阿里云
应用实时监控服务

应用监控

文档版本：20190914

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。
2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。
3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。
4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的”现状“、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。
5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含”阿里云”、Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。
6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	**禁止:** 重置操作将丢失用户配置数据。
	该类警示信息可能导致系统重大变更甚至故障，或者导致人身伤害等结果。	**警告:** 重启操作将导致业务中断，恢复业务所需时间约10分钟。
	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	**说明:** 您也可以通过按Ctrl + A选中全部文件。
>	多级菜单递进。	设置 > 网络 > 设置网络类型
粗体	表示按键、菜单、页面名称等UI元素。	单击 确定。
`courier` 字体	命令。	执行 `cd /d C:/windows` 命令，进入Windows系统文件夹。
*## *	表示参数、变量。	`bae log list --instanceid` *`Instance_ID`*
[]或者[a|b]	表示可选项，至多选择一个。	`ipconfig` *`[-all|-t]`*
{}或者{a|b}	表示必选项，至多选择一个。	`swich` *`{stand | slave}`*

目录

法律声明.....I
通用约定.....I
1 应用监控概述..... 1
2 开始监控 Java 应用..... 3
2.1 以普通方式安装探针.....3
2.1.1 为 Java 应用安装探针..... 3
2.1.2 为 Docker 中的应用安装探针.....8
2.1.3 为容器服务 Kubernetes 版 Java 应用安装探针.....11
2.1.4 为开源 Kubernetes 环境中的应用安装探针.....15
2.2 以快速方式安装探针..... 20
2.2.1 为 Java 应用快速安装探针.....20
2.2.2 为 EDAS 中的应用快速安装探针.....23
2.2.3 为 ECS 中的应用快速安装探针..... 26
3 开始监控 PHP 应用..... 31
3.1 为 PHP 应用安装探针..... 31
3.2 为容器服务 Kubernetes 版 PHP 应用无侵入安装探针.....35
4 控制台功能.....41
4.1 应用总览.....41
4.2 应用详情.....44
4.2.1 JVM 监控..... 44
4.2.2 主机监控.....46
4.2.3 内存快照.....48
4.3 应用接口调用监控.....51
4.4 MQ 监控..... 54
4.5 调用链路查询.....55
4.6 应用监控 3D 拓扑图.....57
4.7 ### 实时诊断..... 65
4.8 自定义配置.....68
5 使用教程.....73
5.1 使用全息排查诊断业务问题..... 73
5.2 使用线程剖析诊断代码层面的问题..... 76
5.3 诊断服务端报错问题..... 80
5.4 诊断应用卡顿问题..... 85
5.5 使用主动诊断排查 RT 类错误.....90
6 参考信息.....95
6.1 应用监控兼容性列表..... 95
6.2 探针版本说明..... 97
6.3 关键统计指标说明..... 100
6.4 ARMS-SDK 使用说明.....104

1 应用监控概述

ARMS 应用监控是一款应用性能管理（Application Performance Management，简称 APM）产品。您无需修改代码，只需为应用安装一个探针，ARMS 就能够对应用进行全方位监控，帮助您快速定位出错接口和慢接口、重现调用参数、发现系统瓶颈，从而大幅提升线上问题诊断的效率。

自动发现应用拓扑

ARMS 应用监控探针能够自动发现应用的上下游依赖关系。具体而言，该探针能够有效捕获、智能计算、自动展示不同应用之间通过 RPC 框架（例如 Dubbo、HTTP、HSF 等协议）组成的调用链。您可以通过应用拓扑轻松发现系统中的性能瓶颈和异常调用。

3D 拓扑

3D 拓扑图能立体展示应用、服务和主机的健康状况，以及应用的上下游依赖关系，帮助您快速定位诱发故障的服务、被故障影响的应用和关联的主机等，全方位地诊断故障根源，从而快速排除故障。

捕获异常事务和慢事务

您可以进一步获取接口的慢 SQL、MQ 堆积分析报表或者异常分类报表，对错、慢等常见问题进行更细致的分析。

自动发现并监控接口

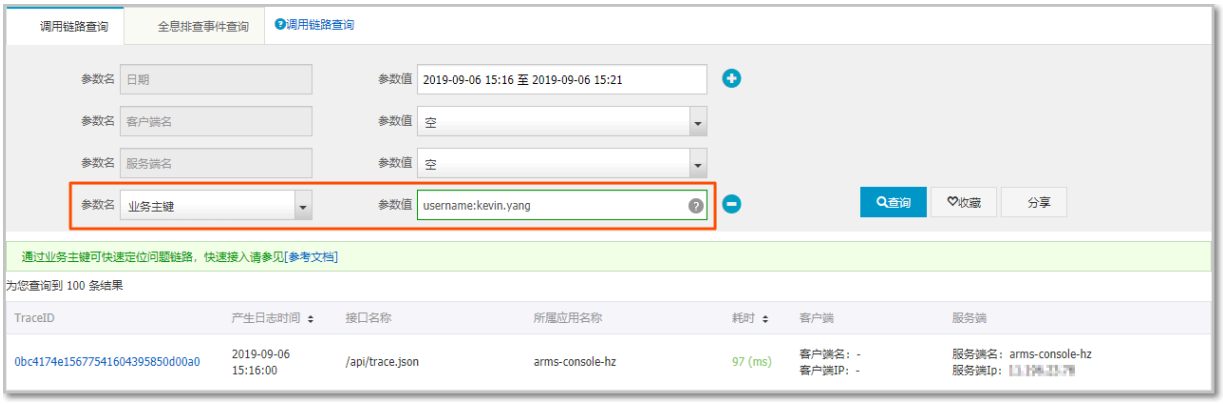
自动发现和监控应用代码中常见的 Web 框架和 RPC 框架，并自动统计 Web 接口和 RPC 接口的调用量、响应时间、错误数等指标。

实时诊断

当您需要密切监控一小段时间内的应用性能时，例如发布应用或者对应用进行压测时，可以使用 ARMS 应用监控的实时诊断功能。开启实时诊断后，ARMS 应用监控会持续监控应用 5 分钟，并在这 5 分钟内全量上报调用链数据。接下来，您就能以出现性能问题的调用链路为起点，通过方法栈瀑布图和线程剖析等功能定位问题原因。

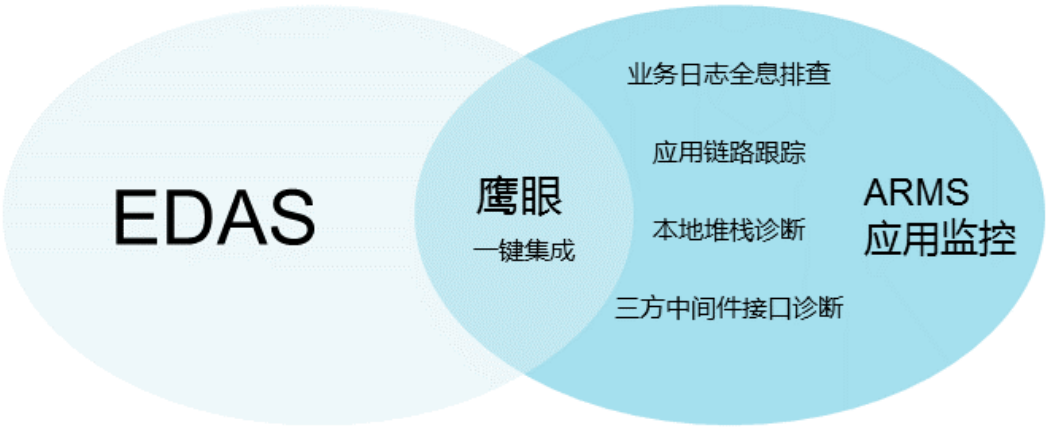
多维排查

您可以查看分布式及本地方法栈明细，并按应用、IP、耗时等维度进行多维分析。您还可以搭配使用 ARMS 自定义监控中的全息排查功能，排除业务单据完整事务。



集成阿里云中间件 PaaS 平台

ARMS 应用监控支持一键集成阿里云 PaaS 平台 EDAS，让运行于阿里中间件分布式架构平台上的应用监控更加有效。



2 开始监控 Java 应用

2.1 以普通方式安装探针

2.1.1 为 Java 应用安装探针

为 Java 应用安装 ARMS 探针后，ARMS 即可对 Java 应用进行应用拓扑、调用链路追踪、异常事务和慢事务监控、SQL 分析等一系列监控。安装探针可采用手动接入方式和一键接入方式。本文将介绍如何使用手动接入方式为 Java 应用安装探针。

前提条件

- 确保您使用的公网服务器安全组已开放 8442、8443、8883 三个端口的 TCP 公网出方向权限，VPC 内不需要开通。为阿里云 ECS 开放出方向权限，请参见#unique_7。

  > **说明:**
  > ARMS 不仅可接入阿里云 ECS 上的应用，还能接入其他能访问公网的服务器上的应用。

- 确保您使用的第三方组件或框架在应用监控兼容性列表范围内，请参见应用监控兼容性列表。

操作步骤

为 Java 应用安装探针操作步骤如下：

1. 登录 ARMS 控制台，在左侧导航栏中选择应用监控 > 应用列表 。
2. 在应用列表页面右上角单击新接入应用。

3. 在新应用接入页面选择使用语言为 Java，选择使用环境为默认，选择接入方式为手动接入。

4. 采用以下方法之一下载探针，然后在控制台下载探针页签中单击下一步。

   - 方法一：手动下载。在下载探针页签中单击下载探针按钮，下载最新 ZIP 包。
   - 方法二：Wget 命令下载。根据您的地域使用 Wget 命令下载对应的探针压缩包。

```
# 杭州地域
wget "http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/
ArmsAgent.zip" -O ArmsAgent.zip
# 上海地域
wget "http://arms-apm-shanghai.oss-cn-shanghai.aliyuncs.com/
ArmsAgent.zip" -O ArmsAgent.zip
# 青岛地域
wget "http://arms-apm-qingdao.oss-cn-qingdao.aliyuncs.com/ArmsAgent.
zip" -O ArmsAgent.zip
# 北京地域
wget "http://arms-apm-beijing.oss-cn-beijing.aliyuncs.com/ArmsAgent.
zip" -O ArmsAgent.zip
# 深圳地域
wget "http://arms-apm-shenzhen.oss-cn-shenzhen.aliyuncs.com/
ArmsAgent.zip" -O ArmsAgent.zip
# 新加坡地域
wget "http://arms-apm-ap-southeast.oss-ap-southeast-1.aliyuncs.com/
cloud_ap-southeast-1/ArmsAgent.zip" -O ArmsAgent.zip
# 金融云环境
wget "http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/finance/
ArmsAgent.zip" -O ArmsAgent.zip
```

5. 切换到探针安装包所在目录，并执行以下命令解压安装包到任意工作目录下。

说明:

{user.workspace} 是示例目录。

```
unzip ArmsAgent.zip -d /{user.workspace}/
```

6. 在控制台安装探针页签中查看并保存 LicenseKey。

7. 采用以下方法之一添加 AppName 以及 LicenseKey 参数。

   · 方法一：根据您的应用运行环境修改 JVM 参数。

   说明：

   将 <LicenseKey> 替换成您的 LicenseKey；将 <AppName> 替换成您的应用名称，应用名暂不支持中文；将*{user.workspace}* 替换成实际探针解压的目录。

   - Tomcat 运行环境

     ■ 在 Linux 或 Mac 环境下，请在 *{TOMCAT_HOME}/bin* 目录下的 *setenv.sh* 文件中加入以下配置。

     说明：

     如果您的 Tomcat 版本没有 *setenv.sh* 配置文件，请打开 *{TOMCAT_HOME}/bin/catalina.sh* 文件，找到 JAVA_OPTS 变量定义，并在该变量定义后加入以下配置。

点击下载参考样例：catalina.sh（第 256 行定义）。

```
JAVA_OPTS="$JAVA_OPTS -javaagent:/{user.workspace}/ArmsAgent
/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey=<
LicenseKey> -Darms.appName=<AppName>"
```

■ 在 Windows 环境下，请在 *{TOMCAT_HOME}/bin/catalina.bat* 文件中加入以下配置：

```
set "JAVA_OPTS=%JAVA_OPTS% -javaagent:{user.workspace}
ArmsAgentarms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey
=<LicenseKey> -Darms.appName=<AppName>"
```

- Jetty 运行环境

  在 *{JETTY_HOME}/start.ini* 配置文件中加入以下配置：

```
--exec #打开注释 前面的井号去掉即可 -javaagent:/{user.workspace}/
ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey=<
LicenseKey> -Darms.appName=<AppName>
```

- Spring Boot 运行环境

  启动 Spring Boot 进程时，在启动命令后面加上 `-javaagent` 参数：

```
java -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.
0-SNAPSHOT.jar -Darms.licenseKey=<LicenseKey> -Darms.appName=<
AppName> -jar demoApp.jar
```

说明：

demoApp.jar 为原应用 JAR 包名称，请根据实际情况替换。

- Resin 运行环境

  a. 启动 Resin 进程时，需要在 *conf/resion.xml* 配置文件中添加以下标签：

```
<server-default> <jvm-arg>-javaagent:{user.workspace}/
ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar</jvm-arg> <jvm-
```

```
arg>-Darms.licenseKey=<LicenseKey> </jvm-arg> <jvm-arg>-Darms
.appName=<AppName> </jvm-xxxarg> </server-default>
```

b. 在 *conf/app-default.xml* 文件中添加以下标签：

```
<library-loader path="{user.workspace}/ArmsAgent/plugin"/>
```

- Windows 运行环境

在 Windows 环境下启动 Java 进程时，请在挂载探针路径中使用反斜杠作为分隔符。

```
{CMD}> java -javaagent:{user.workspace}ArmsAgentarms-bootstrap
-1.7.0-SNAPSHOT.jar -Darms.licenseKey=<LicenseKey> -Darms.
appName=<AppName> -jar {user.workspace}demoApp.jar
```

说明：

demoApp.jar 为原应用 JAR 包名称，请根据实际情况替换。

同一台机器上面，部署同一应用的多个实例场景，可以通过 -Darms.agentId（逻辑编号：如 001，002）参数来区分接入的 JVM 进程，例如：

```
    java -javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.
7.0-SNAPSHOT.jar -Darms.licenseKey=<LicenseKey> -Darms.appName=<
AppName>
    -Darms.agentId=001 -jar demoApp.jar
```

· 方法二：

a. 修改 *arms-agent.config* 文件。将 <LicenseKey> 替换成您的 LicenseKey；将 <AppName> 替换成您的应用名称，应用名暂不支持中文字符。

```
arms.licenseKey=<LicenseKey> arms.appName=<AppName>
```

b. 修改 JVM 参数，在 Java 应用的启动脚本中添加以下参数。

```
-javaagent:/{user.workspace}/ArmsAgent/arms-bootstrap-1.7.0-
SNAPSHOT.jar
```

说明：

将 {user.workspace} 替换成实际探针解压的目录。

8. 重启 Java 应用。

结果验证

约一分钟后，若 Java 应用出现在应用列表中且有数据上报，则说明接入成功。

卸载 Java 探针

1. 删除第 7 步中添加的 AppName、LicenseKey 相关的所有参数。

2. 重启 Java 应用。

快速更改应用名称

如果因为某些原因希望更改应用名称，例如忘记将示例应用名称 Java-Demo 修改为自定义名称，您可以在不重启应用、不重装探针的情况下更改应用名称，详情参见#unique_9/unique_9_Connect_42_section_czl_pxe_z9v。

更多信息

- #unique_10
- #unique_11

2.1.2 为 Docker 中的应用安装探针

为 Docker 中的 Java 应用安装 ARMS 探针后，ARMS 即可对该应用进行应用拓扑、调用链路追踪、异常事务和慢事务监控、SQL 分析等一系列监控。

背景信息

Docker 是一个开放源代码软件项目，让应用程序部署在软件货柜下的工作可以自动化进行，借此在Linux操作系统上，提供一个额外的软件抽象层，以及操作系统层虚拟化的自动管理机制。Docker利用 Linux 核心中的资源分离机制，例如 cgroups，以及 Linux 核心名字空间，来创建独立的容器。

为 Docker 中的应用安装 ARMS 探针后，ARMS 将自动适配该应用运行的环境，不需要针对 Tomcat、Jetty 和 Springboot 等应用配置运行环境。

前提条件

1. 您已成功开通 ARMS 服务，请参见#unique_13。
2. 您已在 Docker 中部署 Java 应用。

操作步骤

如果有一个已部署 Java 应用的镜像 *{original-docker-image:tag}*，可以通过编辑 *Dockerfile* 文件集成已有镜像来形成新的镜像，然后构建、启动新的镜像即可将 Java 应用接入 ARMS 应用监控。具体操作步骤如下：

1. 通过编辑 *Dockerfile* 集成 *{original-docker-image:tag}* 镜像。Dockerfile 示例如下：

```
###################################
##                               ###
##      ARMS APM DEMO Docker     ###
##          For Java             ###
##      withAgent   V0.1         ###
##                               ###
```

```
###################################
FROM {original-docker-image:tag}
WORKDIR /root/
# 请根据所在地域替换探针的下载地址。
RUN wget "http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/
ArmsAgent.zip" -O ArmsAgent.zip
RUN unzip ArmsAgent.zip -d /root/
# LicenseKey 在控制台应用监控接入页面查看。
# AppName 为用户自定义 ARMS 监控应用名称，用户名暂不支持中文。
# 若所有镜像都接入同一个应用监控任务，配置此处的 arms_licenseKey 和
arms_appName 即可。
# 若需将镜像接入其他应用监控任务，可在 docker run 中使用 -e 参数指定该应用的
arms_licenseKey 和 arms_appName 参数，以覆盖此处的配置。
ENV arms_licenseKey=xxx
ENV arms_appName=xxx
ENV JAVA_TOOL_OPTIONS ${JAVA_TOOL_OPTIONS} '-javaagent:/root/
ArmsAgent/arms-bootstrap-1.7.0-SNAPSHOT.jar -Darms.licenseKey='${
arms_licenseKey}' -Darms.appName='${arms_appName}
### for check the args
RUN env | grep JAVA_TOOL_OPTIONS
### 下面可加入用户 自定义 dockerfile 逻辑。
### ......
```

- 将配置中的 `original-docker-image:tag` 替换为您自己的镜像地址。若您没有自定义镜像，可使用系统镜像。
- 根据所在地域替换探针的下载地址。

```
# 杭州地域
wget "http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/
ArmsAgent.zip" -O ArmsAgent.zip
# 上海地域
wget "http://arms-apm-shanghai.oss-cn-shanghai.aliyuncs.com/
ArmsAgent.zip" -O ArmsAgent.zip
# 青岛地域
wget "http://arms-apm-qingdao.oss-cn-qingdao.aliyuncs.com/
ArmsAgent.zip" -O ArmsAgent.zip
# 北京地域
wget "http://arms-apm-beijing.oss-cn-beijing.aliyuncs.com/
ArmsAgent.zip" -O ArmsAgent.zip
# 深圳地域
wget "http://arms-apm-shenzhen.oss-cn-shenzhen.aliyuncs.com/
ArmsAgent.zip" -O ArmsAgent.zip
# 新加坡地域
wget "http://arms-apm-ap-southeast.oss-ap-southeast-1.aliyuncs.com
/cloud_ap-southeast-1/ArmsAgent.zip" -O ArmsAgent.zip
# 金融云环境
```

```
wget "http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/
finance/ArmsAgent.zip" -O ArmsAgent.zip`
```

- 将 `arms_licenseKey` 和 `arms_appName` 分别替换成您的 LicenseKey 和应用名称。获取 LicenseKey 步骤如下：

  a. 登录 ARMS 控制台，在左侧导航栏中选择应用监控 > 应用列表。

  b. 在应用列表页面右上角单击新接入应用。

  c. 在新接入应用页面顶部单击 License Key 右侧的复制图标。

2. 运行 `docker build` 命令来构建镜像。示例如下：

```
docker build -t registry.cn-hangzhou.aliyuncs.com/arms-docker-repo/
arms-springboot-demo:v0.1 -f /{workspace}/Dockerfile /{workspace}/
```

说明：

`registry.cn-hangzhou.aliyuncs.com/arms-docker-repo/arms-springboot-demo:v0.1` 为镜像名称，请根据实际修改。

3. 运行 `docker run` 命令来启动镜像。

使用原有镜像 *{original-docker-image:tag}* 的 `docker run` 启动脚本来启动即可。若需将镜像接入其他应用监控任务，可在 `docker run` 中使用 `-e` 参数指定该应用的 `arms_licenseKey` 和 `arms_appName` 参数，以覆盖 *Dockerfile* 中的配置。示例如下：

```
docker run -d -e "arms_licenseKey=<LicenseKey>" -e "arms_appName=<
AppName>" -p 8081:8080 registry.cn-hangzhou.aliyuncs.com/arms-docker
-repo/arms-springboot-demo:v0.1
```

说明：

- 将 `<LicenseKey>` 替换成您的 LicenseKey，将 `<AppName>` 替换成您的应用名称，应用名暂不支持中文。

- `registry.cn-hangzhou.aliyuncs.com/arms-docker-repo/arms-springboot-demo:v0.1` 为镜像名称，请根据实际修改。

结果验证

约一分钟后，若您的应用出现在应用列表中且有数据上报，则说明接入成功。

2.1.3 为容器服务 Kubernetes 版 Java 应用安装探针

只需安装 ARMS 应用监控组件（探针），即可对部署在容器服务 Kubernetes 版中的 Java 应用进行监控，查看应用拓扑、接口调用、异常事务和慢事务等方面的监控数据。本文介绍如何为容器服务 Kubernetes 版 Java 应用安装探针。

前提条件

- #unique_15
- #unique_16：本文示例中的命名空间名称为 `arms-demo`

安装 ARMS 应用监控组件

首先需要安装 ARMS 应用监控组件 ack-arms-pilot。

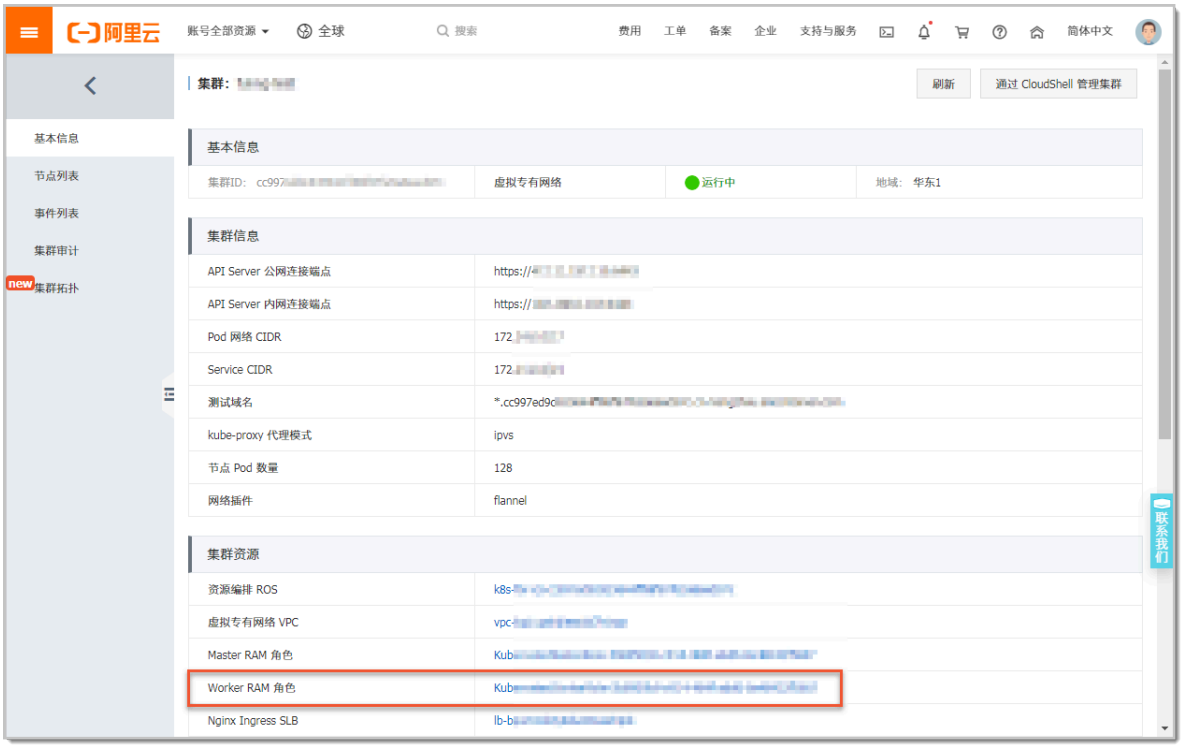
1. 登录容器服务 Kubernetes 版控制台。
2. 在左侧导航栏选择市场 > 应用目录，在右侧页面单击 ack-arms-pilot。
3. 在应用目录 - ack-arms-pilot 页面上，在右侧的创建面板中选择前提条件中创建的集群和命名空间，并单击创建。

为容器服务 Kubernetes 版授权

接下来要为容器服务 Kubernetes 版授予 ARMS 资源的访问权限。

1. 使用主账号登录容器服务 Kubernetes 版控制台。
2. 在左侧导航栏选择集群 > 集群，在集群列表页面上的目标集群右侧操作列单击管理。

3. 在目标集群的基本信息页面上，单击集群资源区域的 Worker RAM 角色链接。

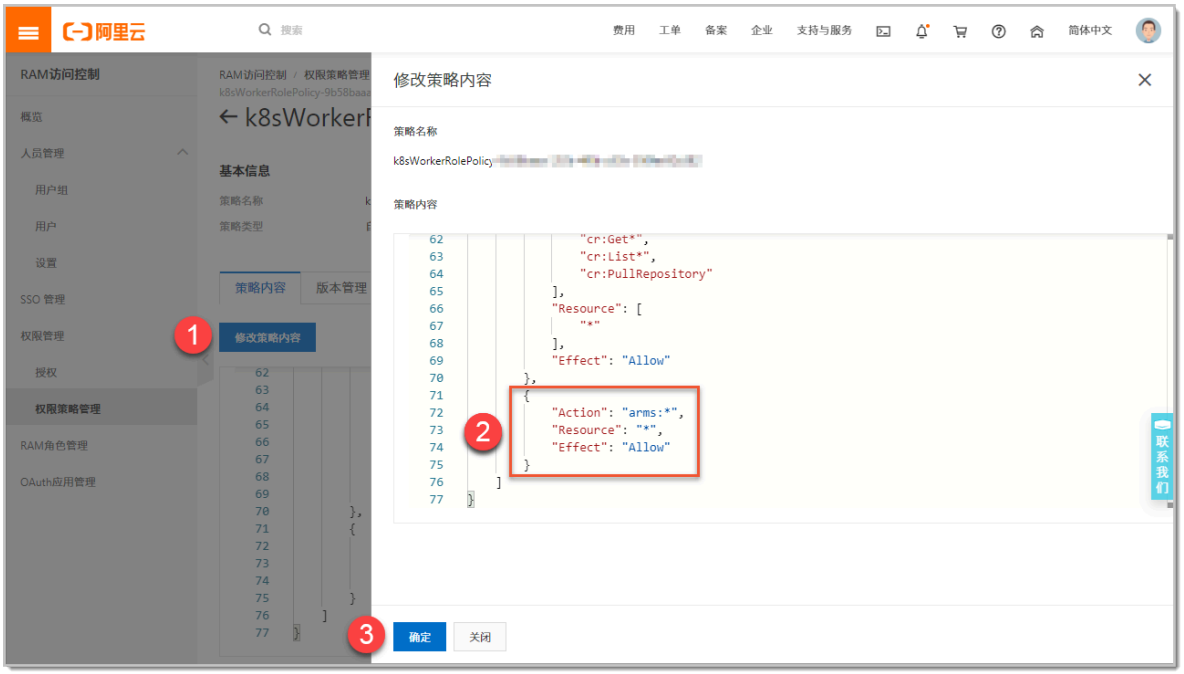


4. 在 RAM 访问控制控制台的 RAM 角色管理页面上，单击权限管理页签上的目标权限策略名称链接。

5. 在策略内容页签上单击修改策略内容，并在右侧的修改策略内容面板将以下内容添加到策略内容中，最后单击确定。

```
{
    "Action": "arms:*",
    "Resource": "*",
    "Effect": "Allow"
```

```
}
```



