if a URL is configured, the agent checks whether HTTP Code 200 is returned by the configured URL.

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

View health check status

Log on to the EDAS console, click **Applications** in the left-side navigation pane, and click the application name to 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 Abnormal: Displayed when any other code than Code 200 is returned by the configured URL in Step 2.

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

If the agent detects no configured URL in Step 2, the **Normal** state is still displayed, followed by an exclamation mark. When you point the cursor over it, the prompt "Configure a health check URL to check the application status accurately" is displayed.”

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/[.war package name]/_ehc.html. To manually configure a health check URL, follow these steps:

Click **Modify** next to the health check URL in the **Basic Information** tab of the Application Details page.

Enter a correct health check URL and click **Save**.

If you have configured corresponding container settings, configure a health check URL in the format of http://127.0.0.1:[custom port number/[configured path]/_ehc.html according to the container configuration. *"_ehc.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/_ehc.html if no container setting is 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.

Modify basic information

In the **Application Setting** dialog box that appears, click the **Basic Information** tab, enter the Application Name and click **Modify**.

Use Docker to deploy applications

Deployment process

You can use Docker to deploy applications on EDAS to improve your resource usage.

Perform the following steps to deploy an application on EDAS using Docker:

Create a Docker host cluster, and add a Docker instance to the cluster.

For details, see Create a cluster and Add a cluster host.

Note:

You do not need to manually install EDAS Agent on ECS where the Docker application is to be installed.

The Docker application deployed on EDAS shares the IP address and network with the host at present. It will have an independent IP address and network in VPC in the future.

Create an application.

Use Docker to create an application. For details, see the Docker-related content in Publish an application.

Deploy an application.

You can upload a WAR package or create a Docker image to deploy an application using Docker.

Upload a WAR package

The method for deploying an application using Docker is the same as that for deploying an EDAS common application. For details, see Publish an application. After the WAR package is uploaded, it runs in a Docker container.

Create a Docker image

i. Prepare for deployment
   a. Activate an image repository
   b. Prepare a packaging environment
ii. Create custom images
   a. Compile a Dockerfile
   b. Use the local Docker command
iii. Deploy images

This section mainly describes how to use a Docker image to deploy an application.

EDAS provides open APIs for deploying a Docker application. You can build a Jenkins continuous integration platform by yourself. After the code is committed by calling the open APIs of EDAS, the application is automatically packed as a Docker image and deployed on EDAS. For details, see Build continuous integration.

Note:

After deploying an application, you can monitor and manage it on the application management page of the EDAS console.

You can monitor and manage Docker containers in **Cluster Management** under **Resources** of the EDAS console.

Prepare for deployment

Before using a Docker image to deploy applications, complete the following preparations.

1. Activate the image repository
2. Prepare a packaging environment

Activate the image repository

Log on to the image repository console at https://cr.console.aliyun.com/.

Create the default namespace and set the default password as prompted.

Prepare a packaging environment

Perform the following steps to prepare a packaging environment:

1. Prepare Docker
2. Configure the image accelerator
3. Download the EDAS basic image

Prepare Docker

In current packaging mode, the Docker service must be available and able to access the Alibaba Cloud image repository.

If you are a Linux user, use Centos 7.0/Ubuntu 12.04 or later versions to ensure that the kernel version of the operating system meets the basic requirements for running Docker.

Method 1(Recommended): Convert an ECS instance to a Docker host using the EDAS console.

Select a host in the cluster and log on to the host.

Method 2: Install Docker locally.

- Install Docker on Linux
  - Document: https://docs.docker.com/engine/installation/linux/
  - Command line: curl -sSL http://acs-public-mirror.oss-cn-hangzhou.aliyuncs.com/docker-engine/internet | sh -
- Install Docker on Mac: https://docs.docker.com/docker-for-mac/

Configure an image accelerator

Access the image repository console and obtain your image acceleration channel **Image accelerator** .

Configure an image accelerator.

Configure the image accelerator address based on the local packaging environment and the help information of the specific operating system.

If the operating system is CentOS, copy the following command and paste and execute it on the host in which Docker is installed and image is to be packaged.

```
sudo mkdir -p /etc/docker
sudo tee /etc/docker/daemon.json <<-'EOF'
{
"registry-mirrors": ["URL of image repository"]

}
EOF
sudo systemctl daemon-reload
sudo systemctl restart docker
```

Restart the Docker service.

- Linux: The service is automatically restarted when the preceding command is run.
- Mac system: If Docker Engine is used, click **restart** in the lower right corner.

Download EDAS basic image

To create an image, download the basic image of the corresponding version of the application.

The version is the value set in the **Application Runtime Environment** field when the application is created, for example, EDAS-Container 3.2.

Confirm the application version.

Click the created Docker application and click **Software Version** on the left-side menu bar to confirm the application version.

Find the corresponding source.

Access the URL https://dev.aliyun.com/detail.html?repoId=5610 to find the corresponding version.

Pull the image.

Run the following command to pull the image.

Replace 3.2 with the image version to be pulled. If the update time on the right changes, you are advised to update the basic image and repackage the image.

docker pull registry.aliyuncs.com/edas/edas-container:3.2

Create custom images

Constraints and suggestions

- To ensure that you can use all functions of EDAS, do NOT modify Tomcat default boot scripts, directories or the path of logs.
- If a new version EDAS application is published, you need to re-package and publish the image to obtain new functions and features of EDAS. If the last packed image has been published for a long time, it is recommended that you update the basic image and re-package and publish it. For details, see Download an EDAS basic image.
- Directory list
    - Tomcat directory: /home/admin/taobao-tomcat-production-7.0.59.3/
    - Log directory: /root/logs/
    - WAR package directory: /home/admin/taobao-tomcat-production-7.0.59.3/deploy/

Create a custom basic image

Use any of the following methods to create a custom basic image:

Method 1: Compile a Dockerfile

After pulling the expected basic image, compile a Dockerfile to prepare a standard image. The following examples describe how to **publish a WAR package** and **modify Tomcat configurations**.

Note: Do not use the CMD command to create the boot script for the custom image.

