ALIBABA CLOUD

阿里云

弹性容器实例

日志与监控

文档版本：20211122

阿里云

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
危险	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。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会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，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	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。	单击**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。	在**结果确认**页面，单击**确定**。
Courier字体	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	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。	`switch {active|stand}`

目录

1.采集日志

1.1. 通过阿里云日志服务采集日志

本文介绍在ASK集群中，如何通过阿里云日志服务SLS采集容器的标准输出和文件日志。

前提条件

- 已创建ASK集群。具体操作，请参见创建Serverless Kubernetes集群。
- 已开通日志服务。

  登录日志服务控制台时，如果没有开通日志服务，将收到相关提示，您可以根据页面提示开通。

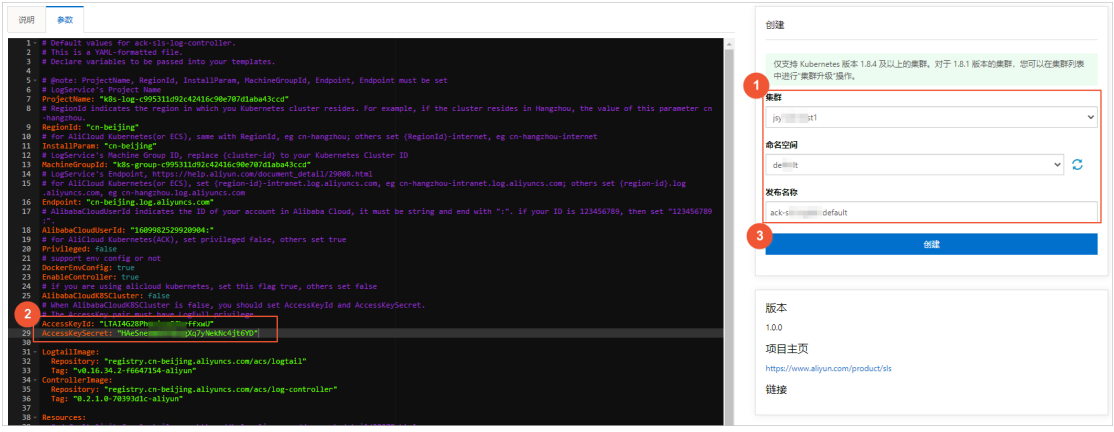
背景信息

阿里云日志服务SLS（Log Service）是针对日志数据的一站式服务，您无需开发就能快捷完成日志数据采集、消费、投递以及查询分析等功能。更多信息，请参见日志服务简介。

如果您使用的是阿里云ASK集群，支持通过SLS CRD和环境变量两种方式进行配置，将容器日志采集到阿里云日志服务SLS中。

方式一：通过SLS CRD配置

1. 登录容器服务管理控制台。
2. 在集群中安装alibaba-log-controller。
   i. 在左侧导航栏，选择**市场>应用目录**。
   ii. 在**阿里云应用**页签下，找到**ack-sls-logtail**应用，然后单击该应用。
   iii. 配置参数，并选择集群进行安装。
      a. 选择要安装的集群和命名空间，输入发布名称。
      b. 单击**参数**页签，在下方YAML模板中填写AccessKey和AccessKeySecret。
      c. 单击**创建**。



   iv. 查看安装结果。

   单击集群名称进入集群信息页面，在左侧导航栏选择**应用>Helm**，查看对应发布（默认发布名称为ack-sls-logtail-default）的状态是否为**已部署**。
3. 创建日志配置CRD。

连接集群，参考YAML示例编写日志配置CRD的YAML配置文件（命名为log.yaml），然后执行命令创建日志配置CRD。

采集的日志分为标准输出（包括错误输出）和文件日志两种：

- 标准输出CRD YAML示例

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
 name: test-stdout    #资源名称，集群内唯一
spec:
 project: k8s-log-c326bc86****  #Project名称，可自定义，推荐使用集群ID命名
 logstore: test-stdout   #Logstore名称，如果不存在则自动创建
 shardCount: 2   #可选配置，Shard数量，默认为2，取值范围为1~10
 lifeCycle: 90 #可选配置，Logstore中日志的保留时间，默认为90天，取值范围为1~7300，7300天表示永久存储
 logtailConfig:
  inputType: plugin    #采集的数据源类型，file表示文件日志，plugin表示标准输出
  configName: test-stdout      #采集配置的名称，与metadata.name持一致
  inputDetail:
   plugin:
    inputs:
     - type: service_docker_stdout
       detail:
         Stdout: true
         Stderr: true
#          IncludeEnv:
#           aliyun_logs_test-stdout: "stdout"
```

- 文件日志CRD YAML示例

```
apiVersion: log.alibabacloud.com/v1alpha1
kind: AliyunLogConfig
metadata:
 name: test-file  #资源名称，集群内唯一
spec:
 project: k8s-log-c326bc86****  #Project名称，可自定义，推荐使用集群ID命名
 logstore: test-file    #Logstore名称，如果不存在则自动创建
 logtailConfig:
  inputType: file    #采集的数据源类型，file表示文件日志，plugin表示标准输出
  configName: test-file #采集配置的名称，与资源名metadata.name持一致
  inputDetail:
   logType: common_reg_log     #对于分隔符类型的日志，logType可以设置为json_log
   logPath: /log/   #日志文件夹
   filePattern: "*.log"   #文件名, 支持通配符，例如log_*.log
   dockerFile: true #采集容器内的文件，dockerFile设置为true
   #用作解析时间的key''
   #timeKey: 'time'
   #时间解析方式''
   #timeFormat: '%Y-%m-%dT%H:%M:%S'
   #避免不同采集配置中存在相同采集目录而导致冲突
   #dockerIncludeEnv:
   # aliyun_logs_test-file: "/log/*.log"
```

执行以下命令创建日志配置CRD。

```
kubectl create -f log.yaml
```

> 说明
>
> 创建日志配置CRD后，您可以在日志服务控制台查看对应生成的日志库及logtail配置。如果后续需要更新配置，可以直接编辑CRD，系统将自动同步配置到日志服务SLS。

4. 部署应用。

   完成日志配置CRD后，后续创建的业务Pod的日志将被采集到日志服务SLS中。

   以下为Pod YAML配置文件示例，可以实现通过while循环不断打印标准输出和日志文件。

```
apiVersion: v1
kind: Pod
metadata:
 labels:
  app: sls
 name: eci-sls-demo
 namespace: default
spec:
 containers:
 - args:
  - -c
  - mkdir -p /log;while true; do echo hello world; date; echo hello sls >> /log/busy.log; sleep 1;
    done
  command:
  - /bin/sh
  image: busybox:latest
  imagePullPolicy: Always
  name: sls
```

5. 在日志服务控制台查看日志。

   在集群对应的Project中找到目标日志库，单击日志库名称即可查看日志。更多信息，请参见查看配置效果。

方式二：通过环境变量配置

1. 登录容器服务管理控制台。
2. 在左侧导航栏，单击**集群**。
3. 在**集群列表**页面，找到要配置的集群，单击集群名称。
4. 在集群详情页面的左侧导航栏，选择**工作负载>无状态**。
5. 新建或者修改Pod的YAML配置模板，设置环境变量来传入日志相关配置。