为 Java 应用开启 ARMS 应用监控

以下步骤分别对应创建新应用和已有应用这两种情况。

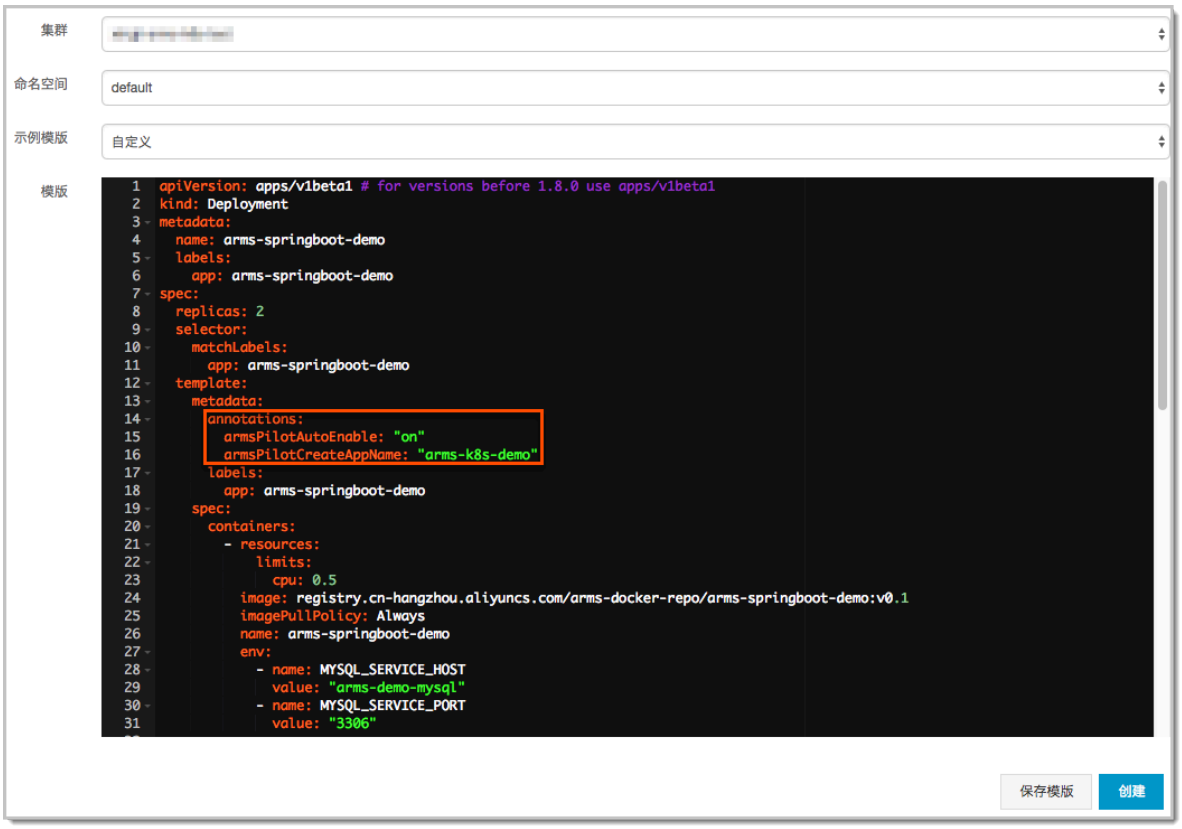
如需在创建新应用的同时开启 ARMS 应用监控，请按以下步骤操作：

1. **在容器服务 Kubernetes 版控制台左侧导航栏选择应用 > 无状态。**
2. **在无状态（Deployment）页面右上角单击使用模板创建。**
3. **在使用模板创建页面上选择集群、命名空间和示例模板，并在模板（YAML 格式）中将以下 `annotations` 添加到 *spec > template > metadata* 层级下。**

> **说明：**
> **请将 *<your-deployment-name>* 替换为您的应用名称。**

```
annotations:
  armsPilotAutoEnable: "on"
```

```
  armsPilotCreateAppName: "<your-deployment-name>"
```



创建一个无状态（Deployment）应用并开启 ARMS 应用监控的完整 YAML 示例模板如下：

如需为现有应用开启 ARMS 应用监控，请按以下步骤操作：

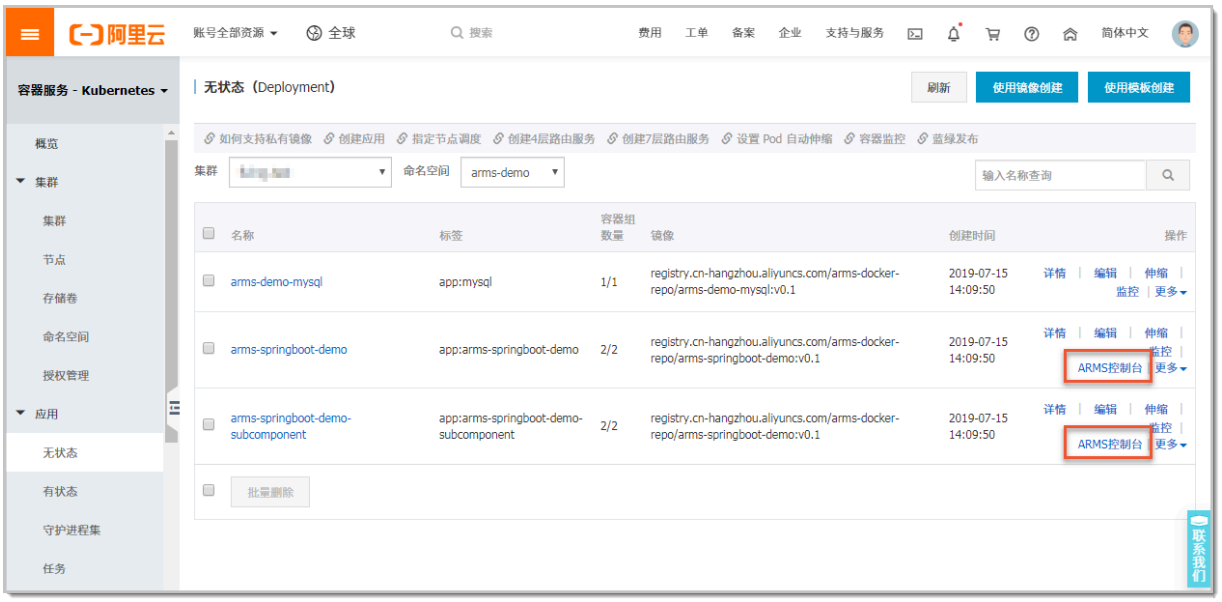
1. **在容器服务 Kubernetes 版控制台左侧导航栏选择应用 > 无状态或应用 > 有状态。**
2. **在无状态（Deployment）或有状态（StatefulSet）页面上，选择集群和命名空间，并在目标应用右侧操作列中选择更多 > 查看 Yaml。**
3. **在编辑 YAML 对话框中将以下 `annotations` 添加到 *spec > template > metadata* 层级下，并单击更新。**

> **说明：**
> **请将 *<your-deployment-name>* 替换为您的应用名称。**

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

预期结果

在无状态（Deployment）或有状态（StatefulSet）页面上，目标应用的操作列将出现 ARMS 控制台按钮。



说明:

若操作列没有出现 ARMS 控制台按钮，请检查您是否已授权容器服务访问 ARMS 资源。

后续步骤

如果因为某些原因希望更改应用名称，例如忘记将示例应用名称 Java-Demo 修改为自定义名称，您只需修改 Deployment 内的 `armsPilotCreateAppName` 参数并重启 Pod，即可在不重启应用、不重装探针的情况下更改应用名称，详见#unique_9/unique_9_Connect_42_section_iw6_l4v_c66。

#unique_15
#unique_16
#unique_17

2.1.4 为开源 Kubernetes 环境中的应用安装探针

借助 ARMS 应用监控，您可以对开源 Kubernetes 环境的应用进行应用拓扑、接口调用、异常事务和慢事务监控、SQL 分析等监控。本文将帮助您将开源 Kubernetes 环境中的应用接入 ARMS 应用监控。

前提条件

- 确保您的 Kubernetes api-server 组件接口版本在 1.10 及以上。
- 确保您的集群联通公网。
- 您已成功开通 ARMS 服务，参见开通 ARMS 服务。

操作步骤

ARMS 应用监控目前仅支持无状态（Deployment）和有状态（StatefulSet）两种类型的应用接入。将开源 Kubernetes 环境中的无状态（Deployment）类型的应用接入 ARMS 应用监控的操作步骤如下：

1. 安装 arms-pilot。

   a. 采用以下方法之一下载 arms-pilot。

   - 方法一：手动下载最新安装包。
   - 方法2： 执行以下 Wget 命令下载 arms-pilot 安装包。

   ```
   wget 'http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/arms-pilot/arms-pilot-0.1.1-community.tgz' -O arms-pilot-0.1.1.tgz
   ```

   b. 执行以下命令解压 arms-pilot 安装包。

   ```
   tar zxvf arms-pilot-0.1.1.tgz
   ```

   c. 执行以下命令安装 arms-pilot。

   ```
   helm install ./arms-pilot --namespace arms-pilot-system
   ```

2. 获取 ARMS 的 LicenseKey。具体步骤如下：

   a. 登录 ARMS 控制台，在左侧导航栏中选择应用监控 > 应用列表。

   b. 在应用列表页面右上角单击新接入应用。

   c. 在新接入应用页面然后查看并保存 LicenseKey。

3. 修改目标无状态（Deployment）应用的 YAML 文件。

   a. 执行以下命令查看目标无状态（Deployment）应用的配置。

   ```
   ### 查看指定无状态（Deployment）类型应用的配置
   ```

```
kubectl get deployment {deployment 名称} -o yaml
```

说明:

若您不清楚 `{deployment 名称}`，请先执行以下命令查看所有无状态（Deployment）应用，在执行结果中找到目标无状态（Deployment）应用，再查看目标无状态（Deployment）应用的配置。

```
### 查看所有无状态（Deployment）类型应用的配置
kubectl get deployments --all-namespace
```

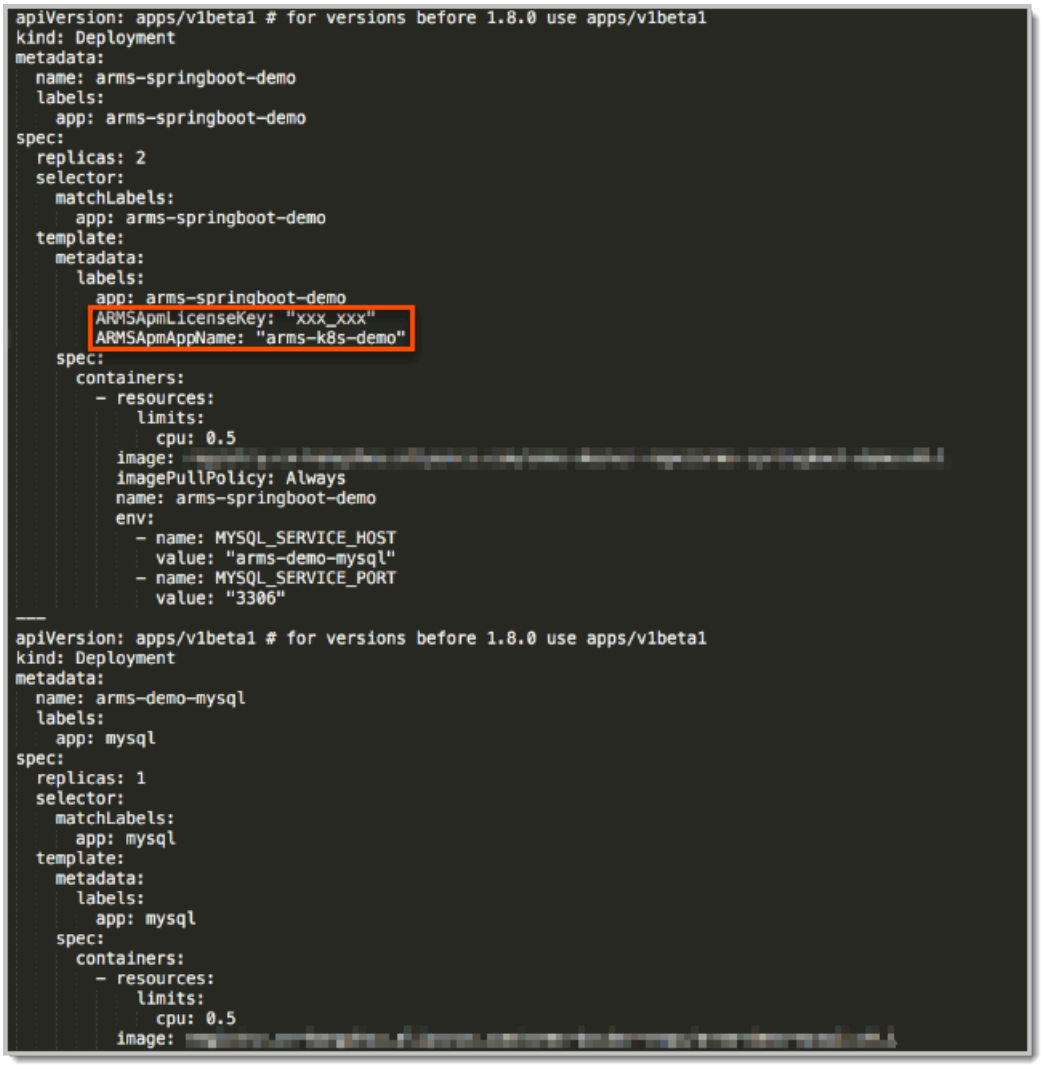
b. 启动编辑目标无状态（Deployment）应用的 YAML 文件。

```
kubectl edit deployment {deployment名} -o yaml
```

c. 在 YAML 文件中的 *spec -> template -> metadata -> labels* 层级下加入以下内容。

- 请将 `xxx` 分别替换成您的 LicenseKey 和应用名称，应用名暂不支持中文。
- 将 LicenseKey 中的符号 `@` 替换为符号 `_` 。

```
ARMSApmAppName: xxx
ARMSApmLicenseKey: xxx
```



示例：

在开源环境中创建一个无状态（Deployment）应用并接入 ARMS 应用监控的完整 YAML 文件如下。

```
apiVersion: apps/v1beta1 # for versions before 1.8.0 use apps/
v1beta1
kind: Deployment
metadata:
  name: arms-springboot-demo
  labels:
    app: arms-springboot-demo
spec:
  replicas: 2
  selector:
    matchLabels:
      app: arms-springboot-demo
  template:
    metadata:
      labels:
        app: arms-springboot-demo
        ARMSApmLicenseKey: "xxx_xxx"
```

```
        ARMSApmAppName: "arms-k8s-demo"
    spec:
      containers:
        - resources:
            limits:
              cpu: 0.5
          image: registry.cn-hangzhou.aliyuncs.com/arms-docker-
repo/arms-springboot-demo:v0.1
          imagePullPolicy: Always
          name: arms-springboot-demo
          env:
            - name: MYSQL_SERVICE_HOST
              value: "arms-demo-mysql"
            - name: MYSQL_SERVICE_PORT
              value: "3306"
---
apiVersion: apps/v1beta1 # for versions before 1.8.0 use apps/
v1beta1
kind: Deployment
metadata:
  name: arms-demo-mysql
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - resources:
            limits:
              cpu: 0.5
          image: registry.cn-hangzhou.aliyuncs.com/arms-docker-
repo/arms-demo-mysql:v0.1
          name: mysql
          ports:
            - containerPort: 3306
              name: mysql
---
apiVersion: v1
kind: Service
metadata:
  labels:
    name: mysql
  name: arms-demo-mysql
spec:
  ports:
    # the port that this service should serve on
    - name: arms-mysql-svc
      port: 3306
      targetPort: 3306
  # label keys and values that must match in order to receive
traffic for this service
  selector:
    app: mysql
---
```

```
```

d. 保存配置。

应用将自动重启，以上配置生效。

2 ~ 5 分钟后，若您的应用出现在 ARMS 控制台的应用监控 > 应用列表中且有数据上报，则说明接入成功。

卸载 arms-pilot

若您不再需要监控开源 Kubernetes 环境的应用，可执行以下命令卸载 arms-pilot。

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

相关文档

为开源 Kubernetes 环境中的应用安装探针常见问题

2.2 以快速方式安装探针

2.2.1 为 Java 应用快速安装探针

ARMS 提供一键接入方式为 Java 应用安装探针，操作简单，安装成功后无需重启应用即可开始监控，适用于新手用户。当应用重启时，探针会自动加载，该 Java 应用将自动接入 ARMS 应用监控。

前提条件

- 确保您使用的第三方组件或框架在应用监控兼容性列表范围内，请参见应用监控兼容性列表。
- 若您的应用已经按照手动接入方式接入 ARMS 应用监控，则需先卸载探针才能正常使用一键接入方式。请参见卸载探针。

操作步骤

具体操作步骤如下：

1. 登录 ARMS 控制台，在左侧导航栏中选择应用监控 > 应用列表 。
2. 在应用列表页面右上角单击新接入应用。

3. 在新接入应用页面选择使用语言为 Java，选择使用环境为默认环境，选择接入方式为一键接入。 然后查看并保存 LicenseKey。

4. 运行您所在地域对应的安装脚本。

   - 将 `<licenseKey>` 替换为您的 LicenseKey。
   - 将 `Java-Demo` 替换成您的应用名，应用名暂不支持中文。

```
# 杭州地域
wget -O- http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/
install.sh | sh && ~/.arms/supervisor/cli.sh <licenseKey> Java-Demo
# 上海地域
wget -O- http://arms-apm-shanghai.oss-cn-shanghai.aliyuncs.com/
install.sh | sh && ~/.arms/supervisor/cli.sh <licenseKey> Java-Demo
# 青岛地域
wget -O- http://arms-apm-qingdao.oss-cn-qingdao.aliyuncs.com/install
.sh | sh && ~/.arms/supervisor/cli.sh <licenseKey> Java-Demo
# 北京地域
wget -O- http://arms-apm-beijing.oss-cn-beijing.aliyuncs.com/install
.sh | sh && ~/.arms/supervisor/cli.sh <licenseKey> Java-Demo
# 深圳地域
wget -O- http://arms-apm-shenzhen.oss-cn-shenzhen.aliyuncs.com/
install.sh | sh && ~/.arms/supervisor/cli.sh <licenseKey> Java-Demo
# 新加坡地域
wget -O- http://arms-apm-ap-southeast.oss-ap-southeast-1.aliyuncs.
com/cloud_ap-southeast-1/install.sh | sh && ~/.arms/supervisor/cli.
sh <licenseKey> Java-Demo
# 金融云环境
```

```
wget -O- http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/
finance/install.sh | sh && ~/.arms/supervisor/cli.sh <licenseKey>
Java-Demo
```

说明:

- 执行安装脚本后，该脚本会自动下载最新探针。
- 若您的服务器只有一个 Java 进程，安装脚本会默认选择该进程安装探针；若您的服务器有多个 Java 进程，请根据提示选择一个进程安装探针。

结果验证

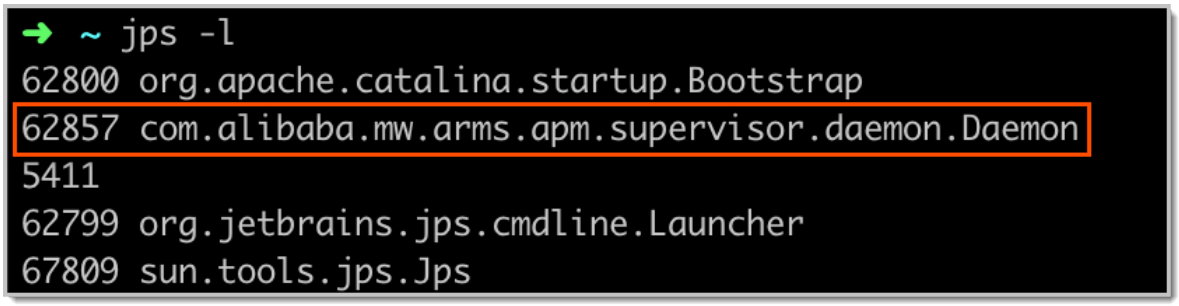
约一分钟后，若您的应用出现在应用列表中且有数据上报，则说明接入成功。

卸载探针

当您不需要 ARMS 探针采集数据时，可按照以下步骤卸载探针：

1. 执行 `jps -l` 命令，并在执行结果中找到 `com.alibaba.mw.arms.apm.supervisor.daemon.Daemon` 对应的进程号。

   在本示例中，`com.alibaba.mw.arms.apm.supervisor.daemon.Daemon` 对应的进程号为：62857。

2. 执行命令 `kill -9 进程号` 。例如：`kill -9 62857` 。
3. 重新启动您的应用。

快速更改应用名称

如果因为某些原因希望更改应用名称，例如忘记将示例应用名称 Java-Demo 修改为自定义名称，您可以在不重启应用、不重装探针的情况下更改应用名称，详情参见#unique_9/unique_9_Connect_42_section_j8u_9a8_b5t。

常见问题

1. 如果在执行一键接入 Java 应用脚本时出现以下 getcwd 相关错误该怎么处理?

```
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.initialize
SystemClass(System.java:1119)
```

可能原因是执行脚本过程中误删了当前目录。解决办法为：先执行 `cd`，然后重新运行脚本。

2. 使用一键接入方式安装探针后，在哪里查看日志？

   日志的默认目录为：*/root/.arms/supervisor/logs/arms-supervisor.log*，若此目录下没有日志，请执行命令`ps -ef |grep arms`查看日志所在目录。

更多信息

- #unique_10
- #unique_11
- #unique_24

2.2.2 为 EDAS 中的应用快速安装探针

借助 ARMS 应用监控，您可以对 EDAS 应用进行应用拓扑、接口调用、异常事务和慢事务监控、SQL 分析等监控。ARMS 与 EDAS 进行了功能集成，通过在 EDAS 控制台简单操作即可将 EDAS 应用快速接入 ARMS 应用监控。

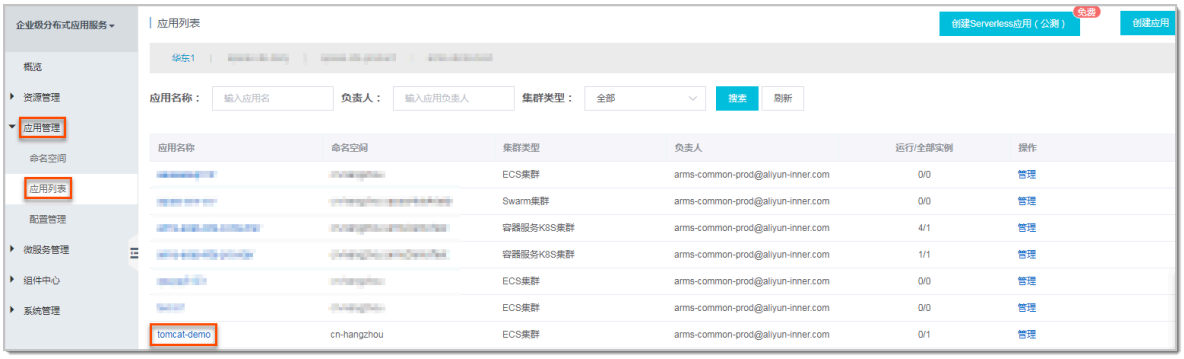
在 EDAS 产品的组件中心开通 ARMS

1. 登录 EDAS 控制台，在左侧导航栏中选择组件中心 > 组件概览。
2. 在组件概览页面顶部的分类区域单击微服务，然后单击应用监控 右侧的立即开通。

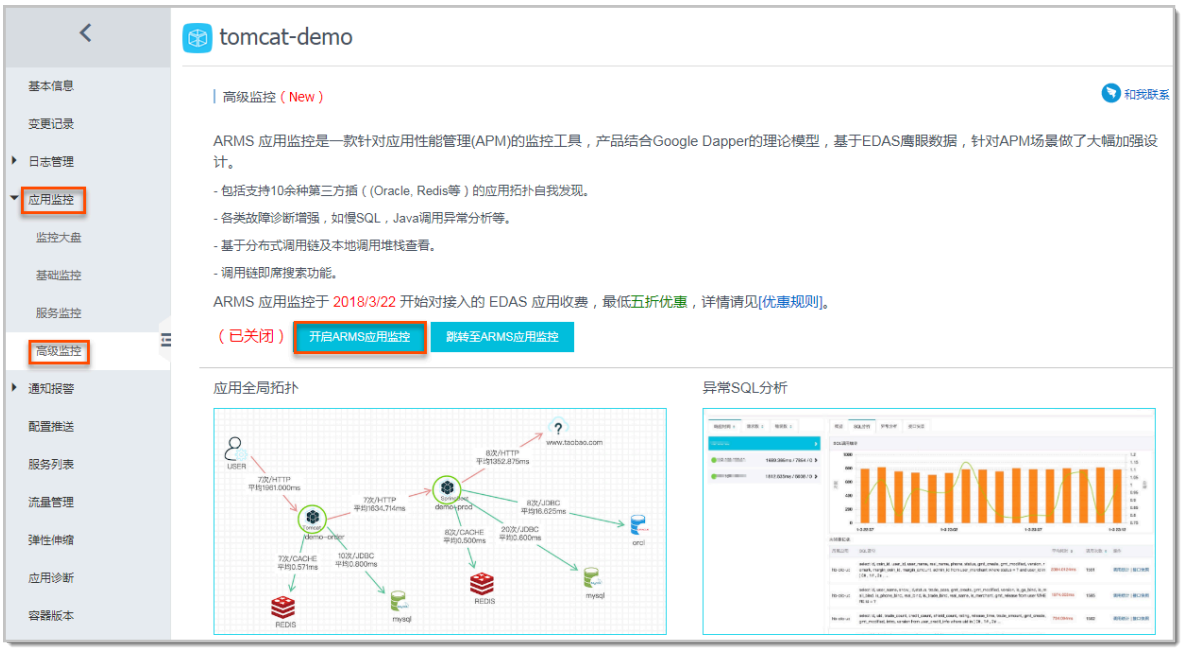
3. 在安全授权提示对话框中单击确定，即可开通 ARMS 应用实时监控服务。

在 EDAS 中开启 ARMS 应用监控

1. 登录 EDAS 控制台，在左侧导航栏中选择应用管理 > 应用列表，单击要开启 ARMS 应用监控的应用名称，进入应用详情页面。



2. 在应用详情页面左侧导航栏中选择应用监控 > 高级监控，单击开启 ARMS 应用监控。



3. 在提示对话框中单击确认。

4. 在提示对话框中单击前去重启应用，令 ARMS 应用监控生效。

在 ARMS 中查看应用

成功开启 ARMS 应用监控并且重启应用后，单击跳转至 ARMS 应用监控，可跳转到 ARMS 应用列表页。

在应用列表页面输入应用名称即可查看应用详情。更多 ARMS 应用监控的使用方法请参见应用监控概述。

2.2.3 为 ECS 中的应用快速安装探针

借助 ARMS 应用监控，您可以对云服务器 ECS 上的应用进行应用拓扑、接口调用、异常事务和慢事务、SQL 分析等监控。ARMS 与 ECS 进行了数据联通，通过在 ARMS 控制台上简单操作即可快速为同阿里云账户下的 ECS 中的应用安装探针。

