Alibaba Cloud Application Real-time Monitoring Service

Application monitoring

Issue: 20200616

Legal disclaimer

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

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.
2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Alibaba Cloud.
3. The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through Alibaba Cloud-authorized channels. You shall pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.
4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides the document in the context that Alibaba Cloud products and services are provided on an "as is", "with all faults" and "as available" basis. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not bear any liability for any errors or financial losses incurred by any organizations, companies, or individuals arising from their download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, bear responsibility for any indirect, consequential, exemplary, incidental, special, or punitive damages, including lost profits arising from the use or trust in this document, even if Alibaba Cloud has been notified of the possibility of such a loss.

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.
6. Please contact Alibaba Cloud directly if you discover any errors in this document.

Document conventions

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

Style	Description	Example
{} or {a|b}	This format is used for a required value, where only one item can be selected.	switch {active|stand}

Contents

2 Start monitoring Java applications

2.1 Monitor Java applications (general mode)

2.2 Install agent with regular method

2.3 Install agent with shortcut method

2.4 Monitor applications in Alibaba Cloud Container Service for Kubernetes clusters

2.5 Monitor applications in open-source Kubernetes clusters

2.6 Monitor applications in Docker clusters

2.7 Start monitoring the applications deployed in other environments (such as self-built IDC)

3 Start monitoring PHP applications

3.1 Install the ARMS agent for common PHP applications

After installing the application real-time Monitoring Service (ARMS) PHP agent, you can monitor PHP applications. For example, you can view the monitoring data of application topology, traces, abnormal transactions, and slow transactions, and conduct SQL analysis. The new version of the ARMS agent for PHP has deeply optimized the application performance. ARMS can reduce the CPU and memory usage of the agent to about 5%.

Install the ARMS agent for common PHP applications

1. Select the public network download address or VPC download address based on your network environment, and use the Wget command to download the compressed installation package.

   The following figure shows the download URLs in the public network.

   ```
   # China (Hangzhou)
   wget "http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
   # China (Shanghai)
   wget "http://arms-apm-shanghai.oss-cn-shanghai.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
   # China (Qingdao)
   wget "http://arms-apm-qingdao.oss-cn-qingdao.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
   # China (Beijing)
   wget "http://arms-apm-beijing.oss-cn-beijing.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
   # China (Zhangjiakou)
   wget "http://arms-apm-zhangjiakou.oss-cn-zhangjiakou.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
   # China (Shenzhen)
   wget "http://arms-apm-shenzhen.oss-cn-shenzhen.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
   # China (Hong Kong)
   wget "http://arms-apm-hongkong.oss-cn-hongkong.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
   ```

   The following figure shows the download link in the VPC.

   ```
   # China (Hangzhou)
   wget "http://arms-apm-hangzhou.oss-cn-hangzhou-internal.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
   # China (Shanghai)
   wget "http://arms-apm-shanghai.oss-cn-shanghai-internal.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
   # China (Qingdao)
   ```

```
wget "http://arms-apm-qingdao.oss-cn-qingdao-internal.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
# China (Beijing)
wget "http://arms-apm-beijing.oss-cn-beijing-internal.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
# China (Zhangjiakou)
wget "http://arms-apm-zhangjiakou.oss-cn-zhangjiakou-internal.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
# China (Shenzhen)
wget "http://arms-apm-shenzhen.oss-cn-shenzhen-internal.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
# China (Hong Kong)
wget "http://arms-apm-hongkong.oss-cn-hongkong-internal.aliyuncs.com/grey/arms-php-agent.zip" -O arms-php-agent.zip
```

2. Run the following command to extract and move the installation package to /usr/local/arms/arms-php-agent table of contents

```
unzip arms-php-agent.zip
mkdir -p /usr/local/arms
mv arms-php-agent /usr/local/arms/arms-php-agent
```

3. In the left-side navigation pane, choose **application monitoring** > **application List**, and in **application List** at the top of the page, select the target region, and in the upper-right corner, click **application Access**.
4. In **application Access** copy LicenseKey at the top of the page.
5. In php.ini add the following code to the configuration file.

```
extension=/usr/local/arms/arms-php-agent/arms-x.x.so
[ARMS]
arms.enable=1
arms.app_name=<yourAppName>
arms.license_key=<yourLicenseKey>
arms.sock_path=/arms.sock
```

Note:

- arms-x.x.so in x.x the version of your PHP application. Currently, PHP applications of versions 5.4 to 7.3 are supported.
- Customize Settings <yourAppName>, which will display your PHP application in the ARMS console.
- Will <yourLicenseKey> replace with step4 the LicenseKey.
- If the corresponding version of your PHP application is configured with with-config-file-scan-dir, you can /etc/php/7.2/php-fpm/conf.d create a new item in the directory arms.ini file, content and above in php.ini the content added in the file is consistent.

6. Run the following command to start Hercules and report your PHP application data.

```
cd /usr/local/arms/arms-php-agent/
sudo ./hercules service install
sudo ./hercules service start
```

7. Restart the service.

   - If you are using an Ngnix server, restart the PHP-FPM.
   - If you are using the Apache server, restart Apache2 service.

Uninstall the ARMS agent

1. Delete step5 added in php.ini file content or arms.ini file.
2. Restart the service.

   - If you are using an Ngnix server, restart the PHP-FPM.
   - If you are using the Apache server, restart Apache2 service.

3. Run the following commands to stop and uninstall the Hercules service:

```
sudo ./hercules service stop
sudo ./hercules service uninstall
```

4. Run the following command to delete the arms Agent Directory:

```
sudo rm -rf /usr/local/arms/arms-php-agent
```

   You have completely uninstalled ARMS PHP agent.

Related tasks

Install the arms agent for multi-site and standalone PHP applications

After installing the application real-time Monitoring Service (ARMS) PHP agent, you can monitor PHP applications. For example, you can view the monitoring data of application topology, traces, abnormal transactions, and slow transactions, and conduct SQL analysis. This topic describes how to install the arms agent for single-server, multi-site PHP applications.

3.2 Install the arms agent for multi-site and standalone PHP applications

After installing the application real-time Monitoring Service (ARMS) PHP agent, you can monitor PHP applications. For example, you can view the monitoring data of application topology, traces, abnormal transactions, and slow transactions, and conduct SQL analysis. This topic describes how to install the arms agent for single-server, multi-site PHP applications.

Install the arms agent for multi-site and standalone PHP applications

1. Make sure that the arms agent has been installed for a common application. For more information, see Install the ARMS agent for common PHP applications.
2. Modify the Apache or Nginx configuration file.
   - For an Apache single-site multi-site application, add `php_value arms.app_name "<yourAppNewName>"` configuration, where `<yourAppNewName>` replace it with the name of your PHP application. For example, in the following code:

```
<VirtualHost *:80>
    ServerName www.example.com
    DocumentRoot /home/www/html
    php_value arms.app_name "example"
    <Directory "/home/www/html">
        Options FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
<VirtualHost *:80>
    ServerName www.test.com
    DocumentRoot /home/www/test
    php_value arms.app_name "test"
    <Directory "/home/www/test">
        Options FollowSymLinks
        AllowOverride All
        # Require all denied
        Require all granted
    </Directory>
</VirtualHost>
```

   - For a single-site application of Nginx, add `fastcgi_param PHP_VALUE "arms.app_name=<yourAppNewName>"` configuration, where `<yourAppNewName>` replace it with the name of your PHP application. For example, in the following code:

```
server {
        listen      80;
        server_name  localhost;
        location / {
                try_files $uri $uri/ /index.php? $query_string;
            }
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   /usr/share/nginx/html;
        }
        location ~ \.php$ {
            fastcgi_pass   localhost:9000;
            fastcgi_index  index.php;
            fastcgi_param PHP_VALUE "arms.app_name=example"
            fastcgi_param SCRIPT_FILENAME /var/www/html/$fastcgi_script_name;
            include        fastcgi_params;
        }
    }
 server {
        listen      80;
```

```
    server_name www.example.com
    location / {
            try_files $uri $uri/ /index.php? $query_string;
      }
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
      root   /usr/share/nginx/html;
    }
    location ~ \.php$ {
      fastcgi_pass   localhost:9000;
      fastcgi_index  index.php;
      fastcgi_param PHP_VALUE "arms.app_name=test"
      fastcgi_param SCRIPT_FILENAME /var/www/test/$fastcgi_script_name;
      include        fastcgi_params;
    }
  }
```

Uninstall the ARMS agent

1. Delete Install the ARMS agent for common PHP applicationsdocument's step5 the added php.ini file content or arms.ini file.
2. Delete the step2 the configuration content to be added in the Apache or Nginx configuration file.
3. Restart the service.
   - If you are using an Ngnix server, restart the PHP-FPM.
   - If you are using the Apache server, restart Apache2 service.
4. Run the following commands to stop and uninstall the Hercules service:

   ```
   sudo ./hercules service stop
   sudo ./hercules service uninstall
   ```

5. Run the following command to delete the arms Agent Directory:

   ```
   sudo rm -rf /usr/local/arms/arms-php-agent
   ```

   You have completely uninstalled ARMS PHP agent.

Related tasks

Install the ARMS agent for common PHP applications

After installing the application real-time Monitoring Service (ARMS) PHP agent, you can monitor PHP applications. For example, you can view the monitoring data of application topology, traces, abnormal transactions, and slow transactions, and conduct SQL analysis. The new version of the ARMS agent for PHP has deeply optimized the application performance. ARMS can reduce the CPU and memory usage of the agent to about 5%.

4 Function specification

4.1 Application overview

When connected to Application Real-Time Monitoring Service (ARMS), your application is monitored by ARMS in all rounds. You can view the key health metrics of your application on the **Application Overview** page. The topology allows you to preview the upstream and downstream dependent components of the component, and the 3D topology allows you to view the health status of your application, services, and hosts.

Portal

1. Log on to the ARMS console. In the left-side navigation pane, choose **Application Monitoring** > **Applications**.
2. On the **Applications** page, click the name of the target application to access the **Application Overview**.

   At the top of the **Application Overview** page, you can click tabs **Overview Analysis**, **Topology Graph**, and **3D Topo (Probation)** to view the corresponding information.

Features

- **Key application metrics**

  The **Overview Analysis** tab shows these key metrics:

  - Total number of requests, average response time, error count, real-time number of instances, number of full garbage collection (GC) activities, number of slow SQLs, number of exceptions within the specified time range, and how they change compared with the previous day and previous week.
  - Application Support Services: time sequence curves of the application support service requests and average response time.
  - Application Dependent Services: time sequence curves of the application dependent service requests, average response time, and number of application instances, and statistics of the HTTP status code.
  - System info: time sequence curves of CPU, memory, and load.
  - Slow call: time sequence curves and details of slow calls.
  - Statistical analysis: analysis on slow API calls and exception types.

- **Application topology**

  On the **Topology Graph** tab, you can get a better view of the upstream and downstream components of your application and their call relations, allowing you to effectively identify the bottlenecks of your application.

- **3D Topo (Probation)**

  The **3D Topo (Probation)** tab provides a comprehensive view of the health conditions for your application, service, and host, and the upstream and downstream dependencies of the application. With the 3D topology, you can quickly identify the services that caused failures, applications influenced by the failures, and associated hosts. This enables you to thoroughly investigate the root cause of the failures, and then fix the issues quickly. For more information about the 3D topology function, see 3D topology.

4.2 Application details

4.2.1 JVM monitoring

The application monitoring function of Application Real-Time Monitoring Service (ARMS) provides the Java Virtual Machine (JVM) monitoring function. It monitors heap metrics, non-heap metrics, direct buffer metrics, memory-mapped buffer metrics, garbage collection (GC) details, and the number of JVM threads. This topic describes the JVM monitoring feature and how to monitor JVM metrics.

Features

ARMS' JVM monitoring feature can help you monitor the following metrics.