Example 1: Add the local application /tmp/edas-app.war to the image and publish the application in image mode.

```
FROM registry.aliyuncs.com/edas/edas-container:3.2
ADD /tmp/edas-app.war /home/admin/taobao-tomcat-production-7.0.59.3/deploy/
```

Example 2: Replace server.xml and install the system monitoring tool SAR.

```
FROM registry.aliyuncs.com/edas/edas-container:3.2
ADD /tmp/server.xml /home/admin/taobao-tomcat-production-7.0.59.3/conf/server.xml
RUN yum install -y sysstat
```

After compiling Dockerfile, publish the image.

Method 2: Use the local Docker command

Similar to the file copy command cp, run the following command to copy the local file /tmp/app.war to the WAR package directory in the packaging environment.

Format:

docker cp	**Local file path**	**edas-build-package:container file path**
docker cp	/tmp/app.war	edas-build-package:/home/admin/taobao-tomcat-production-7.0.59.3/deploy/

Take the preceding command as an example, run the following command to copy the local file /tmp/app.war to the WAR package deployment directory of the container:

docker cp /tmp/app.war edas-build-package:/home/admin/taobao-tomcat-production-7.0.59.3/deploy/

Method 3: Download the remote file in the Docker container

Run docker exec -it edas-build-package bash to go to the container.

Run wget *http://anything-you-want/local storage path* to download the file.

Deploy a WAR package

- Do not modify the default deployment directories listed in Constraints and suggestions.
- See the procedure in Copy the local file to the image / Download the remote file in the image to copy the WAR package to the deployment directory.

Deploy images

Basic image environment list

The following basic services are built in the EDAS Docker container to meet the baseline requirements for running EDAS applications.

- CentOS 7
- JDK 7 (default) or JDK 8 (optional)
- Ali-Tomcat 7

Perform the following steps to deploy an image:

1. Create a container image repository
2. Log on to the remote repository
3. Update/Create a basic image container
4. Package the local image
5. Upload the local image to the image repository

Create a container image repository

Log on to the Docker Image Repository Console, and click **Image List** in the left-side navigation pane.

On the Image Repository List page, click **Create Image Repository** in the upper right corner.

Based on the project features, configure **Repository Name**, **Repository Type** and other required parameters, and set **Code Source** to **Local Repository**.

Click **Create Image Repository**.

Note: If the code is confidential, set **Repository Type** to **Private**.

The preceding configuration is used for the following examples. To modify the configuration, see Paths and command reference.

Log on to the remote repository

Log on to the Image Repository console. On the **Image List** page, click **Modify Docker Logon Password** to set the logon password.

Click **Manage** for the image repository you just created.

Find the code for logging on to Alibaba Cloud Docker Registry provided on the page, copy the code to the terminal, and enter the password.

```
> docker login --username=**********@aliyun.com registry.cn-hangzhou.aliyuncs.com
< Password:
< Login Succeeded
```

If "Login Succeeded" is displayed, the login succeeds.

Update/Create a basic image container

To re-package the basic image, initialize the entire packaging environment again and clear data. Back up data before performing the preceding operations.

(Skip this step if it is creation for the first time.) Back up the code and change file.

(Skip this step if if it is creation for the first time.) See Download an EDAS basic image to update the local basic image.

```
docker pull registry.aliyuncs.com/edas/edas-container:3.0.0
```

Run the following commands to initialize the packaging environment and clear existing data. (Replace the image version number with your actual version number.)

```
docker rm -f -v edas-build-package
docker run --name edas-build-package -d --restart=always registry.aliyuncs.com/edas/edas-container:3.0.0
```

Package the local image

After creating a custom basic image, based on different packaging methods, execute the following shell command to package the image. The packaged image is stored locally.

Note: Replace the last image address with the image repository name created in creating a container image repository, and enter a version number that is easy to identify.

The following command is used as an example. Package the service registry.cn-hangzhou.aliyuncs.com/edas/demo-frontend-service with the version number 20161111.

For Method 1: Compile a Dockerfile

```
docker build -t demo-frontend-service:20161111 -f /tmp/Dockerfile .
docker tag demo-frontend-service:20161111 registry.cn-hangzhou.aliyuncs.com/edas/demo-frontend-service:20161111
```

For Method 2: Use the local Docker command or Method 3: Download the remote file in the Docker container

```
export IMAGE_ID=`docker ps -a -f name=edas-build-package --format {{.ID}}` && docker commit $IMAGE_ID registry.cn-hangzhou.aliyuncs.com/edas/demo-frontend-service:20161111
```

Upload the local image to the image repository

Run the following command to push the local image to the remote server for deployment. If a message is displayed indicating that you have not logged on or do not have the permission, refer to Log on to the remote repository.

The following command is used as an example to show how to push the local image to the remote service repository registry.cn-hangzhou.aliyuncs.com/edas/demo-frontend-service with the version number 20161111.

```
docker push registry.cn-hangzhou.aliyuncs.com/edas/demo-frontend-service:20161111
```

Paths and command reference

EDAS console: https://edas-intl.console.aliyun.com

Image repository console: https://cr.console.aliyun.com

Image network accelerator: https://cr.console.aliyun.com/#/accelerator

Docker quick installation: https://cr.console.aliyun.com/#/accelerator

Log on to the image repository (China East 1 (Hangzhou)):

docker login —username=**@aliyun.com registry.cn-hangzhou.aliyuncs.com`

Log on to the image repository (China North 2 (Beijing)):

docker login —username=**@aliyun.com registry.cn-beijing.aliyuncs.com

Pull the packaged image (demo of this guide):

docker pull registry.cn-hangzhou.aliyuncs.com/edas/demo-frontend-service:20161111

Pull the basic image (V3.2):

docker pull registry.aliyuncs.com/edas/edas-container:3.2

Pull the basic image (V3.0):

docker pull registry.aliyuncs.com/edas/edas-container:3.0.0

Reset the packaging environment:

docker rm -f -v edas-build-package

Create a packaging environment:

docker run —name edas-build-package -d —restart=always registry.aliyuncs.com/edas/edas-container:3.0.0

Package a locally published file (Dockerfile):

docker build -t demo-frontend-service:20161111 -f /tmp/Dockerfile . && docker tag demo-frontend-service:20161111 registry.cn-hangzhou.aliyuncs.com/edas/demo-frontend-service:20161111`

Package a locally published file (container):

export IMAGE_ID=`docker ps -a -f name=edas-build-package —format {{.ID}} | docker commit $IMAGE_ID registry.cn-hangzhou.aliyuncs.com/edas/demo-frontend-service:20161111

Remotely push the package:

docker push registry.cn-hangzhou.aliyuncs.com/edas/demo-frontend-service:20161111

Log on to the packaging environment:

docker exec -it edas-build-package bash

Download the network file:

wget https://www.taobao.com/robots.txt

Publish a WAR package (in Dockerfile mode):

```
FROM registry.aliyuncs.com/edas/edas-container:3.2
ADD /tmp/edas-app.war /home/admin/taobao-tomcat-production-7.0.59.3/deploy/
```

Publish a WAR package (in container mode):

docker cp /local WAR package path/app.war edas-build-package:/home/admin/taobao-tomcat-production-7.0.59.3/deploy/

Logs

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.