前提条件

您已经在要部署应用的地域下购买 ECS，并成功部署应用。

操作步骤

1. 登录 ARMS 控制台，在左侧导航栏中选择应用监控 > 应用列表 。
2. 在应用列表页面右上角单击新接入应用。
3. 在新接入应用页面选择使用语言为 Java，选择使用环境为云服务器 ECS。

4. 首次接入时需要先进行 ARMS 访问 ECS 授权，请使用主账号完成授权。

   a. 在弹出的提示框中单击进入 RAM 进行授权。

   b. 在云资源访问授权页面选中 AliyunARMSAccessingECSRole 权限后单击同意授权。

   c. 在同步 ECS 页面单击确定同步。

   d. 关闭同步 ECS页面，完成授权。

   授权成功后，新接入应用页面中将显示此账号下所有 ECS 实例。

5. 在请选择您要安装探针的应用区域单击目标 ECS 实例操作列的安装探针，并在弹出的提示对话中单击确认。

在 ECS 上安装探针成功后，ARMS 将获取此 ECS 上的所有进程信息并显示在目标 ECS 实例下方的进程列表中。

说明:

若成功安装探针后，ECS 进程信息不准确，请单击 ECS 实例左侧的 - 号然后单击 + 号刷新信息。若探针安装失败请参见常见问题进行处理。

6. 探针安装成功后，在下方的弹框中编辑目标进程的应用名称，然后单击操作列的启用应用监控。

说明:

当多个进程的应用名称相同时，表现为一个监控任务下的多个实例。

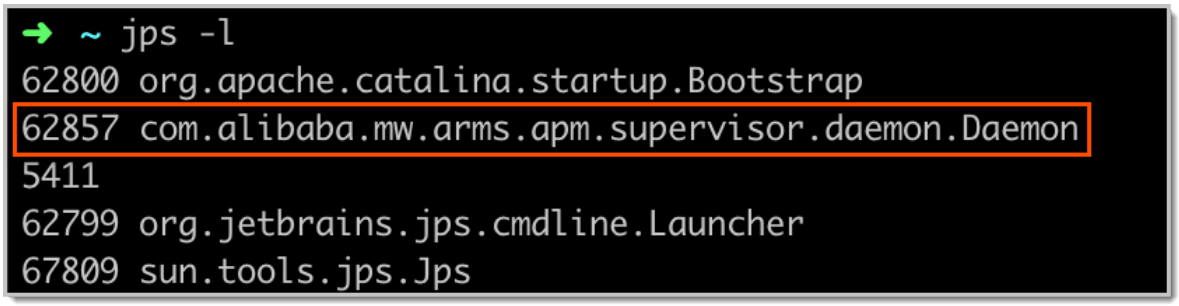
约一分钟后，若您的应用出现在应用列表中且有数据上报，则说明接入成功。

卸载探针

当您不需要监控 ECS 上的应用时，可以卸载 ECS 上的探针。卸载之后，ARMS 将停止对该 ECS 上所有应用的监控。操作步骤如下：

1. 在安装探针的 ECS 中执行 `jps -l` 命令查看所有进程，在执行结果中找到 `com.alibaba.mw.arms.apm.supervisor.daemon.Daemon` 对应的进程号。

   本示例中，对应的进程号为：62857。



2. 执行命令 `kill -9 进程号`。例如：`kill -9 62857`。
3. 重新启动您的应用。

常见问题

探针安装失败怎么处理?

1. 确保您的 ECS 可以访问所在地域的探针下载链接。

   首先确保 ECS 可以访问外网，且能够访问所在地域的探针下载链接。

```
# 杭州地域
http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/install.sh
# 上海地域
http://arms-apm-shanghai.oss-cn-shanghai.aliyuncs.com/install.sh
# 青岛地域
http://arms-apm-qingdao.oss-cn-qingdao.aliyuncs.com/install.sh
# 北京地域
http://arms-apm-beijing.oss-cn-beijing.aliyuncs.com/install.sh
# 深圳地域
http://arms-apm-shenzhen.oss-cn-shenzhen.aliyuncs.com/install.sh
# 新加坡地域
http://arms-apm-ap-southeast.oss-ap-southeast-1.aliyuncs.com/
cloud_ap-southeast-1/install.sh
```

2. 确保您的 ECS 可以访问 ARMS 控制台。

```
#国内
https://arms.console.aliyun.com/

#新加坡
```

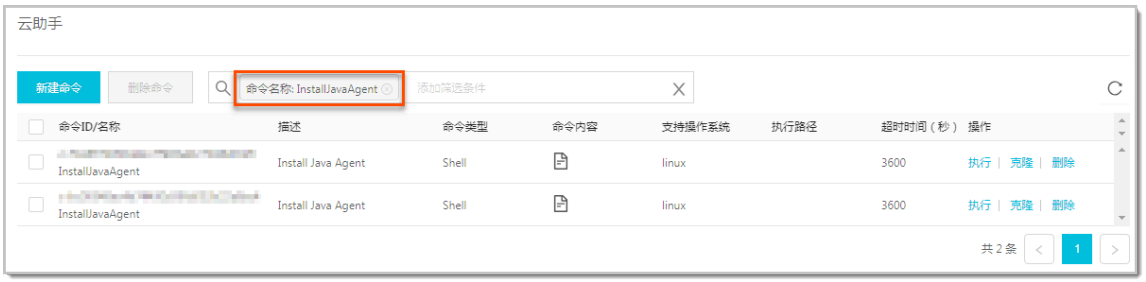
```
https://arms-ap-southeast-1.console.aliyun.com
```

3. 登录 ECS 控制台，并完成以下检查工作。

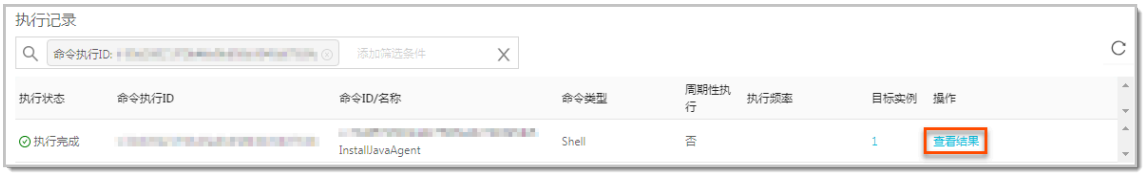
   a. 在左侧导航栏中选择运维与监控 > 云助手。

   b. 在云助手页面的搜索框中选择命令名称，并输入 `InstallJavaAgent`。

   若查找结果不存在，请联系 ARMS 钉钉服务账号 arms160804。



   c. 在云助手页面的执行记录页签的搜索框中输入 `InstallJavaAgent` 命令对应的 ID，在查找结果中单击该记录右侧操作列的查看结果，查看 `InstallJavaAgent` 命令是否执行成功。若未执行成功，根据详细执行结果排查问题（如 ECS 磁盘满、未安装 Java Agent 等问题，可以通过清理磁盘或安装 Java Agent 解决），若不能解决请将详细执行结果反馈给 ARMS 钉钉服务账号 arms160804。



快速更改应用名称

如果因为某些原因希望更改应用名称，例如忘记将示例应用名称 Java-Demo 修改为自定义名称，您可以在不重启应用、不重装探针的情况下更改应用名称，详情参见#unique_9/unique_9_Connect_42_section_zs3_iqg_tlj。

相关文档

为 ECS 中的应用快速安装探针的常见问题

3 开始监控 PHP 应用

3.1 为 PHP 应用安装探针

为 PHP 应用安装 ARMS 探针并重启应用后，ARMS 即可对 PHP 应用进行应用拓扑、调用链路追踪、异常事务和慢事务监控、SQL 分析等一系列监控。

前提条件

- 确保您使用的公网服务器安全组已开放 8442、8443、8883 三个端口的 TCP 公网出方向权限，VPC 内不需要开通。为阿里云 ECS 开放出方向权限，请参见#unique_7。

  > 说明:
  > ARMS 不仅可接入阿里云 ECS 上的应用，还能接入其他能访问公网的服务器上的应用。

- 确保您的应用使用的第三方组件或框架在应用监控兼容性列表范围内，请参见应用监控兼容性列表。

接入 PHP 探针

1. 登录 ARMS 控制台，在左侧导航栏中选择应用监控 > 应用列表 。
2. 在应用列表页面右上角单击新接入应用。
3. 在新接入应用页面选择使用语言为 PHP。

4. 采用以下任意一种方法下载 PHP 探针，完成后单击下一步。

   · 方法一：手动下载。单击下载探针按钮，下载最新 ZIP 包。

   · 方法二：Wget 命令下载。根据所在地域使用 Wget 命令下载 Agent 压缩包。

```
# 杭州地域
wget "http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/arms-php
-agent.zip" -O arms-php-agent.zip

# 上海地域
wget "http://arms-apm-shanghai.oss-cn-shanghai.aliyuncs.com/arms-php
-agent.zip" -O arms-php-agent.zip

# 青岛地域
wget "http://arms-apm-qingdao.oss-cn-qingdao.aliyuncs.com/arms-php-
agent.zip" -O arms-php-agent.zip

# 北京地域
wget "http://arms-apm-beijing.oss-cn-beijing.aliyuncs.com/arms-php-
agent.zip" -O arms-php-agent.zip

# 深圳地域
wget "http://arms-apm-shenzhen.oss-cn-shenzhen.aliyuncs.com/arms-php
-agent.zip" -O arms-php-agent.zip

# 新加坡地域
wget "http://arms-apm-ap-southeast.oss-ap-southeast-1.aliyuncs.com/
cloud_ap-southeast-1/arms-php-agent.zip" -O arms-php-agent.zip

# 金融云环境
wget "http://arms-apm-hangzhou.oss-cn-hangzhou.aliyuncs.com/finance/
arms-php-agent.zip" -O arms-php-agent.zip
```

5. 切换到安装包所在目录，运行以下命令解压安装包到任意工作目录下。

```
unzip arms-php-agent.zip /{user.workspace}/
```

说明:

`{user.workspace}` 是示例路径，请按需修改。

6. 在安装探针页签查看并保存 LicenseKey。

7. 采用以下方法之一安装和配置探针。

   - 自动安装。执行以下命令自动安装探针。

```
cd arms-php-agent
./install.sh <licenseKey> PHP-Demo
```

> **说明:**
>
> - 将 `<licenseKey>` 替换为您的 LicenseKey。

- 将 PHP-Demo 替换成您的应用名称。应用名称暂不支持中文字符。

- 手动安装。

  a. 切换到探针解压目录。

  ```
  cd arms-php-agent
  ```

  b. 修改 *arms-agent.conf* 文件中的配置项。

  ```
  LicenseKey=<licenseKey>
  AppName=PHP-Demo
  PluginRootDir=<arms-php-agent>
  ```

  说明:

  - 将 `<licenseKey>` 替换为您的 LicenseKey。
  - 将 PHP-Demo 替换成您的应用名称。应用名称暂不支持中文字符。
  - 将 `<arms-php-agent>` 替换为解压后 *arms-php-agent* 目录下 *plugins* 的绝对路径。

  c. 执行以下命令获取 PHP 拓展安装目录。

  ```
  php -i | grep ^extension_dir
  ```

  若返回结果如下所示，则安装目录为 `/usr/lib/php/extensions/no-debug-non-zts-20160303`。

  ```
  extension_dir => /usr/lib/php/extensions/no-debug-non-zts-
  20160303 => /usr/lib/php/extensions/no-debug-non-zts-20160303
  ```

  d. 将 *arms.so* 文件拷贝到上一步获取的 PHP 拓展安装目录。

  ```
  sudo cp arms.so /usr/lib/php/extensions/no-debug-non-zts-
  20160303
  ```

  e. 新增动态链接库路径。

  ```
  sudo vi /etc/ld.so.conf
  ```

  f. 新增动态链接共享库和静态档案库。

  ```
  <php-agent-dir>/lib
  ```

  说明:

`<php-agent-dir>` 为探针解压后的绝对路径。

g. 加载动态链接库。

```
sudo ldconfig
```

8. 在 *php.ini* 文件末尾添加以下内容 。

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

说明:

- 若使用脚本安装，请使用脚本提示的替换内容。
- 若使用手动安装，则需将 `<php_extension_dir>` 替换为 PHP 拓展安装目录，默认安装目录为 `/usr/lib64/xxx`，`<php-agent-dir>` 替换为解压后探针目录的绝对路径。

9. 重启您的 PHP 应用。

结果验证

约一分钟后，若您的 PHP 应用出现在应用列表中且有数据上报，则说明接入成功。

卸载 PHP 探针

当您不需要 ARMS 监控 PHP 应用时，可按照以下步骤卸载探针。

1. 修改 *php.ini* 文件，删除以下四行：

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

2. 重启您的 PHP 应用。

卸载探针后，您的应用数据将不会上报 ARMS。

3.2 为容器服务 Kubernetes 版 PHP 应用无侵入安装探针

只需安装 ARMS 应用监控组件（探针），即可监控部署在容器服务 Kubernetes 版中的 PHP 应用，查看应用拓扑、接口调用、异常事务和慢事务等方面的监控数据。本文介绍如何为容器服务 Kubernetes 版 PHP 应用安装探针。

前提条件

- #unique_15

- #unique_16：本文示例中的命名空间名称为 arms-php-demo

> **注意：**
> PHP 应用监控处于公测期，使用不会产生费用。

安装 ARMS 应用监控组件

首先需要安装 ARMS 应用监控组件 ack-arms-pilot。

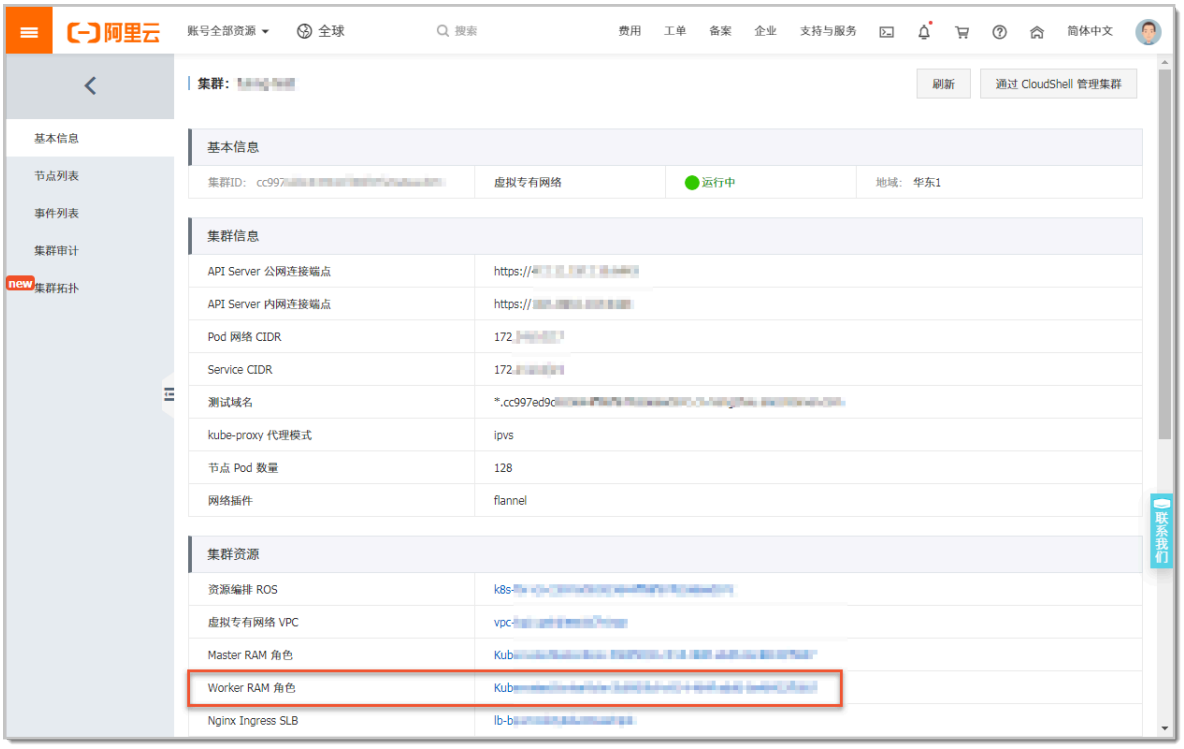
1. 登录容器服务 Kubernetes 版控制台。
2. 在左侧导航栏选择市场 > 应用目录，在右侧页面单击 ack-arms-pilot。
3. 在应用目录 - ack-arms-pilot 页面上，在右侧的创建面板中选择前提条件中创建的集群和命名空间，并单击创建。

为容器服务 Kubernetes 版授权

接下来要为容器服务 Kubernetes 版授予 ARMS 资源的访问权限。

1. 使用主账号登录容器服务 Kubernetes 版控制台。
2. 在左侧导航栏选择集群 > 集群，在集群列表页面上的目标集群右侧操作列单击管理。

3. 在目标集群的基本信息页面上，单击集群资源区域的 Worker RAM 角色链接。

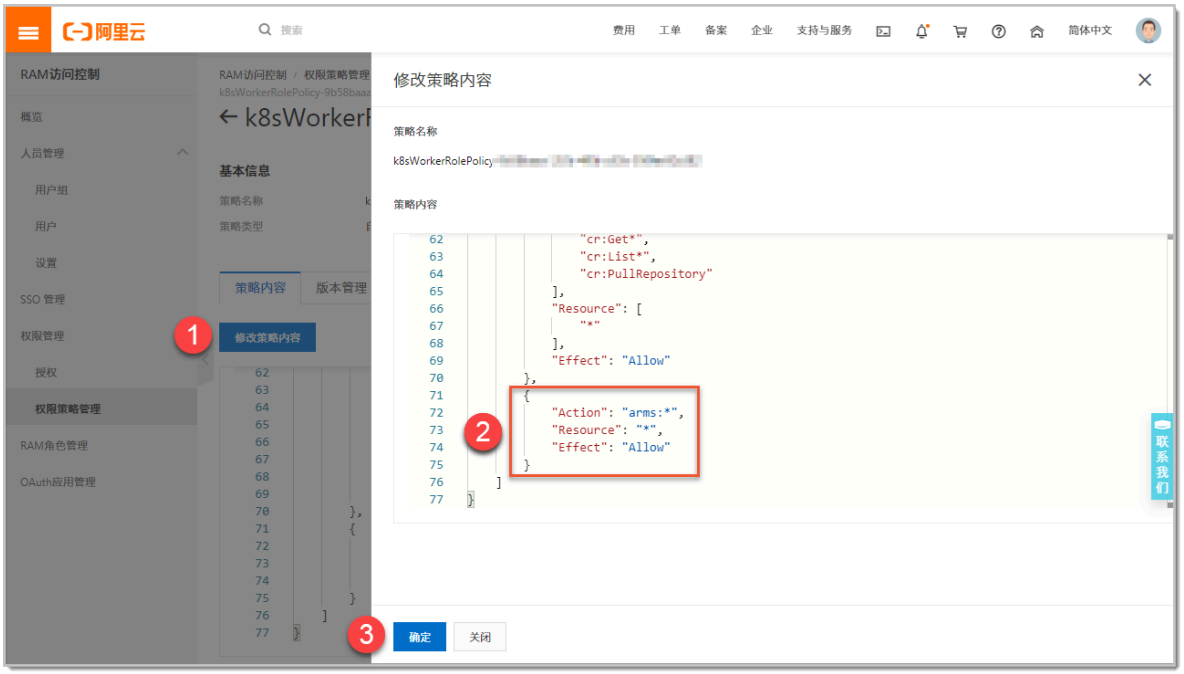


4. 在 RAM 访问控制控制台的 RAM 角色管理页面上，单击权限管理页签上的目标权限策略名称链接。

5. 在策略内容页签上单击修改策略内容，并在右侧的修改策略内容面板将以下内容添加到策略内容中，最后单击确定。

```
{
    "Action": "arms:*",
    "Resource": "*",
    "Effect": "Allow"
```

```
}
```



为 PHP 应用开启 ARMS 应用监控

以下步骤分别对应创建新应用和已有应用这两种情况。

如需在创建新应用的同时开启 ARMS 应用监控，请按以下步骤操作：

1. 在容器服务 Kubernetes 版控制台左侧导航栏选择应用 > 无状态。
2. 在无状态（Deployment）页面右上角单击使用模板创建。
3. 在使用模板创建页面上选择集群、 命名空间和示例模板，并在模板（YAML 格式）中将以下 `annotations` 添加到 *spec > template > metadata* 层级下。

> **说明:**
> 请将 *<your-deployment-name>* 替换为您的应用名称。

```
annotations:
  armsPilotAutoEnable: "on"
  armsPilotCreateAppName: "<your-deployment-name>"
  armsAppType: PHP
```

4. （本步骤仅限首次安装时）请修改安装 arms-pilot 的命名空间下的 *ConfigMap arms-php.ini* 文件，该文件内容为 *php.ini* 默认配置。

> **说明:**

在 extension=/usr/local/arms/arms-php-agent/arms-7.2.so 配置中，*arms-7.2.so* 中的 7.2 为您的 PHP 版本，可使用的值为 5.4、5.5、5.6、7.0、7.1、7.2。

5. 将 *arms-php.ini ConfigMap* 配置项添加到 *php.ini* 文件的 *spec > template > spec > containers* 配置下，将 mountPath 设置为您的 *php.ini* 文件路径。

```
volumeMounts:
        - name: php-ini
          mountPath: /etc/php/7.2/fpm/php.ini
          subPath: php.ini
```

```
volumes:
      - name: php-ini
        configMap:
          name: arms-php.ini
```

创建一个无状态（Deployment）应用并开启 ARMS 应用监控的完整 YAML 示例模板如下：

如需为现有应用开启 ARMS 应用监控，请按以下步骤操作：

1. 在容器服务 Kubernetes 版控制台左侧导航栏选择应用 > 无状态或应用 > 有状态。
2. 在无状态（Deployment）或有状态（StatefulSet）页面上，选择集群和命名空间，并在目标应用右侧操作列中选择更多 > 查看 Yaml。
3. 在编辑 YAML 对话框中将以下 annotations 添加到 *spec > template > metadata* 层级下，并单击更新。

说明：

请将 <*your-deployment-name*> 替换为您的应用名称。

```
annotations:
  armsPilotAutoEnable: "on"
  armsPilotCreateAppName: "<your-deployment-name>"
  armsAppType: PHP
```

4. 挂载 *arms-php.ini ConfigMap* 项到 *php.ini* 文件 *spec > template > spec > containers* 下，将 mountPath 设置为您的 *php.ini* 文件路径。

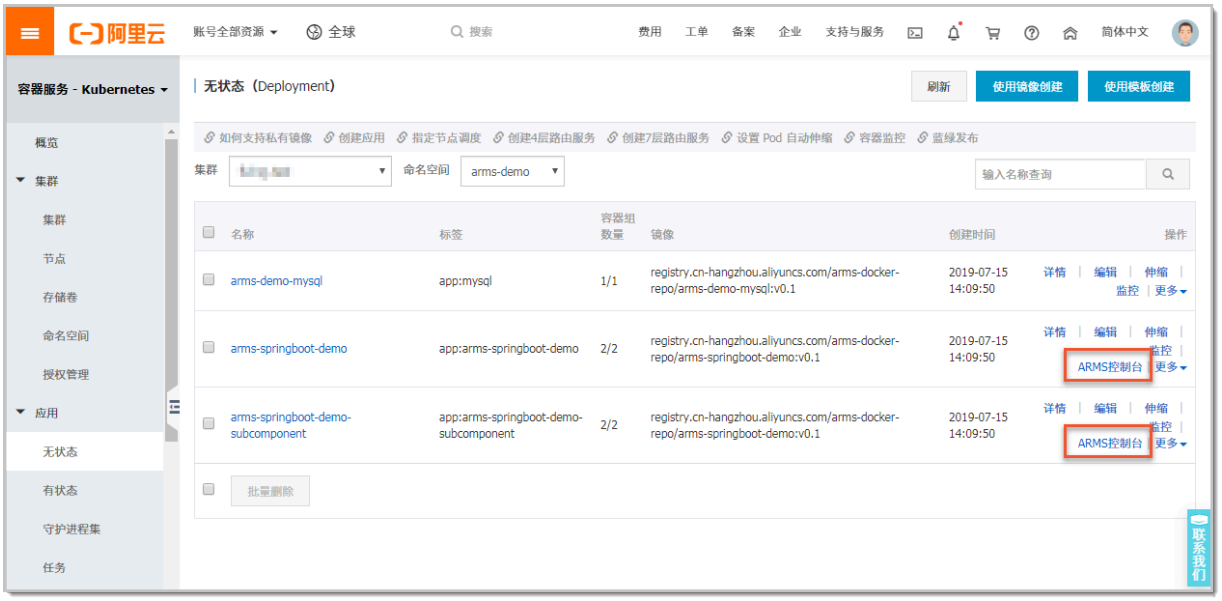
```
volumeMounts:
        - name: php-ini
          mountPath: /etc/php/7.2/fpm/php.ini
          subPath: php.ini
```

```
volumes:
      - name: php-ini
        configMap:
```

```
        name: arms-php.ini
```

预期结果

在无状态（Deployment）或有状态（StatefulSet）页面上，目标应用的操作列将出现 ARMS 控制台按钮。



说明:

若操作列没有出现 ARMS 控制台按钮，请检查您是否已授权容器服务访问 ARMS 资源。

#unique_15
#unique_16
#unique_32

4 控制台功能

4.1 应用总览

当应用成功接入 ARMS 后，ARMS将全方位监控您的应用。您可以在应用总览页面快速查看应用的健康状况关键指标，通过应用拓扑图预览应用的上下游依赖组件，通过 3D 拓扑图查看应用、服务和主机的健康状况。