- Heap memory
  - heap_init: initial bytes of the heap memory
  - heap_max: maximum bytes of the heap memory
  - heap_commited: submitted bytes of the heap memory
  - heap_used: used bytes of the heap memory
- Non-heap memory
  - non_heap_init: initial bytes of the non-heap memory
  - non_heap_max: maximum bytes of the non-heap memory
  - non_heap_commited: submitted bytes of the non-heap memory
  - non_heap_used: used bytes of the non-heap memory

- Direct buffer
  - direct_capacity: total size of direct buffer (in bytes)
  - direct_used: used size of the direct buffer (in bytes)
- Memory-mapped buffer
  - mapped_capacity: total size of memory-mapped buffer (in bytes)
  - mapped_used: used size of memory-mapped buffer (in bytes)
- GC details
  - GcPsMarkSweepCount: GC Parallel Scavenge MarkSweep amount
  - GcPsScavengeCount: GC Parallel Scavenge amount
  - GcPsMarkSweepTime: GC Parallel Scavenge MarkSweep time
  - GcPsScavengeTime: GC Parallel Scavenge time
- Number of JVM threads
  - ThreadCount: total number of threads
  - ThreadDeadLockCount: number of deadlocked threads
  - ThreadNewCount: number of new threads
  - ThreadBlockedCount: number of blocked threads
  - ThreadRunnableCount: number of treads that can run
  - ThreadTerminatedCount: number of terminated threads
  - ThreadTimedWaitCount: number of threads in timed waiting
  - ThreadWaitCount: number of waiting threads

Procedure

1. Log on to the ARMS console. In the left-side navigation pane, choose **Application Monitoring** > **Applications**.
2. On the **Applications** page, click the name of the target application.
3. On the **Application Details** page, select the target node and click the **JVM Monitoring** tab on the right side of the page.

   The **JVM Monitoring** tab shows the time sequence curves of the instantaneous times of GC, instantaneous time consumption of GC, heap memory details, non-heap memory details, and number of JVM threads.

   - Click **Instantaneous** and **Accumulated** in the upper-right corner of **Instantaneous Count/Min** and **Instantaneous Duration/Min** panels. You can alternatively view

the time sequence curves of the instantaneous or accumulated times and time consumption of GC. By default, the instantaneous values are displayed.

- Click a metric name (for example, the number of instantaneous GC times) on each monitoring panel to enable or disable the visibility of this metric in the chart.

> **Note:**
> Each chart must have at least one visible metric.

4.2.2 Host monitoring

The application monitoring function of Application Real-Time Monitoring Service (ARMS) provides the host monitoring function. It monitors metrics such as the CPU, memory, disk, load, network traffic, and network data packets. This topic describes the host monitoring feature and how to view host monitoring metrics.

Features

Host monitoring helps you monitor the following metrics.

- CPU
  - SystemCpuIdle: idle CPU usage in the last five seconds
  - SystemCpuSystem: system CPU usage in the last five seconds
  - SystemCpuUser: user CPU usage in the last five seconds
  - SystemCpuIOWait: I/O await CPU usage in the last five minutes
- Memory
  - SystemMemFree: idle system memory (in KB)
  - SystemMemTotal: total system memory (in KB)
  - SystemMemUsed: used system memory (in KB)
  - SystemMemBuffers: current memory in the cache of system buffer (in KB)
  - SystemMemCached: current memory in the cache of system page (in KB)
  - SystemMemSwapFree: idle memory of system swap (in KB)
  - SystemMemSwapTotal: total memory of system swap (in KB)
- Disk
  - SystemDiskTotal: total bytes of the system disk
  - SystemDiskFree: idle bytes of the system disk
  - SystemDiskUsedRatio: usage of system disk

- Load
  - SystemLoad: load on the system
- Network traffic
  - SystemNetInBytes: average bytes per second received through the network in the last 30 seconds
  - SystemNetInErrs: average errors per second received through the network in the last 30 seconds
  - SystemNetOutBytes: average bytes per second sent through the network in the last 30 seconds
  - SystemNetOutErrs: average errors per second sent through the network in the last 30 seconds
- Network packets
  - SystemNetInPackets: average messages per second received through the network in the last 30 seconds
  - SystemNetOutPackets: average messages per second sent through the network in the last 30 seconds

Procedure

1. Log on to the ARMS console. In the left-side navigation pane, choose **Application Monitoring** > **Applications**.
2. On the **Applications** page, click the name of the target application.
3. On the **Application Details** page, select the target node and click the **Host Monitoring** tab on the right side of the page.

   The **Host Monitoring** tab shows the time sequence curves of the CPU, memory, disk, load, network traffic, and network packets.

   - You can enable or disable the visibility of a metric in the chart by clicking its name (for example, the system CPU usage) on the monitoring panel.

     Note:
     Each chart must have at least one visible metric.
   - Click the two alert icons in the upper-right corner of the monitoring panel. You can view existing alert points and create new alerts. For how to create alerts, see #unique_20.

4.2.3 Memory snapshot

The application monitoring function of Application Real-Time Monitoring Service (ARMS) provides the memory snapshot feature. After creating the memory snapshot, you can view specific information of multiple memory metrics within the specified time range through detailed logs. This topic describes the scenarios and methods for using memory snapshots.

Scenarios

ARMS provides JVM monitoring to give you a clear view of multiple memory metrics within the specified time range. Although the chart can show you that the memory is overused, it cannot provide detailed information for you to identify the root cause. You can create a memory snapshot and view detailed memory usage through the log.

Procedure

1. Log on to the ARMS console. In the left-side navigation pane, choose **Application Monitoring** > **Applications**.
2. On the **Applications** page, click the name of the target application.
3. On the **Application Details** page, select the target node and click the **JVM Monitoring** tab on the right side of the page.
4. In the upper-right corner of the **JVM Monitoring** tab, click **Create Memory Snapshot**.

   Note:
   When you click **Create Memory Snapshot**, if the previous snapshot job is still running, you will get an error message. Wait until the previous snapshot job is finished. Currently, you can only create memory snapshots for Linux systems.

5. In the **Create Memory Snapshot** dialog box, select an IP address from the IP drop-down list and click **OK**.

   Note:
   If you create a snapshot at the machine level under an application, then the IP address of this machine is selected by default in the **IP** field.

   Notice:

> The running time of a snapshot job varies from a few minutes to half an hour. When a snapshot job is running, you may experience brief freezing. Be cautious in using this function.

The created snapshot is displayed in the snapshot job list of **Snapshot History**.

The information of a snapshot job includes:

- IP address
- Creation time of snapshot
- Running time of snapshot job
- **Delete**: Used to delete a snapshot
- **Detail**: Used to view details of a memory snapshot

Green indicates a successful snapshot job, and red indicates a failed one.

6. Click **Detail** to open the **Snapshot Details** dialog box and view the details.
   - **Dominator Tree**: Lists the top 5 objects that use the most memory, in descending order.

     The indentation of lines represents the dominance relation between objects. If the Number 1 object on the list takes up a small portion or uses little memory, it means no object is overusing memory. Otherwise, modify the memory object to shrink it or quickly release it.
   - **Histogram**: Lists the top 20 classes that use the most memory, in descending order.

4.3 API call monitoring

This topic explains the API call monitoring function of application monitoring.

Features

On the **Interface Invocation** page of application monitoring, you can view the detailed statistics of the APIs of the application. Application Real-Time Monitoring Service (ARMS) can automatically discover and monitor the APIs provided in the following web frameworks and RPC frameworks:

- Tomcat 7+
- Jetty 8+
- Resin 3.0+
- Undertow 1.3+

- WebLogic 11.0+
- SpringBoot 1.3.0+
- HSF 2.0+
- Dubbo 2.5+

API overview

The **Overview** tab of the **Interface Invocation** page shows all APIs detected by ARMS. You can sort the list by response time, request count, or error count. Select a service to view its detailed topology and the line charts of request count, response time, and error count on the **Overview** tab.

SQL analysis

The **SQL Analysis** tab shows the list of SQL requests initiated within the code snippets of the selected service on the left side. Through this tab, you can identify which SQL request is the cause for the slow response of a service. You can also click **Interface Snapshot** of an SQL request to view the complete trace where the SQL is executed.

Exception analysis

The **Exception Analysis** tab shows the exceptions thrown within the code snippets of the selected service on the left side. You can also click **Interface Snapshot** of an exception to view the complete trace where the exception stack is located.

Interface snapshot

In the service trace snapshot, you can view the call stack of a single call, details of the executed SQLs, details of the thrown exceptions , and the parameters of the API.

4.4 MQ monitoring

The MQ Monitoring page in application monitoring of Application Real-Time Monitoring Service shows the message publishing and subscription of Message Queue for RocketMQ (RocketMQ) topics.

Portal

Follow these steps to access the MQ Monitoring page of ARMS application monitoring.

1. Log on to the ARMS console. In the left-side navigation pane, choose **Application Monitoring** > **Applications**.
2. On the **Applications** page, click the name of the target application.

3. Click **MQ Monitoring** in the left-side navigation pane.
4. Click a search result link on the right side of the page.

When you complete the preceding steps, the Overview tab of the MQ Monitoring page appears.

Features

The MQ Monitoring page:

- Shows the message publishing and subscription relationship between your application and the MQ data source in the topology graph.
- Shows the statistics of message publishing, including request count, response time, and error count.
- Shows the statistics of message subscription, including latency, request count, response time, and error count.
- Provides the interface snapshots of message publishing and subscription. You can view the complete trace through the trace ID and diagnose the problems.

Available actions

- In the time picker in the upper-right corner of the page, select the start time and end time of the statistics you want to view.
- Click the **Publishing Statistics** and **Subscription Statistics** tabs to view the statistics of message publishing and subscription.
- Click the **Interface Snapshot** tab to view the interface snapshots of message publishing and subscription. You can view the complete trace through the trace ID and diagnose the problems when needed.
- Click **Back to Overview** to go back to the MQ Monitoring page.

4.5 Trace query

On the **Invocation Trace Query** page, you can query the details about a specific trace by trace ID, or query traces by multiple filtering criteria.

TraceId: the unique ID of each trace, through which you can accurately query the calling data.

Trace: supports distributed traces, and supports querying inter-service traces and local service traces by using the local call method stack.

- Distributed trace (inter-service trace): a trace between services.
- Local trace (method stack): the local method stack of one-time inter-service traces.

Query traces by using either of the following methods:

- Precise query: Select **TraceId** from the **Parameter Name** drop-down list. Enter the specific trace ID in the **Parameter Value** field, and click **Query**.
- Advanced query: Query traces by setting multiple criteria.

Portal

Follow these steps to access the **Invocation Trace Query** page.

1. In the left-side navigation pane, choose **Application Monitoring** > **Invocation Trace Query**.

Query traces

You can query a trace by specifying the trace ID or using multiple filtering criteria.

Table 4-1: Query fields

Query field	Description
Call Type	• HTTP Entry Point: The client uses HTTP protocol for calling. • Dubbo Call: The client uses Dubbo for calling. • HSF Call: The client uses HSF for calling.
Takes Longer Than	The consumed time for the call is longer than the specified milliseconds.
Show Abnormal Calls Only	Check this option to single out calls that throw exceptions.
Client Name/Client IP	The name and IP of the application that initiates the call.
Server Name/Server IP	The name and IP address of the called application.

Query field	Description
Interface Name	The name of the interface called by the application. Prefix fuzzy match is supported. For example, you can use keyword api or resource to search for the /api/ResourceQuery interface.

View inter-service traces (distributed traces)

Go to the **Distributed Invocation Trace** page by clicking the trace ID of the trace you want to view.

Field description

- Status: whether the call is normal. Red indicates that an exception exists in the local trace of the service. Green indicates that the call is normal.
- IP Address: the IP address of this application.
- Call Type: the type of the call, which corresponds to the Call Type option of the ad hoc query.
- Timeline: the time consumption of each inter-service trace, and time distribution of this trace relative to that of the entire trace.

View local trace of the service (method stack)

On the **Invocation Trace** page, click the magnifier icon in the **Method Stack** column to go to the Method Stack page.