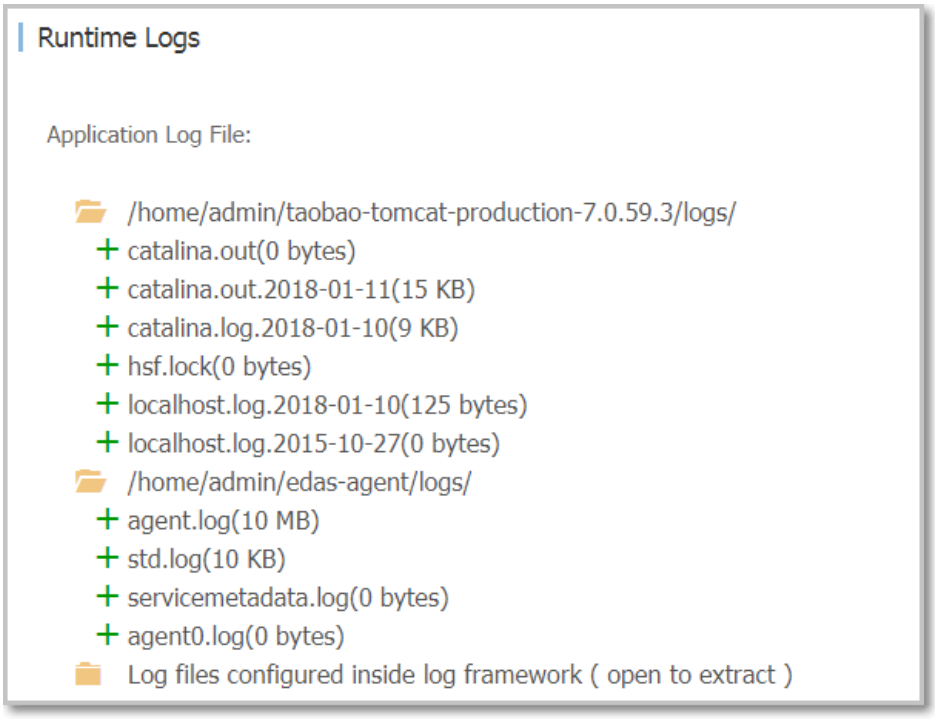
In the left-side navigation pane, select **Logs** > **Log Directories**. Alternatively, you can also go to the **Instance Information** tab of the application details page, and click **Logs** in the **Actions** column.

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.

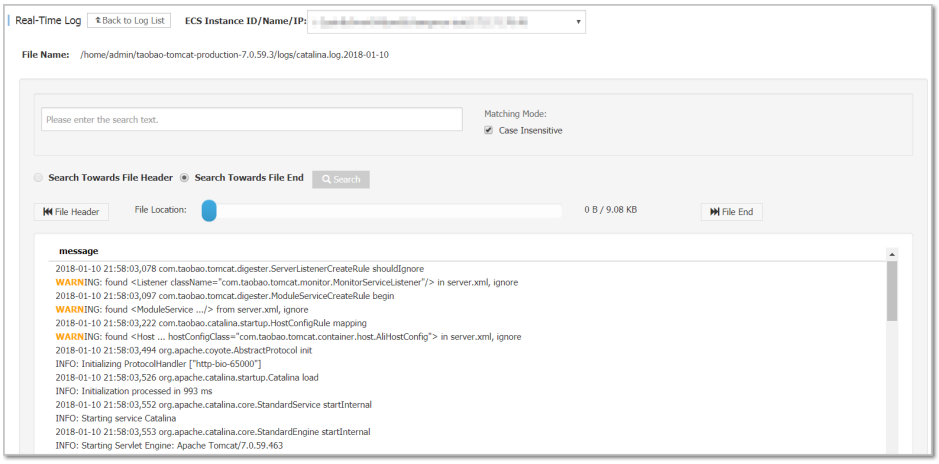
EDAS Agent logs directory.

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.

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 status and quickly discover and locate problems.

Terms

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

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

Creation time: The time for creating a call trace.

Sequence number: Used for trace sampling.

Flag: (Optional) Used for debugging or marking.

Process ID: (Optional) Used for single-host multi-process applications.

RpcId: Calls and flags the log tracking sequence and nesting relationship. It is transmitted across systems.

Service dimension: Service monitoring monitors data in the 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.

Application drilldown: Views metrics of upstream/downstream applications associated with the target metric.

Types of data

- **RPC Call Overview**: Displays the RPC services (including the HSF and other custom services) provided by an application as the server.

- **RPC Call Source**: Displays the client that calls an application as the server.
- **RPC Call Dependency**: Displays the RPC services (including the HSF and other customer services) that are called by an application as the client.

Types of monitoring reports

- **Block (default)**: A report of the block type displays data in a table together with a graph. Information contained includes the monitored object, time, QPS, elapsed time, errors, and results. By default, the graph shows the data for the last hour, and the table shows the data for the last five minutes.
- **Chart**: A report of the multi-chart type displays data in graphs. Information contained includes the monitored object, time, QPS, elapsed time, errors, and results. By default, the graphs show the data for the last hour, and the latest data is also listed.
- **Table**: A report of the table type displays data in a table. Information contained includes the monitored objects, QPS, elapsed time, errors, and results. Data within the last minute is displayed.

Metric description

- **Errors**: Indicates the number of RPC call errors per minute. This metric is calculated using the formula: Errors = Total number of errors within the minute/60.
- **Result**: Indicates the return result in the format of "Result: QPS", where "Result" indicates the RPC result. The HTTP result is consistent with the HTTP ErrorCode.

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.

Infrastrucure 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.

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.