功能入口

1. 登录 ARMS 控制台，在左侧导航栏中选择应用监控 > 应用列表 。
2. 在应用列表中选择您想查看的应用，进入应用总览页面。

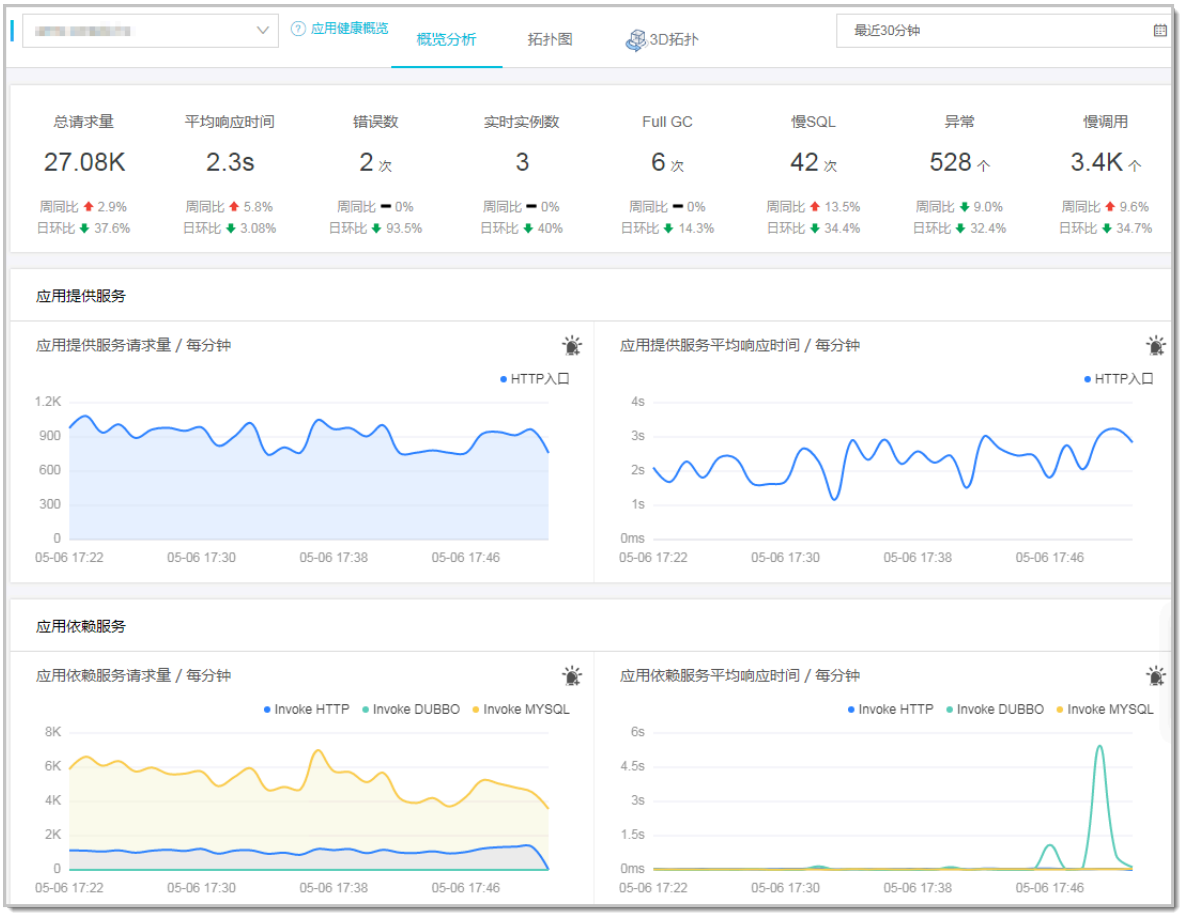
   您可以在应用总览页顶部选择概览分析、拓扑图和 3D 拓扑页签查看相应信息。

功能介绍

- 应用关键指标

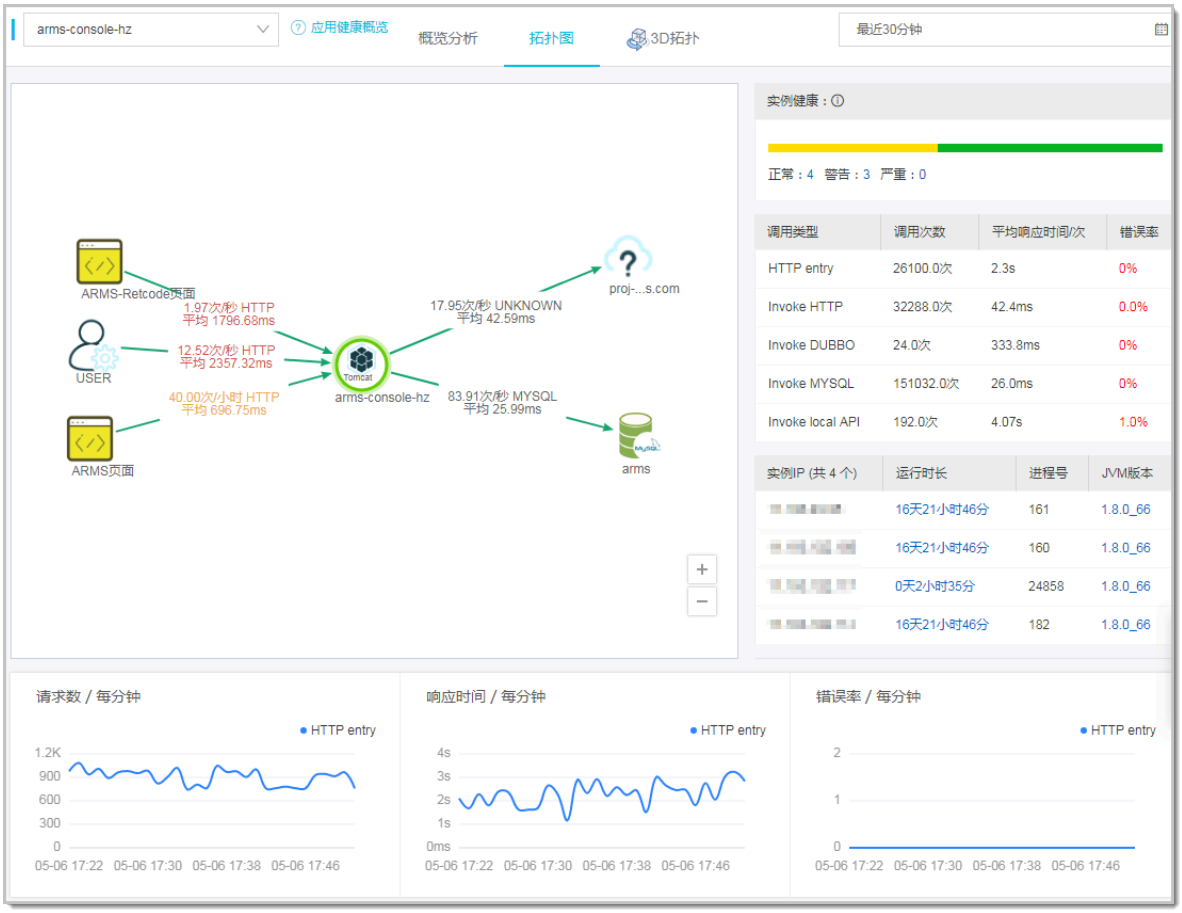
  概览分析页签上展示以下关键指标：

  - 选定时间内的总请求量、平均响应时间、错误数、实时实例数、FullGC 次数、慢 SQL 次数、异常次数和慢调用次数，以及这些指标和上一天的环比、上周的同比升降幅度。
  - 应用提供服务：应用提供服务的请求量和平均响应时间的时序曲线。
  - 应用依赖服务：应用依赖服务的请求量、平均响应和应用实例数的时序曲线，以及 HTTP-状态码统计。
  - 系统信息：CPU、MEM 和负载的时序曲线。
  - 慢调用：慢调用的时序曲线和调用详情。
  - 统计分析：接口慢调用分析和异常类型分析。



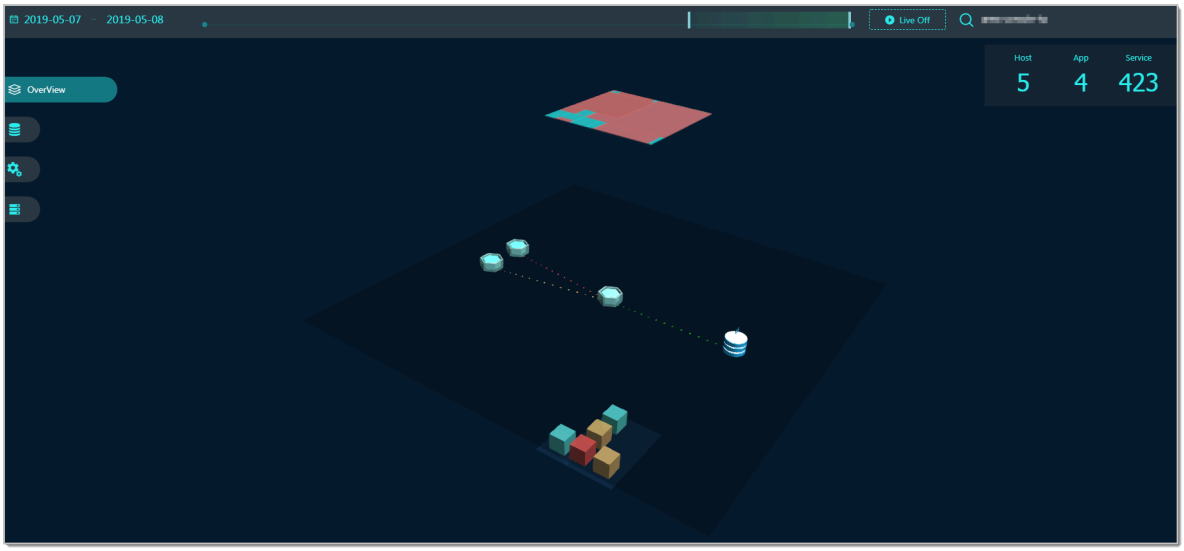
- 应用拓扑

  在拓扑图页签上，您可以通过拓扑图更加直观地看到应用的上下游组件以及与它们的调用关系，从而更快速地找出应用的瓶颈。



- 3D 拓扑

  在 3D 拓扑页签上，立体地展示了应用、服务和主机的健康状况，以及应用的上下游依赖关系。借助 3D 拓扑图，您可以快速定位诱发故障的服务、被故障影响的应用和关联的主机等，全方位地诊断故障根源，从而快速排除故障。3D 拓扑详细介绍请参见#unique_35。



4.2 应用详情

4.2.1 JVM 监控

ARMS 应用监控提供 JVM 监控功能，用于监控堆内存指标、非堆内存指标、直接缓冲区指标、内存映射缓冲区指标、GC（垃圾收集）累计详情和 JVM 线程数等 JVM 指标。本文将介绍 JVM 监控功能和查看 JVM 监控指标的操作步骤。

功能介绍

ARMS 的 JVM 监控功能可以帮助您监控以下指标。

- 堆内存
  - heap_init：堆内存初始字节数
  - heap_max：堆内存最大字节数
  - heap_commited：堆内存提交字节数
  - heap_used：堆内存使用字节数

- 非堆内存
  - non_heap_init：非堆内存初始字节数
  - non_heap_max：非堆内存最大字节数
  - non_heap_commited：非堆内存提交字节数
  - non_heap_used：非堆内存使用字节数
- 直接缓冲区
  - direct_capacity：直接缓冲区总大小（字节）
  - direct_used：直接缓冲区已使用大小（字节）
- 内存映射缓冲区
  - mapped_capacity：内存映射缓冲区总大小（字节）
  - mapped_used：内存映射缓冲区已使用大小（字节）
- GC（垃圾收集）累计详情
  - GcPsMarkSweepCount：垃圾收集 PS MarkSweep 数量
  - GcPsScavengeCount：垃圾收集 PS Scavenge 数量
  - GcPsMarkSweepTime：垃圾收集 PS MarkSweep 时间
  - GcPsScavengeTime：垃圾收集 PS Scavenge 时间
- JVM 线程数
  - ThreadCount：线程总数量
  - ThreadDeadLockCount：死锁线程数量
  - ThreadNewCount：新建线程数量
  - ThreadBlockedCount：阻塞线程数量
  - ThreadRunnableCount：可运行线程数量
  - ThreadTerminatedCount：终结线程数量
  - ThreadTimedWaitCount：限时等待线程数量
  - ThreadWaitCount：等待中线程数量

操作步骤

1. 登录 ARMS 控制台，在左侧导航栏中选择应用监控 > 应用列表 。
2. 在应用列表页面选择您想查看的应用。

3. 在应用详情页面选择您想查看的节点，并在页面右侧单击 JVM 监控页签。

   JVM 监控页签内展示了 GC 瞬时次数、GC 瞬时耗时、堆内存详情、非堆内存详情和 JVM 线程数的时序曲线。

   - 单击 GC 瞬时次数/每分钟和 GC 瞬时耗时的瞬时值/每分钟面板右上角的瞬时值和累计值按钮，可以切换查看 GC 瞬时次数和 GC 瞬时耗时的瞬时值或累计值的时序曲线，默认为瞬时值。
   - 单击各监控面板上的指标名称（例如 GC 瞬时次数），可打开或关闭该指标在图表中的可见性。

   > **说明:**
   > 每个图表必须至少有一个指标设为可见。

4.2.2 主机监控

ARMS 应用监控提供主机监控功能，用于监控 CPU、MEM（内存）、Disk（磁盘）、Load（负载）、网络流量和网络数据包的各项指标。本文将介绍主机监控功能和查看主机监控指标的操作步骤。

功能介绍

主机监控功能可以帮助您监控以下指标。

- CPU
  - SystemCpuIdle：最近 5 秒的空闲 CPU 使用率
  - SystemCpuSystem：最近 5 秒的系统 CPU 使用率
  - SystemCpuUser：最近 5 秒的用户 CPU 使用率
  - SystemCpuIOWait：最近 5 秒等待 IO 完成的 CPU 使用率
- MEM（内存）
  - SystemMemFree：系统空闲内存（单位 kb）
  - SystemMemTotal：系统总内存（单位 kb）
  - SystemMemUsed：系统已使用内存（单位 kb）
  - SystemMemBuffers: 当前系统缓冲区缓存中的内存（单位 kb）
  - SystemMemCached: 当前系统页面缓存中的内存（单位 kb）
  - SystemMemSwapFree：系统 swap 空闲内存（单位 kb）
  - SystemMemSwapTotal：系统 swap 总内存（单位 kb）
- Disk（磁盘）
  - SystemDiskTotal：系统磁盘总字节数
  - SystemDiskFree：系统磁盘空闲字节数
  - SystemDiskUsedRatio：系统磁盘使用率
- Load（负载）
  - SystemLoad：系统负载
- 网络流量
  - SystemNetInBytes：最近 30 秒平均每秒网络接收的字节数
  - SystemNetInErrs：最近 30 秒平均每秒网络接收的错误数
  - SystemNetOutBytes：最近 30 秒平均每秒网络发送的字节数
  - SystemNetOutErrs：最近 30 秒平均每秒网络发送的错误数
- 网络数据包
  - SystemNetInPackets：最近 30 秒平均每秒网络接收的报文数
  - SystemNetOutPackets：最近 30 秒平均每秒网络发送的报文数

操作步骤

1. 登录 ARMS 控制台，在左侧导航栏中选择应用监控 > 应用列表 。
2. 在应用列表页面选择您想查看的应用。

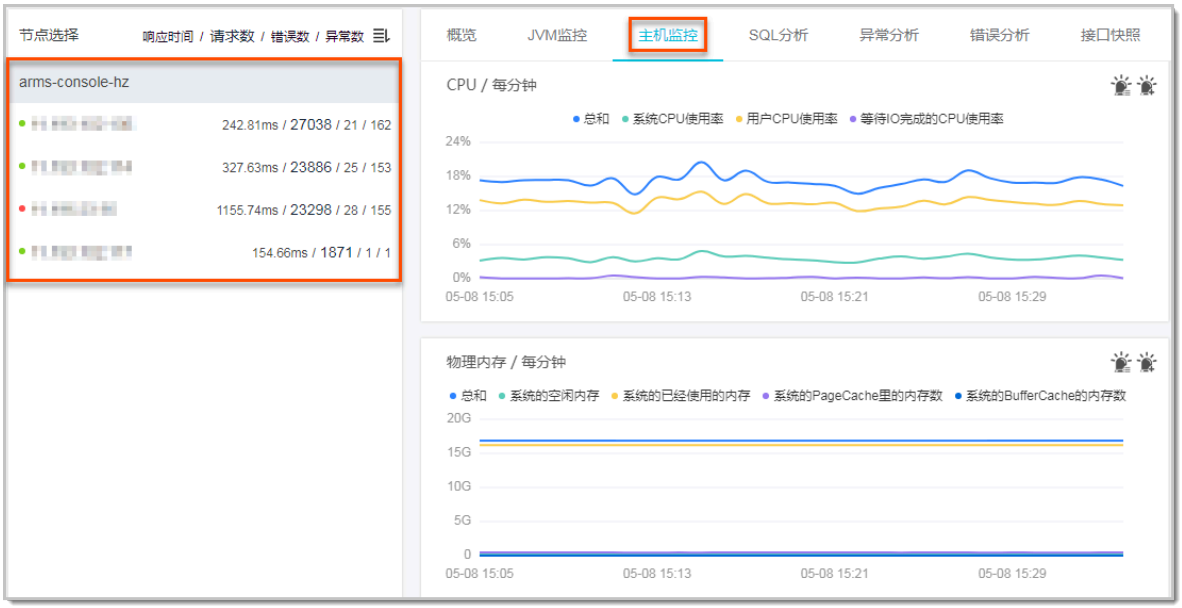
3. 在应用详情页面选择您想查看的节点，并在页面右侧单击主机监控页签。

   主机监控页签内展示了 CPU、MEM（内存）、Disk（磁盘）、Load（负载）、网络流量和网络数据包的时序曲线。

   - 单击各监控面板上的指标名称（例如系统 CPU 使用率），可打开或关闭该指标在图表中的可见性。

     > **说明:**
     > 每个图表必须至少有一个指标设为可见。

   - 单击监控面板右上角的两个报警图标，可以查看已有报警的报警点和创建新的报警。创建报警请参见#unique_39和#unique_40。



4.2.3 内存快照

ARMS 应用监控提供内存快照功能。创建内存快照后，您可以通过详细日志来查看指定时间段内多项内存指标的具体信息。本文将介绍内存快照的使用场景及使用方法。

应用场景

借助 ARMS 的 JVM 监控，您可以直观地看到指定时间段内的多项内存指标。虽然图表能体现出内存使用量过大的情况，但无法显示具体信息，因此不能帮助您排查问题的原因。此时您可以创建内存快照，通过详细的日志查看内存占用的详细信息。

操作步骤

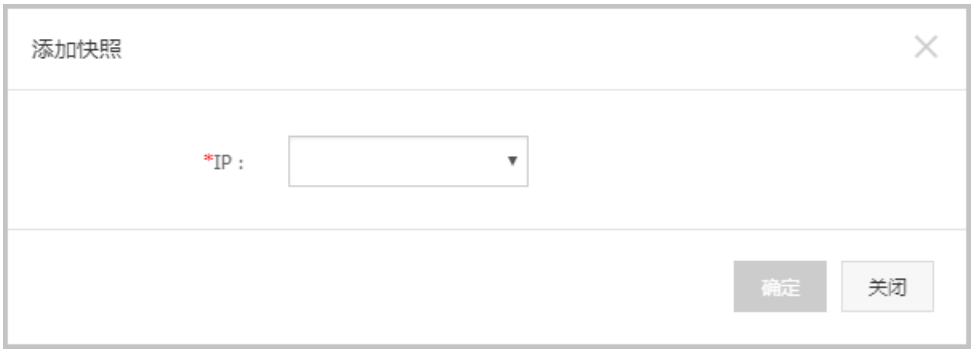
1. 登录 ARMS 控制台，在左侧导航栏中选择应用监控 > 应用列表 。
2. 在应用列表页面选择您想查看的应用。

3. 在应用详情页面选择您想查看的节点，并在页面右侧单击 JVM 监控页签。

4. 在JVM 监控页签右上角，单击创建内存快照。

> **说明:**
> 如果单击创建内存快照时，上一个快照任务仍在运行，则系统会弹出错误消息。请您耐心等待上一个快照任务运行完毕。目前仅支持为 Linux 系统新建内存快照。

5. 在创建内存快照对话框中选择一个 IP，并单击确定。



> **说明:**
> 如果在应用下的机器层面上新建快照，则 IP 字段会默认选中该机器的 IP 地址。

> **注意:**

快照任务的运行时间从几分钟到半小时不等。快照任务运行期间，应用可能会出现短暂的卡顿现象，请谨慎使用。

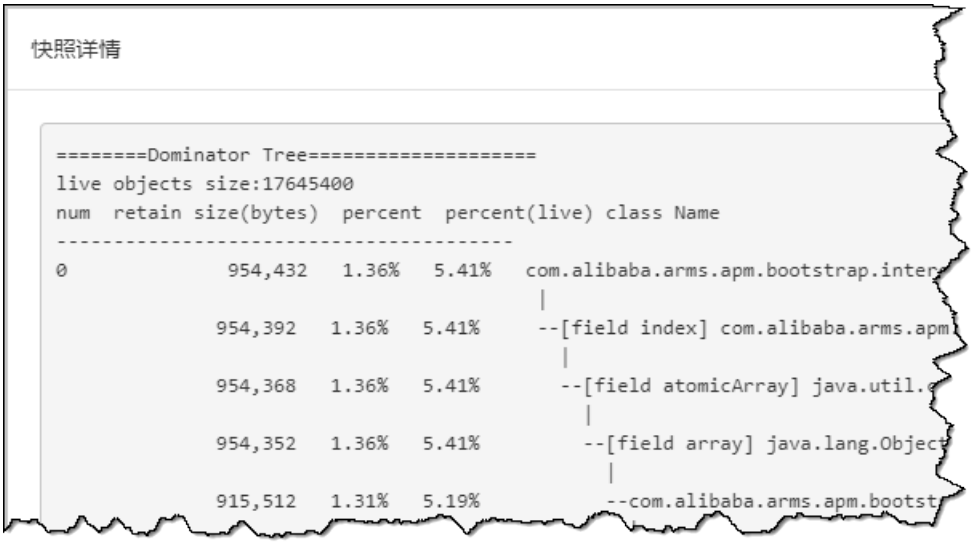
创建成功后快照会显示在历史快照的快照任务列表中。

每个快照任务的信息依次为：

- **IP**
- **创建快照的时间点**
- **快照任务的运行时间**
- **删除：用于删除快照**
- **查看详情：用于查看内存快照的详细信息**

绿色表示快照任务执行成功，红色表示快照任务执行失败。

6. 单击查看详情，即可打开快照详情对话框，查看内存快照的详细信息。



- Dominator Tree：按从大到小的顺序列出占用内存大小前 5 位的对象。

  上下行的缩进表示对象之间的支配关系。如果排名第一的对象占比例较小或支配的内存较小，则表示没有大内存对象。否则，需要修改该内存对象，减少其大小或者将其快速释放。
- Histogram：按从大到小的顺序列出占用内存大小前 20 位的类。

4.3 应用接口调用监控

本文介绍了应用监控中的接口调用监控功能。

功能介绍

在应用监控的接口调用页面上，您可以查看该应用下的调用接口的调用详情。ARMS 可自动发现与监控以下 Web 框架和 RPC 框架中提供的接口：

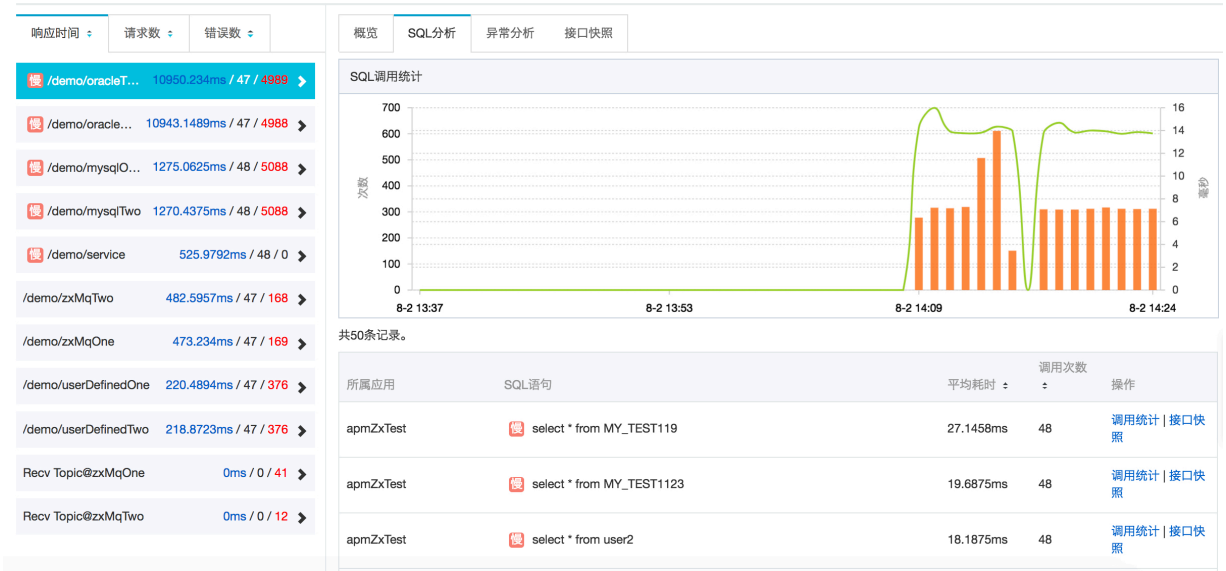
- Tomcat 7+
- Jetty 8+
- Resin 3.0+
- Undertow 1.3+
- WebLogic 11.0+
- SpringBoot 1.3.0+
- HSF 2.0+
- Dubbo 2.5+

接口概览

接口调用页面的概览标签页列出了被 ARMS 探针自动发现的所有接口。您可以按照响应时间、请求数或错误数对该列表排序。选中一个服务，即可在概览标签页上查看该服务的详细调用拓扑，以及请求数、耗时、错误数的时序曲线。

SQL 分析

SQL 分析标签页展示的是左侧选中服务的代码段内所发起的 SQL 请求列表。借助此标签页，您可以找出是哪一个 SQL 造成某个服务过慢。您还可以单击某个 SQL 中的接口快照来查看一个 SQL 执行逻辑所处的完整代码链路。



异常分析

异常分析标签页展示的是左侧选中服务的代码段内所抛出的 Java 异常。您还可以单击某个异常中的接口快照来查看一个异常堆栈所处的完整代码链路。

接口快照

在服务链路快照中，您可以看到该服务接口中单次调用的调用堆栈、执行的明细 SQL、抛出的具体异常信息，以及接口中的参数详情。

4.4 MQ 监控

ARMS 应用监控的 MQ 监控可展示消息队列 RocketMQ 的 Topic 发布和订阅消息的情况。

功能入口

请按照以下步骤进入 ARMS 应用监控的 MQ 监控页面。

1. 登录ARMS 控制台，在左侧导航栏中选择应用监控 > 应用列表 。
2. 在应用列表页面，单击目标应用的名称。
3. 在左侧导航栏中单击 MQ 监控。
4. 在页面右侧单击查询结果的链接。