```
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
  app: alpine
 name: alpine
spec:
 replicas: 2
 selector:
  matchLabels:
   app: alpine
 template:
  metadata:
   labels:
    app: alpine
  spec:
   containers:
   - image: alpine
    imagePullPolicy: Always
    args:
    - ping
    - 127.0.0.1
    name: alpine
    env:
    #配置环境变量
    #配置Project，如果使用K8s集群默认的project，则可以不填
    - name: aliyun_logs_test-stdout_project
     value: k8s-log-xxx
    - name: aliyun_logs_test-file_project
     value: k8s-log-xxx
    #配置机器组，如果使用K8s集群默认的机器组，则可以不填
    - name: aliyun_logs_test-stdout_machinegroup
     value: k8s-group-app-alpine
    - name: aliyun_logs_test-file_machinegroup
     value: k8s-group-app-alpine
    #配置标准输出和错误输出的Logstore和路径
    - name: aliyun_logs_test-stdout
     value: stdout
    #将/log/*.log目录下的日志收集到名为aliyun_logs_test-file的Logstore
    - name: aliyun_logs_test-file
     value: /log/*.log
    #设置日志保留时间，只对单个Logstore生效
    - name: aliyun_logs_test-stdout_ttl
     value: "7"
    #设置日志分区数，只对单个Logstore生效
    - name: aliyun_logs_test-stdout_shard
     value: "2"
```

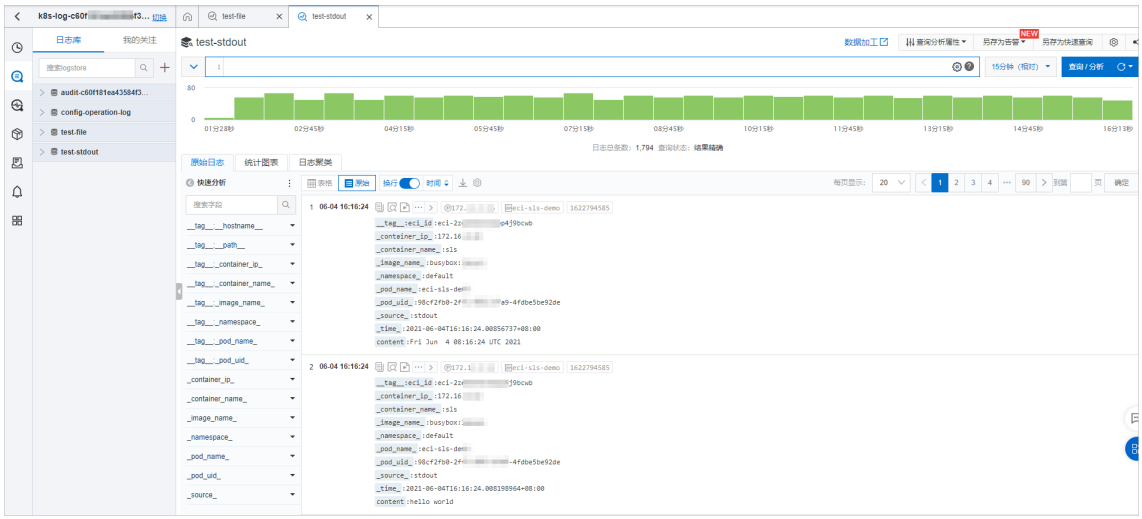
在上述示例中，所有与日志配置相关的环境变量采用 `aliyun_logs_` 为前缀，其中环境变量 `aliyun_logs_test-stdout` 表示创建一个名为test-stdout的Logstore，将容器的标准输出采集到test-stdout这个Logstore中，采集路径为stdout。

6. 单击**创建**。
7. 在日志服务控制台查看日志。

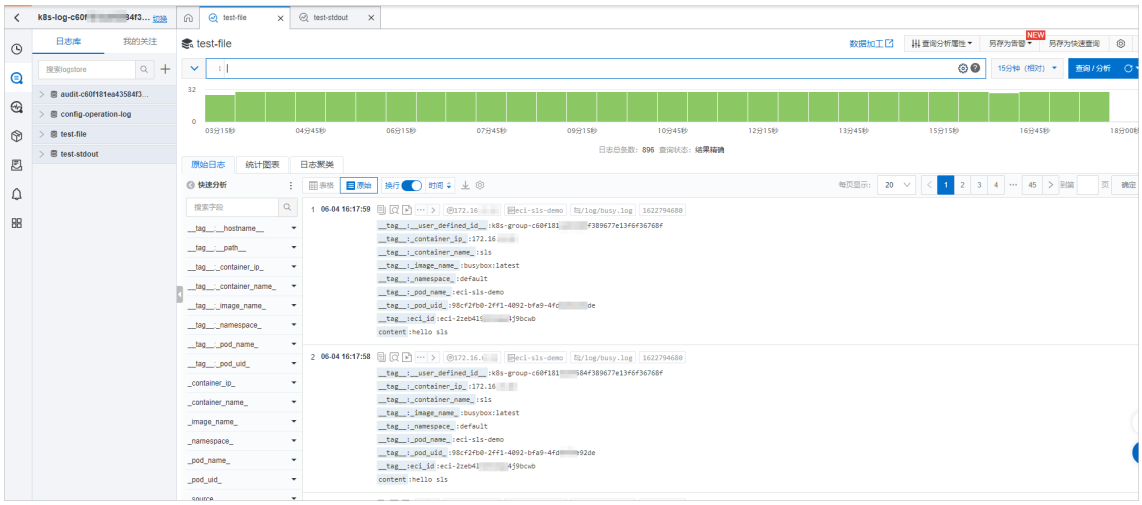
   在集群对应的Project中找到目标日志库，单击日志库名称即可查看日志。更多信息，请参见查看配置效果。

查看配置效果

1. 登录日志服务控制台。
2. 单击Projcect名称。
3. 找到目标日志库，单击日志库名称查看日志。
   - 标准输出



   - 文件日志



1.2. 通过Sidecar方式采集日志

阿里云日志服务支持在ECI中通过Sidecar模式采集日志。本文介绍如何安装Sidecar和配置Logtail，实现容器日志的采集。

前提条件

- 已创建ASK集群。具体操作，请参见创建Serverless Kubernetes集群。
- 已开通日志服务。

  登录日志服务控制台时，如果没有开通日志服务，将收到相关提示，您可以根据页面提示开通。

背景信息

阿里云日志服务SLS支持在ECI中通过Sidecar模式采集日志，即在每个ECI实例中，除业务容器外，运行一个Sidecar容器作为日志Agent，用于采集业务容器产生的日志。

> **注意**
>
> Sidecar模式基于Logtail实现，Logtail必须和业务容器共享日志目录。

采集的日志可以分为以下两种：

- 标准输出

  采集标准输出依赖于ECI的stdlog卷。创建Pod时可以将该卷挂载到Sidecar容器上，Sidecar可以直接以文件的方式访问ECI基础组件收集的标准输出日志。关于如何挂载stdlog卷，请参见挂载stdlog。

- 文件日志

  采集文件日志可以通过Pod内共享的Volume实现。Pod内同一个Volume可以挂载到多个容器，Sidecar可以直接收集业务容器输出到对应Volume内的文件日志。

步骤一：部署Sidecar容器

1. 创建一个包含Sidecar容器的Deployment。

   YAML内容示例如下，请根据实际情况将占位符变量替换为实际值。

```
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
  app: nginx-log-sidecar-demo
 name: nginx-log-sidecar-demo
spec:
 replicas: 2
 selector:
  matchLabels:
   app: nginx-log-sidecar-demo
 template:
  metadata:
   labels:
    app: nginx-log-sidecar-demo
  spec:
   containers:
    - name: nginx-log-demo
      image: registry-vpc.${RegionId}.aliyuncs.com/log-service/docker-log-test:latest
      command:
       - /bin/mock_log
      args:
       - '--log-type=nginx'
       - '--stdout=false'
       - '--stderr=true'
```