Field description

- Called Method: the method for calling the local method stack. After the call method is expanded, the next-layer calls of the method are displayed.
- Line Number: the number of the line where the local method code is located.
- Expansion information:
  - Parameters: the input parameters of the call.
  - SQL: the SQL statements called by the database.
  - Exception: the details of exceptions and other information.
- Timeline: the time distribution of every method call of the local trace.

4.6 3D topology

The application monitoring function of Application Real-Time Monitoring Service provides the 3D topology, which shows the health condition of applications, services, and hosts, in addition to the upstream and downstream dependencies of the applications. With the 3D topology, you can quickly identify the services that caused failures, applications influenced by the failures, and associated hosts. This enables you to thoroughly investigate the root cause of the failures, and then fix the issues quickly.

Quick start

Portal

Follow these steps to access the 3D topology function in ARMS application monitoring.

1. In the left-side navigation pane, choose **Application Monitoring** > **Applications**.
2. On the Applications page, take the following actions as needed.
   - To view the 3D topology of all applications, click **View 3D Topology of All Applications (Beta)** at the top.
   - To view the 3D topology of a specific application, find the application and click **3D Topology** in the **Actions** column.

> **Note:**
>
> Applications are displayed in a list on the Applications page by default. If the page is in the card view, click the 3D Topology icon in the upper-right corner of the application card.

Overview

On the **Overview** page that appears by default, you can see all the content of the service layer, application layer, and host layer. In the upper-right corner of the page, you can find the number of hosts, applications, and services.

Figure 4-1: Overview layer of ARMS 3D Topology

On the overview layer, you can perform the following operations:

- In the upper-left corner, click the time range section, and select the specific start time and end time in the time picker that appears.
- In the timeline at the top of the page, drag the slider to change the time range of the current view.
- In the search box in the upper-right corner of the page, enter your keywords and press Enter to search.
- Drag with your mouse to view the data on all three layers from different dimensions.
- Click any object in the view to check metrics related to that object on the right-side panel .

Service

The service layer shows the services that your applications depend on.

Figure 4-2: Service layer of ARMS 3D Topology

Services under each application are grouped into a block. The more the services are called, the bigger their block is. Different states of the services are indicated in different colors.

- : Service calls are normal.
- : The error rate of the service is relatively high.
- : No data is returned from the service.

Note:
The response time threshold of services is configurable. In the left-side navigation pane, click the triangle icon next to Service to open the Threshold Settings box. Drag the slider in the Threshold Settings box to set the threshold.

After you click a service, the right-side panel shows the following information:

- The name of the service
- QPS: the queries per second
- RT(ms): the response time in milliseconds
- ErrQps: the error queries per second

Note:
In the QPS, RT, and ErrQps sections, the left-side numbers are the average value within the selected time range, and the corresponding line chart is on the right side.

Application

The application layer shows the applications and their upstream and downstream dependencies, including the middleware they depend on. By checking the direction of the connecting lines, you can determine the calling and called objects.

Figure 4-3: Application layer of ARMS 3D Topology

After you click an application, the right-side panel shows the following information of that application:

- Application Name
- QPS: the queries per second
- RT(ms): the response time in milliseconds
- ErrQps: the error queries per second

Note:
In the QPS, RT, and ErrQps sections, the left-side numbers are the average value within the selected time range, and the corresponding line chart is on the right side.

Docker/ECS

The Docker/ECS layer shows the host information of your applications.

Figure 4-4: Host layer of ARMS 3D Topology

Each cube is a host. All hosts are grouped by applications. Different states of the hosts are indicated in different colors.

- : Normal
- : Slow
- : Alerting
- : Abnormal
- : Offline

After you click a host, the right-side panel shows the following information of that host:

- IP address and basic information of the host:
  - IDC
  - Unit
  - Host
  - CPU: Number of CPU cores
  - JVM: JVM version
  - Tomcat: Tomcat version
- CPU: CPU usage
- MEM: Memory usage
- LOAD: Load

Note:

In the CPU, MEM, and LOAD sections, the left-side numbers are the average value within the selected time range, and the corresponding line chart is on the right side.

4.7 Application diagnosis

4.7.1 ### Real-time diagnosis

The real-time diagnosis function provided by application monitoring of Application Real-Time Monitoring Service (ARMS) is suitable for scenarios that need close monitoring of application performance and location of the cause of problems within a short time. This topic describes how to use the real-time diagnosis function.

Context

When you need to closely monitor the application performance for a short period of time, such as releasing an application or performing stress tests on the application, you can use the real-time diagnosis function of ARMS application monitoring. After real-time diagnosis is enabled, ARMS application monitoring continuously monitors the target application for five minutes and reports all data of the traces during this period. Next, you can use the method stack waterfall diagram and thread profiling to identify the causes of the exception based on the trace that shows performance problems.

Quick start

Portal

Follow these steps to go to ARMS application monitoring for real-time diagnosis.

1. Log on to ARMS console.
2. In the left-side navigation pane, choose **Application Monitoring** > **Applications**.
3. On the **Applications** page, click the name of the target application.
4. In the left-side navigation pane, click **Real-time Diagnosis**.

Enable and disable real-time diagnosis

The first time you access the **Real-time Diagnosis** page, real-time diagnosis is automatically enabled. To enable real-time diagnosis in other cases, click **Enable real-time diagnosis** in the upper-right corner.

Real-time diagnosis is automatically started for five minutes and then disabled. To disable the real-time diagnosis in advance, click **Terminate Real-time Diagnosis** in the upper-right corner.

View real-time monitoring data

In the **Real-time Requests Distribution** and **Requests by Response Time** sections, you can view the statistics of the last 1000 requests captured as of the current time point.

In the chart of the **Real-time Requests Distribution** section, select a time range. Data of the selected time range can be set as visible. That is, the chart only displays data from this time range. Click **Reset** in the upper-right corner of the chart and the default view can be restored.

Filter monitoring data

You can filter request monitoring data displayed on the page by interface name and IP address.

1. Click the **+** icon at the top of the **Real-time Requests Distribution** section.
2. Select an interface or IP from the drop-down list, and click **Query**.
   Only the request monitoring data of the selected interface is displayed on the page.

View information of traces

On the **Traces** and **Interface Aggregation** tabs, you can view all information of the trace captured in the corresponding period. Click a TraceId to access the **Link Invocation** page.

Use the local method stack waterfall diagram and thread profiling to locate the causes for the exception. For more information, see Trace query and Use ARMS TProf to analyze code problems.

4.8 Application Settings

4.8.1 Custom configuration

Some common settings of application monitoring, such as the sampling rate of traces, agent switch, and slow SQL threshold, can be directly configured on the **Custom Configuration** tab.

Prerequisites

#unique_30

Portal

1. Log on to ARMS console.
2. In the left-side navigation pane, choose **Application Monitoring** > **Applications**.
3. On the **Applications** page, click the name of the target application.
4. In the left-side navigation pane, click **Application Settings** and then the **Custom Configuration** tab on the right.

Note:

You must click **Save** at the bottom of the page for the settings to take effect.

Configure trace sampling settings

In the **Invocation Trace Sampling Settings** section, you can turn on or off the sampling of traces, and set the sampling rate. You only need to enter the number of the percentile in the **Sampling Rate Settings** field. For example, when you enter 10, the sampling rate is 10%.

Notice:

The modification takes effect immediately without restarting the application. If sampling is turned off, the trace data will not be captured. Proceed with caution.

Configure the agent switch and log level

In the **Agent Switch Settings** section, you can turn on or off the master switch of the agent and other plug-in switches, and configure the log level.

Notice:
The modifications to the master switch of the agent and log level take effect immediately without restarting the application. If the master switch of the agent is turned off, Application Real-Time Monitoring Service (ARMS) cannot monitor your applications. Proceed with caution. To make changes to each plug-in switch take effect, you must manually restart the application.

Configure threshold settings

In the **Threshold Settings** section, you can set the slow SQL query threshold, interface response time threshold, and throttling threshold.

Configure advanced settings

In the **Advanced Settings** section, you can set the interface to be filtered and the maximum length of the method stack.

- **Invalid Interface Invocation Filtering**: Enter an interface whose call status does not need review. This interface is then hidden from the **Interface Invocation** page.
- **Method stack maximum length**: Default value: 128 entries; upper limit: 400 entries.
- **Maximum Length of Captured SQL Statements**: Default value: 1024 characters; lower limit: 256 characters; upper limit: 4096 characters.
- **Capture SQL Bound Variables**: Captures the variable value bound with **PrepareStatement**. This setting takes effect without restarting the application.
- **Exception filtering**: The exception entered here is not displayed in the chart on the Application Details and Exception Analysis tabs.
- **Error Count Filtering**: By default, HTTP status codes greater than 400 are counted as errors. You can customize a threshold value greater than 400.
- **Compress Traces**: Turning on this switch compresses invocation traces and reduces the used storage space.
- **Maximum Length of Request Input Parameters**: Default value: 512 characters; upper limit: 2048 characters.
- **Show Percentiles**: Allows for turning on or off quantile statistics.
- **Enable application emergency alert**: Supports alerts for emergencies such as thread deadlocks and out-of-memory (OOM). The Versions of the ARMS agent must be 2.5.8 +.

Configure thread settings

In the **Thread Settings** section, you can turn on or off the switch of the thread diagnosis method stack and the thread profiling master switch. You can also set the trigger threshold of the slow call listener.

> **Note:**
> The listener is started only when the service call response time exceeds the threshold (1,000 ms by default) and lasts until the call ends or the consumed time exceeds 15 seconds. We recommend that you set the threshold to the 99th percentile of the call response time For example, if 100 calls are listed in ascending order by response time, the 99th one is represented by the 99th percentile.

Configure memory snapshot settings

In the **Memory Snapshot Settings** section, you can turn on or off memory snapshots. If you turn it on, a memory dump (at most one time a day) will be created in the case of memory leaks.

Configure URL convergence rules

In the **URL Convergence Settings** section, you can enable or disable the convergence function. You can also set the convergence threshold and convergence rules. URL convergence means that a series of similar URLs are displayed as individual objects. For example, the first half is displayed as a series of URLs in /service/demo. The convergence threshold is the minimum number of conditions for URL convergence. For example, when the threshold is 100, URLs converge only when 100 of them meet the regular expression of the rules.

5 Tutorials

5.1 Use ARMS TProf to analyze code problems

Application Real-Time Monitoring Service (ARMS) TProf is a code-based diagnosis tool. It can automatically capture stack snapshots of slow calls and truly restore the first scene of code execution.

Scenarios

Here are typical scenarios of the use of ARMS TProf:

- If slow calls occur during the peak traffic of promotions and sales, ARMS TProf can help you quickly locate the faulty code.
- If large amounts of slow calls occur in the system, ARMS TProf can automatically save the first scene.
- If occasional slow calls cannot recur because the business is too complex, ARMS TProf can restore the real code execution trajectory.

Procedure

Set thread profiling parameters

1. In the left-side navigation pane of the console, choose **Application Monitoring** > **Applications**.
2. On the **Applications** page, click the name of the target application.
3. In the left-side navigation pane, click **Application Setup** and then the **Custom Configuration** tab on the right of the page.
4. In the **Thread Profiling Settings** section, you can turn the thread profiling control switch on or off, and set the trigger threshold for slow calls.

Note:

- The listener is started only when the service call response time exceeds the threshold (1,000 ms by default), and it lasts until the call ends or the consumed time exceeds 15 seconds.

- We recommend that you set the threshold to the 99th percentile of the call response time For example, if 100 calls are listed in ascending order by response time, the 99th one is represented by the 99th percentile.

View thread profiling details through interface snapshots

1. In the left-side navigation pane of the console, choose **Application Monitoring** > **Applications**.
2. On the **Applications** page, click the name of the target application.
3. In the left-side navigation pane, click **Interface Invocation**, and then the **Interface Snapshot** tab on the right of the page.
4. On the **Interface Snapshot** tab, click a TraceId link. The **Link Invocation** tab opens in a new window.
5. Click the magnifier icon in the **Thread Profiling** column. The **Thread Profiling** dialog box appears.