完成以上步骤后，即可进入 MQ 监控页面的概览页签。

功能介绍

MQ 监控页面具备以下功能：

- 在拓扑图中展示应用与 MQ 数据源之间的消息发布和订阅关系。
- 展示消息发布的统计数据，包括请求数、响应时间和错误数。
- 展示消息订阅的统计数据，包括消息请求数、响应时间和错误数。
- 提供关于消息发布和订阅的接口快照，您可以通过 TraceId 链接查看完整调用链以及诊断问题原因。

可用操作

- 在页面右上角的时间选择框内选择需要查看统计数据的起止时间。
- 单击发布端统计和订阅端统计页签，查看消息发布和订阅的统计数据。
- 单击接口快照页签，查看关于消息发布和订阅的接口快照，必要时可通过 TraceId 链接查看完整调用链以及诊断问题原因。
- 单击返回总览，回到 MQ 监控页面。

4.5 调用链路查询

在调用链路查询页面，您可以通过 TraceId 精确查询调用链路详细情况，或结合多种条件筛选查询调用链路。

TraceId：每条调用链路的唯一标识 ID，可用于精确查询调用信息。

调用链路：支持分布式调用链路，及本地调用方法堆栈查看服务间调用链路、服务本地调用链路。

- 分布式调用链路（服务间调用链路）：服务与服务间的调用链路。
- 本地调用链路（方法栈）：一次服务间调用链路中的本地方法栈。

采用以下方法之一查询调用链路。

- 精确查询：在参数名下拉列表中选择 TraceId，在参数值输入框中输入具体的 TraceId，单击查询。
- 高级查询：结合以下多种条件查询调用链路。

功能入口

请按照以下步骤进入调用链路查询页面。

1. 登录 ARMS 控制台，并在页面左上角选择所需地域。
2. 在左侧导航栏中选择应用监控 > 调用链路查询。

查询调用链路

要查询调用链路，您可以通过指定具体的 TraceId 进行精确查询，或结合多个条件进行筛选。

表 4-1: 查询字段

查询字段	描述
调用类型	· HTTP 入口：客户端使用 HTTP 协议调用 · 提供 Dubbo：客户端通过 Dubbo 方式调用 · 提供 HSF：客户端通过 HSF 方式调用
耗时大于	调用的耗时大于指定毫秒数
仅显示异常调用	勾选即可筛选出抛异常的调用
客户端名/客户端 IP	调用发起应用的名称、IP
服务端名/服务端 IP	请求被调用的应用的名称、IP
接口名称	应用调用的接口名称，支持前缀模糊匹配，例如 /api/ResourceQuery 可搜索 api、Resource 等分词

查看服务间调用链路（分布式调用链路）

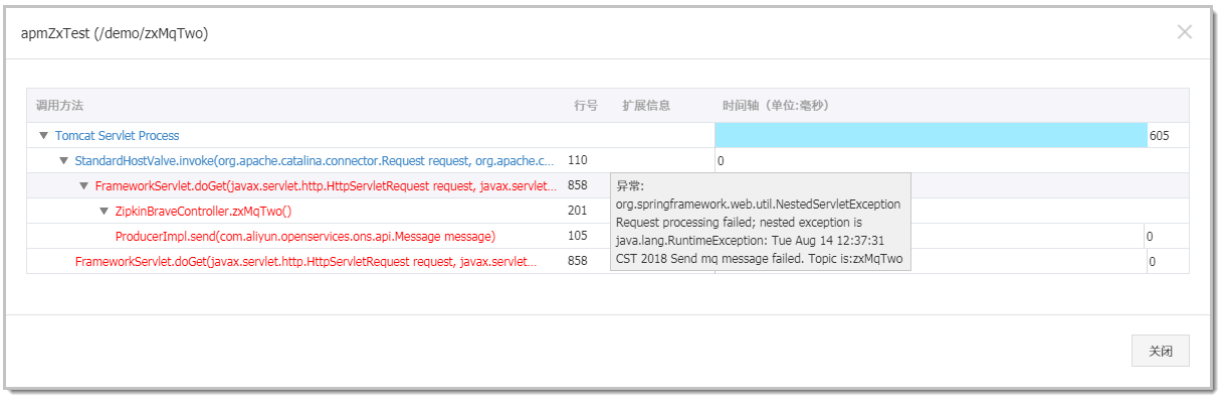
单击需要查看的 TraceID 名称，进入调用链路页面。

字段说明

- 状态：红色表示该服务调用的本地调用链路中存在异常，绿色表示正常。
- IP 地址：该应用的 IP 地址。
- 调用类型：该次调用的调用类型，与即席查询的调用类型选项对应。
- 时间轴：各服务间调用链路的耗时，以及相对于整条调用链路的耗时分布。

查看服务本地调用链路（方法栈）

在调用链路页面，单击方法栈列的放大镜按钮，进入查看本地调用链路（方法栈）页面。



字段说明

- 调用方法：本地方法栈调用方法，展开后显示的是该方法的下一层调用。
- 行号：本地方法的代码所在行数。
- 扩展信息：
  - 参数：调用的输入参数等
  - SQL：数据库调用的 SQL 语句等
  - 异常：抛错的信息等
- 时间轴：本地调用链路每次方法调用的时间分布。

4.6 应用监控 3D 拓扑图

ARMS 应用监控 3D 拓扑图能立体展示应用、服务和主机的健康状况，以及应用的上下游依赖关系。借助 3D 拓扑图，您可以快速定位诱发故障的服务、被故障影响的应用和关联的主机等，全方位地诊断故障根源，从而快速排除故障。

快速入门

功能入口

请按照以下步骤进入 ARMS 应用监控 3D 拓扑图。

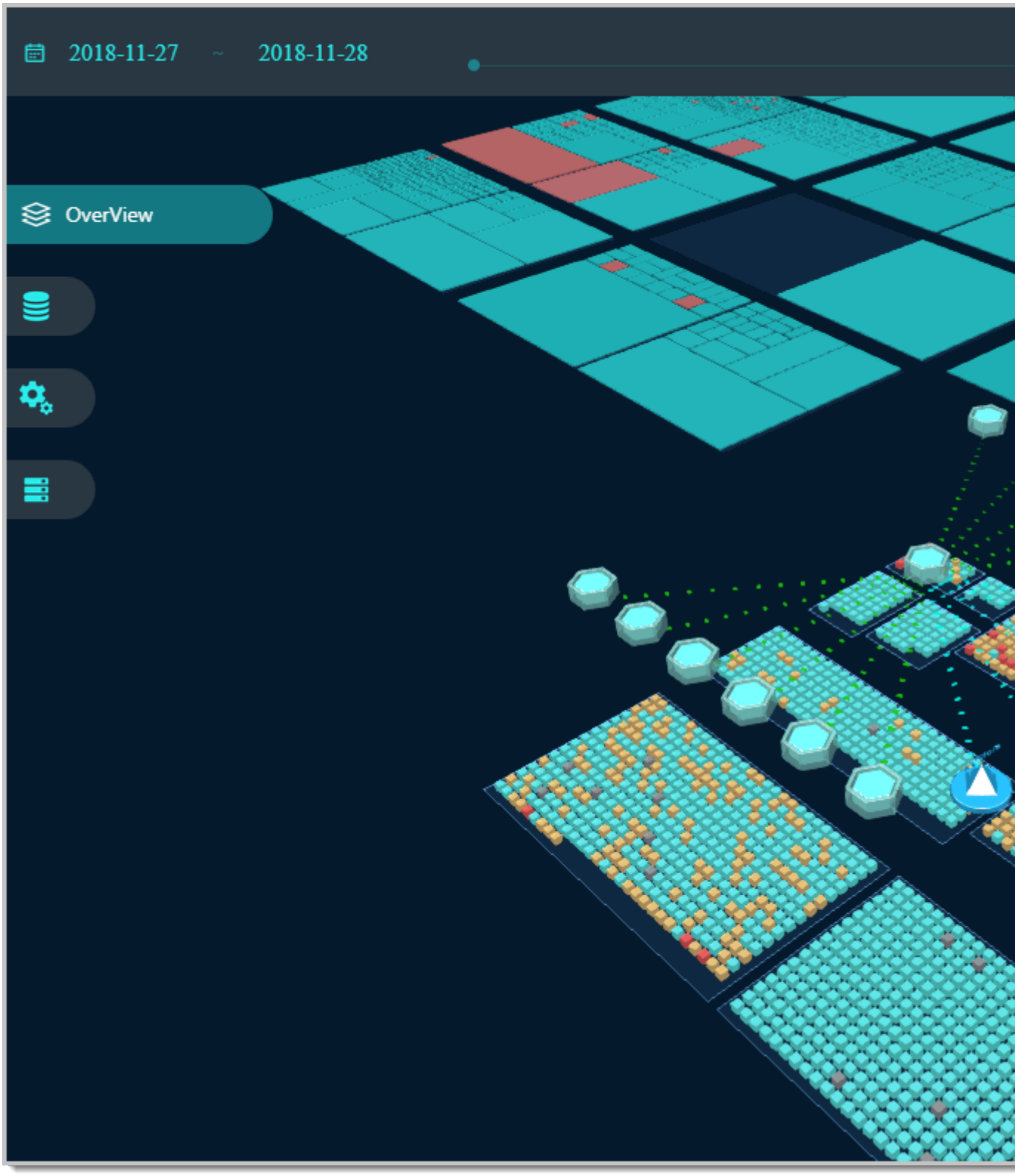
1. 登录 ARMS 控制台。
2. 在左侧导航栏中选择应用监控 > 应用列表。
3. 在应用列表页面上按需执行以下操作。
   - 如需查看全部应用的 3D 拓扑，则单击顶部的 3D 拓扑查看全部应用（试用）。
   - 如需查看单个应用的 3D 拓扑，则单击操作列中的 3D 拓扑。

> **说明:**
>
> 应用列表页面默认为列表视图，如果已经切换为卡片视图，则单击应用卡片右上角的 3D 拓扑图标。

总览层（Overview）

在默认展示的 Overview 页面上，您可以看到服务层、应用层和主机层的全部内容。页面右上角显示的是主机、应用和服务的数量。

图 4-1: ARMS 3D 拓扑图 Overview 页面



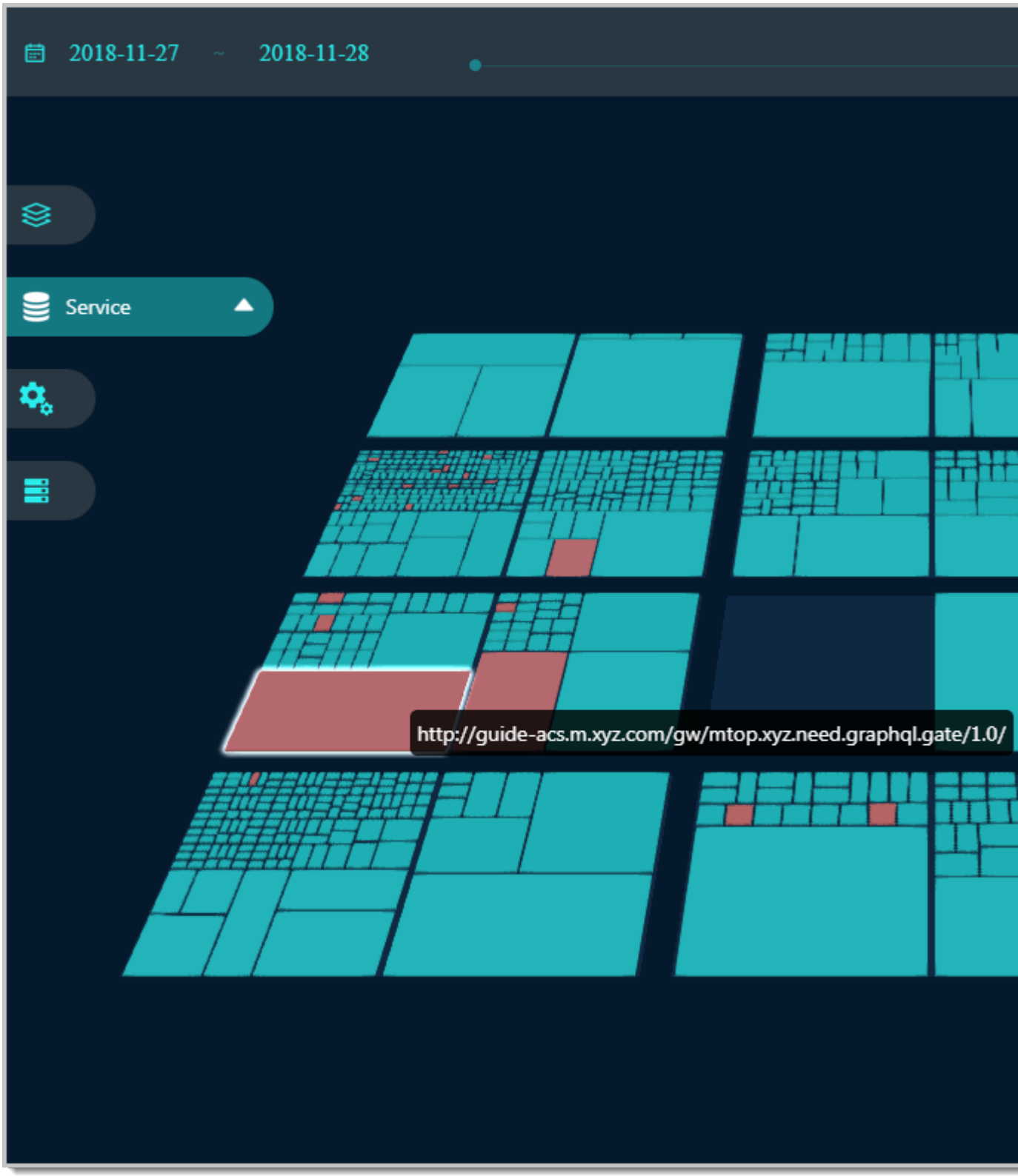
在总览层，您可以执行以下操作：

- 在页面左上角单击时间范围区域，然后在弹出的时间范围选择器内选择精确的起止时间。
- 在页面顶部的时间轴上，随意拖动时间滑块来改变当前视图对应的时间范围。
- 在页面右上角的搜索框内，输入关键字并按回车进行搜索。
- 用鼠标拖放，以任意角度查看三层数据。
- 单击视图中的任意对象，在右侧面板中查看该对象的相关指标。

服务层（Service）

服务层展示应用所依赖的服务信息。

图 4-2: ARMS 3D 拓扑图服务层



每个应用下的服务对应一个板块。调用次数越多，所占面积越大。服务的不同状态以不同颜色表示。

- ：服务调用正常
- ：服务出错率较高
- ：服务无返回数据

说明:

服务的响应时长阈值是可以配置的。在左侧导航栏中单击服务层（Service）右侧的三角形图标，即可展开耗时阈值设置框。在该设置框中拖动滑块即可设置阈值。

单击一个服务后，右侧面板将展示该服务的以下信息：

- 服务名称
- QPS：Query Per Second（每秒查询数）
- RT(ms)：Response Time（响应时间，单位为毫秒）
- ErrQps：Error QPS（每秒错误查询数）

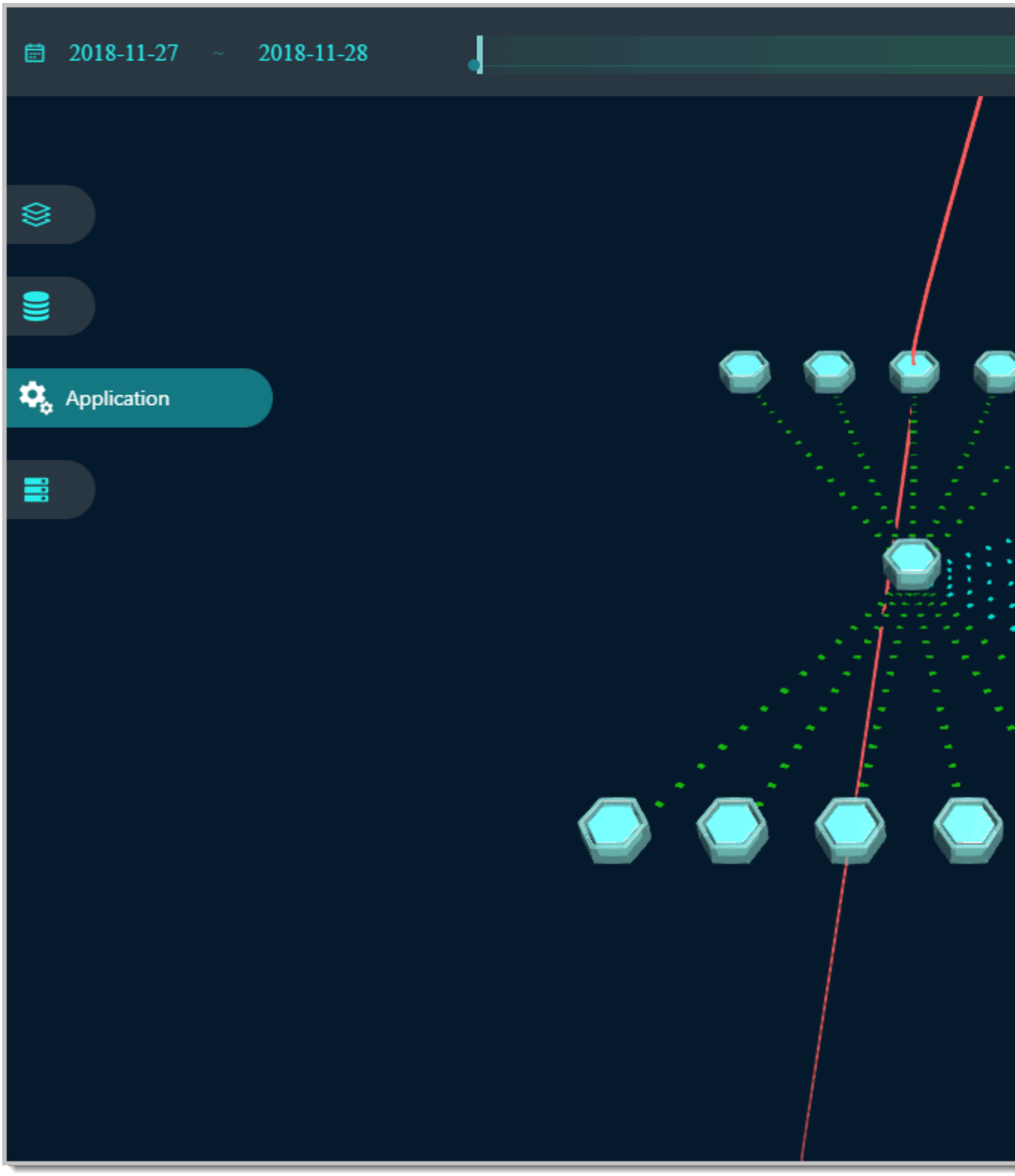
说明:

在 QPS、RT 和 ErrQps 区域框中，左侧的数值是所选时间范围内的平均值，右侧是所选时间范围内的曲线图。

应用层（Application）

应用层展示应用及其上下游依赖关系，包括依赖的中间件在内。通过连接线的流向，您可以看到调用方向。

图 4-3: ARMS 3D 拓扑图应用层



单击一个应用后，右侧面板将展示该应用的以下信息：

- 应用名称
- QPS：Query Per Second（每秒查询数）
- RT(ms)：Response Time（响应时间，单位为毫秒）
- ErrQps：Error QPS（每秒错误查询数）

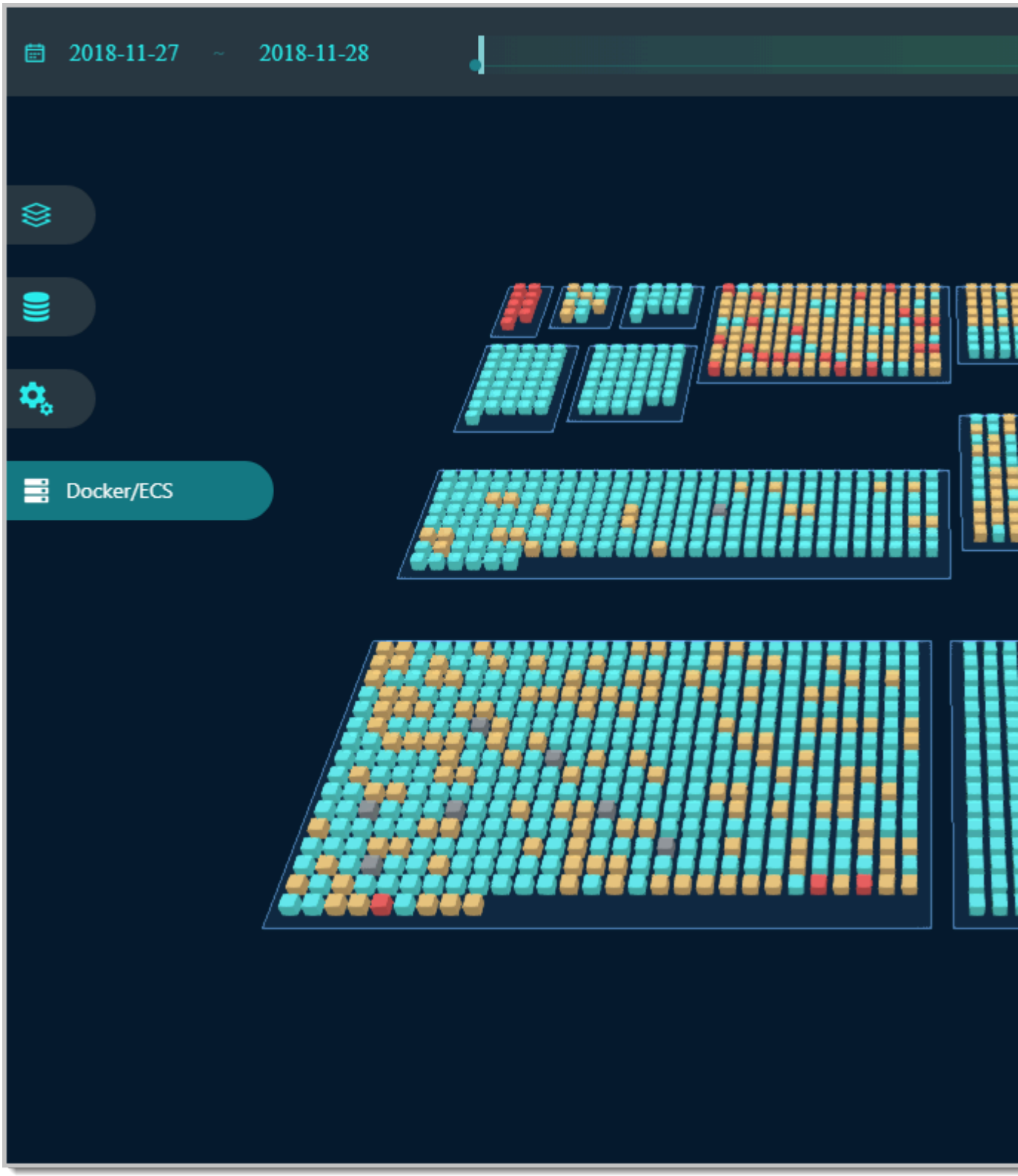
说明:

在 QPS、RT 和 ErrQps 区域框中，左侧的数值是所选时间范围内的平均值，右侧是所选时间范围内的曲线图。

主机层（Docker/ECS）

主机层展示应用的主机详情。

图 4-4: ARMS 3D 拓扑图主机层



每个方块代表一个主机。所有主机按应用分区。主机的不同状态以不同颜色表示。

- ：正常
- ：缓慢
- ：警告
- ：异常
- ：离线

单击一个主机后，右侧面板将展示该主机的以下信息：

- 主机 IP 地址及基础信息：
  - IDC：数据中心
  - Unit：单元
  - Host：主机
  - CPU：核数
  - JVM：JVM 版本
  - Tomcat：Tomcat 版本
- CPU：CPU 使用率
- MEM：内存使用率
- LOAD：负载

说明：

在 CPU、MEM 和 LOAD 区域框中，左侧的数值是所选时间范围内的平均值，右侧是所选时间范围内的曲线图。

4.7 ### 实时诊断

ARMS 应用监控的实时诊断功能适用于在短时间内密切监控应用性能和定位问题原因的场景。本文介绍实时诊断功能的使用方法。

背景信息

当您需要密切监控一小段时间内的应用性能时，例如发布应用或者对应用进行压测时，可以使用 ARMS 应用监控的实时诊断功能。开启实时诊断后，ARMS 应用监控会持续监控应用 5 分钟，并在这 5 分钟内全量上报调用链数据。接下来，您就能以出现性能问题的调用链路为起点，通过方法栈瀑布图和线程剖析等功能定位问题原因。