```
        - '--path=/var/log/nginx/access.log'
        - '--total-count=100000000'
        - '--logs-per-sec=100'
      imagePullPolicy: Always
      volumeMounts:
        - mountPath: /var/log/nginx
          name: nginx-log
    - name: logtail
      image: registry-vpc.${RegionId}.aliyuncs.com/log-service/logtail:latest
      env:
        - name: ALIYUN_LOGTAIL_USER_ID
          value:  "${Aliuid}"
        - name: ALIYUN_LOGTAIL_USER_DEFINED_ID
          value: nginx-log-sidecar
        - name: ALIYUN_LOGTAIL_CONFIG
          value: /etc/ilogtail/conf/${RegionId}/ilogtail_config.json
        - name: aliyun_logs_machinegroup
          value: k8s-group-app-alpine
      imagePullPolicy: Always
      volumeMounts:
        - mountPath: /var/log/nginx
          name: nginx-log
        - mountPath: /stdlog
          name: stdlog
  volumes:
    - emptyDir: {}
      name: nginx-log
    - name: stdlog      #标准输出日志输出到stdlog卷
      flexVolume:
        driver: alicloud/pod-stdlog
```

2. 获取Pod的配置信息。
3. 查看日志。
   - 通过kubectl命令查看日志
   - 通过弹性容器实例控制台查看日志

> **说明**
>
> Pod的stdlog卷中的标准输出日志是ECI底层组件记录的，其格式与原生K8s的日志格式一致。K8s会在每行标准输出日志前增加前缀。配置日志格式解析时，您需要配置去除该前缀。具体操作，请参见解析JSON格式的日志。

步骤二：配置Logtail采集日志

部署Sidecar容器后，您需要在日志服务控制台配置Logtail。

1. 登录日志服务控制台。
2. 在**接入数据**区域，单击**正则>文本日志** 。

> 说明
>
> Logtail支持通过极简模式、正则模式、分隔符模式、JSON模式等模式采集文本日志。本文以正则模式为例介绍Logtail配置。更多信息，请参见采集文本日志。

3. 选择日志空间，单击**下一步**。

   请选择Project和Logstore，您也可以直接单击立即创建准备流程。

   如果您是通过日志库下的**数据接入**后的加号进入采集配置流程，系统会直接跳过该步骤。

4. 创建机器组。

   如果您已有可用的机器组，可直接单击**使用现有机器组**。

   i. 确认已创建机器组，单击**确认安装完毕**。

   ii. 在**创建机器组**页签中，配置相关参数，单击**下一步**。

   机器组标识选择**用户自定义标识**，将CreateContainerGroup时配置的环境变量ALIYUN_LOGTAIL_USER_DEFINED_ID填入**用户自定义标识**框中。

5. 配置机器组。

   选择一个机器组，将该机器组从源机器组移动到应用机器组。

6. 设置Logtail配置。

   - 标准输出
   - 文本日志

7. 查询分析配置。

   默认已设置索引，您也可以根据业务需求，重新设置索引。具体操作，请参见配置索引。

8. 查看采集的日志。

   完成上述配置后，日志服务将开始采集日志。

1.3. 采集Job任务的日志

本文介绍在ACK集群中使用ECI来运行Job任务时，如何采集Job任务的日志。

前提条件

- 已创建一个ACK集群并在该集群上部署了虚拟节点。具体操作，请参见创建Kubernetes托管版集群和部署虚拟节点。
- 已为Namespace添加 `alibabacloud.com/eci=true` 的Label。

  通过配置Namespace Label的方式可以将Pod调度到ECI上运行。更多信息，请参见调度Pod到ECI。
- 已创建NAS文件系统，并已添加挂载点。具体操作，请参见管理文件系统和管理挂载点。

背景信息

在ACK集群中，对于在标准节点的ECS上运行的Job任务，可以通过DaemonSet的方式来采集日志。但是对于在虚拟节点的ECI上运行的Job任务，由于ECI不支持DaemonSet，当Job任务运行结束后，Pod会立即退出，此时日志可能还未被采集完成。

针对上述场景，您可以采用以下方式来采集ECI上运行的Job任务的日志：

1. 为Job任务挂载NAS文件系统，将输出的日志保存到NAS文件系统上。
2. 将收集了Job任务日志的NAS文件系统挂载到另一个Pod上，获取NAS文件系统中存储的Job任务日志。

> **说明**
>
> 如果您使用了阿里云日志服务SLS，通过配置环境变量的方式，为Job任务挂载Volume，可以直接同步阿里云日志服务。更多信息，请参见自定义配置ECI日志采集。

操作步骤

下文以名为vk的Namespace为例说明操作步骤，该Namespace已添加 `alibabacloud.com/eci=true` 的Label，部署在该Namespace下的Pod将被调度到ECI上运行。实际配置时请替换为您要使用的Namespace。

1. 准备Job任务的YAML配置文件。

   ```
   vim job.yaml
   ```

   以下为一个计算π值的Job任务配置示例：

   ```
   apiVersion: batch/v1
   kind: Job
   metadata:
    name: pi
   spec:
    template:
     spec:
      containers:
      - name: pi
       image: resouer/ubuntu-bc
       command: ["sh", "-c", "echo 'scale=1000; 4*a(1)' | bc -l > /eci/a.log 2>&1"] #输出结果重定向到指定文件
       volumeMounts:
       - name: log-volume
         mountPath: /eci
         readOnly: false
      restartPolicy: Never
      #挂载NAS文件系统用于存储日志
      volumes:
      - name: log-volume
       nfs:
          path: /eci
          server: 04edd48c7c-****.cn-hangzhou.nas.aliyuncs.com
          readOnly: false
    backoffLimit: 4
   ```

2. 部署Job任务到ECI上。

   ```
   kubectl apply -f job.yaml -n vk
   ```

3. 查看Pod状态，确认Job任务是否已正常运行。

   ```
   kubectl get pod -n vk
   ```

4. 准备用于采集Job任务日志的Pod配置文件。

```
vim log-collection.yaml
```

YAML文件内容示例如下：

```
apiVersion: v1
kind: Pod
metadata:
 name: log-collection
spec:
 containers:
 - image: nginx:latest
  name: log-collection
  command: ['/bin/sh', '-c', 'echo &(cat /eci/a.log)'] #查看Job任务的日志文件
  volumeMounts:
  - mountPath: /eci
   name: log-volume
 restartPolicy: Never
 #挂载存储了Job任务日志的NAS文件系统
 volumes:
 - name: log-volume
  nfs:
   server: 04edd48c7c-****.cn-hangzhou.nas.aliyuncs.com
   path: /eci
   readOnly: false
```

5. 部署Pod，查看Job任务的日志文件。

```
kubectl apply -f log-collection.yaml -n vk
```

1.4. ECI中日志采集的自定义配置

用户自定义设置

根据业务需要，您可能需要将ECI的日志收集到自定义项目下的自定义日志库里。对于不同的应用和服务，您可能还需要将ECI实例加入不同的机器组。对于项目、日志库和机器组等自定义配置需求，您可以通过两种办法实现：

- **通过日志服务控制台（API）手动设置**

用户可以自行登录日志服务控制台，创建自定义项目，创建自定义日志库，以及自定的机器组，为日志库创建自定义config并应用到选择的机器组。这样日志内容就可以导向新的日志库了。

如果觉得通过日志服务控制台配置太繁琐，您可以通过ECI代创建和配置。

- **通过ECI自定义**

ECI除了具备为用户生成所有默认设置外，还支持为用户生成自定义的配置。比如项目名、日志库名、配置名、机器组名、以及日志收集目录等。具体的参数通过ECI的容器的环境变量传入，格式如下：

Logstore名和配置名

首先是配置名

```
-name: aliyun_logs_{配置名}
-value: {日志采集路径}
```

> 注意
>
> 如需采集标准输出，请将日志采集路径设置为stdout。

默认情况下，logstore的名字和配置名同名，如果需要设置配置所输出的logstore的名字，可以采用如下的方式自定义：

```
-name: aliyun_logs_{配置名}_logstore
-value: {logstore 名称}
```

日志库名约束

- 日志库名称仅支持小写字母、数字、连字符（-）和下划线（_）。
- 必须以小写字母和数字开头和结尾。
- 名称长度为3-63个字符。

> 注意
>
> 校验不通过的，会直接忽略，使用ECI默认的。

项目名

设置日志收集所属的project，方式如下。

```
-name: aliyun_logs_{配置名}_project
-value: {project 名称}
```

默认情况下，对于ECI的API用户，每个地域会有一个默认的project，对于k8s的用户，默认project为每个集群一个，命名方式为“k8s-log-{k8s集群id}”

项目名约束

- 项目名称仅支持小写字母、数字和连字符（-）。
- 必须以小写字母和数字开头和结尾。
- 名称长度为3-63个字符。

> 注意
>
> 校验不通过的，会直接忽略，使用ECI默认的。

Logstore设置分区数

什么是分区（Shard），请参见分区。

设置方法：

```
-name: aliyun_logs_{配置名}_shard
-value: {shard数值}
```

默认值为2，可选范围是[1,10]。

Logstore设置日志保留时间

设置方法：

```
-name: aliyun_logs_{配置名}_ttl
-value: {ttl数值}
```

默认值为90，可选范围是[1,3650]。

机器组名称

非必填参数，

默认情况，

对于ECI API 用户，ECI实例会加入到ECI帮用户创建的默认机器组，一个region对应一个。

对于 kubernetes 用户，ECI实例会加入集群默认的机器组，命名格式“k8s-group-{k8s集群id}”。

自定义设置的格式如下：

```
-name: aliyun_logs_{配置名}_machinegroup
-value: {机器组名称}
```

机器组名约束

- 机器组名称仅支持字母、数字、连字符（-）和下划线（_）。
- 必须以小写字母和数字开头和结尾。
- 名称长度为3-63个字符。

> **注意**
>
> 校验不通过的，会直接忽略，使用ECI默认的。

自定义Tag

设置方法：

```
-name: aliyun_logs_{配置名}_tags
-value: {Tag名称}={Tag值}
```

用户Volume日志收集

对于标准输出和错误输出，只需要通过环境变量进行设置就可以收集；如需要收集任意的自定的目录下的日志文件，需要依赖Volume才可以进行收集。

Volume的标准日志收集目录为Volume挂载的目录下的子目录，具体取决于用户自己的设定。

比如：

用户有个EmptyDirVolume，挂载到了容器的/pod/data/目录下，那么Volume的日志收集可以指定是/pod/data/下的任意子目录下的任意文件。通过这种方式，用户可以灵活的调整挂载目录并配合自己的业务，实现自定义的日志收集目录。

创建EmptyDirVolume