Note:

- The actual response time is the actual execution time for the service call, which is not affected by thread profiling.
- The listening response time is the time consumed for TProf listening In general, the listening response time is approximately the actual response time minus the trigger threshold for slow calls.

View thread profiling details through multi-dimensional query

1. In the left-side navigation pane of the console, choose **Application Monitoring** > **Multi-dimensional Query**.
2. On the **Traces and Events**tab, select **Contain only thread profiling snapshots** from the **Parameter Name** drop-down list and then click **Query**.
3. Click a TraceId link in the search results. The **Link Invocation** tab opens in a new window.
4. Click the magnifier icon in the **Thread Profiling** column.

FAQ

Why are some slow calls not listened to?

A: If a large number of slow calls occur in a short period of time, to ensure the stability of the business system, ARMS limits listening resources for some slow calls.

5.2 Diagnose errors on the server

It is challenging to analyze the causes of webpage errors, which are one of the most common problems of Internet applications. After being installed on an application, the Application Real-Time Monitoring Service (ARMS) agent can automatically capture, collect, count, and track exceptions without modifying the application code. You can use the agent to accurately locate all exceptions in the application and perform online diagnosis.

Problem description

Webpage errors, frequently taking the form of a "5XX" error, are one of the most common problems for Internet applications. "5XX" errors usually occur on the server side. The server side has the most complex business logic and it is the most error-prone part of the entire network-request link. Errors on the server side prove to be the most challenging for cause analysis. O&M engineers or development engineers often have to log on to the target server to view logs and determine the causes.

For applications with less complex logic and short uptimes, logging on to the server to view logs can solve most of these problems. However, this method has been proven useless many times in the following scenarios.

- When you want to know the time and frequency of a specific type of error in a distributed application cluster.
- A system has been running for a long time, but you do not care about the residual exceptions. You just want to know about today's additional exceptions as compared with yesterday's, and additional exceptions after the release of the system compared with before the release of the system.
- When you view the web requests and relevant parameters associated with an exception.
- Customer Services provides the number of an order that the user fails to place for cause analysis of the failure.

Solution

Install an ARMS agent on the application. The ARMS agent can automatically capture, collect, count, and track exceptions without modifying the application code. The agent presents a clear picture of various errors.

Prerequisites

Make sure you have installed the ARMS agent for your application.

Step 1: View statistics of application exceptions

The installed ARMS agent collects and shows the application's total requests, average response time, count of errors, real-time instances, number of full garbage collection (GC) activities, slow SQLs, exceptions, and slow calls within the selected period. The agent also shows how these metrics change when compared with their counterparts in the previous day and previous week. Follow these steps to view the statistics of application exceptions.

1. In the left-side navigation pane of the ARMS console, choose **Application Monitoring** > **Applications**.
2. On the **Applications** page, click the name of your application.
3. On the **Application Overview** page, the total number of exceptions and how it changes from the previous day and previous week are displayed at the top of the **Overview Analysis** tab.
4. Scroll the page to the **Exception Type** section at the bottom of the **Overview Analysis** tab. Here you can view the number of times for which each type of exception occurs.
5. In the left-side navigation pane, click **Application Details**, and then **Exception Analysis** on the right of the page. You can view exception statistics, error count, and exception details on this tab.

Step 2: Diagnose the causes of the exception

The statistics of application exceptions are insufficient for locating the causes for exceptions. The abnormal stack in the log contains the code snippet for the call. However, it does not contain the complete upstream and downstream information and request parameters of this call. The ARMS agent's bytecode enhancement technology allows you to capture complete upstream and downstream call snapshots of exceptions at a low performance consumption. Then you can further identify the specific causes for the exceptions.

1. On the **Exception Analysis** tab, click **Interface Snapshot** in the **Actions** column of a certain type of exception.

   Call trace related to this exception type is displayed on the **Interface Snapshot** tab.
2. On the **Interface Snapshot** tab, click the TraceId of an incorrect call.

   To find the target trace, see Trace query .

3. On the page that appears, view the trace information of the target exception. Click the magnifier icon in the **Method Stack** column to view the detailed request parameters and exception log of the call. This allows you to retrieve the context information of the exception.

At this point, the causes for the exception are revealed. This effectively helps you with the subsequent code optimization. You can also return to the **Interface Invocation** page to view other exceptions in the list and solve them one by one.

5.3 Diagnose application access problems

The locating and troubleshooting of the causes for application access problems present many challenges. Application Monitoring of Application Real-Time Monitoring Service (ARMS) provides a set of solutions, such as thread profiling, tracing diagnosis, and API monitoring. These solutions help you quickly and accurately locate all slow calls in an application and solve the application access problems.

Analysis

Website freezing and slow webpage loading are among the most common problems of Internet applications. Troubleshooting and solving these problems is complex and takes a long time. Here are the major reasons:

- Overly long application tracing
  - The cause can be any failure at any stage of the long trace. The failure can happen between the front-end webpage and the back-end gateway, or between the web application server and the back-end database.
  - Applications using the microservice architecture have even more complex traces. Different components of applications might be maintained by different teams and persons, which makes troubleshooting more difficult.
- Incomplete or low-quality logs
  - Most methods of troubleshooting online problems rely on application logs. However, the locations of problems are often unpredictable, and "slow response" often occurs . To find the true causes of "slow response", you need to print the log in every place where errors can occur and record each call. The cost is very high for doing so.

- Insufficient monitoring
  - Rapid business development and fast application iteration lead to frequent API changes of applications and increased dependencies. This further contributes to the deterioration of the quality of code. Applications need a comprehensive monitoring system that automatically monitors every API of the application and records abnormal calls.

Solution

After being installed, the ARMS agent can use ARMS application monitoring features, such as thread profiling, trace diagnosis, API monitoring, to monitor all slow calls. This does not change the code of your application.

Prerequisites

Make sure you have installed the ARMS agent for your applicartion.

Step 1: View the statistics of slow SQLs

The installed ARMS agent collects and shows the application's total requests, average response time, count of errors, real-time instances, number of full garbage collection (GC) activities, slow SQLs, exceptions, and slow calls within the selected period. The agent also shows how these metrics change when compared with their counterparts in the previous day and previous week. Follow these steps to view the statistics of slow SQLs.

1. In the left-side pane of the ARMS console, choose **Application Monitoring** > **Applications**.
2. On the **Applications** page, click the name of your application.
3. On the **Application Overview** page, the total number of exceptions and how these changed from the previous day and previous week are displayed at the top of the **Overview Analysis** tab.

   In this example, there are 42 times of slow SQLs.

Step 2: Find and locate slow APIs

On the **Interface Invocation** page, ARMS shows all APIs provided by the monitored application, and the number of calls and consumed time on this API. Slow APIs are marked to help you quickly locate them.

1. Click **Interface Invocation** in the left-side pane of the ARMS console.

2. In the **Interface Selection** section of the **Interface Invocation** page, select the slow API that is called the most frequently. View detailed information of the API on the right.

Step 3: View and locate the problem code

After locating a slow API, you need to find the problem code to eliminate the problem. A snapshot is a complete record of an entire trace. The snapshot includes the code and time consumption of each call, and helps you precisely locate the problem code.

1. Click the **Interface Snapshot** tab on the **Interface Invocation** page.

   On the **Interface Snapshot** tab, you can view snapshots of all APIs corresponding to this API.
2. On the **Interface Snapshot** tab, click the TraceId of a trace, and then click the magnifier icon in the **Method Stack** column to view the problem code.

   To find the target trace, see Trace query.

   In this example, most of the time of this 705 ms call is spent in calling the SQL `SELECT * FROM l_employee`.

At this point, the causes for a specific slow call are revealed. This effectively helps you with the subsequent code optimization. You can also return to the **Interface Invocation** page to view other slow calls in the list and solve them one by one.

5.4 Use active diagnosis to troubleshoot response time errors

Locating and troubleshooting response time errors is a long and complex process. Application Real-Time Monitoring Service (ARMS) provides the active diagnosis feature to help you quickly and accurately locate response time errors. Active diagnosis of ARMS shortens the response time of applications.

Background

Response time errors are likely caused by the long response time of downstream applicatio ns, uneven traffic, an excessive number of full garbage collection (GC) activities, or an excessive load. To locate and troubleshoot response time errors, you need to:

- Know the server that causes this spike in response time.
- Conduct an SQL analysis on the time consumption of applications.
- Check the number of full GC activities of the application and whether there is a spike of time consumption.

- Check whether there are memory leaks.
- Check the exception log.
- Inspect whether the response time of downstream applications follows the same trend.

Prerequisites

Make sure you have installed the ARMS agent for your application.

Step 1: View exception information

The installed ARMS agent collects and shows the application's total requests, average response time, count of errors, real-time instances, number of full garbage collection (GC) activities, slow SQLs, exceptions, and slow calls within the selected period. Follow these steps to view the exception information of the application.

1. In the left-side pane of the ARMS console, choose **Application Monitoring** > **Applications**.

   On the **Applications** page, if the application has an exception, the status bar is displayed in red.
2. On the **Applications** page, click the red dot in the status bar of the target application.
3. View exception details on the **Key Events** page.

   The **Key Events** page shows the time sequence curves of the response time and average response time of the current application and the downstream applications. Information of abnormal APIs and the Trace IDs of the top 5 abnormal calls are also displayed.

Step 2: Diagnose the causes of the exception

The statistics of application exceptions are insufficient for locating the causes for exceptions. You can use SQL analysis, tracing analysis, or interface snapshot to quickly locate the causes of an exception.

1. On the **Key Events** page, click the name of the dependent service. For example, click **Invoke MYSQL**. Then, in the **Application Dependent Services** section of the **Overview** page, view the details of downstream applications.

   The **Application Dependent Services** section shows the time sequence curves of the request volume, average response, number of application instances, and HTTP status code statistics of the application dependent services. In this example, the response time of the downstream application reaches 1680 ms. It can be concluded that the application's spike in response time is caused by the downstream application's spike in response time.

2. Return to the **Key Events** page and click the API that contributes the most to the spike. For example, click **/Xxxdata/... page**, and then the **SQL Analysis** tab on **/Xxxdata/... page** to view API information.

   The **SQL Analysis** tab shows the SQL call statistics and SQL statement details. For more information, see SQL analysis. In this example, it can be concluded that the cause of the spike in response time of the current application is overly slow SQL calls of the downstream application.

3. Return to the **Key Events** page and click the Trace ID of an application whose time consumption is among the top 5. Then click the magnifier icon in the Method Stack column to view the problem code.

   To find the target trace, see Trace query.

   In this example, it can be seen that most of the time of the 536 ms call is spent in `SELECT t1.id...` of this SQL call.

At this point, the causes for the exception are revealed. This effectively helps you with the subsequent code optimization. You can also return to the **Key Events** page to view other slow calls in the list and solve them one by one.

6 Reference

6.1 Supported components and frameworks

This topic lists out third-party Java and PHP application components and frameworks supported by Application Real-Time Monitoring Service (ARMS). If the components or frameworks used by the application to be monitored are not supported, configure a universal Filter interceptor to collect monitoring data.

Supported Java components and frameworks

Components	JDK 1.7	JDK 1.8
Dubbo	2.5.X+	2.5.X+
Google HTTP Client	1.10.X+	1.10.X+
GRPC-Java	1.15+	1.15+
HttpClient 3	3.X+	3.X+
HttpClient 4	4.X+	4.X+
JDK HTTP	1.7.X+	1.7.X+
Jetty	8.X+	8.X+
Lettuce	4.0+	4.0+
MariaDB	1.3+	1.3+
Memcached	2.8+	2.8+
MongoDB	3.7+	3.7+
MyBatis	3.X+	3.X+
MySQL JDBC	5.0.X+	5.0.X+
OKHttp	2.X+	2.X+
Oracle JDBC	10.2.X+	10.2.X+
PostgreSQL JDBC	9.4+	9.4+
Redis	2.X+	2.X+
Resin	3.0+	3.0+
Spring	4.X+	4.X+
Spring Boot	1.3.X+	1.3.X+

SQL Server JDBC	6.4+	6.4+
Thrift	0.8+	0.8+
Tomcat	7.X+	7.X+
Undertow	1.3X+	1.3X+
WebLogic	12.X+	12.X+

If the components or frameworks of an application are not supported, configure a universal Filter interceptor to capture data. Follow these steps:

1. Add the Arms-sdk-1.7.1.jar to the pom.xml file.

```
<dependency>
<groupId>com.alibaba.arms.apm</groupId>
<artifactId>arms-sdk</artifactId>
<version>1.7.1</version>
</dependency>
```

Note:

If you cannot retrieve the pom.xml, download Arms-sdk-1.7.1.jar.