快速入门

功能入口

请按照以下步骤进入 ARMS 应用监控实时诊断。

1. 登录 ARMS 控制台。
2. 在左侧导航栏中选择应用监控 > 应用列表。
3. 在应用列表页面单击目标应用的名称。
4. 在左侧导航栏中单击实时诊断。

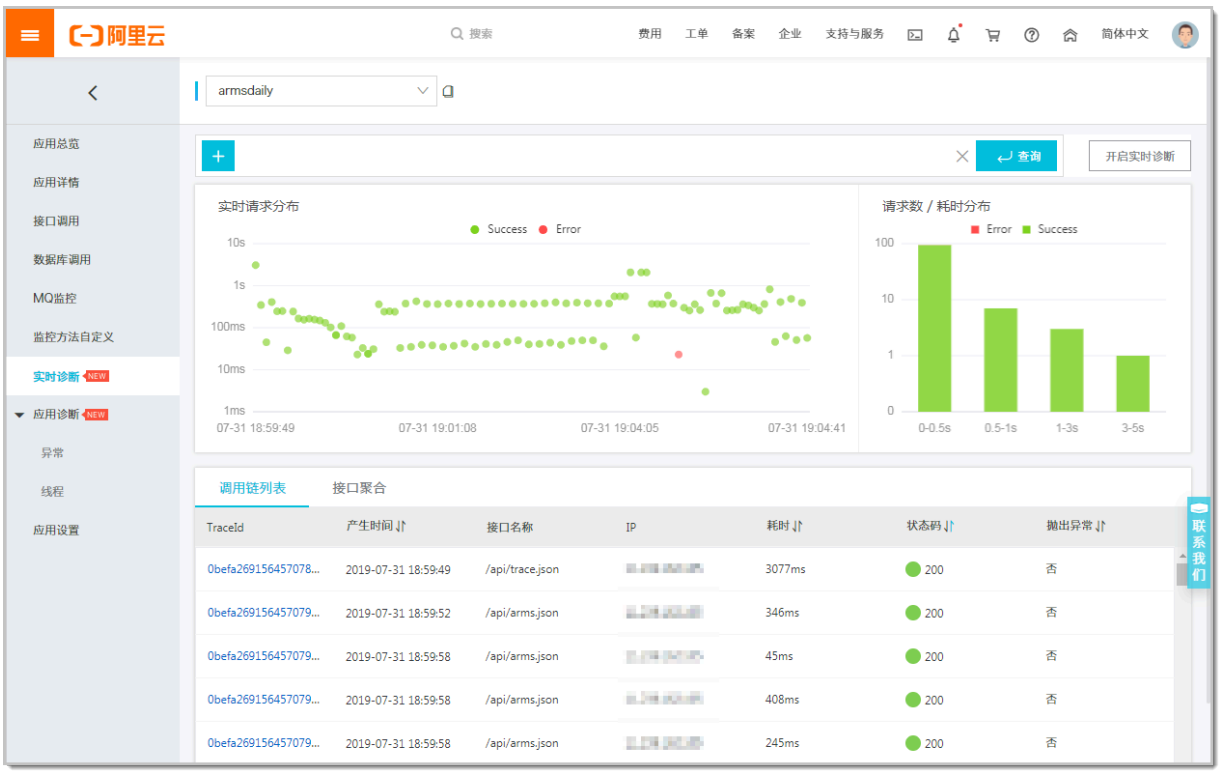
开启和终止实时诊断

首次进入实时诊断页面时，默认自动开启实时诊断。其他情况下，如需开启实时诊断，请单击页面右上角的开启实时诊断。

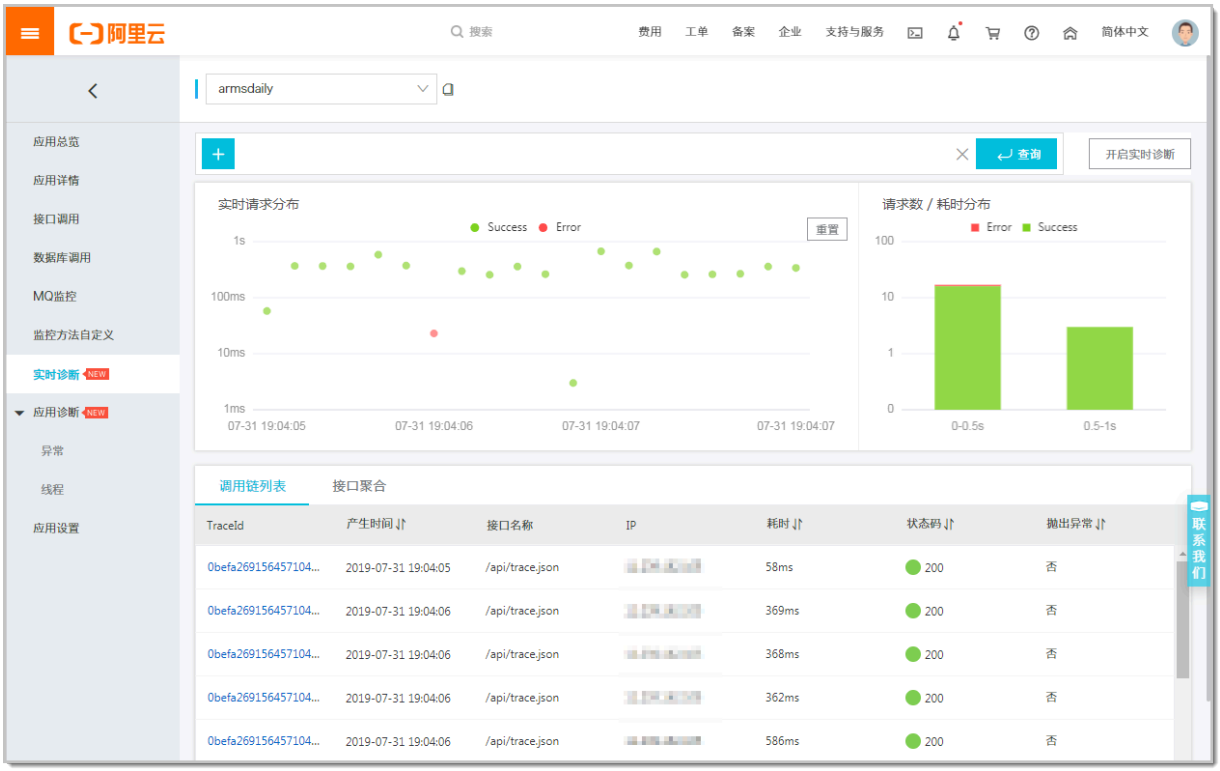
实时诊断将于自动开启 5 分钟后自动终止。如需提前终止实时诊断，请单击页面右上角的终止实时诊断。

查看实时监控数据

在实时请求分布和请求数/耗时分布区域，您可以查看截至当前时间点捕捉到的最后 1000 次请求统计数据。



在实时请求分布区域的图表中，用鼠标框选一段时间区间，即可将所选时间区间设为数据可视时间区间，即图表中仅显示该时间区间内的数据。此后，单击图表右上角的重置即可恢复为默认视图。

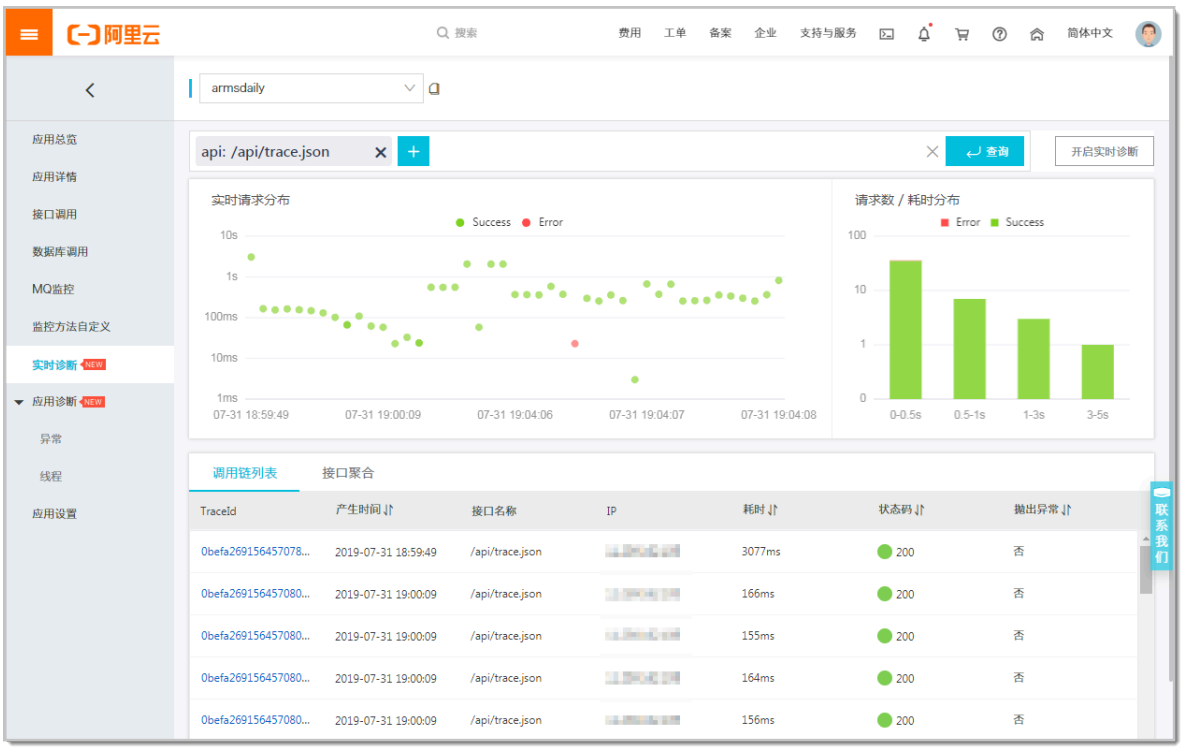


筛选监控数据

您可以按照接口名称和 IP 筛选页面上显示的请求监控数据。

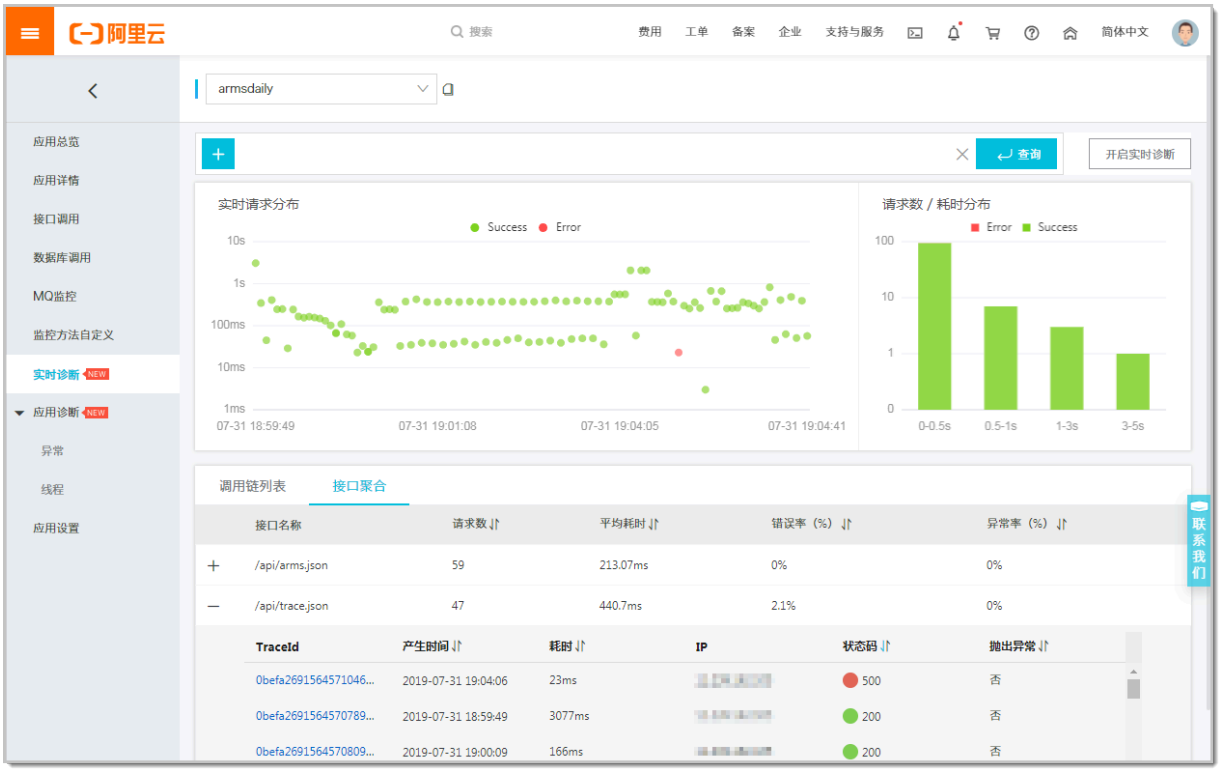
1. **在实时请求分布区域上方单击 + 图标。**
2. **在下拉框中选择一个 API 或 IP，并单击查询。**

   仅选中接口的请求监控数据会显示在页面上。



查看调用链信息

在调用链列表和接口聚合页签上，您可以查看相应时间区间内捕捉到的全部调用链信息。单击一个TraceId，即可进入调用链路页面，并借助本地方法栈瀑布图和线程剖析等功能定位问题原因，具体方法参见调用链路查询和使用线程剖析诊断代码层面的问题。



4.8 自定义配置

应用监控的一些常用设置，例如调用链采样率、Agent 开关、慢 SQL 阈值等，可直接在自定义配置页签上配置。

前提条件

#unique_50

功能入口

1. 登录 ARMS 控制台。
2. 在左侧导航栏中选择应用监控 > 应用列表。
3. 在应用列表页面单击目标应用的名称。
4. 在左侧导航栏中单击应用设置，并在右侧单击自定义配置页签。

> **说明:**
> 设置完毕后，在页面底部单击保存方可生效。

配置调用链采用设置

在调用链采样设置区域框，可以打开或关闭调用链采样，并设置采样率。采样率设置字段输入百分比的数字部分即可，例如输入 `10` 代表采样 10%。

> **注意:**
> 修改即时生效，无需重启应用。如果关闭采样，则调用链数据将不会被采集，请谨慎操作。

配置 Agent（探针）开关和日志级别

在 Agent 开关配置区域框，可以打开或关闭探针总开关以及各插件开关，并配置日志级别。

> **注意:**
> 探针总开关和日志级别的修改即时生效，无需重启应用。如果关闭探针总开关，则系统将无法监控您的应用，请谨慎操作。要使对各插件开关的修改生效，必须手动重启应用。

配置阈值设置

在阈值设置区域框，可以设置慢 SQL 查询阈值、接口响应时间阈值和限流阈值。

配置高级设置

在高级设置区域框，可以设置需过滤的接口、方法堆栈最大长度等。

- 无效接口调用过滤：输入不需要查看调用情况的接口，从而将其从接口调用页面隐去。
- 方法堆栈最大长度：默认为 128 条，最大值为 400 条。
- 采集 SQL 最大长度：默认为 1024 个字符，最小值为 256，最大值为 4096。
- 采集 SQL 绑定值：捕获 `PrepareStatement` 参数绑定的变量值，无需重启应用即可生效。
- 异常过滤：此处输入的异常不会显示在应用详情和异常分析页面的图表中。
- 错误数过滤：默认情况下，大于 400 的状态码会计入错误数，您可以自定义大于 400 但不计入的 HTTP 状态码。
- 启用调用链压缩：打开开关则压缩调用链，以减少占用的存储空间。
- 请求入参最大长度设置：默认为 512，最大值为 2048。
- 开启分位数：是否开启分位数统计功能。
- 开启应用紧急事件报警：支持针对线程死锁、OOM 等紧急事件的报警。探针版本须为 2.5.8+。

配置线程设置

在线程设置区域框，可以打开或关闭线程诊断方法栈开关、线程剖析总控开关，并设置慢调用监听触发阈值。

说明:

服务调用耗时超过慢调用监听触发阈值（默认值为 1000 毫秒）时才会启动监听，并一直持续到该次调用结束或超过 15 秒。建议将此阈值设为调用耗时的第 99 百分位数。假设有 100 次调用，则按耗时从小到大排序，排在第 99 位的耗时就是第 99 百分位数。

配置内存快照设置

在内存快照设置区域框，可以启用或停用内存快照功能。打开此开关后，出现内存泄漏时将自动转储内存（一天至多一次）。

配置 URL 收敛规则

在 URL 收敛设置区域框中，可以打开或关闭收敛功能的开关，并设置收敛阈值和收敛规则。URL 收敛是指将具有相似性的一系列 URL 作为一个单独的个体展示，例如将前半部分都为 /service/demo/ 的一系列 URL 集中展示。收敛阈值是指要进行 URL 收敛的最低数量条件，例如当阈值为 100 时，则符合规则正则表达式的 URL 达到 100 时才会对它们进行收敛。

调用链路查询

在调用链路查询页面，您可以通过 TraceId 精确查询调用链路详细情况，或结合多种条件筛选查询调用链路。

应用接口调用监控

本文介绍了应用监控中的接口调用监控功能。

#unique_52

内存快照

ARMS 应用监控提供内存快照功能。创建内存快照后，您可以通过详细日志来查看指定时间段内多项内存指标的具体信息。本文将介绍内存快照的使用场景及使用方法。

5 使用教程

5.1 使用全息排查诊断业务问题

全息排查用于通过业务主键快速定位问题链路，需要和自定义监控功能搭配使用。本文介绍了使用全息排查的方法。

前提条件

- 已成功创建一个应用监控任务，请参见#unique_23。
- 已在应用的 *pom* 文件中添加以下依赖：

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

Maven 仓库地址为：`https://oss.sonatype.org`。

> 说明：
> 如果无法获取 pom 依赖，请直接下载 arms-sdk-1.7.1.jar。

获取 TraceId 与 RpcId

满足上述前提条件后，即可通过以下代码获取 TraceId 与 RpcId：

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

打印日志

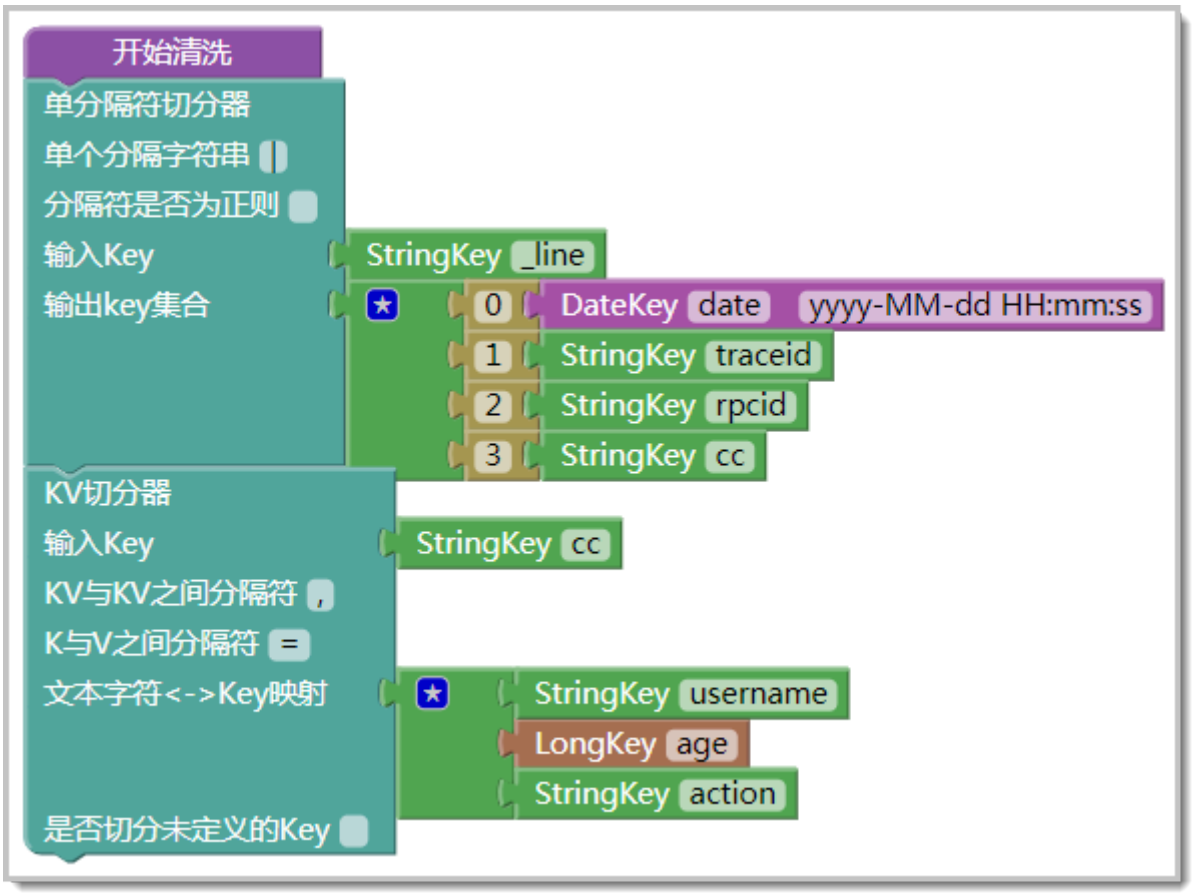
获取 TraceId 与 RpcId 后，您可以根据业务需求打印和输出业务日志。以下是包含 TraceId 与 RpcId 的样例业务日志。该日志输出到文件 */home/admin/logs/example/example.log* 中，但您也可以按需将其输出到日志服务 Log Service（简称 LOG，原 SLS）、MQ 等其他通道中。

```
2018-07-12 11:37:40|1e057c401531366659965100 5d1201|0|username=xiao,age
=22,action=login
2018-07-12 11:37:40|1e057c401531366659965100 5d1201|0|username=xiao,age
=22,action=search
2018-07-12 11:37:40|1e057c401531366659965100 5d1201|0|username=xiao,age
=22,action=cart
```

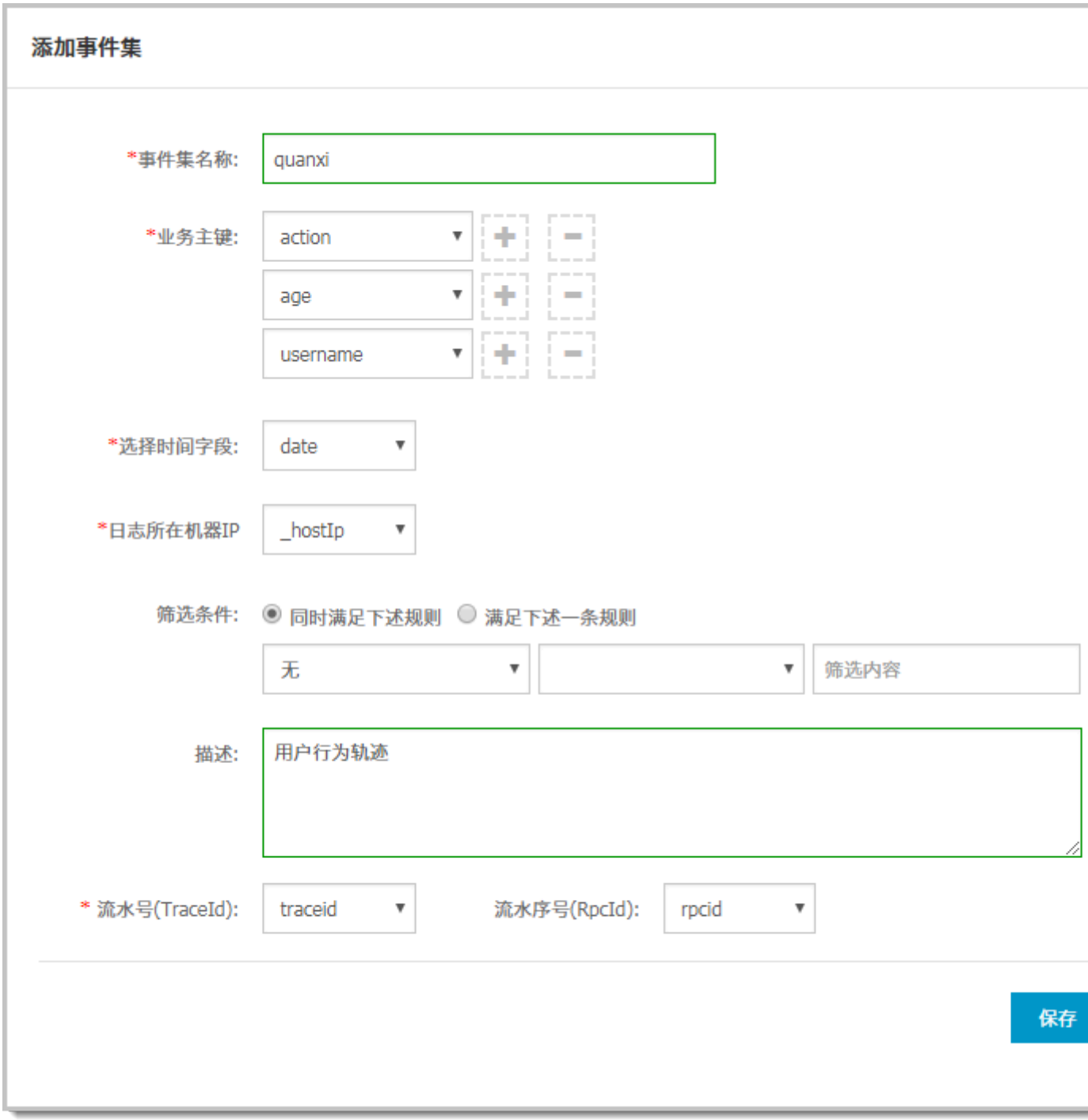
以上每条业务日志均表示用户的一条行为轨迹。

配置全息排查事件集

使用上方的样例业务日志作为数据源，参见#unique_55创建一个自定义监控任务，并按照下图方案自定义切分日志。



随后进入创建自定义监控任务的#unique_56步骤，并按照下图配置事件集。



- 业务主键：表示搜索业务事件所使用的字段。在本文的示例中，业务主键是行为（action）和用户名称（username）。
- 选择时间字段：必须选择业务时间，不能选择系统时间。
- 流水号：设置 TraceId。
- 流水序号：设置 RpcId。

配置好事件集后，请启动自定义监控任务。

全息排查功能的使用案例

本案例的业务日志表示用户的行为轨迹，对应的应用为购物网站。假设用户 kevin.yang 投诉称在 2018-07-12 14:20 以后下单失败，那么您可以通过以下两种方法来定位问题原因。

- 方法一：调用链路查询。
  1. 在左侧导航栏中单击应用监控 > 调用链路查询，进入实例列表页面的调用链路查询页签。
  2. 在标签页上的日期参数值中输入日期范围，在最下方的参数名下拉列表中选择业务主键，并在右侧的参数值中输入业务主键的值，例如本例中的 `username:kevin.yang`，然后单击查询。指定时间区间内的所有调用链路显示在搜索结果中。
  3. 在搜索结果中，单击异常调用链路的 TraceId，然后单击业务轨迹标签页，显示该 TraceId 下的所有业务事件，并根据业务事件定位问题原因。
- 方法二：全息排查事件查询。
  1. 在左侧导航栏中单击应用监控 > 调用链路查询，进入实例列表页面的调用链路查询页签。
  2. 在左侧导航栏中单击多维查询，并单击全息排查事件查询标签页。
  3. 在标签页上的日期参数值中输入日期范围，在全息排查事件集下拉框中选择前面配置的事件集，然后单击查询。指定时间区间内的所有调用链路显示在搜索结果中。
  4. 在搜索结果中，单击调用链查询，然后单击业务轨迹标签页，显示该 TraceId 下的所有业务事件，并根据业务事件定位问题原因。

更多信息

- 创建全息排查事件集

5.2 使用线程剖析诊断代码层面的问题

ARMS TProf 线程剖析是代码级的诊断工具，能够自动捕获慢调用的堆栈快照，真实还原代码执行的第一现场。

使用场景

其典型使用场景有：

- 当促销活动流量峰值出现慢调用时，ARMS TProf 可为您快速定位问题代码。
- 当系统出现大量慢调用时，ARMS TProf 可为您自动保存第一现场。
- 当业务太复杂，偶发性慢调用无法复现时，ARMS TProf 可为您还原代码真实执行轨迹。

使用方法

设置线程剖析参数

1. 登录 ARMS 控制台。
2. 在控制台左侧导航栏中选择应用监控 > 应用列表 。
3. 在应用监控列表页面，单击目标应用的名称。
4. 在左侧导航栏中单击应用设置，并在页面右侧单击自定义配置标签页。
5. 在线程剖析设置区域，可以打开或关闭线程剖析总控开关，并设置慢调用触发阈值。

> 说明:
>
> - 服务调用耗时超过该阈值（默认值为 1000 毫秒）时才会启动监听，并一直持续到该次调用结束或超过 15 秒。
> - 建议将此阈值设为调用耗时的第 99 百分位数。假设有 100 次调用，则按耗时从小到大排序，排在第 99 位的耗时就是第 99 百分位数。

通过接口快照查看线程剖析详情

1. 在控制台左侧导航栏中选择应用监控 > 应用列表 。
2. 在应用监控列表页面，单击目标应用的名称。
3. 在左侧导航栏中单击接口调用，并在页面右侧单击接口快照标签页。
4. 在接口快照标签页上单击一个 TraceId 链接。调用链路标签页在新窗口打开。

5. 在线程剖析栏中单击放大镜图标。线程剖析对话框打开。

说明:

- 实际耗时是服务调用的实际执行时间，不受线程剖析影响。
- 监听耗时是能够被 TProf 监听到的耗时。通常情况下，监听耗时 ≈ 实际耗时 - 慢调用触发阈值。

通过多维查询查看线程剖析详情

1. 在控制台左侧导航栏中选择应用监控 > 多维查询 。

2. 在调用链路查询标签页的参数名下拉框中选择仅含线程剖析快照，并单击查询。

3. 在搜索结果中单击一个 TraceId 链接。调用链路标签页在新窗口打开。

4. 在线程剖析栏中单击放大镜图标。

常见问题

为什么有的慢调用没有被监听?