```
'Volume.1.Name': 'default-volume',
'Volume.1.Type': 'EmptyDirVolume',
```

将Volume挂载至容器目录

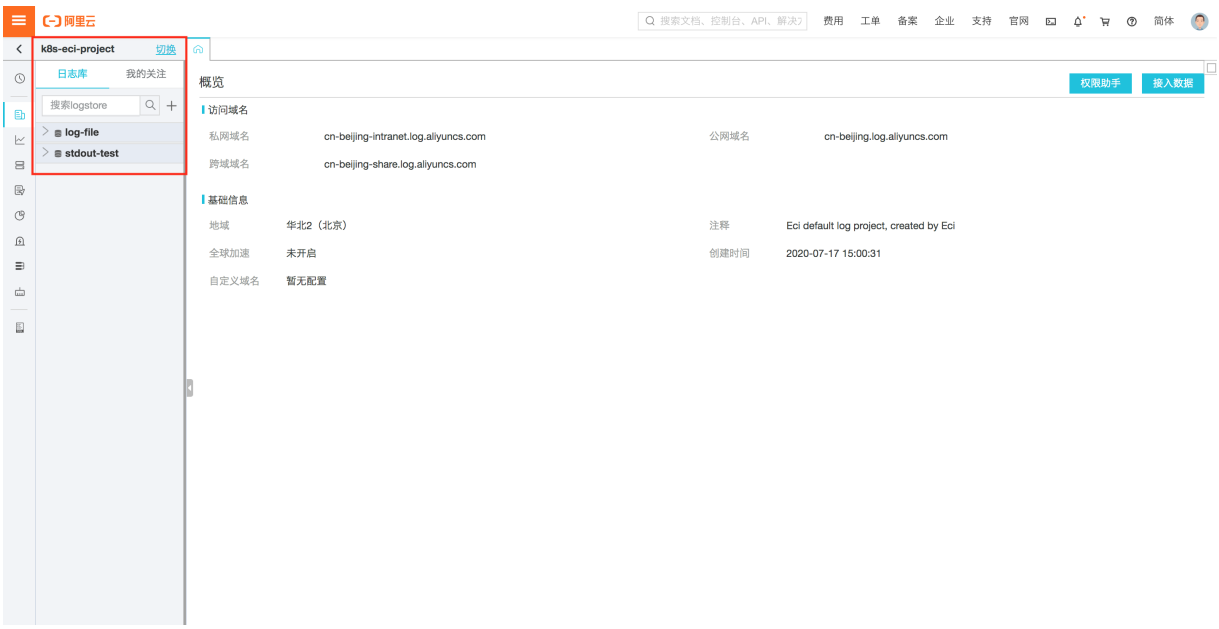
```
'Container.1.VolumeMount.1.Name': 'default-volume',
'Container.1.VolumeMount.1.MountPath': '/pod/data/',
'Container.1.VolumeMount.1.ReadOnly': False,
```

配置日志仓库

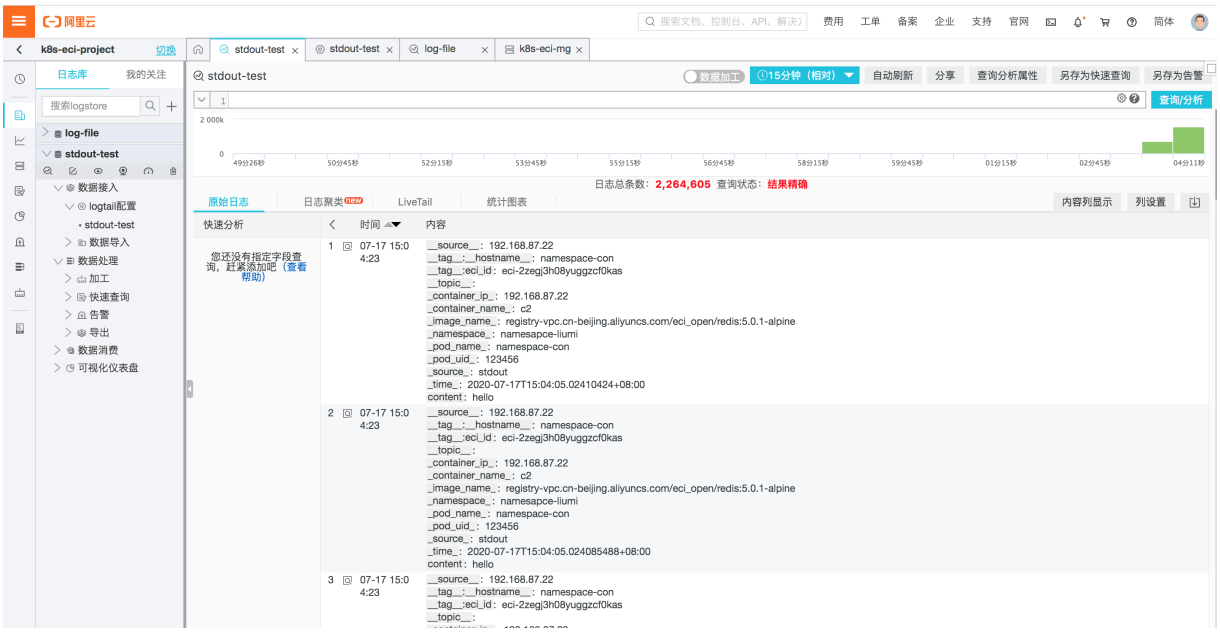
'aliyun_logs_stdout-test' 为ECI的容器的标准输出的收集目录，'aliyun_logs_log-file' 为Volume的日志收集目录，模糊匹配/pod/data/目录下的任意日志文件。

```
'Container.1.EnvironmentVar.1.Key': 'aliyun_logs_log-file',
'Container.1.EnvironmentVar.1.Value': '/pod/data/*.log',
'Container.1.EnvironmentVar.2.Key': 'aliyun_logs_stdout-test',
'Container.1.EnvironmentVar.2.Value': 'stdout',
'Container.1.EnvironmentVar.3.Key': 'aliyun_logs_log-file_project',
'Container.1.EnvironmentVar.3.Value': 'k8s-eci-project',
'Container.1.EnvironmentVar.4.Key': 'aliyun_logs_stdout-test_project',
'Container.1.EnvironmentVar.4.Value': 'k8s-eci-project',
'Container.1.EnvironmentVar.5.Key': 'aliyun_logs_log-file_machinegroup',
'Container.1.EnvironmentVar.5.Value': 'k8s-eci-mg',
'Container.1.EnvironmentVar.6.Key': 'aliyun_logs_stdout-test_machinegroup',
'Container.1.EnvironmentVar.6.Value': 'k8s-eci-mg',
```

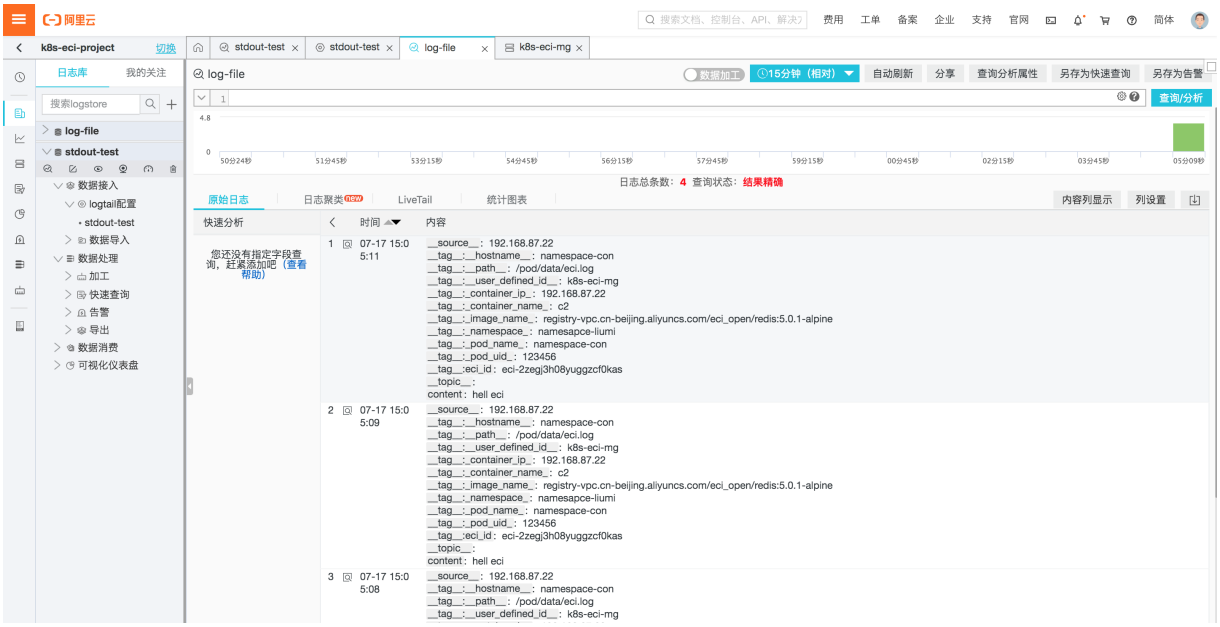
效果



标准输出日志：



日志文件收集：



1.5. 解析JSON格式的日志

对于JSON格式的标准输出日志，K8s添加时间戳、来源等信息后，会导致JSON解析失败。本文介绍如何通过阿里云日志服务SLS的Processor能力解决上述JSON格式日志解析的问题。

背景信息

ECI采集的标准输出为原生K8s的日志，K8s会在每行日志之前增加时间戳、来源等信息，但这可能会破坏您原生的日志格式，例如：您的标准输出日志为JSON格式，K8s添加前缀后，JSON解析就会失败。如下所示：