2. Configure the ARMS Filter interceptor in web.xml.

```
<filter>
 <filter-name>EagleEyeFilter</filter-name>
 <filter-class>com.alibaba.arms.filter.EagleEyeFilter</filter-class>
</filter>
<filter-mapping>
 <filter-name>EagleEyeFilter</filter-name>
 <url-pattern>/*</url-pattern>
</filter-mapping>
```

3. Install the ARMS agent for Java applications.
4. Restart the application to make the configuration take effect.

Supported PHP components and frameworks

Item	Version requirement
PHP version	PHP 5.4, 5.5, 5.6, 7.0, 7.1, and 7.2 NTS
Nginx	PHP-FPM
Apache	Apache2handler
Runtime environment of the ARMS agent for PHP	Glibc-2.12 and later versions

6.2 Versions of the ARMS agent

This topic describes the version history of the ARMS agents for Java and for PHP.

Versions of the ARMS agent for Java applications

Version	Date	Revision
2.5.8	August 2, 2019	• Supported the dual-state alert function. This function sets alert rules for metrics with two states only: yes or no. • Supported Chinese DM database plug-ins.
2.5.7.2	July 30, 2019	• Supported JVM metaspace metrics. • Supported the definition of which HTTP status codes to ignore. By default, status codes greater than 400 are counted as errors. You can also customize a threshold greater than 400.
2.5.7	July 11, 2019	Upgraded the Fastjson version that is relied on to eliminate security vulnerabilities.
2.5.6.1	June 28, 2019	• Supported Dubbo and MariaDB plug-ins. • Supported capturing the variable values bound to **PrepareStatement**, which takes effect without restarting the application. • Optimized memory and fixed several bugs. • Removed Log4j log dependency to avoid conflicts.
2.5.6	June 7, 2019	• Supported quantile statistics. • Optimized functions and fixed several bugs.
2.5.5	June 3, 2019	• Supported HSF and HTTP calls. • Optimized functions and fixed several bugs.
2.5.3	March 15, 2019	• Supported reporting thread metrics when running an application. • Supported the Spring-Data-Redis plug-in. • Supported the Druid database connection pool plug-in.

Version	Date	Revision
2.5.2	February 21, 2019	• Introduced the collection of the number of file handles. • Supported reporting garbage collection (GC) time and instantaneous numbers of times. • Supported customizing the maximum length of request parameters.
2.5.1	January 14, 2019	• Supported trace compression. • Supported creating an application monitoring task without using the console. • Optimized functions and fixed several bugs.
2.5.0	December 28, 2018	• Supported one-click access without restarting the application. • Improved host monitoring and supported the Windows system. • Supported Spring-webflux. • Optimized functions and fixed several bugs.
2.4.6	October 26, 2018	• Supported the gRPC Remote Procedure Call (gRPC) , Thrift, and XMemcached plug-ins. • Supported topology views of API calls. • Supported topology views that cover the front end and back end.
2.4.5	September 17, 2018	• Supported the Lettuce plug-in (JRE 1.8 +). • Supported the MongoDB plug-in. • Supported capturing exception details.
2.4.4	August 6, 2018	• Supported reporting data of application thread profiling. • Supported Memcached caching. • Supported custom configuration of exception filtering.

Version	Date	Revision
2.4.3.1	June 29, 2018	• Supported WebLogic servers. • Supported Undertow servers. • Optimized memory usage by the ARMS agent. • Shortened the time for starting and loading the ARMS agent. • Eliminated the occasional problem that JVM monitoring metrics and host monitoring metrics cannot be reported.
2.4.3	May 18, 2018	• Supported capturing the monitoring metrics of Message Queue for RocketMQ (RocketMQ). • Supported customizing the monitoring methods. • Prevented frequent log output in throttling scenarios. • Supported customizing the maximum length of the method stack. • Optimized the sampling function by excluding abnormal traces.
2.4.2	April 19, 2018	• Supported reading custom configuration details. • Supported retrieving trace information through SDKs. • Supported capturing JVM metrics such as the thread and GC times, and time consumption. • Supported monitoring HSF calls. • Supported capturing host monitoring metrics such as CPU, memory, network, and disk. • Eliminated the problem that the **./shutdown.sh** process may get stuck in the Tomcat environment.
2.4.1	March 24, 2018	• Supported JVM monitoring, such as reporting heap memory and non-heap memory. • Supported PlayFrameWork 1.4.4. • Supported custom configuration of parameters, such as the sampling rate, agent switch, log level, and threshold.

Version	Date	Revision
2.4.0	February 14, 2018	• Supported the PostgreSQL database. • Supported connecting ARMS with Alibaba Cloud Elastic Compute Service (ECS) instances in each region over the internal network. • Ended the beta phase of ARMS application monitoring.

Versions of the ARMS agent for PHP applications

Version	Date	Revision
2.0.2	July 31, 2019	• Fixed issues in the transmission of network modules under high concurrency. • Redesigned the logics for transmission and re-connection. • Reduced memory usage. • Fixed several bugs.
2.0.1	July 23, 2019	• Used the Arms-agent process as the daemon. • Supported the Redis and MongoDB plug-ins. • Fixed several bugs.
2.0.0	July 5, 2019	• Used a new and much more reliable network model. • Optimized the display of exception information. • Optimized the memory usage. • Fixed several bugs.
1.1.0	April 30, 2019	• Supported the GCC 4.4.7 environment. • Introduced the TCP connection heartbeat. • Fixed host monitoring bugs. • Supported showing the ARMS version by running the **php -m** command. • Used DNS to resolve the domain names of collectors. • Optimized functions and fixed several bugs. **Notice:** We recommend that you upgrade the ARMS agent from 1.x.x to 2.x.x.

Version	Date	Revision
1.0.1	March 15, 2019	• Supported Laravel 5.x. • Supported the PDO plug-in. • Removed unnecessary logs. • Fixed several bugs. **Notice:** We recommend that you upgrade the ARMS agent from 1.x.x to 2.x.x.

6.3 Key statistical metrics

This topic explains the meanings of key statistical metrics on each page in application monitoring of Application Real-Time Monitoring Service (ARMS).

Terms

Here are the basic terms for this topic:

- Apdex

  Application Performance Index (Apdex) is an internationally accepted standard for evaluating application performance. According to Apdex, the user experience of an application is defined at three levels:

  - - Satisfied (0 to T)
  - - Tolerating (T to 4T)
  - - Frustrated (greater than 4T)

  Image source: apdex.org

  The following formula is used to calculate the Apdex score::

  ```
  Apdex score = (Satisfied samples + Tolerating samples/2)/Total samples
  ```

  ARMS uses the average response time of an application in calculation, and defines T at 500 ms.

- Instances

  An instance is a machine where the monitored application is deployed. The granularity of an instance is JVM. In the following figure, "a3" is an application, and each row under a3 is a machine where a3 is deployed. Each machine is an instance.

Related statistics pages

- The Application Monitoring > Applications page

  In the left-side navigation pane of the console, choose **Application Monitoring** > **Applications**. You can view the Apdex satisfaction trend curve of each application.

  Figure 6-1: Apdex satisfaction trend

- The Application Overview page

  On the Applications page, click the name of the target application to access the **Application Overview** page. Choose the corresponding menus in the left-side navigation pane. Then you can view statistics for other dimensions on other pages.

  - **Overview Analysis**
    - Services provided by the application: requests and average response time
    - Services the application depends on: requests and average response time
    - System information: CPU, memory, and load
    - Statistical analysis: slow call analysis, average response time, exception type, and times of occurrence
  - **Topology** tab
    - Application topology
    - Instance health status: Green indicates **Normal**, yellow indicates **Alerting**, and red indicates **Severe**.
    - Type of call:

Type of call	Description	Remarks
HTTP entry point	The client uses the HTTP protocol to call the entry point of the application	Service entry call

Type of call	Description	Remarks
Dubbo call	Calls generated by Dubbo consumers	Service entry call
HSF calls	Calls generated by HSF consumers	Service entry call
HTTP call	HTTP call initiated by this application to other services	Inter-service calls
HSF provision	Calls generated by HSF producers	Inter-service calls
Dubbo provision	Calls generated by Dubbo producers	Inter-service calls
MySQL call	Calls initiated for operating on MySQL	Database call
Oracle call	Calls initiated for operating on Oracle	Database call
Redis call	Calls initiated for operating on Redis	Database call

- Application Details page

  This page shows details of calling the current application. Click different tabs to view the detailed analysis of different dimensions, such as instance response time, request count, error count, instance overview, SQL analysis, exception analysis, and interface snapshot.

- API call page

  This page shows the statistical information of APIs opened by the current application. Click different tabs to view the detailed analysis of different dimensions, such as instance response time, request count, error count, instance overview, SQL analysis, exception analysis, and interface snapshot.

- Database call page

  This page shows application-related database call information. Click different tabs to view the detailed analysis of instance response time, request count, error count, instance overview, SQL analysis, or exception analysis.

Key statistical metrics on related tabs

- **Response Time** tab

Reported field	Description
Response time	The average response time of applications and instance calls, or the average execution response time of database operations

- **Request Count** tab

Reported field	Description
Request count	The number of requests for calling applications or instances , or the number of times of performing database operations

- **Error Count** tab

Reported field	Description
Error count	The number of incorrect application or instance calls, or the number of abnormal executions in database operations

- **Overview** tab

Reported field	Description
Request count	The number of requests for calling applications or instances , or the number of times of performing database operations
Response time	The average response time of applications and instance calls, or the average execution response time of database operations
Error rate	(The number of abnormal application or instance calls, or the number of abnormal executions in database operations)/Number of requests
Performance overview	The column chart and the left side of Y axis show the number of requests. The line chart and the right side of Y axis show the response time.

- **SQL Analysis** tab

Reported field	Description
SQL call statistics	The column chart and the left side of Y axis show the number of database requests. The line chart and the right side of Y axis show the database response time.

Reported field	Description
Average time consumption	The average time consumed for this database call
Number of database calls	The number of times this type of database has been called

- **Exception Analysis** tab

Reported field	Description
Exception statistics	The column chart shows the number of exceptions of the application, instance, and database.
Exception type	Type of collected errors
Exception details	Error details
Average time consumption	The average time consumed by this incorrect call
Error count	The number of times this exception type has occurred

- **Interface Snapshot** tab

Reported field	Description
Time consumption	Consumed time for calling the API of an application or instance
Status	The return status of the API call of an application or instance . Green indicates a success response, and red indicates an exception.
TraceId	The index ID of an application or instance call. You can click it to jump to the details page of this trace.

6.4 ARMS SDKs

You can use Application Real-Time Monitoring Service (ARMS) SDKs to dynamically obtain the TraceId and its properties in the service code.

Prerequisites

- An application monitoring job is created in the ARMS console, and the ARMS agent has been installed and started in the Java program for monitoring. For more information about how to install the ARMS agent for Java applications, see Install the ARMS agent for Java applications.

- Arms-sdk-1.7.1.jar is introduced to the program.

```
 <dependency>
<groupId>com.alibaba.arms.apm</groupId>
<artifactId>arms-sdk</artifactId>
<version>1.7.1</version>
</dependency>
```

Note:

If you cannot retrieve the pom.xml file, download arms-sdk-1.7.1.jar.

Retrieve TraceId and RpcId

After the preceding prerequisites are met, run the following code to retrieve the TraceId and RpcId:

```
Span span = Tracer.builder().getSpan();
String traceId = span.getTraceId();
String rpcId = span.getRpcId();
```

Pass through the service baggage

To pass through a service baggage, write procedures for adding and retrieving the service baggage into the code.