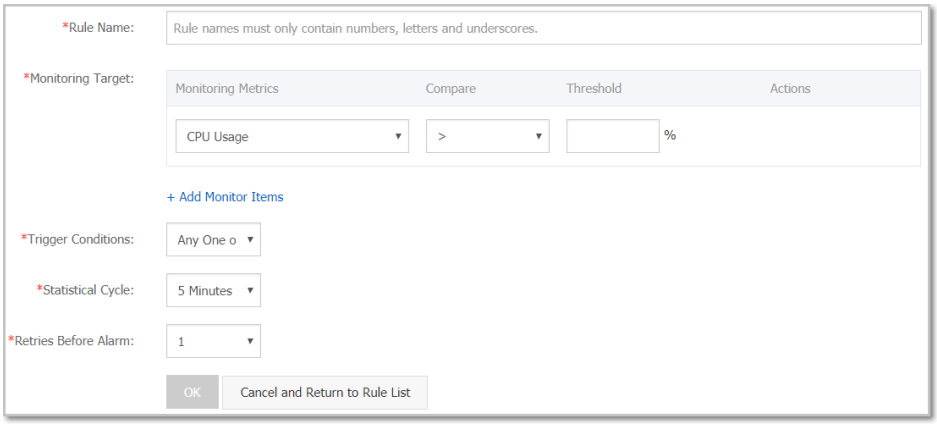
Configure alarm policies

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.