```
2020-04-02T15:40:05.440500764+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:07.442412564+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:09.442774495+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:11.443799303+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:13.445099622+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:15.445934358+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:17.447064707+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:19.448112987+08:00 stdout F {"key1":"val1","key2":"val2"}
2020-04-02T15:40:21.449393263+08:00 stdout F {"key1":"val1","key2":"val2"}
```

操作步骤

通过阿里云日志服务SLS的Processor能力，可以帮助您解决JSON格式日志的解析问题。操作步骤如下：

1. 登录日志服务控制台。
2. 在Project列表区域，找到目标Project，单击名称。

   ECI自动创建的Project名称默认为eci-log。
3. 找到目标Logstore，单击图标，然后单击**logtail配置**。
4. 修改Logtail配置。

   采用极简模式并启用插件处理，其中插件配置内容如下：

```
{
  "processors": [
    {
      "type": "processor_anchor",
      "detail": {
        "SourceKey": "content",
        "Anchors": [
          {
            "Start": "stdout F ",
            "Stop": "",
            "FieldName": "json_content",
            "FieldType": "string",
            "ExpondJson": false
          }
        ]
      }
    },
    {
      "type": "processor_json",
      "detail": {
        "SourceKey": "json_content",
        "KeepSource": false,
        "ExpandConnector": ""
      }
    }
  ]
}
```

执行结果

保存Logtail配置后，您可以看到正常解析的日志内容，如下图所示。

2.ASK接入ARMS监控

2.1. ASK接入ARMS应用监控

借助ARMS应用监控，您可以对ASK集群中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。本文介绍如何为ASK集群接入ARMS应用监控。

前提条件

- 已创建ASK集群。
- 已开通ARMS服务。具体操作，请参见开通ARMS。

> **说明**
>
> 应用监控是ARMS的付费子产品，您可以免费试用15天，试用结束后需开通基础版或者专家版方可继续使用。更多信息，请参见ARMS定价页。

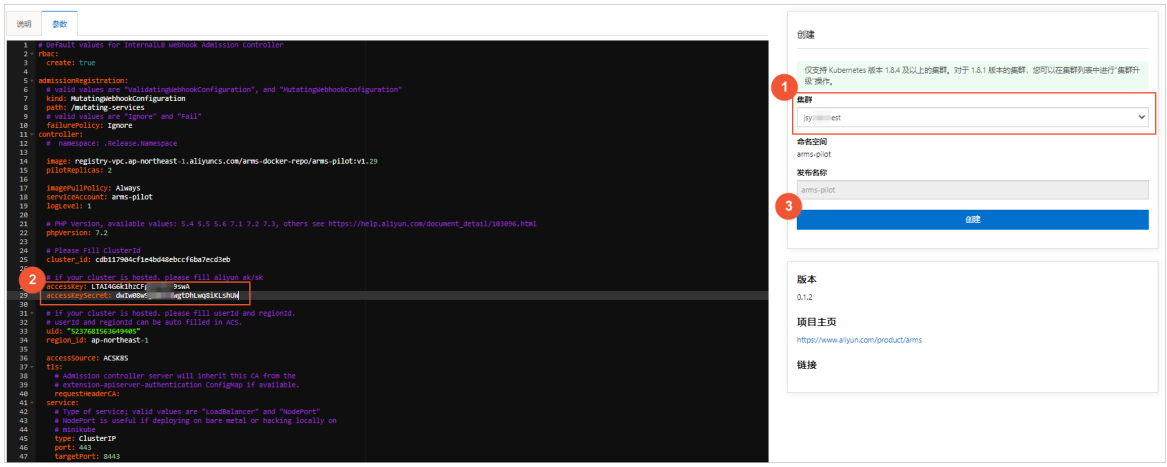
背景信息

应用实时监控服务ARMS（Application Real-Time Monitoring Service）是一款应用性能管理（APM）产品，包含应用监控、Prometheus监控等功能模块，能帮助您实现全栈式性能监控和端到端全链路追踪诊断，实现轻松高效的应用运维。

为ASK集群安装ARMS应用监控组件（探针）后，ARMS可以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。更多信息，请参见应用监控概述。

步骤一：安装应用监控组件

1. 登录容器服务管理控制台。
2. 在左侧导航栏，选择**市场>应用目录**。
3. 在**阿里云应用**页签下，找到**ack-arms-pilot**应用，然后单击该应用。
4. 配置参数，并选择集群进行安装。
    i. 选择要安装的ASK集群。
    ii. 单击**参数**页签，在下方模板中填写AccessKey和AccessKeySecret。
    iii. 单击**创建**。



5. 查看安装结果。

   单击集群名称进入集群信息页面，在左侧导航栏选择**应用>Helm**，查看arms-pilot的状态是否为**已部署**。

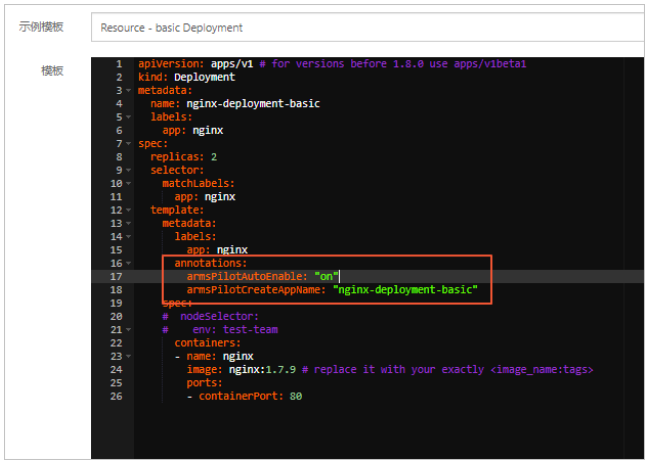
步骤二：为应用开启应用监控

您可以在应用的YAML配置文件中添加Annotation开启应用监控。Annotation请添加在spec>template>metadata下。

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

新建应用开启监控

1. 在容器服务管理控制台的**集群**页面，找到要新建应用的集群，单击集群名称。
2. 在左侧导航栏，选择**工作负载>无状态**。
3. 单击右上角的**使用YAML创建资源**。
4. 选择命名空间和示例模板，并在模板中添加Annotation至spec>template>metadata下，然后单击**创建**。



以Java应用Spring Cloud Eureka Server为例，开启ARMS应用监控的完整YAML示例模板如下：

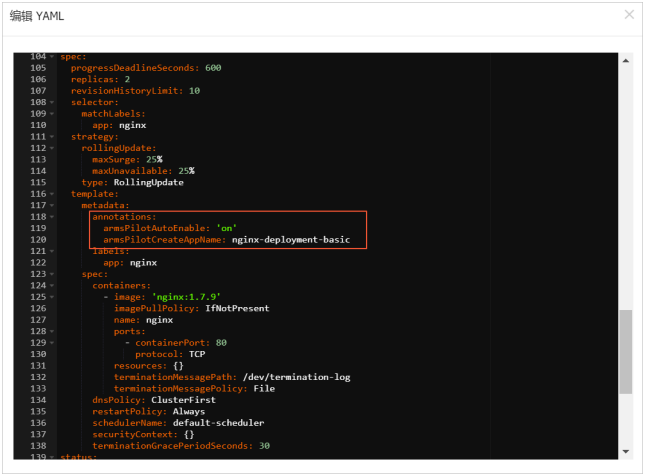
```
apiVersion: apps/v1
kind: StatefulSet
```