1. Add a service baggage to the service code.

```
Map<String, String> baggage = new HashMap<String, String>();
baggage.put("key-01", "value-01");
baggage.put("key-02", "value-02");
baggage.put("key-03", "value-03");
Span span = Tracer.builder().getSpan();
span.withBaggage(baggage);
```

2. Retrieve the service baggage from the service code.

```
Span span = Tracer.builder().getSpan();
Map<String, String> baggage = span.baggageItems();
```

7 FAQ about updating the ARMS agent for Java applications

This topic describes how to update the Application Real-Time Monitoring Service (ARMS) agent for monitoring Java applications in Enterprise Distributed Application Service (EDAS) and Container Service, and on other platforms.

How do I update the ARMS agent for applications in EDAS?

To update the ARMS agent for Java applications in EDAS, re-deploy the applications.

How do I update the ARMS agent for applications in Container Service?

To update the ARMS agent for applications in Container Service, restart the pods of the applications.

How do I update the ARMS agent for applications on other platforms?

To update the ARMS agent for other applications than those in EDAS and Container Service, uninstall the ARMS agent and then install it again.

To uninstall the ARMS agent, perform the following steps as needed:

- Uninstall the ARMS agent for Java applications on Elastic Compute Service (ECS) instances
- Uninstall the ARMS agent for Java applications on other platforms

  Delete the parameters related to **AppName** and **LicenseKey** in the configuration file or startup script. For more information, see#unique_41/unique_41_Connect_42_uninstall .

8 Application monitoring FAQ

This topic summarizes the frequently asked questions about using ARMS to application monitoring.

Overview

- FAQ about manually installing the security center Agent for Java applications
  - Is the ARMS Agent compatible with agents of other APM products, such as Pinpoint?
  - What can I do if a OutOfMemoryError error occurs when an application starts after the EDAs Agent is installed?
  - How do I test the network connectivity?
  - How can I check whether ARMS Agent is successfully installed?
  - Why is there no monitoring data in the console after ARMS Agent is installed?
  - What do I do if the ARMS Agent is successfully installed but the IP address is incorrectly displayed?
  - What are the common troubleshooting methods for ARMS Agent log (path: ArmsAgent/log) errors?
  - How do I support single-instance multi-instances?
- FAQ about installing the arms Agent for Java applications
  - How can I resolve getcwd errors when running the import script?
  - Where can I view the logs after I install the security center Agent with one click?
- FAQ about installing the security center Agent for Java applications on ECS instances
  - How do I handle an Agent installation failure?
  - What do I do if the ECS instance process information is incorrect after the Agent is installed?
  - What can I do if the process in an ECS instance fails to start application monitoring?
- FAQ about installing the security center Agent for Alibaba Cloud Container Service for Kubernetes Java applications
  - Why is there no data on the Java application in a Kubernetes cluster application monitoring container service is installed with the Agent?

- FAQ about installing the EDAs Agent for Java applications in open-source Kubernetes environments
  - What do I do if an application fails to start?
  - How do I view the Agent log?
- FAQs about changing the name of a Java application without reinstalling the Agent
  - How do I change the application name for a normal Java application that installs Agent in a common way?
  - How do I change the name of a Java application deployed in a Kubernetes cluster of container service?
  - How can I change the name of a Java application deployed in EDAS?
- FAQ about uninstalling the security center Agent
  - How do I uninstall the Agent that is installed in a common way?
  - How do I uninstall the Agent installed in quick mode?
  - How do I uninstall the arms Agent from the Java application on an ECS instance?
  - How do I uninstall the Agent of the Java application in a container service Kubernetes cluster?
  - How do I uninstall the Agent from the Java application in an open-source Kubernetes environment?
  - How do I uninstall the Agent of the Java application in Docker?
  - How do I uninstall the Agent?
  - How do I uninstall the Agent of the PHP application in a container service Kubernetes cluster?
- Other problems
  - What should I do if the data of my application with the OpenFeign component is incomplete in ARMS?

Is the ARMS Agent compatible with agents of other APM products, such as Pinpoint?

The ARMS Agent is incompatible with other APM agents. APM mostly implements bytecode instrumentation based on ASM framework. Installing two agents at the same time is equivalent to instrumentation your code twice, while the instrumentation code implementation for different merchants is different. Code conflicts may cause serious performance problems. We strongly recommend that you do not install multiple APM agents at the same time.

[Back to the top]

What can I do if a OutOfMemoryError error occurs when an application starts after the EDAs Agent is installed?

To properly increase the JVM value, append the heap memory configuration items to the Java startup command. Configuration items following example indicates that the initial heap memory size (Xms) is 512 MB and the maximum heap memory size (Xmx) is 2 GB.

Note:
Please adjust it according to the actual situation. In other environments such as Tomcat, add the check result in the **JAVA_OPTS** add this parameter to.

```
-Xms512M
-Xmx2048M
```

If `OutOfMemoryError: PermGen space` error, please add the following configuration.

```
-XX:PermSize=256M
-XX:MaxPermSize=512M
```

[Back to the top]

How do I test the network connectivity?

Before the ARMS Agent runs, make sure that ports 8883, 8443, and 8442 are properly connected. Can be used **Telnet** command to check whether the target host can connect to the ARMS server. For example, to test the connectivity to the China (Shenzhen) region, log on to the ECS instance where the application is deployed, and run the following command:

```
telnet arms-dc-sz.aliyuncs.com 8883
telnet arms-dc-sz.aliyuncs.com 8443
telnet arms-dc-sz.aliyuncs.com 8442
```

Note:
Replace the server address based on the region. Server addresses correspond to classic network, public networks, and VPCs.

Table 8-1: ARMS application monitoring server addresses in different regions

Region	Classic network and Internet	VPC
China (Hangzhou)	arms-dc-hz.aliyuncs.com	arms-dc-hz-internal.aliyuncs.com

Region	Classic network and Internet	VPC
China (Beijing)	arms-dc-bj.aliyuncs.com	arms-dc-bj-internal.aliyuncs.com
China (Shanghai)	arms-dc-sh.aliyuncs.com	arms-dc-sh-internal.aliyuncs.com
China (Qingdao)	arms-dc-qd.aliyuncs.com	arms-dc-qd-internal.aliyuncs.com
China (Shenzhen)	arms-dc-sz.aliyuncs.com	arms-dc-sz-internal.aliyuncs.com
China (Zhangjiakou-Beijing Winter Olympics)	arms-dc-zb.aliyuncs.com	arms-dc-zb-internal.aliyuncs.com
China (Hong Kong)	arms-dc-hk.aliyuncs.com	arms-dc-hk-internal.aliyuncs.com
Singapore (Singapore)	arms-dc-sg.aliyuncs.com	arms-dc-sg-internal.aliyuncs.com
Regions for Alibaba GovCloud	arms-dc-gov.aliyuncs.com	arms-dc-gov-internal.aliyuncs.com
Finance Cloud of China (Hangzhou)	arms-dc-hz-finance.aliyuncs.com	arms-dc-hz-finance-internal.aliyuncs.com

[Back to the top]

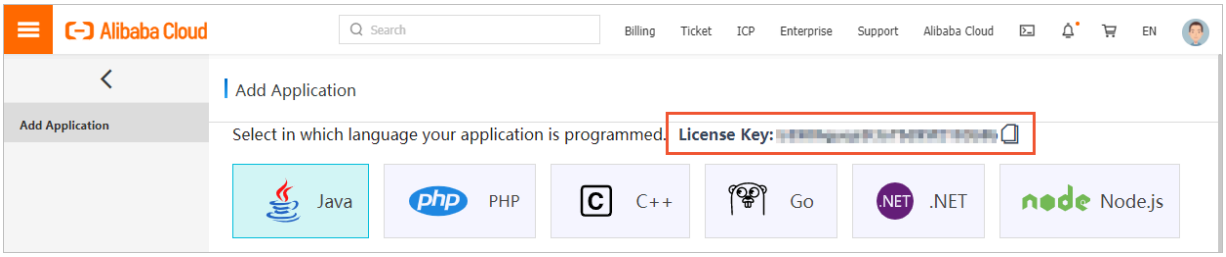
How can I check whether ARMS Agent is successfully installed?

Use **PS** command to check whether ARMS Agent is successfully installed in the command line startup parameter.

```
ps -ef | grep 'arms-bootstrap'
```

Upon successful installation, the result is shown in the following figure.

At the command line, **Darms.licenseKey** the value of must be the same as that of ARMS **application Access** the License Key displayed on the page is the same.



[Back to the top]

Why is there no monitoring data in the console after ARMS Agent is installed?

1. If the send agent metrics. no metrics., check whether your application has continuous external access requests, including HTTP requests, HSF requests, and Dubbo requests, and whether the development framework is supported by ARMS Agent. For more information about third-party components and frameworks supported by ARMS application monitoring, see #unique_43.
2. Check whether the selected time range for query is correct. Select the last 15 minutes as the query time range and check whether monitoring data is available.
3. If it is through -jar the script is started through the command line. Check the command line settings to make sure **-javaagent** the parameter is in the -jar before. The following is a correct example.

   ```
   java -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey=xxx -Darms.appName=xxx -jar demoApp.jar
   ```

4. If ArmsAgent/log/ appears in the log under "LicenseKey is invalid", Exception, check if the application and the Agent are in the same region.
5. If the application has been started up ArmsAgent if the log sub-directory does not exist in the directory, the arms-bootstrap-1.7.0-SNAPSHOT.jar has not been successfully loaded. Check whether the permissions of the ArmsAgent installation directory are correct.
6. If the following error is reported when your application is started, check arms-bootstrap-1.7.0-SNAPSHOT.jar software package and the corresponding permissions.

   ```
   Error opening zip file or JAR manifest missing: /root/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar
   Error occurred during initialization of VM
   agent library failed to init: instrument
   ```

7. If there is still no monitoring data, copy the Java Agent log (path: ArmsAgent/log) as a compressed file and contact the DingTalk service account. arms160804 solve the problem.

8. Check the JDK Version. If the JDK Version is 1.8.0 _25 or 1.8.0 _31, you may fail to install the Agent. We recommend that you upgrade the JDK or contact DingTalk arms160804.

[Back to the top]

What do I do if the ARMS Agent is successfully installed but the IP address is incorrectly displayed?

1. First through **ifconfig -a** check the network configuration of the current machine. If the machine uses network interface controller, the IP addresses collected by the arms Agent may not be as expected, which is related to the network configuration.
2. Choose one of the following ways to solve the problem.
   - Configure the JVM -DEAGLEEYE.LOCAL.IP=10.XX.XX.XX parameter.

     Note:

     Will 10.XX.XX.XX replace it with the actual IP address.
   - Configure the Agent to obtain the JVM -DNETWORK.INTERFACE=eth0 parameter, where eth0 is network interface controller name.

[Back to the top]

What are the common troubleshooting methods for ARMS Agent log (path: ArmsAgent/log) errors?

LicenseKey is invalid error:

1. Ensure that you have successfully created the application in ARMS and that you entered the correct LicenseKey when installing the Agent.
2. ARMS is a multi-Region environment. Therefore, you also need to check whether the download link of the ARMS Agent is in the same Region as your application.

[Back to the top]

How do I support single-instance multi-instances?

To deploy multiple instances of the same application on the same server, you can use **-Darms.agentId**(Logical number, such as 001 or 002) parameter to distinguish the Accessed JVM process, for example:

```
java -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey=<LicenseKey> -Darms.appName=<AppName> -Darms.agentId=001 -jar demoApp.jar
```

[Back to the top]

How can I resolve getcwd errors when running the import script?

If the following error message is returned when you run the script for one-click Java application access:

```
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory Error occurred during initialization of VM java.lang.Error: Properties init: Could not determine current working directory. at java.lang.System.initProperties( Native Method) at java.lang.System.initializeSystemClass(System.java:1119)
```

This may be because the current directory was deleted by mistake when the script was executed. To solve this problem, run the `CD` and then run the script again.