答：如果短时间内出现大量慢调用，为了优先保证业务系统稳定性，ARMS 应用监控仅会使用有限的监听资源，选择性地监听部分慢调用。

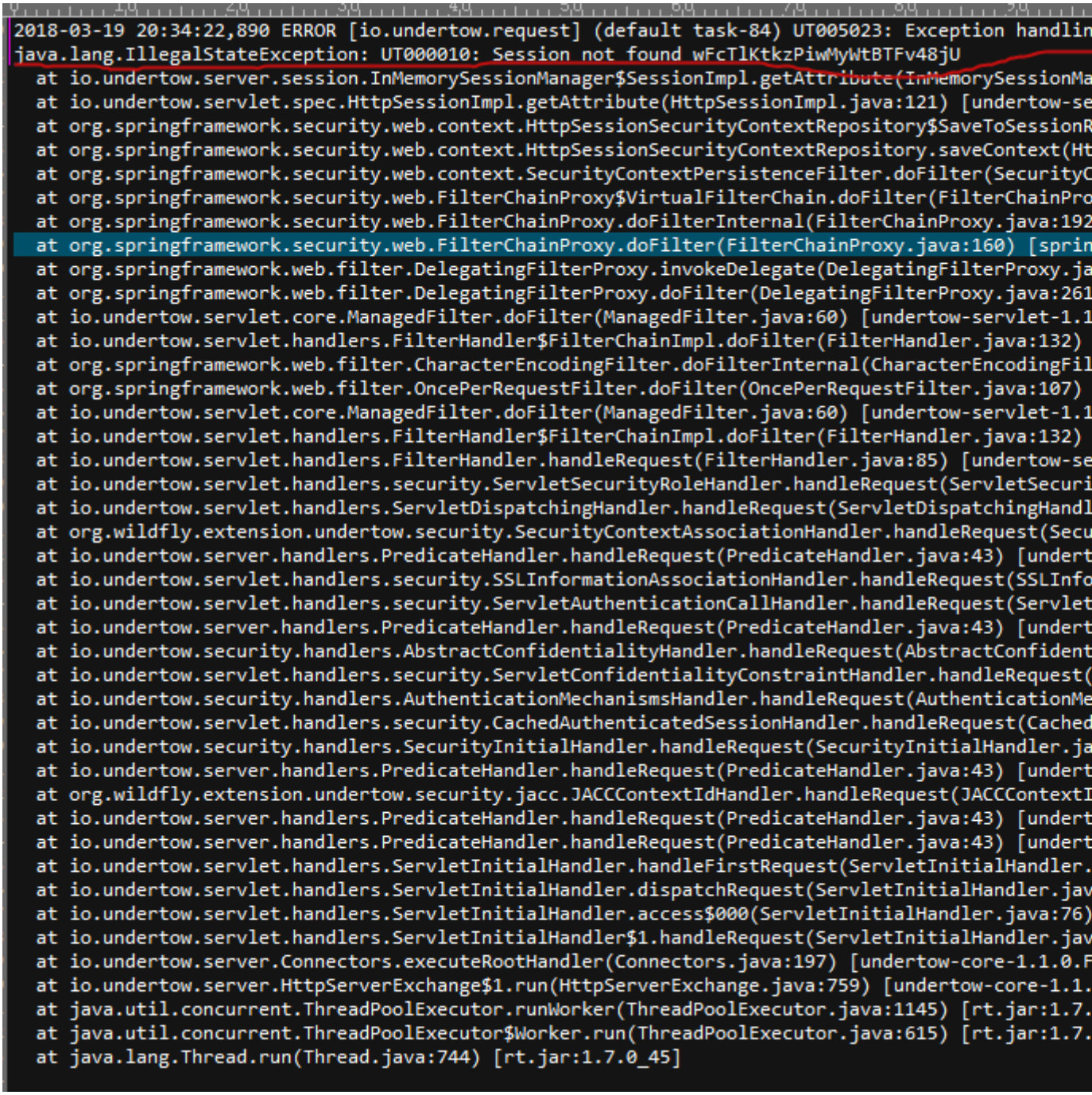
5.3 诊断服务端报错问题

网页抛错是互联网应用最常见的问题之一，错因分析是一个难点。为应用安装 ARMS 探针后，就能在不改动应用代码的情况下，借助 ARMS 应用监控的异常自动捕捉、收集、统计和溯源等功能，准确定位应用中所有异常并进行线上诊断。

问题描述

网页抛错，尤其是“5XX”错误是互联网应用最常见的问题之一。“5XX”错误通常发生于服务端。服务端是业务逻辑最复杂，也是整条网络请求链路中最容易出错、出了错之后最难诊断原因的地方。运维工程师或研发工程师往往需要登录机器查看日志来定位问题。

图 5-1: 示例：常见的 Java 应用错误日志



对于逻辑不太复杂、上线时间不长的应用来说，登录机器查看日志的方式能够解决大部分网站抛错的问题。但在以下场景中，传统的问题诊断方式往往没有用武之地。

- 在一个分布式应用集群中，需知道某一类错误的发生时间和频率。
- 某系统已运行了很长时间，但是不想关心遗留的异常，只想知道今天和昨天相比、发布后和发布前相比多了哪些异常。
- 查看一个异常对应的 Web 请求和相关参数。
- 客服人员提供了一个用户下单失败的订单号，分析该用户下单失败的原因。

解决方案

为应用安装 ARMS 探针后，即可在不改动应用代码的情况下，利用 ARMS 应用监控的异常自动捕捉、收集、统计和溯源等能力，全面掌握应用的各种错误信息。

步骤一：安装 ARMS 探针

为应用安装 ARMS 探针后，才能对应用进行全方位监控。请根据实际需求选择一种方式来安装探针。

- 为 Java 应用安装探针，请参见#unique_58。
- 为 PHP 应用安装探针，请参见#unique_59。
- 为 ECS 上的应用安装探针，请参见#unique_60。
- 为 EDAS 上的应用安装探针，请参见为 EDAS 中的应用快速安装探针。
- 为容器服务 Kubernetes 版上的应用安装探针，请参见#unique_61。
- 为开源 Kubernetes 中的应用安装探针，请参见#unique_62。

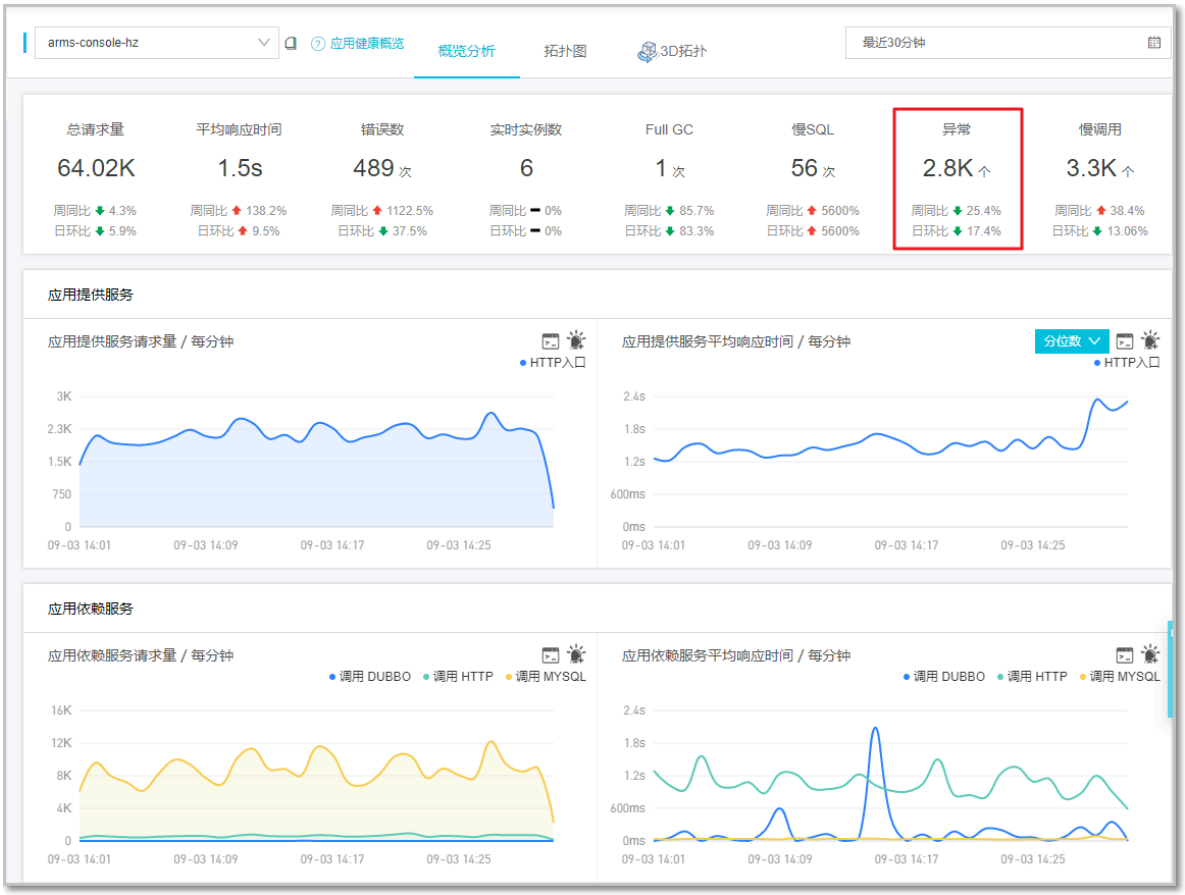
步骤二：查看关于应用异常的统计信息

为应用安装 ARMS 探针后，ARMS 会收集和展示选定时间内应用的总请求量、平均响应时间、错误数、实时实例数、FullGC 次数、慢 SQL 次数、异常次数和慢调用次数，以及这些指标和上一天的环比、上周的同比升降幅度。请按以下步骤查看应用异常的统计信息。

1. 在 ARMS 控制台左侧导航栏中，单击应用监控 > 应用列表。
2. 在应用列表页面上，单击您的应用名称。

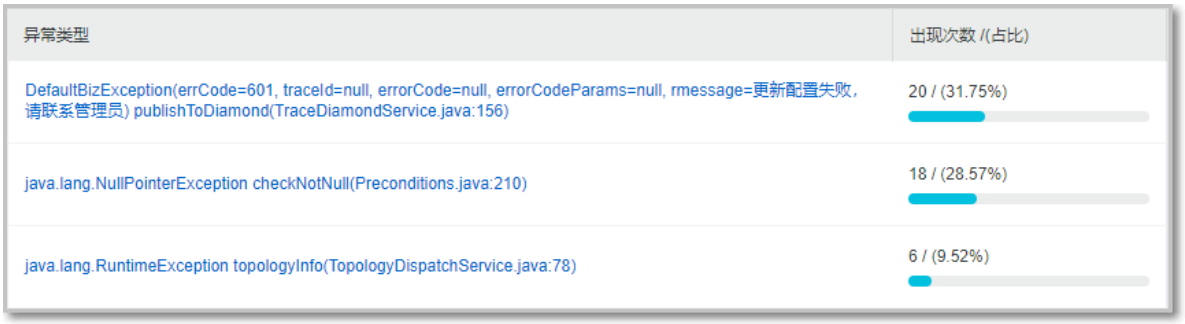
3. 在应用总览页面的概览分析页签顶部，查看异常的总数、周同比和日环比数据。

图 5-2: 异常次数统计



4. 滑动页面至概览分析页签底部的异常类型区域，查看各类型异常出现的次数。

图 5-3: 各类型异常出现的次数



5. 在左侧导航栏中，单击应用详情，然后单击页面右侧的异常分析页签。在该页签上查看异常统计图、错误数、异常详细信息等。

图 5-4: 异常分析页签

步骤三：诊断异常出现的原因

掌握应用异常的统计信息还不足以诊断异常出现的原因。虽然日志中异常堆栈包含调用的代码片段，但并不包含这次调用的完整上下游信息和请求参数。ARMS 探针采用了字节码增强技术，让您能够以很小的性能消耗捕获异常上下游的完整调用快照，进而找出导致异常出现的具体原因。

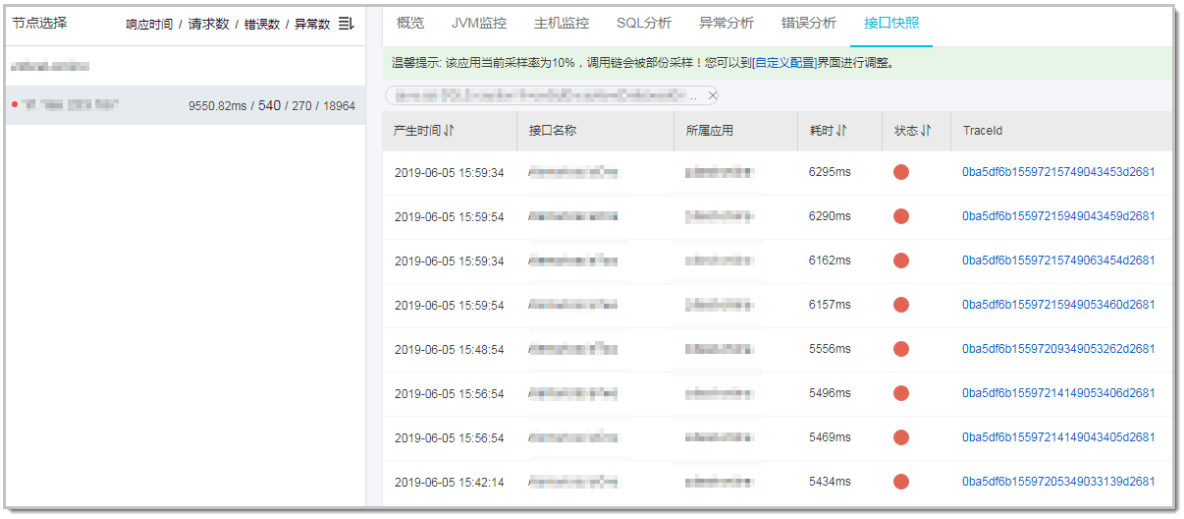
1. 在异常分析页签上，单击某个异常类型的操作列中的接口快照。

   接口快照页签上会展示出与该异常类型相关的调用链路信息。

2. 在接口快照页签上，单击某个错误调用的 TraceId。

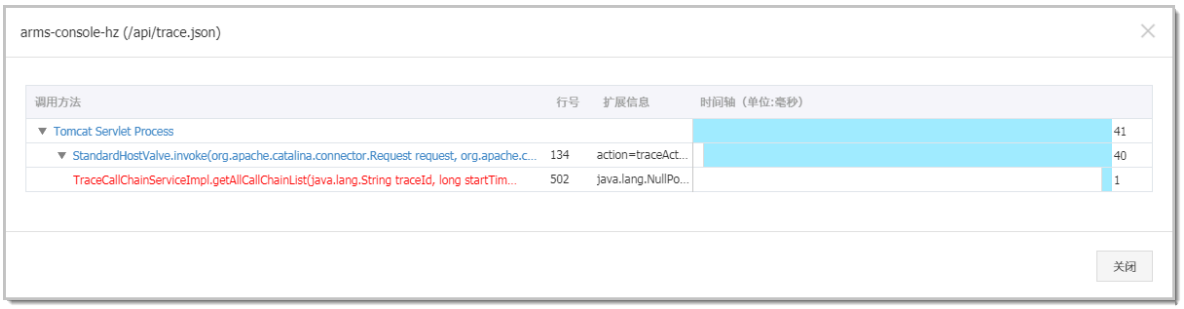
   若您需要查找目标调用链路，请参见#unique_63。

图 5-5: 接口快照页签



3. 在弹出的窗口中，查看异常的调用链路信息。单击方法栈列的放大镜图标，查看调用的详细请求参数和异常日志，从而获得异常的上下文信息。

图 5-6: 异常的完整调用链路信息



操作至此，您已发现了应用异常的原因，这将有效地帮助您进行下一步的代码优化工作。您还可以返回接口调用页面，查看列表中其他异常，逐一解决。

后续操作

为避免在出现问题后被动诊断错误原因，您还可以使用 ARMS 的报警功能针对一个接口或全部接口创建报警，即可在出现问题的第一时间向运维团队发送通知。

创建报警操作步骤请参见#unique_39。

更多信息

- #unique_64
- 应用接口调用监控
- #unique_63
- #unique_52
- #unique_39

5.4 诊断应用卡顿问题

定位、排查应用卡顿问题的原因有诸多难点。针对这类问题，ARMS 应用监控提供线程剖析、调用链路诊断、接口监控等一套解决方案，帮助您快速准确定位应用中所有慢调用，进而解决应用卡顿问题。

问题分析

网站卡顿、页面加载过慢是互联网应用最常见的问题之一。排查、解决网站卡顿、页面加载过慢等问题过程复杂，耗时较长，原因如下：

- 应用链路太长
  - 从前端页面到后台网关，从 Web 应用服务器到后台数据库，任何一个环节出现故障都有可能导致整体卡顿。
  - 采用微服务架构的应用，链路更加复杂，而且不同组件可能由不同的团队和人员维护，加剧了问题排查的难度。
- 日志不全或质量欠佳
  - 应用日志是排查线上问题的主要方法，但出现问题的位置往往无法预期，而且“慢”通常是偶发现象，要真正找到“慢”的原因，需要在每个可能出现问题的地方打印日志，记录每一次调用，但是成本太高。
- 监控不足
  - 业务发展过快、应用快速迭代导致应用频繁修改接口、增加依赖等情况，进而导致代码质量恶化。应用需要一个完善的监控体系来自动监控应用的每一个接口，自动记录出现问题的调用。

解决方案

为应用安装 ARMS 探针后，即可在不改动应用代码的情况下，使用 ARMS 应用监控的线程剖析、调用链路诊断、接口监控等功能，全方位监控应用中所有慢调用。

步骤一：安装 Java 探针

为您的应用安装 ARMS 探针后才能使用 ARMS 应用监控对应用进行全方位监控。请根据实际需求选择一种方式来安装探针。

- 为 Java 应用安装探针，请参见#unique_58。
- 为 PHP 应用安装探针，请参见#unique_59。
- 快速为 ECS 上的应用安装探针，请参见#unique_60。
- 快速为 EDAS 上的应用安装探针，请参见为 EDAS 中的应用快速安装探针。
- 为容器服务 Kubernetes 版上的应用安装探针，请参见#unique_61。
- 为开源 Kubernetes 中的应用安装探针，请参见#unique_62。

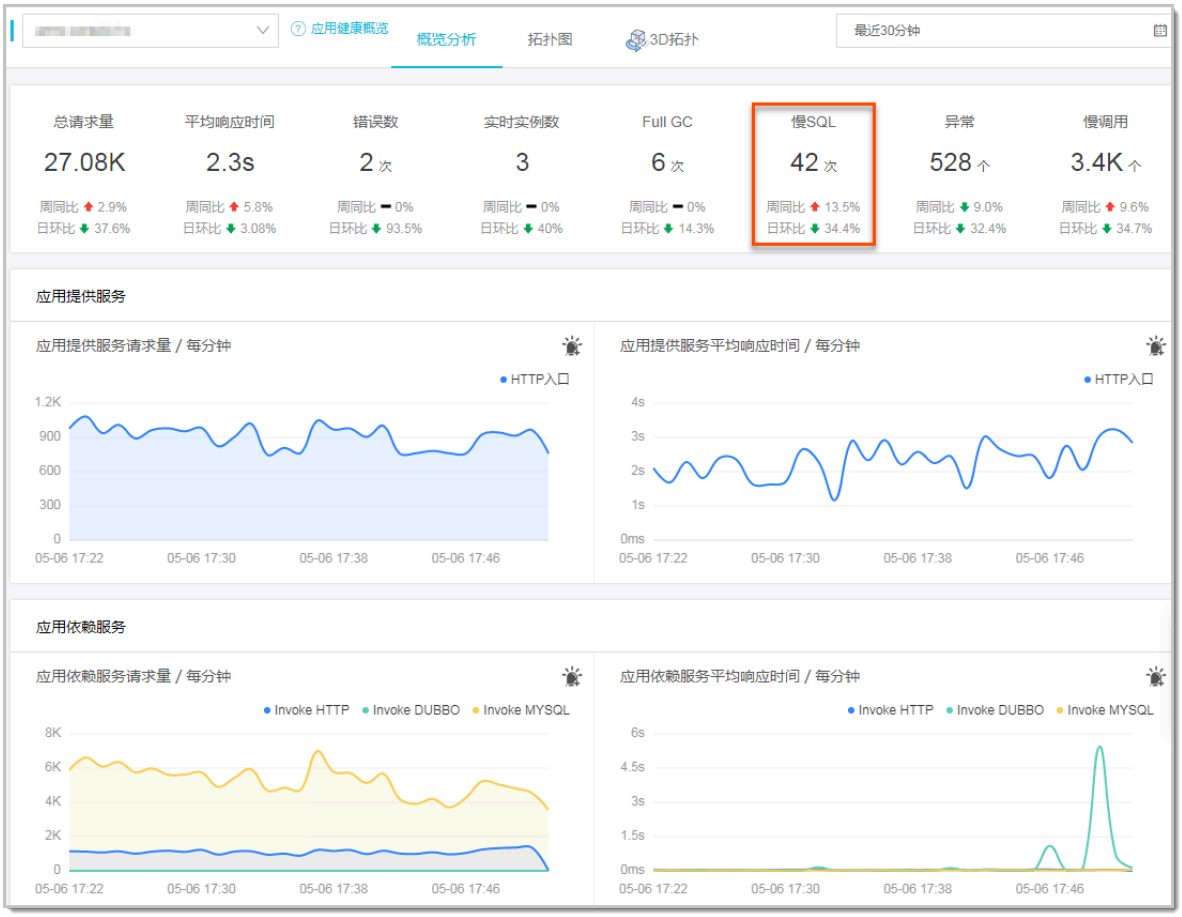
步骤二：查看慢 SQL 的统计信息

为应用安装 ARMS 探针后，ARMS 会收集和展示选定时间内应用的总请求量、平均响应时间、错误数、实时实例数、FullGC 次数、慢 SQL 次数、异常次数和慢调用次数，以及这些指标和上一天的环比、上周的同比升降幅度。请按以下步骤查看慢 SQL 的统计信息。

1. 在 ARMS 控制台左侧导航栏中，选择应用监控 > 应用列表。
2. 在应用列表页面上，单击您的应用名称。

3. 在应用总览页面的概览分析页签顶部，查看慢 SQL 的总数、周同比和日环比数据。

   在本示例中，慢 SQL 次数为 42 次。



步骤三：发现并锁定慢接口

ARMS 在接口调用页面展示了被监控的应用提供的所有接口以及这个接口的调用次数和耗时，慢接口会被标注出来，帮助您发现和锁定慢接口。

1. 在 ARMS 控制台左侧导航栏中单击接口调用。

2. 在接口调用的接口选择区域选中左侧的调用次数最多的慢接口，在右侧查看慢接口的详细信息。

步骤四：查看并锁定问题代码

锁定慢接口后，需要找到问题代码来解决问题。快照是对一次调用的全链路调用的完整记录，包括每一次调用所经过的代码及耗时，可以精准定位问题代码。

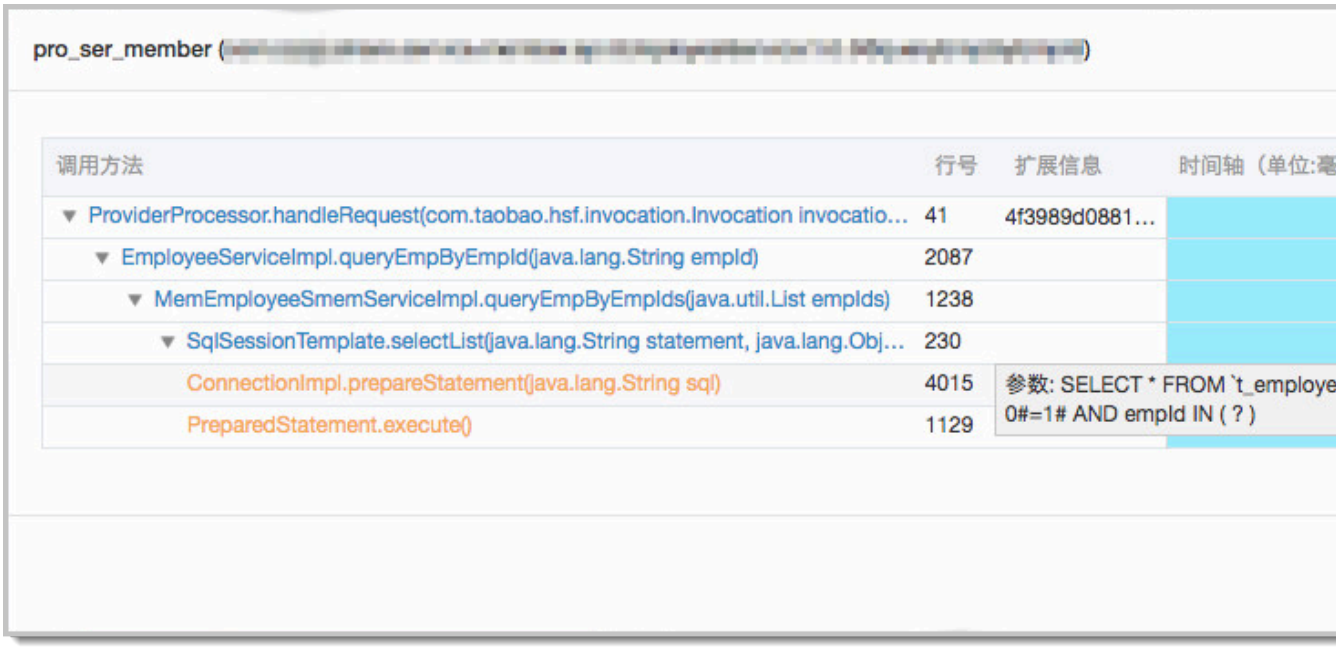
1. 在接口调用页面单击接口快照页签。

   在接口快照页签内，您可以看到这个接口对应的所有接口的快照。

2. 在接口快照页签内单击某一个调用链路的 TraceId，再单击方法栈列的放大镜图标，查看问题代码。

   若您需要查找目标调用链路，请参见#unique_63。

   在本示例中，可以看到在耗时为 705 毫秒的调用中，大部分的时间都消耗在了 `SELECT * FROM l_employee` 这次 SQL 调用中。



操作至此，您已发现了系统中的某个慢调用的原因，这将有效地帮助您进行下一步的代码优化工作。您还可以返回接口调用页面，查看列表中其他慢调用，逐一解决。

后续操作

为避免在出现问题后被动诊断错误原因，您还可以使用 ARMS 的报警功能针对一个接口或全部接口创建报警，即可在出现问题的第一时间向运维团队发送通知。

创建报警操作步骤请参见#unique_39。

更多信息

- #unique_64
- 应用接口调用监控
- #unique_63
- #unique_52
- #unique_39

5.5 使用主动诊断排查 RT 类错误

定位、排查 RT 类错误需要对多项指标逐一排查，过程漫长且复杂。针对此类问题，ARMS 应用监控提供主动诊断功能，帮助您快速准确地定位应用中 RT 类错误，进而解决应用响应时间过长问题。