```
kind: StatefulSet
metadata:
 name: register-server
spec:
 replicas: 3
 serviceName: register-server
 selector:
  matchLabels:
   app: register-server
 template:
  metadata:
   labels:
    app: register-server
   annotations:
    armsPilotAutoEnable: "on"            #开启ARMS应用监控
    armsPilotCreateAppName: "register-server"  #开启ARMS应用监控的应用名称
  spec:
   containers:
    - name: register-server
      image: registry.cn-hangzhou.aliyuncs.com/shuangling/eureka-server:v1
      imagePullPolicy: Always
      env:
      - name: EUREKA_DEFAULT_ZONE
        value: "http://register-server-0.register-server:8000/eureka/,http://register-server-1.register-serv
er:8000/eureka/,http://register-server-2.register-server:8000/eureka/"
      - name: JVM_OPTS
        value: " -Xms1024m -Xmx1536m "
      - name: MY_POD_NAME
        valueFrom:
         fieldRef:
          fieldPath: metadata.name
      ports:
       - name: http
         containerPort: 8000
         protocol: TCP
      readinessProbe:
       httpGet:
        path: /actuator/health
        port: 8001
        scheme: HTTP
       failureThreshold: 3
       initialDelaySeconds: 60
       periodSeconds: 10
       successThreshold: 1
       timeoutSeconds: 10
      volumeMounts:
      - mountPath: /Charts
        name: data
   volumes:
   - name: data
     emptyDir: {}
 podManagementPolicy: "Parallel"
---
apiVersion: v1
kind: Service
```