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 affect 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 also a namespace, for example, a package in Java. The group name may contain a maximum of 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;
    }
}
```

Traffic management

Overview

During application publishing and product iteration, the following scenarios and requirements are typical:

- Gray release

Blue-green release

A/B testing

Marketing: Strategic new product launch, for example, roll out new product to 50% of users and compare with existing product.

The traffic management function is used to allocate traffic that meets the defined rules to specific groups based on the custom solution set on the console, so as to meet the preceding scenarios and requirements.

Configure traffic management

Prerequisites

- EDAS standard/professional/platinum edition is required.
- The application for which the traffic management function is to be used must contain at least two groups. Each group contains at least one application that runs properly. If no group exists, log on to the EDAS console and create a group.

Procedure

In the EDAS Console, select **Applications** in the left-side navigation pane, and then click the name of the application whose traffic is to be managed.

On the application details page, select the **Instance Information** tab, and verify that the groups for traffic management is available.

Select **Traffic Control** in the left-side navigation pane of the application details page.

Click **Create Solution** in the upper-right corner of the traffic management page.

On the **Solution Details** page, enter the solution name and set the traffic management rule.

The traffic management rule can be set to “Parameter Modulo” or “List Diverging”.

Parameter Modulo: Based on the modulo operation, allocate traffic that meets the rule to the target application group.

Rule: Cookie[$key] mod 100 < $percent, perform the modulo operation on 100

using the specified key in Cookie. If the result is smaller than the configured percentage, allocate the traffic to the target application group.

List Diverging: Based on the list, allocate the listed traffic to the target application group.

If the value of the corresponding field of the requested Cookie is in the list, allocate traffic to the target application group.

After the solution is set, click **Save**.

Verify result

Verify the solution status

After the solution is created, you can view it in the solution list. The solution is disabled by default. You can edit or delete the solution. After the solution is enabled, you cannot edit or delete it.

Note: You can configure multiple traffic management solutions for different Cookies. However, only one traffic management solution can be enabled at a time.

Verify the function

Download the compiled application demo: app-latest.war (new version) and app-old.war (old version). Refer to Deploy an application to deploy the WAR packages to the custom group (used for gray traffic) and default group, respectively.

After the packages are successfully deployed, set the value of userId in Cookie on the page. After saving, the page is automatically refreshed.

Set the solution and verify the function as follows:

Rule Settings: List Diverging

Parameter: userId

List: 123456789

Deployed packages in different groups:

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 applciation 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

Rate limiting and degradation overview

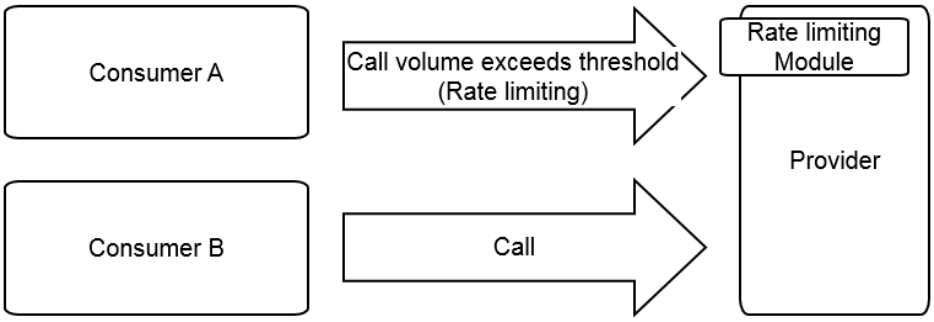
The rate liming and service degradation feature of EDAS solves slow system responses or crashes caused by high pressure of the backend core services. This feature is generally used in high-traffic scenarios, for example, flash sale, shopping spree, great promotion, and anti-click farming scams.