背景信息

RT 类错误可能是由下游应用响应时间太长、流量不均匀、FullGC 过高、负载过高等原因导致。定位、排查 RT 类错误时，需要知道以下信息：

- 导致本次 RT 突增的服务器。
- 应用 SQL 耗时分析。
- 检测应用的 FullGC 的次数、耗时是否有突增。
- 是否存在内存泄漏。
- 检测异常日志。
- 检测下游应用的响应时间是否出现同样的趋势。

步骤一：安装探针

为您的应用安装 ARMS 探针后，ARMS 将对应用进行全方位监控。请根据实际需求选择一种方式来安装探针。

- 为 Java 应用安装探针，请参见#unique_58。
- 为 PHP 应用安装探针，请参见#unique_59。
- 快速为 ECS 上的应用安装探针，请参见#unique_60。
- 快速为 EDAS 上的应用安装探针，请参见为 EDAS 中的应用快速安装探针。
- 为容器服务 Kubernetes 版上的应用安装探针，请参见#unique_61。
- 为开源 Kubernetes 中的应用安装探针，请参见#unique_62。

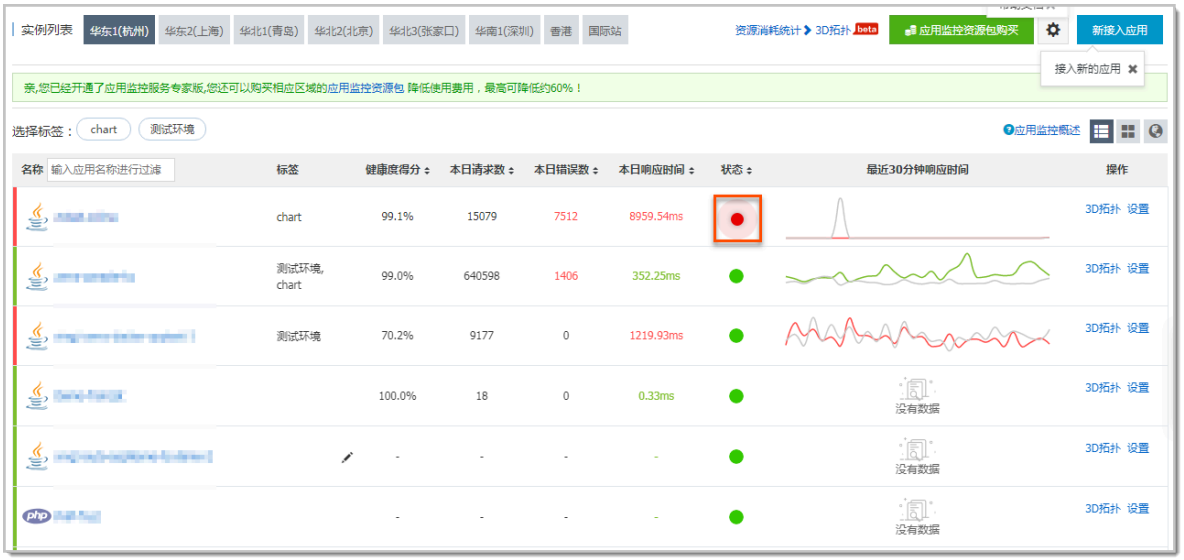
步骤二：查看异常信息

为应用安装 ARMS 探针后，ARMS 会收集和展示选定时间内应用的总请求量、平均响应时间、错误数、实时实例数、FullGC 次数、慢 SQL 次数、异常次数和慢调用次数等指标。请按以下步骤查看应用的异常信息。

1. 在 ARMS 控制台左侧导航栏中，选择应用监控 > 应用列表。

   在应用列表页面，若应用存在异常，则状态栏显示为红色。

2. 在应用列表页面上，单击目标应用状态栏的红点。



3. 在关键事件页面查看异常详情信息。

关键事件页面展示了该应用的响应时间、下游应用响应时间、平均响应时间的时序曲线、异常接口信息以及异常调用 Top 5 的 TraceID。

图 5-7: 关键事件

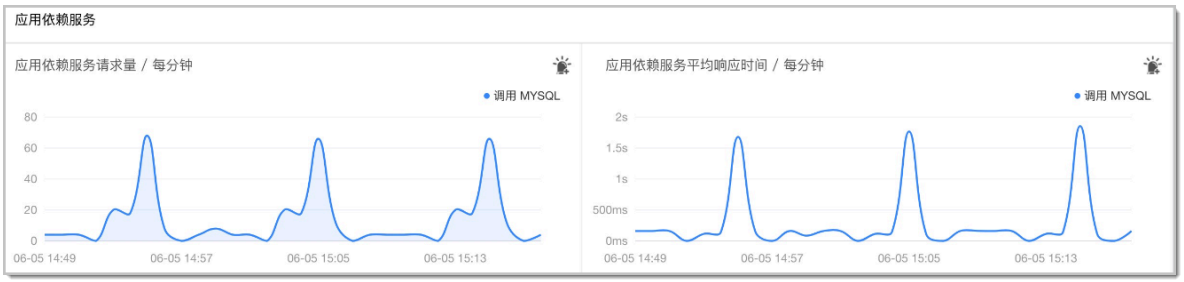
步骤三：诊断异常出现的原因

掌握应用异常的统计信息还不足以诊断异常出现的原因。可以通过 SQL 分析、链路追踪、接口快照等功能快速定位导致异常出现的具体原因。

1. 在关键事件页面单击依赖服务的名称，例如单击调用 MYSQL。然后在概览页的应用依赖服务区域查看下游应用详情。

   应用依赖服务区域展示了应用依赖服务的请求量、平均响应和应用实例数的时序曲线，以及 HTTP-状态码统计等信息。在本示例中，下游服务响应时间达到 1680 ms，可以判断应用 RT 突增是由下游应用 RT 突增导致的。

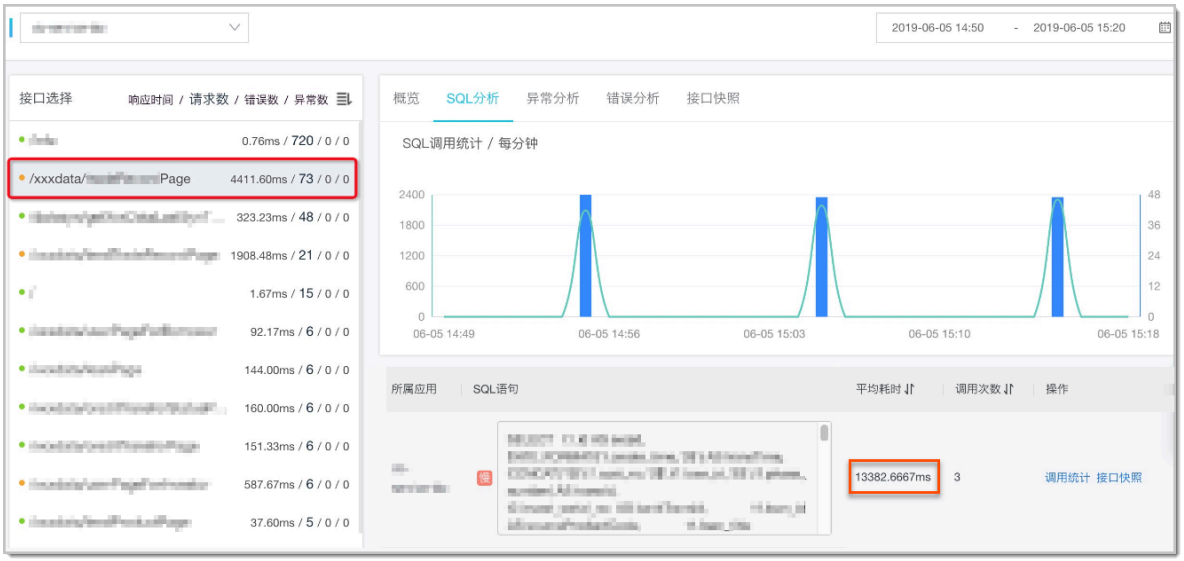
   图 5-8: 应用依赖服务详情



2. 返回关键事件页面，并在该页面单击对突变影响最大的接口，例如单击/xxxdata/...page，然后在/xxxdata/...page的接口详情页面单击SQL 分析页签，查看接口信息。

   SQL 分析页签中展示了 SQL 调用统计、SQL 语句详情等信息，详情请参见 SQL 分析。在本示例中，可以判断下游服务的 SQL 调用太慢导致应用 RT 突增。

   图 5-9: SQL 分析

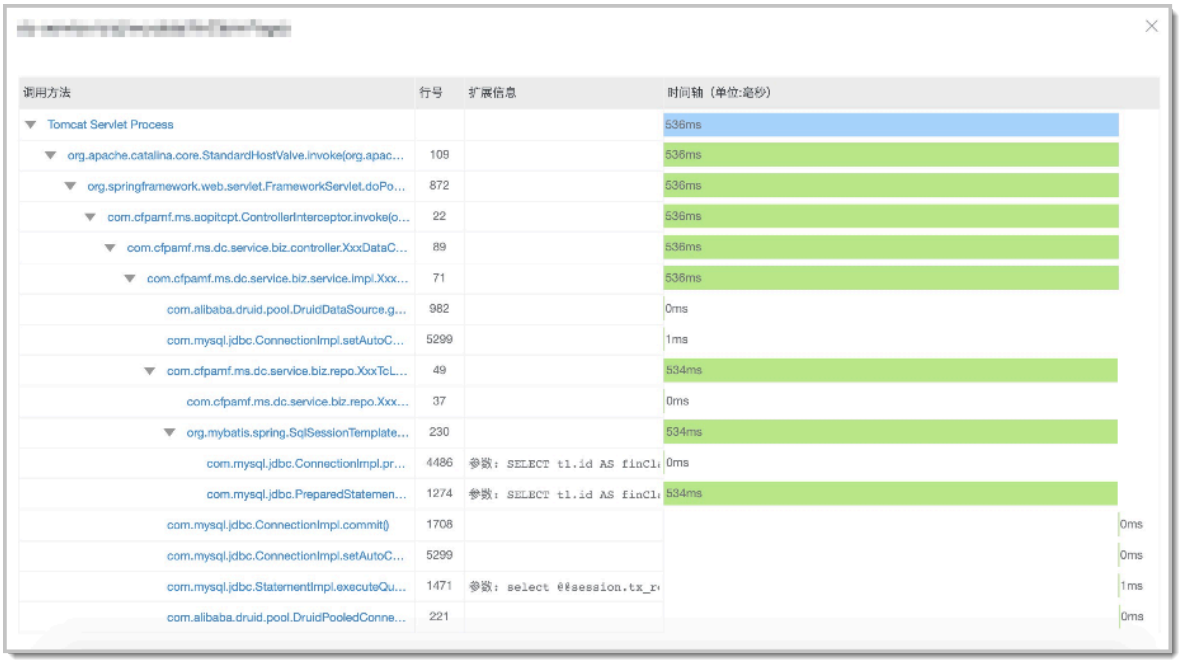


3. 返回关键事件页面，并在该页面单击 Top5 耗时的一个 TraceID，然后再单击方法栈列的放大镜图标，查看问题代码。

   若您需要查找目标调用链路，请参见#unique_63。

   在本示例中，可以看到在耗时为 536 ms 的调用中，大部分的时间都消耗在了 `SELECT t1.id As...` 这次 SQL 调用中。

   图 5-10: 调用链路信息



操作至此，您已发现了系统中的一个慢调用的原因，这将有效地帮助您进行下一步的代码优化工作。您还可以返回关键事件页面，查看列表中其他慢调用，逐一解决。

后续操作

为避免在出现问题后被动诊断错误原因，您还可以使用 ARMS 的报警功能针对一个接口或全部接口创建报警，即可在出现问题的第一时间向运维团队发送通知。

创建报警操作步骤请参见#unique_39。

更多信息

- #unique_64
- 应用接口调用监控
- #unique_63
- #unique_52
- #unique_39

6 参考信息

6.1 应用监控兼容性列表

ARMS 应用监控对接入 Java 和 PHP 应用使用的第三方框架和组件有相关限制，本文将分别列出 ARMS 应用监控对 Java 和 PHP 应用第三方组件和框架的兼容性列表。

支持的 Java 组件和框架

组件	JDK 1.7	JDK 1.8
Dubbo	2.5.X+	2.5.X+
Google HTTP Client	1.10.X+	1.10.X+
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+
MyBatis	3.X+	3.X+
MySQL JDBC	5.0.X+	5.0.X+
Oracle JDBC	10.2.X+	10.2.X+
OKHttp	2.X+	2.X+
Redis	2.X+	2.X+
Spring Boot	1.3.X+	1.3.X+
Spring	4.X+	4.X+
Tomcat	7.X+	7.X+
Undertow	1.3X+	1.3X+
WebLogic	12.X+	12.X+
MemCached	2.8+	2.8+
GRPC-Java	1.15+	1.15+
Lettuce	4.0+	4.0+
MariaDB	1.3+	1.3+
MongoDB	3.7+	3.7+
PostgreSql JDBC	9.4+	9.4+

组件	JDK 1.7	JDK 1.8
Resin	3.0+	3.0+
SQLServer JDBC	6.4+	6.4+
Thrift	0.8+	0.8+

不在支持列表内的框架和组件

如果您的开发框架不在以上 ARMS 支持的 Java 组件和框架列表中，还可以通过配置通用 Filter 拦截器的方式进行数据采集。配置步骤：

1. **在程序中引入** *arms-sdk-1.7.1.jar*。

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

说明：

如果无法获取 pom，请直接下载 arms-sdk-1.7.1.jar。

2. **在** *web.xml* **中配置 ARMS 的 Filter 拦截器。**

```
<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. **在 ARMS 中为 Java 应用安装探针。**
4. **完成配置后，请重启应用令配置生效。**

支持的 PHP 组件和框架

项目名	版本要求
PHP 版本	PHP 5.4、5.5、5.6、7.0、7.1、7.2 NTS版本
Nginx	php-fpm
Apache	apache2handler
PHP 探针运行环境	glibc-2.12 及以上

6.2 探针版本说明

本文主要介绍了 ARMS 应用监控 Java 探针和 PHP 探针的版本更新历史。

Java 探针版本

版本	发布时间	发布说明
2.5.8	2019 年 8 月 2 日	· 支持二元状态报警功能，即针对仅具有是和否、有和无这两种状态的指标设置报警规则。 · 支持国产达梦数据库插件。
2.5.7.2	2019 年 7 月 30 日	· 支持 JVM Metaspace 指标。 · 支持自定义要忽略的 HTTP 状态码。默认情况下，大于 400 的状态码会计入错误数，您可以自定义大于 400 但不计入的 HTTP 状态码。[相关文档]
2.5.7	2019 年 7 月 11 日	升级依赖的有安全漏洞的 Fastjson 版本。
2.5.6.1	2019 年 6 月 28 日	· 支持 Dubbo/Mariadb 插件。 · 自定义配置支持获取 SQL 绑定值：捕获 `PrepareStatement` 参数绑定的变量值，无需重启应用即可生效。[相关文档] · 优化内存和修复若干错误。 · 去除 Log4j 日志依赖，避免冲突。
2.5.6	2019 年 6 月 7 日	· 支持分位数统计功能。 · 优化功能和修复若干错误。
2.5.5	2019 年 6 月 3 日	· 支持 HSF-HTTP 调用。 · 优化功能和修复若干错误。
2.5.3	2019 年 3 月 15 日	· 支持应用运行过程中的线程指标上报。 · 支持 Spring-Data-Redis 插件。 · 支持 Druid 数据库连接池插件。
2.5.2	2019 年 2 月 21 日	· 增加文件句柄数采集。 · 支持 GC 时间及次数瞬时值上报。 · 支持自定义配置请求入参最大长度。[相关文档]
2.5.1	2019 年 1 月 14 日	· 支持调用链压缩。[相关文档] · 支持不通过控制台创建应用监控任务的方式。 · 优化功能和修复若干错误。

版本	发布时间	发布说明
2.5.0	2018 年 12 月 28 日	· 支持一键接入，无需重启应用。 · 完善主机监控，支持 Windows 系统。 · 支持 Spring-webflux。 · 优化功能和修复若干错误。
2.4.6	2018 年 10 月 26 日	· 支持 GRPC、THRIFT、XMemcached 插件。 · 支持接口调用拓扑展示。 · 支持覆盖前后端的拓扑展示。
2.4.5	2018 年 9 月 17 日	· 支持 Lettuce 插件（JRE 1.8+）。 · 支持 MongoDB 插件。 · 采集异常详细信息。
2.4.4	2018 年 8 月 6 日	· 支持应用线程剖析数据上报。 · 支持 Memcached 缓存。 · 支持自定义配置异常过滤。[相关文档]
2.4.3.1	2018 年 6 月 29 日	· 支持 WebLogic 服务器。 · 支持 Undertow 服务器。 · 优化 Agent 内存占用。 · 优化 Agent 启动加载时间。 · 解决 JVM 监控/主机监控指标可能无法上报问题。
2.4.3	2018 年 5 月 18 日	· 支持采集消息队列 RocketMQ 监控指标。 · 支持监控方法自定义。 · 解决限流场景下频繁输出日志的问题。 · 支持自定义配置本地方法堆栈最大长度。[相关文档] · 优化采样功能，不对异常调用链进行采样。
2.4.2	2018 年 4 月 19 日	· 支持自定义配置信息读取。 · 支持通过 SDK 方式实时获取链路信息。 · 支持线程、GC 次数/耗时等 JVM 指标采集。 · 支持 HSF 方法级调用监控。 · 支持主机监控（CPU/物理内存/网络/磁盘）等指标采集。 · 解决 Tomcat 环境下通过 `./shutdown.sh` 停止进程时可能卡住的问题。

版本	发布时间	发布说明
2.4.1	2018 年 3 月 24 日	· 支持 JVM 监控，如堆内存、非堆内存等指标上报。 · 支持 PlayFrameWork 1.4.4 版本。 · 支持自定义配置采样率、探针开关、日志级别、阈值参数等。[相关文档]
2.4.0	2018 年 2 月 14 日	· 支持 PostgreSQL 数据库。 · 支持阿里云各地域的 ECS 与 ARMS 服务器进行内网通讯。 · 支持 ARMS 应用监控正式商用。

PHP 探针版本

版本	发布时间	发布说明
2.0.2	2019 年 7 月 31 日	· 修复高并发情况下网络模块发送问题。 · 重新设计发送及重连逻辑。 · 减少内存占用。 · 修复若干错误。
2.0.1	2019 年 7 月 23 日	· Arms-agent 进程作为守护进程。 · 支持 Redis、Mongodb 插件。 · 修复若干错误。
2.0.0	2019 年 7 月 5 日	· 采用全新网络模型，稳定性大幅提高。 · 优化异常信息显示。 · 优化内存占用。 · 修复若干错误。
1.1.0	2019 年 4 月 30 日	· 支持 GCC 4.4.7 环境。 · 增加 TCP 连接心跳。 · 修复主机监控错误。 · 支持以 `php -m` 命令显示 ARMS 版本。 · DNS 解析 Collector 域名。 · 优化功能和修复若干错误。 注意: 建议尽快将 1.x.x 版本探针升级至 2.x.x。

版本	发布时间	发布说明
1.0.1	2019 年 3 月 15 日	· 支持 Laravel 框架 5.x。 · 支持 PDO 插件。 · 去除多余日志。 · 修复若干错误。 注意: 建议尽快将 1.x.x 版本探针升级至 2.x.x。

6.3 关键统计指标说明

本文说明了 ARMS 应用监控各页面的关键统计指标的含义。

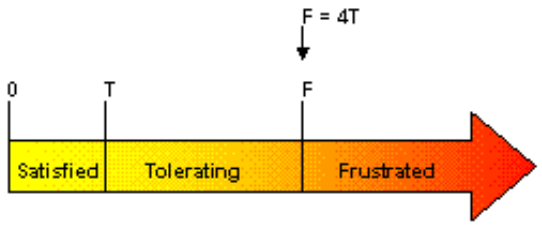
基本概念

本文涉及以下基本概念:

· APDEX 性能指数

APDEX 性能指数（Application Performance Index）是一个国际通用的应用性能计算标准。该标准将用户对应用的使用感受定义为三个等级:

- 满意（0 ~ T）
- 可容忍（T ~ 4T）
- 不满意（大于 4T）



图片来源: apdex.org

计算公式为:

```
Apdex = （满意数 + 可容忍数 / 2） / 总样本量
```

ARMS 取应用的平均响应时间作为计算指标，并将 T 定义为 500 毫秒。

- 实例

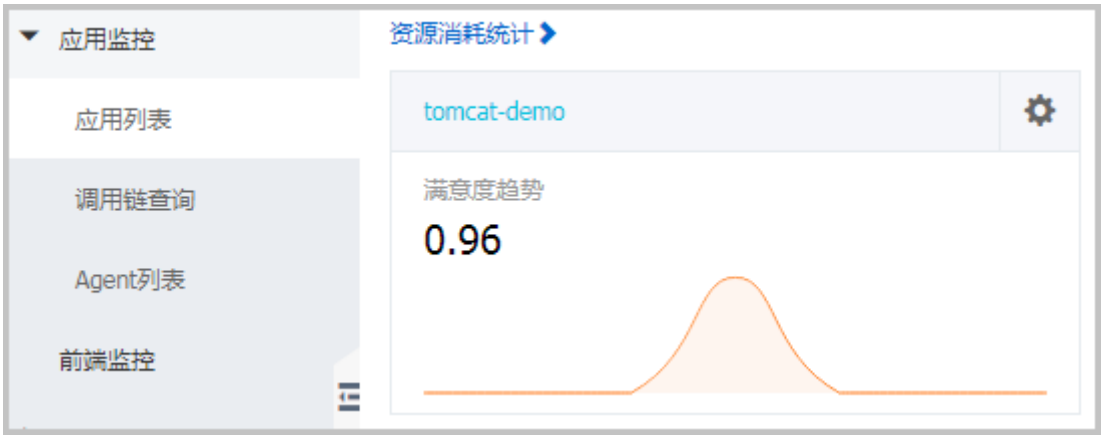
  实例是指被监控的应用所部署的机器，以 JVM 为粒度。例如在下图中，“a3”是一个应用，下方的每一行都是该应用所部署的一台机器，即一个实例。

相关统计页面

- 应用监控列表页面

  在控制台左侧导航栏中选择应用监控 > 应用列表，即可看到各应用的 APDEX 满意度趋势曲线。

  图 6-1: APDEX 满意度趋势



- 应用总览页面

  在应用监控列表页面单击应用名称，即可进入应用总览页面。在左侧导航栏中选择相应菜单，可以在其他页面查看其他维度的统计信息。

  - 概览分析标签页
    - 应用提供的服务：请求量和平均响应时长
    - 应用依赖的服务：请求量和平均响应时长
    - 系统信息：CPU、内存和负载
    - 统计分析：慢接口调用分析和平均响应时间、异常类型和出现次数
  - 拓扑图标签页
    - 应用拓扑图
    - 实例健康： 绿色表示正常，黄色表示警告，红色表示严重。
    - 调用类型：

调用类型	描述	备注
HTTP 入口	客户端使用 HTTP 协议调用该应用的入口	服务入口调用
调用 Dubbo	Dubbo 的消费者产生的调用	服务入口调用
调用 HSF	HSF 服务的消费者产生的调用	服务入口调用
调用 HTTP	该调用为该应用对其他服务发起的 HTTP 调用	服务间调用
提供 HSF	HSF 的生产者产生的调用	服务间调用
提供 Dubbo	Dubbo 的生产者产生的调用	服务间调用
调用 MySQL	对 MySQL 进行操作的调用	数据库调用
调用 Oracle	对 Oracle 进行操作的调用	数据库调用
调用 Redis	对 Redis 进行操作的调用	数据库调用

- 应用详情页面

  此页面展示当前应用的调用详细信息。选择不同标签页，可切换展示实例响应时间、请求数、错误数统计，以及实例概览、SQL 分析、异常分析、接口快照等维度的详细分析。

- 接口调用页面

  此页面展示当前应用所开放的接口的统计信息。选择不同标签页，可切换展示实例响应时间、请求数、错误数统计，以及实例概览、SQL 分析、异常分析、接口快照等维度的详细分析。

- 数据库调用页面

  该部分展示应用所关联的数据库调用情况。选择不同标签页，可切换展示实例响应时间、请求数、错误数统计，以及实例概览、SQL 分析、异常分析等维度的详细分析。

相关标签页的关键统计指标说明

- 响应时间标签页

上报字段	描述
响应时间	应用、实例调用的平均响应时间，或数据库操作的平均执行响应时间

- 请求数标签页

上报字段	描述
请求数	应用、实例调用的请求调用次数，或数据库操作的执行次数

- 错误数标签页

上报字段	描述
错误数	应用、实例调用的错误调用次数，或数据库操作中异常执行次数

- 概览标签页

上报字段	描述
请求数	应用、实例调用的请求调用次数，或数据库操作的执行次数
响应时间	应用、实例调用的平均响应时间，或数据库操作的平均执行响应时间
错误率	（应用、实例调用的异常调用次数，或数据库操作的异常次数）/请求数
性能一览	柱状图与左 Y 轴为请求数统计，折线图与右 Y 轴为响应时间

- SQL 分析标签页

上报字段	描述
SQL 调用统计	柱状图与左 Y 轴为数据库请求数统计，折线图与右 Y 轴为数据库响应时间
平均耗时	本次数据库调用的平均耗时
调用次数	该应用此类型数据库调用次数

- 异常分析标签页

上报字段	描述
异常统计	柱状图为该应用、实例、数据库的异常次数
异常类型	采集到的抛错类型
异常详细信息	抛错的详细信息
平均耗时	本次错误调用的平均耗时
错误数	该异常类型的错误出现的次数

- 接口快照标签页

上报字段	描述
耗时	应用、实例的接口的调用耗时
状态	应用、实例的接口的调用返回状态，绿色表示正常返回，红色表示抛异常
TraceId	应用、实例调用的索引 ID，单击可以跳转到该调用链详情

6.4 ARMS-SDK 使用说明

借助 ARMS 提供的 SDK，您可以在业务代码中动态获取 TraceId 及相关调用链属性。

前提条件

- 您已在 ARMS 控制台上创建应用监控，并已在 Java 程序中挂载和启动应“”用监控的 Agent。详情请参考#unique_23中关于安装 Java 探针的步骤。
- 程序中已引入 arms-sdk-1.7.1.jar。

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

说明:

如果无法获取 Pom，请直接下载 arms-sdk-1.7.1.jar。

获取 TraceId 与 RpcId

满足前提条件后，即可通过以下代码获取上下文的 TraceId 与 RpcId。

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

```
String rpcId = span.getRpcId();
```

透传业务自定义标签

要透传业务自定义标签，需要在代码中写入添加和获取自定义标签的步骤。

1. 在业务代码中添加自定义标签（baggage）。

```
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. 在业务代码中获取自定义标签（baggage）。

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