```
kind: Service
metadata:
 name: register-server
 labels:
  app: register-server
spec:
 clusterIP: None
 type: ClusterIP
 ports:
  - port: 8000
    targetPort: http
    protocol: TCP
    name: http
 selector:
  app: register-server
```

已有应用开启监控

1. 在容器服务管理控制台的**集群**页面，找到应用所在集群，单击集群名称。
2. 在左侧导航栏，选择**工作负载>无状态**，或者选择**工作负载>有状态**。
3. 找到要开启监控的具体应用，单击**更多**，然后选择**查看Yaml**。
4. 编辑YAML内容，添加Annotation至spec> template > metadata下。



5. 单击**更新**。

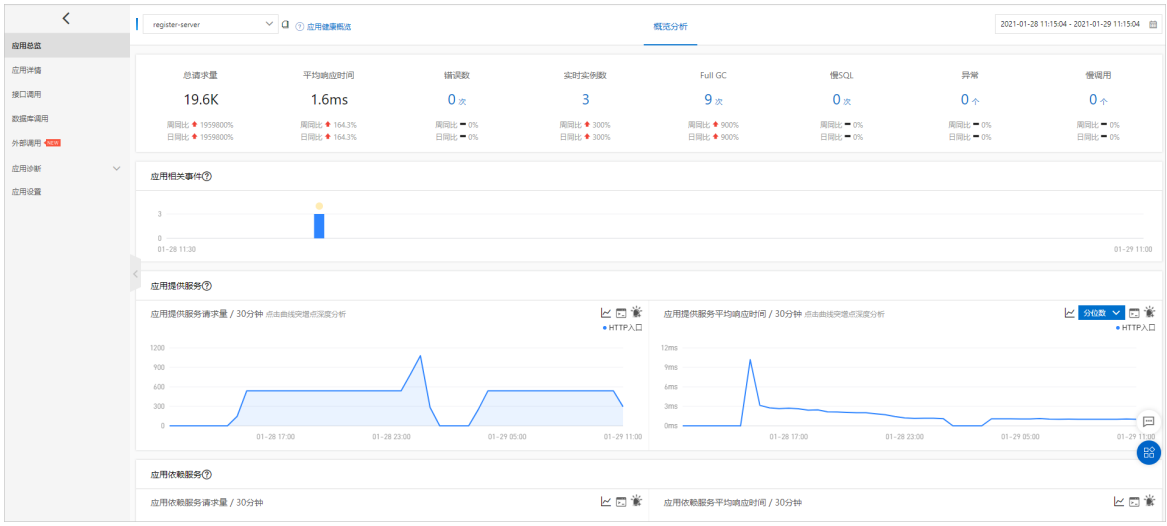
   更新后需要一定时间重新创建容器组，请等待所有容器组滚动更新完成后再查看监控数据。

步骤三：查看应用监控数据

开启ARMS应用监控后，可以在ARMS控制台的**应用监控**页面查看数据。

1. 在容器服务管理控制台的**集群**页面，找到应用所在集群，单击集群名称。
2. 在左侧导航栏，选择**工作负载>无状态**，或者选择**工作负载>有状态**。
3. 找到要查看监控数据的具体应用，单击对应操作列中的**ARMS控制台**。
4. 查看应用监控数据。

   ARMS应用监控包括自动发现应用拓扑、捕获异常事务和慢事务、实时诊断性能等功能。关于如何使用ARMS应用监控，请参见ARMS应用监控。



2.2. ASK接入ARMS Prometheus监控

接入Prometheus监控后，您可以通过ARMS预定义的大盘监控Kubernetes集群的众多性能指标。本文介绍如何为ASK集群接入ARMS Prometheus监控。

前提条件

- 已创建ASK集群。

  > **说明**
  >
  > 如果您集群内使用的安全组不是自动创建生成的，而是手动配置的安全组，请确保该安全组已开放8080、8081和9335端口。

- 已开通ARMS服务。具体操作，请参见开通ARMS。

  > **说明**
  >
  > Prometheus监控是ARMS的付费子产品，您可以免费试用15天，试用结束后需开通专家版方可继续使用。更多信息，请参见ARMS定价页。

背景信息

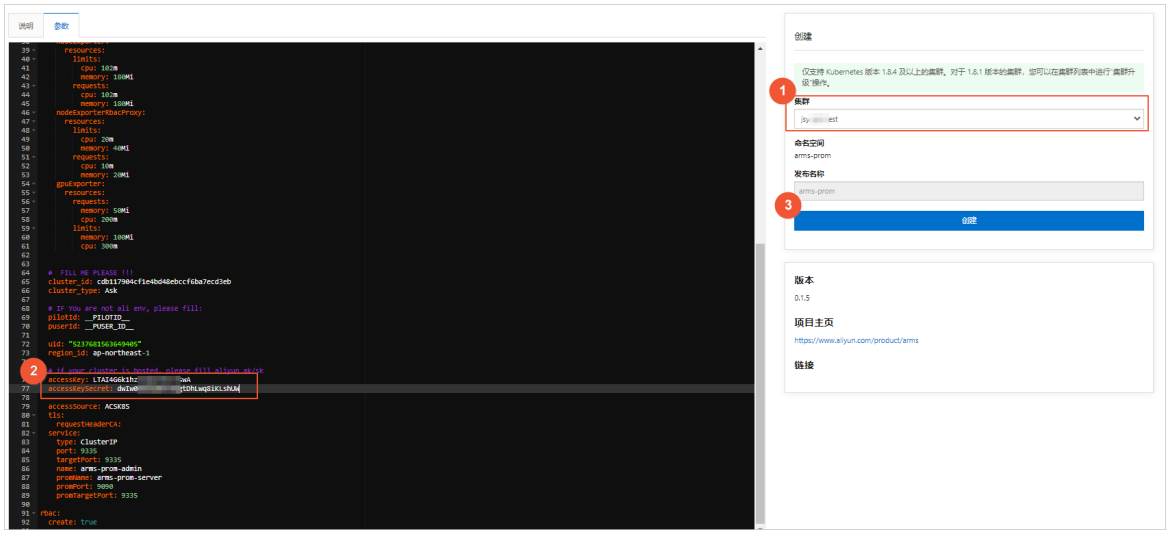
应用实时监控服务ARMS（Application Real-Time Monitoring Service）是一款应用性能管理（APM）产品，包含应用监控、Prometheus监控等功能模块，能帮助您实现全栈式性能监控和端到端全链路追踪诊断，实现轻松高效的应用运维。

ARMS Prometheus监控可以全面对接开源Prometheus生态，支持类型丰富的组件监控，提供多种开箱即用的预置监控大盘，并提供全面托管的Prometheus服务。借助ARMS Prometheus监控，您无需自行搭建Prometheus监控系统，因而无需关心底层数据存储、数据展示、系统运维等问题。更多信息，请参见Prometheus监控概述。

安装Prometheus监控组件

1. 登录容器服务管理控制台。
2. 在左侧导航栏，选择**市场>应用目录**。

3. 在**阿里云应用**页签下，找到**ack-arms-prometheus**应用，然后单击该应用。
4. 配置参数，并选择集群进行安装。
   i. 选择要安装的ASK集群。
   ii. 单击**参数**页签，在下方模板中填写AccessKey和AccessKeySecret。
   iii. 单击**创建**。



5. 查看安装结果。

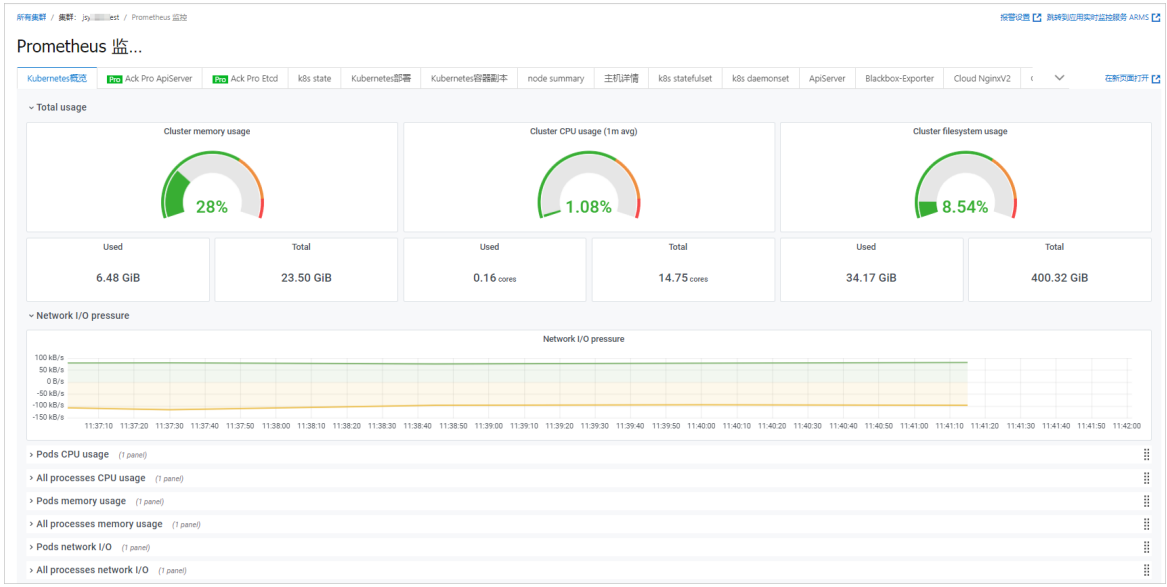
   单击集群名称进入集群信息页面，在左侧导航栏选择**应用>Helm**，查看arms-prom的状态是否为**已部署**。

查看Prometheus监控

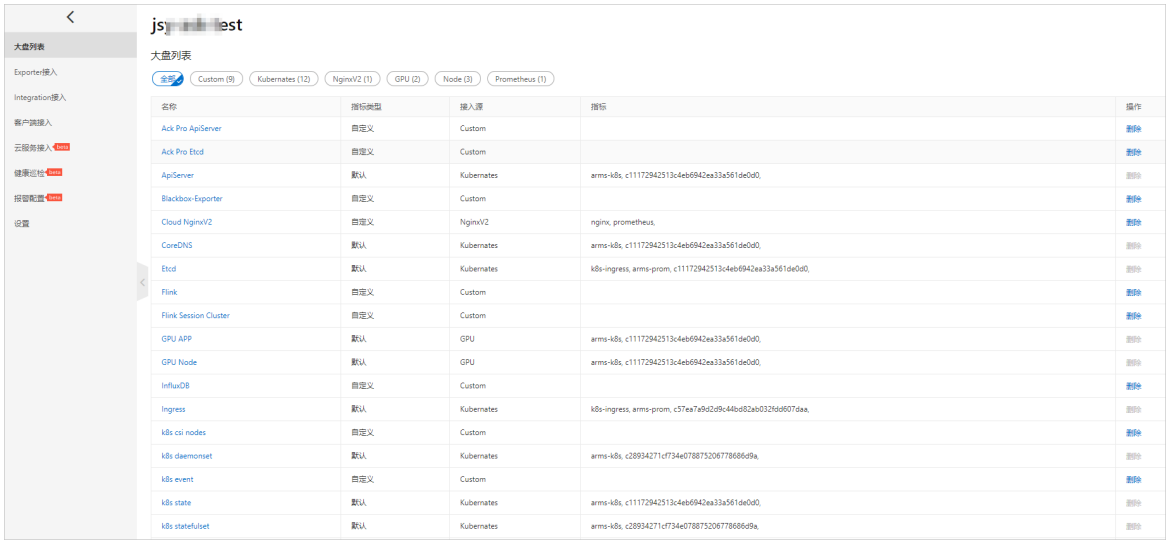
安装prometheus监控组件后，您可以在ARMS控制台的**Prometheus监控**页面查看具体监控信息。

1. 在容器服务管理控制台的左侧导航栏，选择**集群**。
2. 找到要查看监控数据的集群，单击集群名称。
3. 在**集群信息**页面，单击右上角的**Prometheus监控**。
4. 查看Prometheus监控指标。

您可以根据需要切换页签查看各项仪表板展示的监控指标。更多信息，请参见查看Prometheus监控指标。



单击右上角的**跳转到应用实时监控服务 ARMS**，可以跳转到ARMS控制台设置该集群的Prometheus监控大盘。



配置Prometheus监控采集规则

ARMS Prometheus监控兼容并提供三种主流采集规则的实现，包括：

- 标准开源采集规则配置文件prometheus.yaml
- 适合自定义K8s集群内监控的采集规则ServiceMonitor
- 默认采集规则Annotation

三种规则对应的配置方式如下：

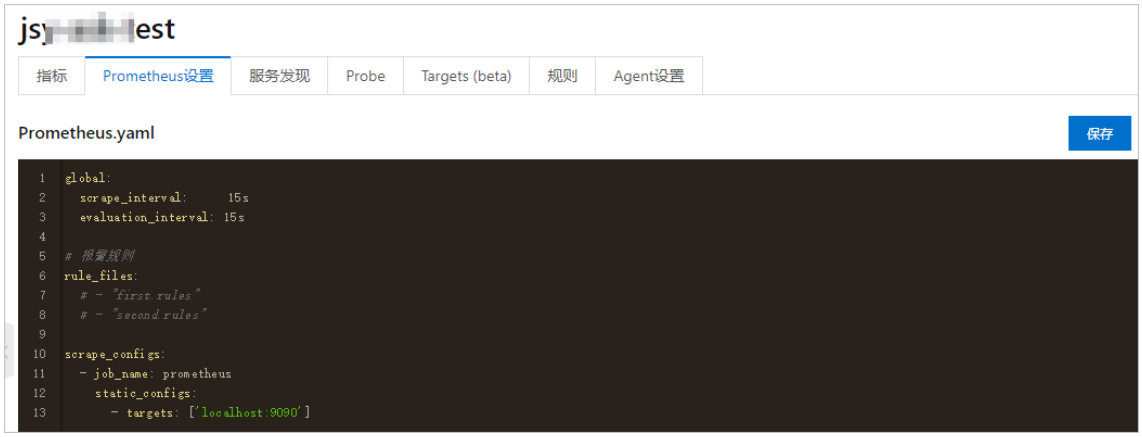
- 编辑prometheus.yaml

  Prometheus监控无需重启，使用prometheus.yaml配置文件即可动态更新采集规则。

  i. 在ARMS控制台的**Prometheus监控**页面。找到要配置的集群，单击**设置**。

  ii. 单击**Prometheus设置**页签。

iii. 编辑prometheus.yaml，然后单击**保存**。



- 添加ServiceMonitor

  添加ServiceMonitor后可以进行K8s集群内应用业务数据的监控。

  i. 在ARMS控制台的**Prometheus监控**页面。找到要配置的集群，单击**设置**。

  ii. 单击**服务发现**页签，然后单击**添加ServiceMonitor**。

  iii. 参考以下示例填写内容，然后单击**确定**。

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
 # 填写一个唯一名称
 name: tomcat-demo
 # 填写目标命名空间
 namespace: default
spec:
 endpoints:
 - interval: 30s
  # 填写Prometheus Exporter对应的Port的Name字段的值
  port: tomcat-monitor
  # 填写Prometheus Exporter对应的Path的值
  path: /prometheus-metrics
 namespaceSelector:
  any: true
 selector:
  matchLabels:
   #填写service.yaml的label字段，用来定位目标service.yaml
   app: tomcat
```