Rate limiting and degradation

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 systems and service unavailability. By adjusting the traffic threshold, the maximum traffic volume of the system is controlled to ensure secure and stable operation of the systems.

Basic principle

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



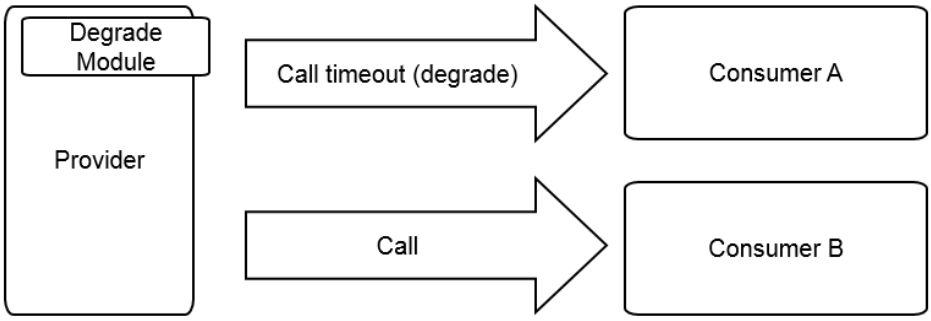
Service degradation

Service degradation is to lower the priority of timed-out service calls from non-core service providers to ensure the availability of core service consumers.

Basic principle

After a degradation module code is configured for a service consumer and a degradation policy is

configured on EDAS, the service consumer is enabled with the degradation 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.



Entry points

The HTTP interfaces, RPC services, and Java methods that are accessed and called by users or other systems are called entry points and displayed on the Entry Points page.

The Rate Limiting and Degradation module detects entry points automatically, based on which you can conveniently configure rate limiting on a specific entry point.

The Entry Points page includes the Entry Points and Dependencies tabs, grouping entry points by service type:

- **Entry Points:** Displays the inbound traffic volume of the current application, and you can apply rate limiting to control the traffic.
- **Dependencies:** Displays the outbound traffic of the services that this application provides, and you can also apply rate limiting and degradation to these services.

The Entry Points tab displays the real-time QPS and accumulated service volume in the last one minute.

The rate limiting status indicates whether rate limiting is enabled depending on whether rate limiting rules are available. If no rate limiting rules are available, rate limiting is disabled.

Note: The system loads the entry points of alive instances in the default group. To check entry points of a specific instance, change the IP address.

Add rate limiting rules

Log on to the EDAS console, click **Applications** on the left-side navigation pane, and select a deployed service provider application.

On the left-side navigation pane of the application details page, click **Rate Limiting and Degradation** > **Entry Points**.

On the **Rate Limiting Monitor** page, select the entry point to apply rate limiting, and click **Add Rate Limiting** in the Actions column.

Set rate limiting rules on the **Configure Rules** page.

Field description:

- **Resource:** This option is already selected on the rate limiting monitoring page and cannot be changed. The QPS and threads of the resource will be restricted after you configure rate limiting rules.
- **Applications:** This option specifies whether the rate limiting applies to all applications or specified applications.
- **Granularity:** This option specifies whether the rate limiting applies at QPS level or thread level. QPS rate limiting restricts the number of requests per second, and thread rate limiting restricts the number of threads. Generally, the larger the number of threads, the larger the QPS value is. However, the QPS of a thread is usually larger than 1. The thread sends requests continuously and the response time is only 10 ms to 99 ms.
- **Rate Limiting Threshold:** Rate limiting is triggered when the value exceeds the threshold.

Click **OK**.

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 upperlimit 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.

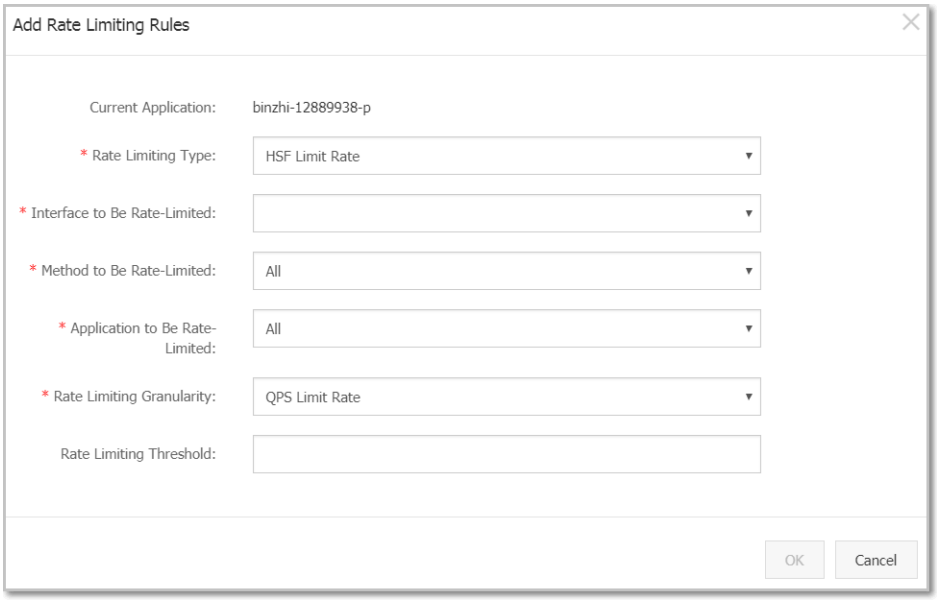
Click **Application Configuration Guide** in the upper right corner of the page.