[Back to the top]

Where can I view the logs after I install the security center Agent with one click?

The default directory for logs is /root/.arms/supervisor/logs/ if there are no logs in this directory, run `ps -ef |grep arms` view the directory where the logs are located.

[Back to the top]

How do I handle an Agent installation failure?

1. Make sure that your ECS instance can access the Internet and the Agent download link in the region where your instance is located can be accessed.

```
# China (Hangzhou)
http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/install.sh
# China (Shanghai)
http://arms-apm-shanghai.oss-cn-shanghai.aliyuncs.com/install.sh
# China (Qingdao)
http://arms-apm-qingdao.oss-cn-qingdao.aliyuncs.com/install.sh
# China (Beijing)
http://arms-apm-beijing.oss-cn-beijing.aliyuncs.com/install.sh
# China (Shenzhen)
http://arms-apm-shenzhen.oss-cn-shenzhen.aliyuncs.com/install.sh
# Singapore
http://arms-apm-ap-southeast.oss-ap-southeast-1.aliyuncs.com/cloud_ap-southeast-1/install.sh
```

2. Make sure that your ECS instance can access the console.

```
# China
https://arms.console.aliyun.com/
#Singapore
```

```
https://arms-ap-southeast-1.console.aliyun.com
```

3. Login ECS console, and complete the following checks.

   a. In the left-side navigation pane, choose **maintenance and monitoring** > **send remote commands (Cloud Assistant)**.

   b. In **cloud Assistant** select **command name**, input InstallJavaAgent command and press Enter.

   > **Note:**
   > If the search results do not exist, contact the ARMS DingTalk service account arms160804.

   c. In **cloud Assistant** page's **execution Records** click the tab and enter InstallJavaAgent the ID of the corresponding command. Click the record ID **operation** column's **view Results**, you can view InstallJavaAgent indicates whether the command was executed successfully. If the execution fails, troubleshoot based on the detailed execution results. If the ECS disk is full or Agent is not installed, you can clear the disk or install Agent to solve the problem. If the problem persists, submit the detailed execution results to ARMS DingTalk account. arms160804.

[Back to the top]

What do I do if the ECS instance process information is incorrect after the Agent is installed?

If no ECS instance process information is displayed after you successfully install the Agent on the ECS or the ECS instance process information is inaccurate, click on the left of the ECS instance **-**and click **+**refresh data. If the problem cannot be solved, contact the ARMS DingTalk service account arms160804.

[Back to the top]

What can I do if the process in an ECS instance fails to start application monitoring?

Check the IP addresses of the /root/.arms/supervisor/logs/arms-supervisor.log whether the log contains Error log level information, and according to this information, troubleshoot. If the problem cannot be solved, contact the ARMS DingTalk service account arms160804.

[Back to the top]

Why is there no data on the Java application in a Kubernetes cluster application monitoring container service is installed with the Agent?

1. In the left-side navigation pane, choose **application** > **container Group**, in **container Group (Pod)** page, from **cluster** list and **namespace** in the left-side navigation pane, select the cluster and namespace of your application.
2. In The Pod list, locate the Pod of the target application. **Edit**, in **edit YAML** dialog box to check whether initContainers exists in the YAML file.

   - If it does not exist, the arms-init-container has not been injected. Run the following command: step3.
   - If yes, arms-init-container has been injected. Run the following command: step6.
3. In **container Group (Pod)** page, from **cluster** list and **namespace** in the list, select the cluster and **arms-pilot** namespace and check whether any Pod list with the prefix **arms-pilot** pod.

   - If yes, perform the step4.
   - Otherwise, install arms-pilot in the application market. For more information, see install ARMS application monitoring.
4. In the left-side navigation pane, choose **application** > **stateless**, in **Deployment** page from **cluster** list and **namespace** select the cluster and namespace where your application is located from the drop-down list. **Operation** column's **more** > **view Yaml files**, in **edit YAML** dialog box to check whether the following Annotations exist in the YAML file:

   ```
   annotations:
     armsPilotAutoEnable: 'on'
     armsPilotCreateAppName: <your-deployment-name>
   ```

   - If yes, perform the step5.
   - If it does not exist, it is in the **edit YAML** in the dialog box spec > template > metadata add the Annotations above in the hierarchy, and add `<your-deployment-name>` with the name of your application and then click **update**.
5. In the left-side navigation pane, choose **application** > **container Group**, in **container Group (Pod)** page, from **cluster** list and **namespace** in the list, select the cluster and

arms-pilot namespace. Click **log** check whether the STS error message is displayed in the Pod log of arms-pilot. `"Message":"STS error"`.

- If an STS error is reported, you must authorize the cluster where the application is located and restart the Pod where the application is located. For more information, see authorize Alibaba Cloud Container Service for Kubernetes.
- If no STS error is reported, contact ARMS DingTalk service account: arms160804.

6. In **edit YAML** dialog box to check whether the following exists in the YAML file: **javaagent** parameter.

```
-javaagent:/home/admin/.opt/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar
```

- If it exists, **container Group (Pod)** find the Pod where the application is located. **Terminal** enter **command line** page and run the following command to check whether there is a log file suffixed with. log, and then contact ARMS DingTalk service account: arms160804.

```
cd /home/admin/.opt/ArmsAgent/logs
```

- If it does not exist, contact ARMS DingTalk service account: arms160804.

[Back to the top]

What do I do if an application fails to start?

Run the following command to view the arms-pilot-system of the log for troubleshooting.

```
kubectl logs -f {arms-pilot-arms-pilot-XXX} -n arms-pilot-system
```

[Back to the top]

How do I view the Agent log?

View logs on the Worker of the Kubernetes cluster. /home/admin/.opt/ArmsAgent/logs/xxxx.log.

[Back to the top]

How do I change the application name for a normal Java application that installs Agent in a common way?

Common Java applications are Java applications other than those deployed on ECS instances of Alibaba Cloud. If you installed the Agent in a common way, the Agent directory is the location you customized.

Check the Agent Version in the Version file in the Agent directory. For example, 2.5.8_cf020486_20190816150025 the Agent version is 2.5.8, and the release date of the Agent is August 16, 2019.

- If your Agent version is earlier than 2.5.8.1, uninstall the Agent and then reinstall it. Use a new application name when you reinstall the Agent.
  - To uninstall the Agent that is manually installed, see how do I uninstall the Agent installed in a common way.
  - To uninstall the EDAs Agent that is automatically installed by using the script, see how to uninstall the Agent installed in quick mode.
  - For more information about how to uninstall the ECS Agent, see how to uninstall the Agent from the Java application in an ECS instance.
- If your Agent version is 2.5.8.1 or later, you can change the name of the Java application without reinstalling the Agent. For more information, see the following procedure:

> **Note:**
> Agents downloaded after August 20, 2019 support this function by default.

1. Download and extract the Supervisor using one of the following methods.

> **Note:**
> Replace the region in the download URL with the region where your application is located.

   - Public network:

```
wget http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/ArmsSupervisor.zip -O ArmsSupervisor.zip
unzip ArmsSupervisor.zip
```

   - VPC address (used when the public endpoint cannot be downloaded)

```
wget http://arms-apm-hangzhou.oss-cn-hangzhou-internal.aliyuncs.com/ArmsSupervisor.zip -O ArmsSupervisor.zip
unzip ArmsSupervisor.zip
```

2. Run the following command to change the application name:

```
cd ArmsSupervisor
```

```
./attach.sh </path/to/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar> <PID> <
NewAppName> <LicenseKey>
```

- </path/to/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar>: indicates the path of a file with the same name under the Agent.
- <PID>: Target process ID, which can be used jps/ps command.
- <NewAppName>: new application name.
- <LicenseKey>:ARMS LicenseKey of the application in the application monitoring, which can be obtained from the console.

If the standard output log of the application displays the following information, the application name is successfully changed:

[Back to the top]

How do I change the application name for a normal Java application that installs Agent in a quick way?

If you installed the Agent quickly, the Agent directory is ~/.arms/supervisor/agent. Note that the account you use must be the same as the application account.

Change your application name as follows:

1. Log on to the machine where the application is located and run the following command:

```
cd ~/.arms/supervisor
./cli.sh <LicenseKey> <NewAppName>
```

- <LicenseKey>:ARMS LicenseKey of the application in the application monitoring, which can be obtained from the console.
- <NewAppName>: new application name.

2. Select the correct process from all Java processes listed in the program.

Note:

If only one process is available, this process is selected by default.

3. Open ~/.arms/attach.info file, modify the old application name to the new application name and save the file.

Warning:

When modifying a file, do not add extra spaces. Otherwise, the modification will fail due to incompatibility with the new application name modified in the preceding steps.

Wait a moment after your application name is changed. No monitoring data will be reported using the old application name, and monitoring data will be reported using the new application name.

How do I change the name of a Java application deployed on an ECS instance?

If your Java application is deployed on an ECS instance, the Agent directory is /.arms/agent.

Change your application name as follows:

1. Log on to the ECS instance where your application is located and run the following command with the root account:

```
su <USER> -c "./attach.sh /.arms/agent/arms-bootstrap-1.7.0-SNAPSHOT.jar <PID> <NewAppName> <LicenseKey>"
```

   - <PID>: Target process ID, which can be used jps/ps command.
   - <NewAppName>: new application name.
   - <LicenseKey>:ARMS LicenseKey of the application in the application monitoring, which can be obtained from the console.

2. Open ~/.arms/attach.info file, modify the old application name to the new application name and save the file.

   Warning:
   When modifying a file, do not add extra spaces. Otherwise, the modification will fail due to incompatibility with the new application name modified in the preceding steps.

If the standard output log of the application displays the following information, the application name is successfully changed:

How do I change the name of a Java application deployed in a Kubernetes cluster of container service?

Check the Agent Version in the Version file in the Agent directory. For example, 2.5.8_cf020486_20190816150025 the Agent version is 2.5.8, and the release date of the Agent is August 16, 2019.

- If your Agent version is earlier than 2.5.8.1, uninstall the Agent and then reinstall it. Use a new application name when you reinstall the Agent.
  - To uninstall the Agent that is manually installed, see how do I uninstall the Agent installed in a common way.
  - To uninstall the EDAs Agent that is automatically installed by using the script, see how to uninstall the Agent installed in quick mode.
  - For more information about how to uninstall the ECS Agent, see how to uninstall the Agent from the Java application in an ECS instance.
- If your Agent version is 2.5.8.1 or later, you can change the name of the Java application without reinstalling the Agent. For more information, see the following procedure:

  Note:

  Agents downloaded after August 20, 2019 support this function by default.

  Modify the values of the **armsPilotCreateAppName** parameter and restart the Pod.

  Wait a moment after your application name is changed. No monitoring data will be reported using the old application name, and monitoring data will be reported using the new application name.

[Back to the top]

How can I change the name of a Java application deployed in EDAS?

Currently, you cannot change the name of a Java application deployed in Enterprise Distributed Application Service (EDAS).

[Back to the top]

How do I uninstall the Agent that is installed in a common way?

In the common way, you need to manually install the security center Agent for your Java application. For more information, see #unique_41. Perform the following steps to uninstall the Agent that is installed in the common mode:

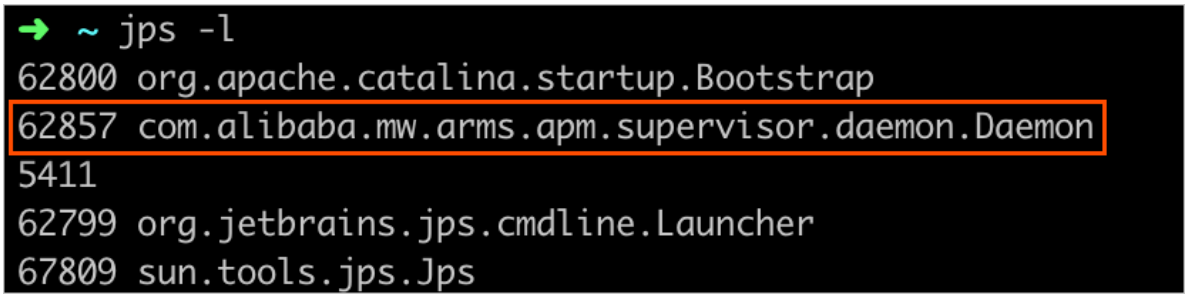
1. When you no longer need ARMS to monitor your Java applications, delete the preceding installation document. step 8 the added **AppName**, **LicenseKey** all relevant parameters.
2. Restart the Java application.