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.



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.

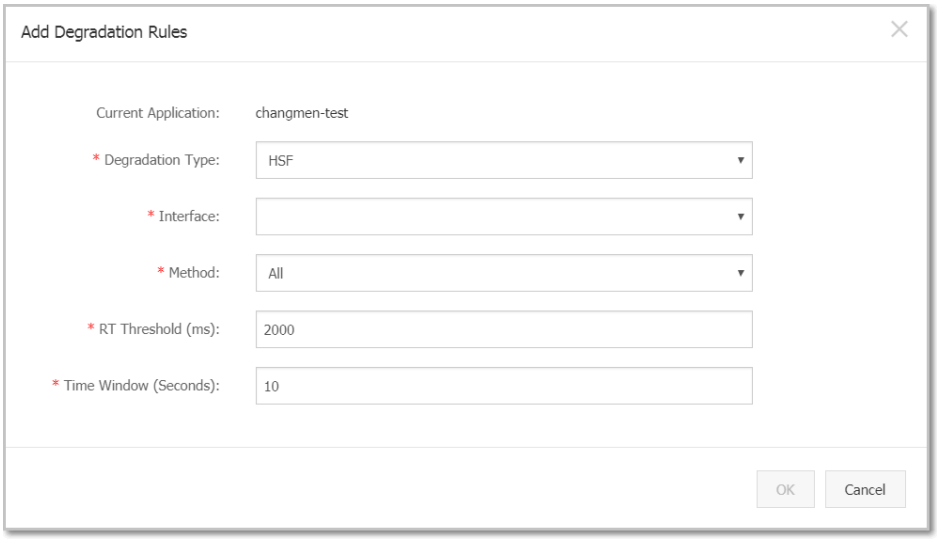
Click **Application Configuration Guide** in the upper-right corner of the degradation rule page.

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.



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.

Rate limiting history

The Rate Limiting Log lists the history of all interfaces are limited in the application within one day.

The list displays the rate limiting history of all interfaces and HTTP operations. To view the history of a certain interface, click **View Trend**.

Note:

- By default, the query result displays rate limiting data within 30 minutes. To view the data within a certain period, specify the time range.
- The minimum time granularity is minute, but rate limiting activities are tracked in seconds. To search rate limiting data to a granularity of seconds, check the logs in logs/home/admin/logs/csp directory.

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 amout 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 blueis 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.

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**. 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.

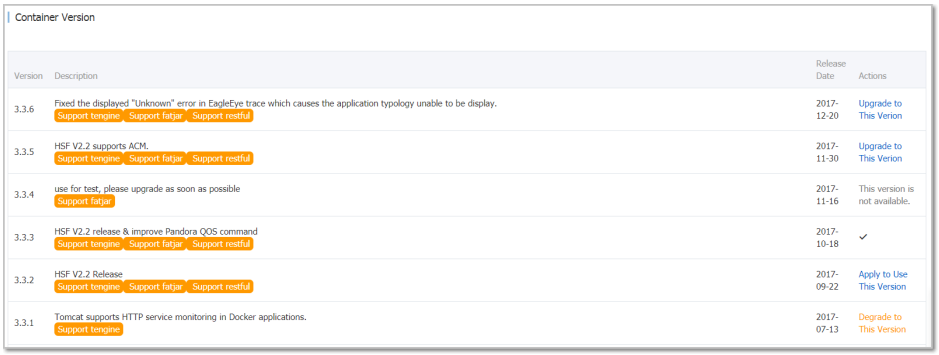
Upgrade or downgrade a container

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.



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 also 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. The new configurations dynamically take effect, without the need for code modification, service republishing, or service restart.

Note: The **Group** in the configuration definition is an existing service group. 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

Digital operations is an important feature of EDAS, and one of its major functions is the distributed trace analysis.

Distributed trace analysis analyzes every service invocation in the distributed system, conducts holographic investigation into all sent and received messages and database accesses, so as to help you precisely identify system bottlenecks and risks.

It includes the following functions:

Application typology

Through the visualized typology, you can easily understand the calling relationships between applications and the relevant performance data.

Trace query

By setting query conditions, you can accurately identify those businesses with poor performances or problems.

Trace details

Based on the query result, you can view the detailed trace information on the slow or malfunctioning businesses and reorganize their dependencies. The information helps you to identify problems such as error-prone points, performance bottlenecks, strong dependencies. You can also evaluate the system capacity based on service invocation ratios

and peak QPS.

Application topology

With the application topology function, you can view the real-time (last second) topology of the calls between applications in the system.

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

View the application topology.

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

Hover your cursor over an application to view the typology of the calls related to this application.

Click 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.

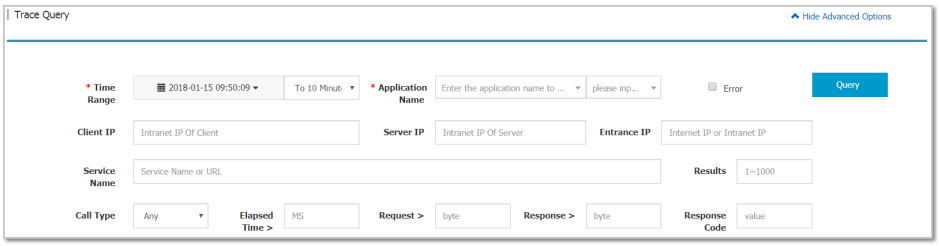
Trace query

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 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 demontrates 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:

0 ms for the client to send the request. 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.

10,077 ms for the server to process the request. The time is taken only by the server to process the request, not including other operations.