[Back to the top]

How do I uninstall the Agent installed in quick mode?

In quick mode, you need to install the security center Agent for your Java application by using a script with one click. For more information, see #unique_45. Perform the following steps to uninstall the quick mode Agent:

1. When you no longer need ARMS to monitor your Java application, run the `jps -l` command to view all processes, and in the execution results, find `com.alibaba.mw.arms.apm.supervisor.daemon.Daemon` the process ID.
   In this example, the `com.alibaba.mw.arms.apm.supervisor.daemon.Daemon` the process number is 62857.

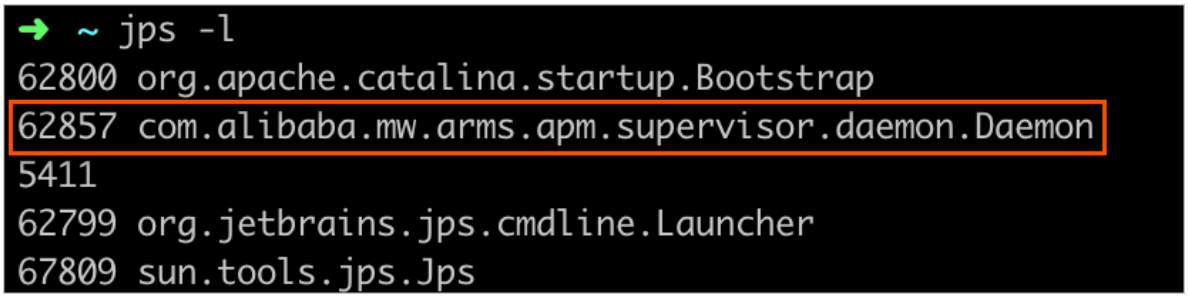


2. Run commands `kill -9 <process id>`.
   For example: `kill -9 62857`.
3. Execution `rm -rf /.arms /root/.arms`.
4. Restart your application.

[Back to the top]

How do I uninstall the arms Agent from the Java application on an ECS instance?

1. When you no longer need ARMS to monitor your Java application, run the `jps -l` command to view all processes, and in the execution results, find `com.alibaba.mw.arms.apm.supervisor.daemon.Daemon` the process ID.
   In this example, the `com.alibaba.mw.arms.apm.supervisor.daemon.Daemon` the process number is 62857.



2. Run commands kill -9 <process id>.
   For example: kill -9 62857.
3. Execution rm -rf /.arms /root/.arms.
4. Restart your application.

[Back to the top]

How do I uninstall the Agent of the Java application in a container service Kubernetes cluster?

When you no longer need ARMS to monitor the Java application in a Alibaba Cloud Container Service for Kubernetes cluster, you can uninstall the ARMS Agent as follows:

1. In the left-side navigation pane, choose **Applications** > **Releases**.
2. In **publish** page's **Helm** on the page, from **cluster** select the cluster from which you want to uninstall ARMS Agent.
3. Select the release name corresponding to the ARMS Agent **arms-pilot**, click **operation** column's **delete**.
4. In **delete an application** dialog box, click **confirm**.
5. Restart your business pod.

[Back to the top]

How do I uninstall the Agent from the Java application in an open-source Kubernetes environment?

1. When you no longer need to monitor the Java application in the open-source Kubernetes environment, you can run the following command to uninstall arms-pilot:

   ```
   helm del --purge arms-pilot
   ```

2. Restart your business pod.

[Back to the top]

How do I uninstall the Agent of the Java application in Docker?

1. When you no longer need to monitor the Java application in the Docker cluster, you can delete the step 1 edited Dockerfile the content.
2. Run **docker build** command to build the image.
3. Run **docker run** command to start the image.

[Back to the top]

How do I uninstall the Agent?

When you no longer need ARMS to monitor PHP applications in a Kubernetes cluster, you can uninstall the ARMS Agent as follows:

1. Delete php.ini the following four lines in the file:

```
[arms]
extension=<php_extension_dir>/arms.so
arms.trace_exception=true
arms.config_full_name=/<php-agent-dir>/arms-agent.conf
```

2. Restart the PHP application.

[Back to the top]

How do I uninstall the Agent of the PHP application in a container service Kubernetes cluster?

If you no longer need ARMS to monitor the PHP application in Alibaba Cloud Container Service for Kubernetes cluster, you can uninstall the ARMS Agent as follows:

1. In the left-side navigation pane, choose **Applications** > **Releases**.
2. In **publish** page's **Helm** on the page, from **cluster** select the cluster from which you want to uninstall ARMS Agent.
3. Select the release name corresponding to the ARMS Agent **arms-pilot**, click **operation** column's **delete**.
4. In **delete an application** dialog box, click **confirm**.
5. Restart your business pod.

[Back to the top]

What should I do if the data of my application with the OpenFeign component is incomplete in ARMS?

After your OpenFeign application is connected to ARMS, if the data is incomplete or the data of downstream applications cannot be seen, a possible cause is that the OpenFeign component enables Hystrix using the RxJava asynchronous framework by default. ARMS does not support asynchronous frameworks.

Note:

This topic only applies to scenarios where the ARMS Application Monitoring Java Agent version is earlier than 2.6.0.

You can disable Hystrix and enable the OkHttp request class to solve the following problems:

1. In pom.xml add the following dependencies to the file.

```
<! -- OKHttp supports Feign -->
<dependency>
   <groupId>io.github.openfeign
   <artifactId>feign-okhttp</artifactId>
</dependency>
```

2. Add the following content to the spring cloud configuration file:

```
feign.okhttp.enabled: true
feign.hystrix.enabled: false
```

3. Configure the OkHttp request class.

```
@Configuration
@ConditionalOnClass(Feign.class)
@AutoConfigureBefore(FeignAutoConfiguration.class)
public class FeignClientOkHttpConfiguration {

   @Bean
   public OkHttpClient okHttpClient() {
      return new OkHttpClient.Builder()
            // The connection times out.
            .connectTimeout(20, TimeUnit.SECONDS)
            // The response times out.
            .readTimeout(20, TimeUnit.SECONDS)
            // The write request times out.
            .writeTimeout(20, TimeUnit.SECONDS)
            // Indicates whether to enable automatic reconnection.
            .retryOnConnectionFailure(true)
            // The connection tool.
            .connectionPool(new ConnectionPool())
            .build();
   }
```

[Back to the top]

9 Troubleshooting

9.1 FAQ about installing the ARMS agent for Java applications

This topic lists FAQ about installing the Application Real-Time Monitoring Service (ARMS) agent for Java applications, and provides solutions.

Is the ARMS agent compatible with agents of other Application Performance Management (APM) products, such as Pinpoint?

The ARMS agent is incompatible with agents of other APM products. Most APM technologies use the Automatic Storage Management (ASM) framework for bytecode instrumentation. If you install two agents, your code instrumentation will be implemented twice. Code instrumentation varies with manufacturers and code conflicts may cause serious performance issues. We do not recommend that you install multiple APM agents at the same time.

What can I do if OutOfMemoryError is reported upon startup of an application after the ARMS agent is installed?

In this case, add the corresponding JVM heap memory parameter as appropriate.

If the initial value of heap memory size (Xms) is 512 MB and the maximum value of heap memory size (Xmx) is 2 GB, the configuration items are as follows: Adjust the values according to the actual requirement. In other environments such as Tomcat, add this parameter to JAVA_OPTS of the configuration file.

```
-Xms512M
-Xmx2048M
```

If the `OutOfMemoryError: PermGen space` error is reported, add the following parameters:

```
-XX:PermSize=256M
-XX:MaxPermSize=512M
```

How do I check the network connectivity?

When deploying the ARMS agent, you can run the Telnet command to check if the target host is connected to the ARMS server network.

Table 9-1: Endpoints of ARMS application monitoring servers

Region	Endpoint
China (Hangzhou)	arms-dc-hz.aliyuncs.com
China (Beijing)	arms-dc-bj.aliyuncs.com
China (Shanghai)	arms-dc-sh.aliyuncs.com
China (Qingdao)	arms-dc-qd.aliyuncs.com
China (Shenzhen)	arms-dc-sz.aliyuncs.com

If you created an application in the China (Shenzhen) region of ARMS, you need to test if the environment in China (Shenzhen) is connected to the ARMS server network. The following result indicates that the network is connected:

```
telnet arms-dc-sz.aliyuncs.com 8443Trying 119.23.169.12...Connected to arms-dc-sz.aliyuncs.com.Escape character is '^]'.
```

Note:

You must replace the endpoint according to your region. The port number remains unchanged.

How do I check if the ARMS agent is successfully loaded?

1. Run the following **ps** command to check if the ARMS agent is successfully loaded according to the command line start parameter:

   ```
   ps -ef | grep 'arms-bootstrap'
   ```

   When the ARMS agent is successfully loaded, the result is as follows:

2. The **arms.licenseKey** and **arms.appId** attributes in the command line must be consistent with those displayed on the ARMS application setting page.

What if no data is available after the ARMS agent is loaded?

1. If the log file of the ARMS agent contains "send Agent metrics. no metrics.", verify whether your application has continuous external access requests, including HTTP requests, HSF requests, and Dubbo requests and whether the development framework is supported by the ARMS agent. For more information about third-party components and frameworks supported by the ARMS agent, see#unique_43 .

2. Check whether the selected time range for query is correct. Select the past 15 minutes as the query time range and check whether monitoring data is available.
3. If you start the ARMS agent by running the -jar command, check the setting of the command, and make sure that the **-javaagent** parameter is before -jar.

```
java -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey=xxx -Darms.appId=xxx -jar demoApp.jar
```

4. If the log file in the ArmsAgent/log/ directory contains the error "LicenseKey is invalid", check whether the region of your application is the same as the region of the ARMS agent.
5. After your application is started, if the log sub-directory is unavailable in the ARMS agent directory since arms-bootstrap-1.7.0-SNAPSHOT.jar fails to be loaded, check whether the permission over the ARMS agent installation directory is correct.
6. If the following error is reported when your application is started, check whether the arms-bootstrap-1.7.0-SNAPSHOT.jar package and the corresponding permission are correct.

7. If there is still no monitoring data, pack the log file of the ARMS agent for Java applications in the ArmsAgent/log directory, and contact Customer Services of ARMS at DingTalk service account @ARMS for assistance.
8. Check the JDK version. If the JDK version is 1.8.0_25 or 1.8.0_31, you may fail to install the agent. We recommend that you upgrade the JDK or contact Customer Services of ARMS at DingTalk service account @ARMS.

What can I do if no IP address or an incorrect IP address is displayed after the ARMS agent is successfully loaded?

1. Run the ifconfig -a command to check the network configuration of the current machine. If the machine uses multiple network interface controllers (NICs), the IP address acquired by the ARMS agent is possibly not as expected, which is related to the network configuration.
2. Solve the problem by using either of the following methods:
   - Configure the -DEAGLEEYE.LOCAL.IP=10.XX.XX.XX JVM parameter explicitly.

Note:

Replace 10.XX.XX.XX with the actual IP address.

- Configure the ARMS agent to obtain the -DNETWORK.INTERFACE=eth0 JVM parameter, where eth0 indicates the NIC name.

Troubleshoot common errors contained in the log file (ArmsAgent/log) of the ARMS agent

- If "LicenseKey is invalid" is reported:
  1. First, make sure that your application is successfully created in ARMS and that AppId and LicenseKey were correctly set when the ARMS agent was installed.
  2. ARMS is a multi-region environment. Therefore, you also need to check whether the download link of the ARMS agent is in the same region as your application.