3,002 ms for the client to receive the response. 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	RedisOperations.dump	Y	
	EXISTS	RedisOperations.hasKey	Y	
	EXPIRE	RedisOperations.expire	Y	
	EXPIREAT	RedisOperations.expireAt	Y	
	KEYS	RedisOperations.keys	Y	
	MIGRATE	N		
	MOVE	RedisOperations.move	Y	
	OBJECT	N		
	PERSIST	RedisOperations.persist	Y	
	PEXPIRE	RedisOperations.expire	Y	
	PEXPIREAT	RedisOperations.expireAt	Y	
	PTTL	RedisOperations.getExpire	Y	
	RANDOMKEY	RedisOperations.randomKey	Y	
	RENAME	RedisOperations.rename	Y	key: oldKey：${oldKey};newKey:${newKey}
	RENAMENX	RedisOperations.renameIfAbsent	Y	
	RESTORE	RedisOperations.restore	Y	
	SORT	RedisKeyCommands.sort	Y	key: query:${SortQuery}
	TTL	RedisOperations.getExpire	Y	
	TYPE	RedisOperations.type	Y	
	SCAN	RedisKeyCommands.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	ValueOperation	Y	

		s.set		
	SETNX	ValueOperations.setIfAbsent	Y	
	SETRANGE	ValueOperations.set	Y	
	STRLEN	ValueOperations.size	Y	

Hash-type operations

Data structure/Object	Operation	Spring Data Redis method	EDAS support for EagleEye tracing (Y/N)	Remarks
Hash	HDEL	HashOperations.delete	Y	
	HEXISTS	HashOperations.hasKey	Y	
	HGET	HashOperations.get	Y	
	HGETALL	HashOperations.entries	Y	
	HINCRBY	HashOperations.increment	Y	
	HINCRBYFLOAT	HashOperations.increment	Y	
	HKEYS	HashOperations.keys	Y	
	HLEN	HashOperations.size	Y	
	HMGET	HashOperations.multiGet	Y	
	HMSET	HashOperations.putAll	Y	
	HSET	HashOperations.put	Y	
	HSETNX	HashOperations.putIfAbsent	Y	
	HVALS	HashOperations.values	Y	
	HSCAN	HashOperations.san	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.rightPushIfPrese	Y	

		nt		

Set-type operations

Data structure/Object	Operation	Spring Data Redis method	EDAS support for EagleEye tracing (Y/N)	Remarks
Set	SADD	SetOpertions.add	Y	
	SCARD	SetOpertions.size	Y	
	SDIFF	SetOpertions.difference	Y	
	SDIFFSTORE	SetOpertions.differenceAndStore	Y	
	SINTER	SetOpertions.intersect	Y	
	SINTERSTORE	SetOpertions.intersectAndStore	Y	
	SISMEMBER	SetOpertions.isMember	Y	
	SMEMBERS	SetOpertions.members	Y	
	SMOVE	SetOpertions.move	Y	
	SPOP	SetOpertions.pop	Y	
	SRANDMEMBER	SetOpertions.randomMember randomMembers distinctRandomMembers	Y	
	SREM	SetOpertions.remove	Y	
	SUNION	SetOpertions.union	Y	
	SUNIONSTORE	SetOpertions.unionAndStore	Y	
	SSCAN	SetOpertions.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	ZSetOperYations.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**	

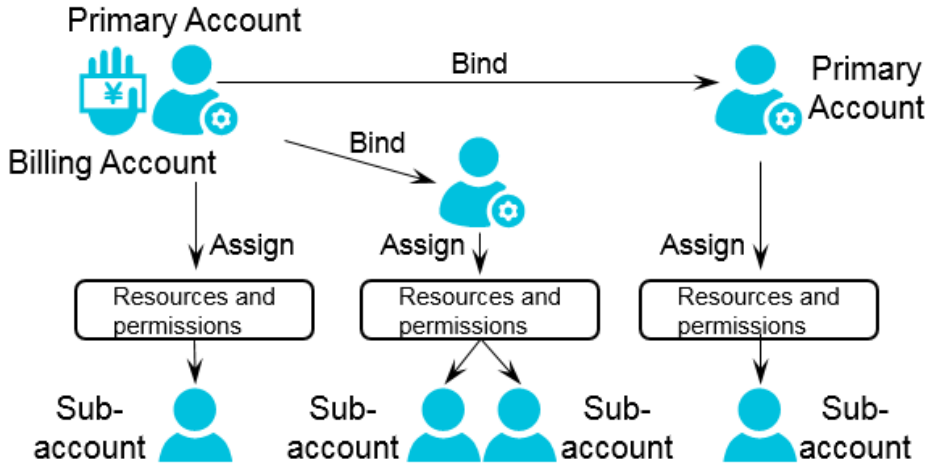
Account management

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.

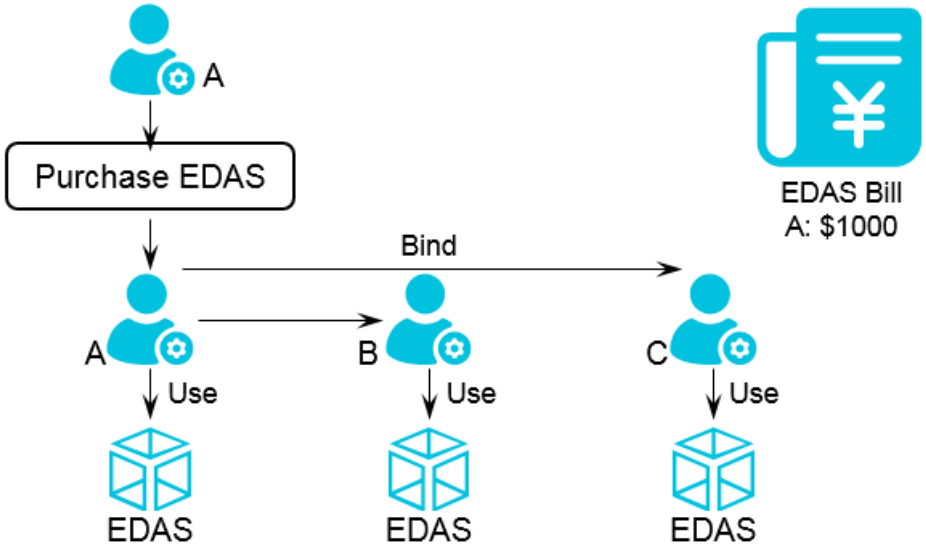
Each primary account is an independent account that owns all resources bought with the account and has full permissions on EDAS except that it cannot bind other primary accounts.

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.

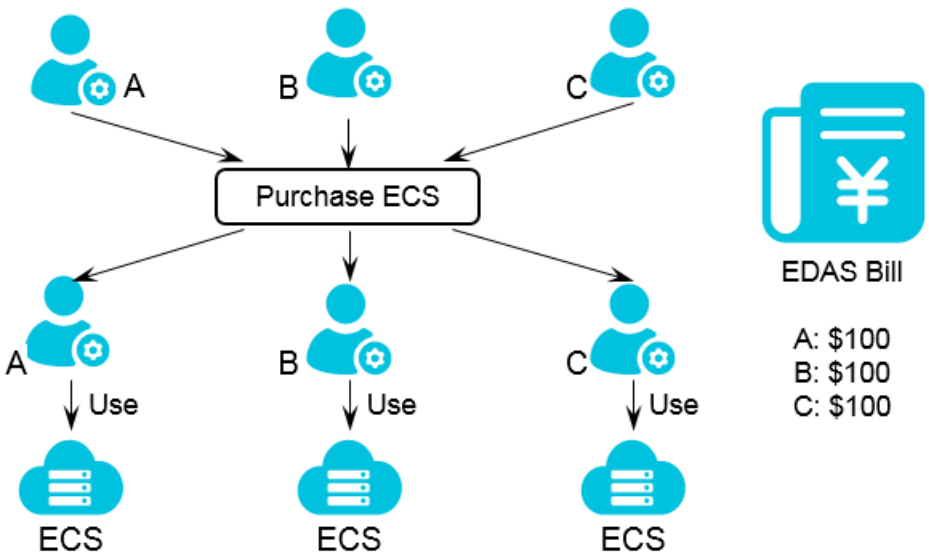
Scenario 1

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.



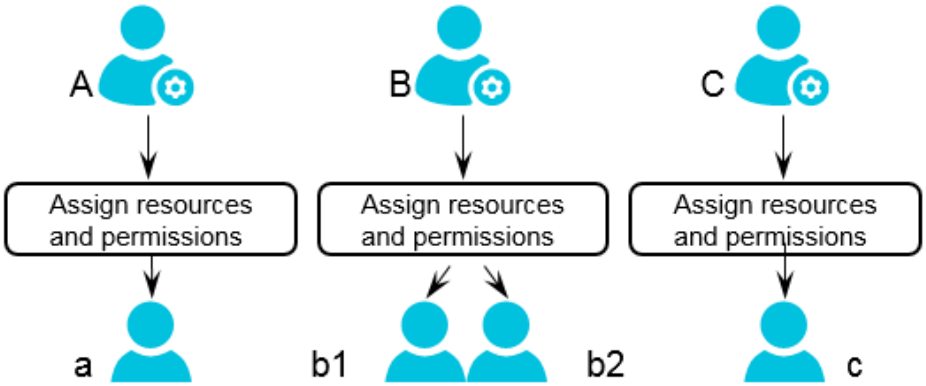
Scenario 2

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.

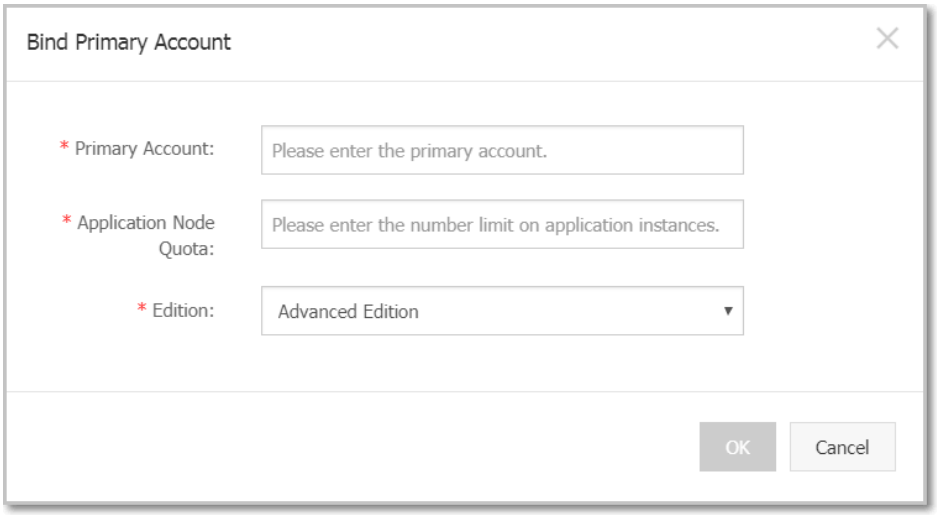
Bind a primary account

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**.



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.

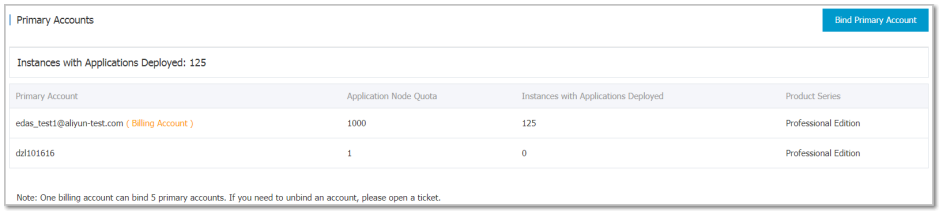
Note:

- 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.



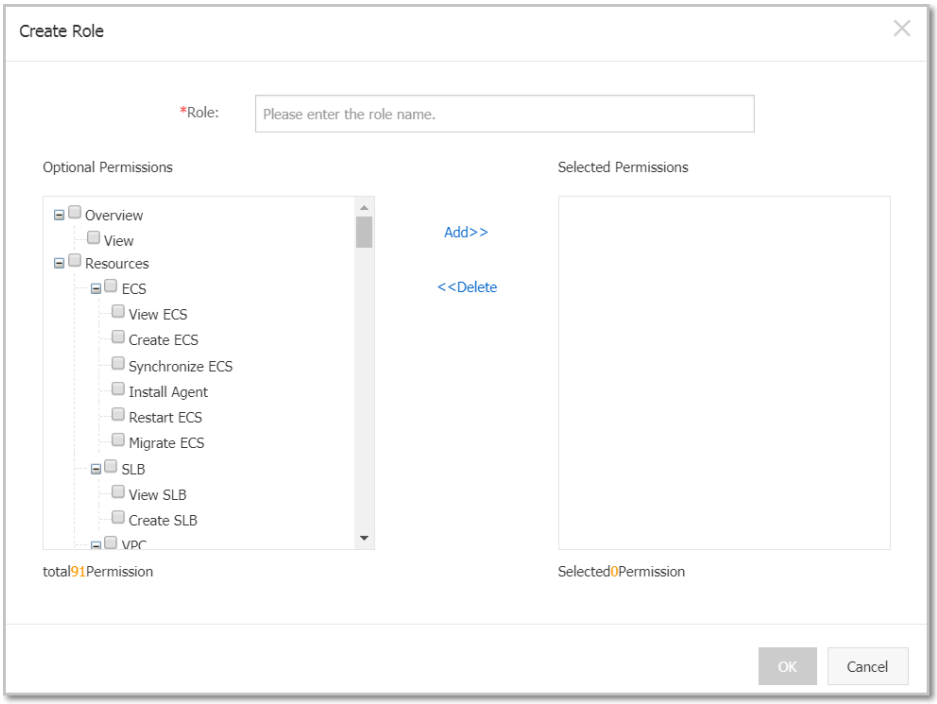
Role management

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**.

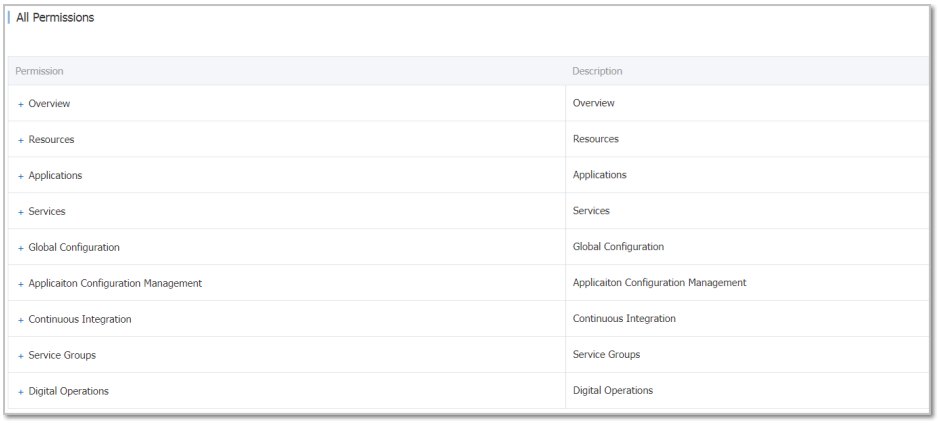


You can view, manage Permissions or Delete roles on the **Roles** page.

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.



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 management. 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.