- 添加Annotation

  在应用的YAML文件里，添加Annotation如下：

```
annotations:
      prometheus.io/scrape: "true"
      prometheus.io/port: "9090"
      prometheus.io/path: "/metrics"
```

2.3. 通过Prometheus监控GPU实例

接入Prometheus监控后，您可以通过预定义的大盘监控Kubernetes集群中ECI GPU实例的性能指标。本文主要为您介绍如何在Prometheus中监控ECI GPU实例。

前提条件

已创建ASK集群，且集群已部署了ARMS Prometheus监控。具体操作，请参见ASK接入ARMS Prometheus监控。

操作步骤

1. 登录容器服务管理控制台。
2. 创建一个ECI GPU实例。

   yaml示例如下：

```
apiVersion: v1
kind: Pod
metadata:
 name: cg-gpu-0
 annotations:
  # 指定GPU实例规格
  k8s.aliyun.com/eci-use-specs : "ecs.gn6i-c4g1.xlarge"
spec:
 containers:
 - image: nginx
  name: cg
  resources:
   limits:
    cpu: 500m
    # 指定容器使用的GPU个数
    nvidia.com/gpu: '1'
  command: ["bash","-c","sleep 100000"]
 dnsPolicy: ClusterFirst
 restartPolicy: Always
```

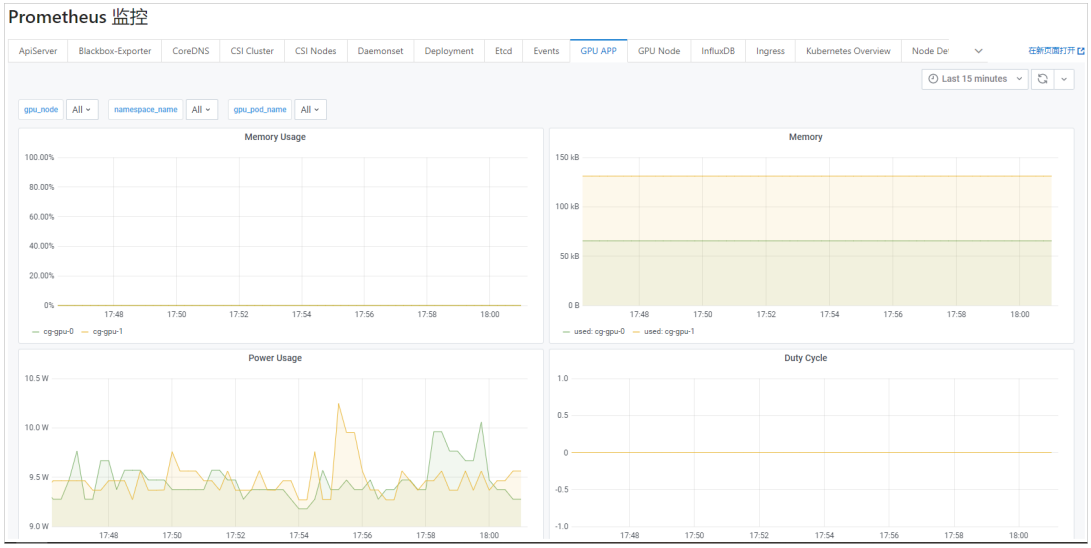
3. 查看GPU指标。
   i. 找到GPU实例所属的集群，单击集群名称。
   ii. 在在**集群信息**页面，单击右上角的**Prometheus监控**。

iii. 单击GPU APP页签或者GPU Node页签，查看对应的监控详情。

ASK集群接入ARMS Prometheus监控后，您无需部署额外插件，即可监控ECI GPU实例。默认情况下，系统已为您提前创建好了对应的监控大盘。

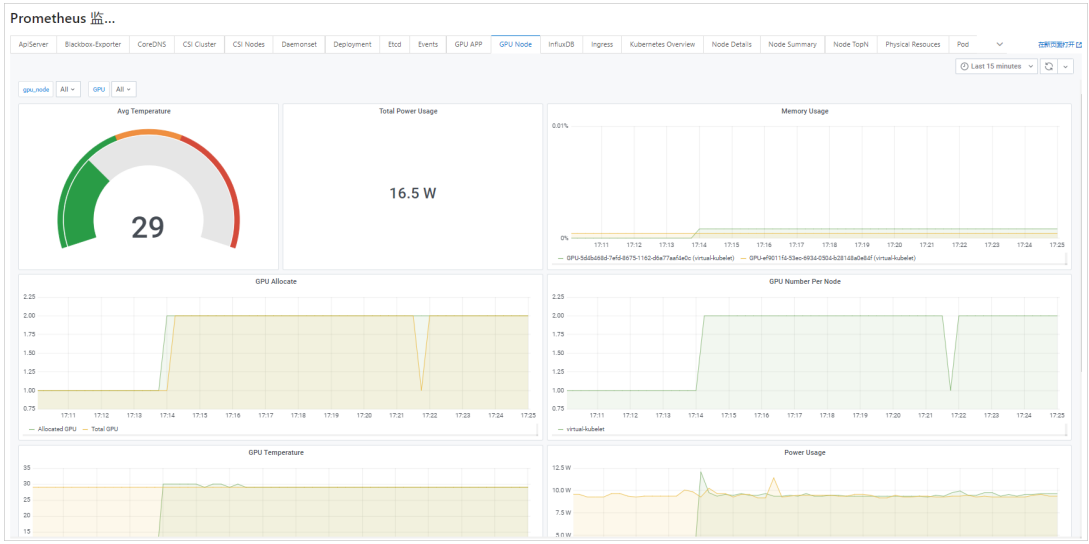
- GPU APP

  在GPU APP大盘中，您可以查看单个Pod对应GPU的数据。如下图所示。



- GPU Node

  在GPU Node大盘中，您可以查看节点上所有GPU的数据。如下图所示。



后续步骤

在使用GPU来做高性能计算时，为了节约成本，您往往需要根据GPU指标（利用率、显存等）来做弹性伸缩。ECI GPU实例支持复用ACK中基于GPU指标的HPA组件，以实现弹性伸缩。具体操作，请参见基于GPU指标实现弹性伸缩。

2.4. 通过Prometheus监控磁盘

在Kubernetes集群中创建ECI实例时，ECI实例会运行在虚拟节点上。由于并不存在真实节点，磁盘是与Pod相关，而不是Node。因此，您需要额外配置Pod级别的磁盘监控，才能监控磁盘相关指标。本文介绍如何在Prometheus中监控ECI实例相关的磁盘。

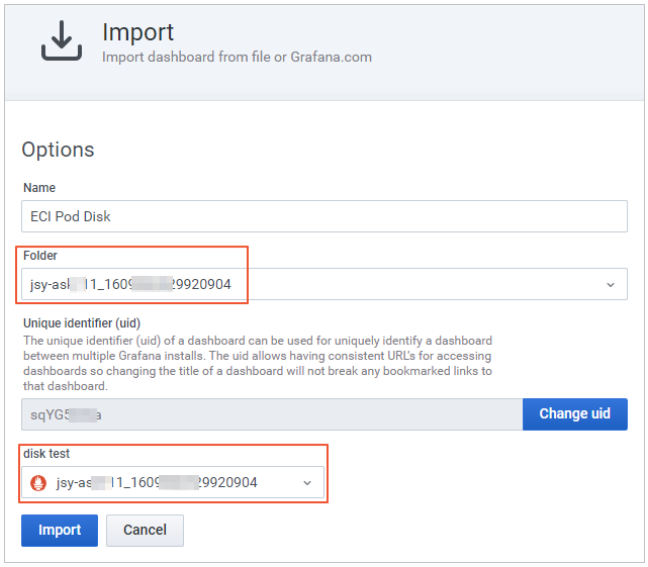
前提条件

已创建ASK集群，且集群已部署了ARMS Prometheus监控。具体操作，请参见ASK接入ARMS Prometheus监控。

操作步骤

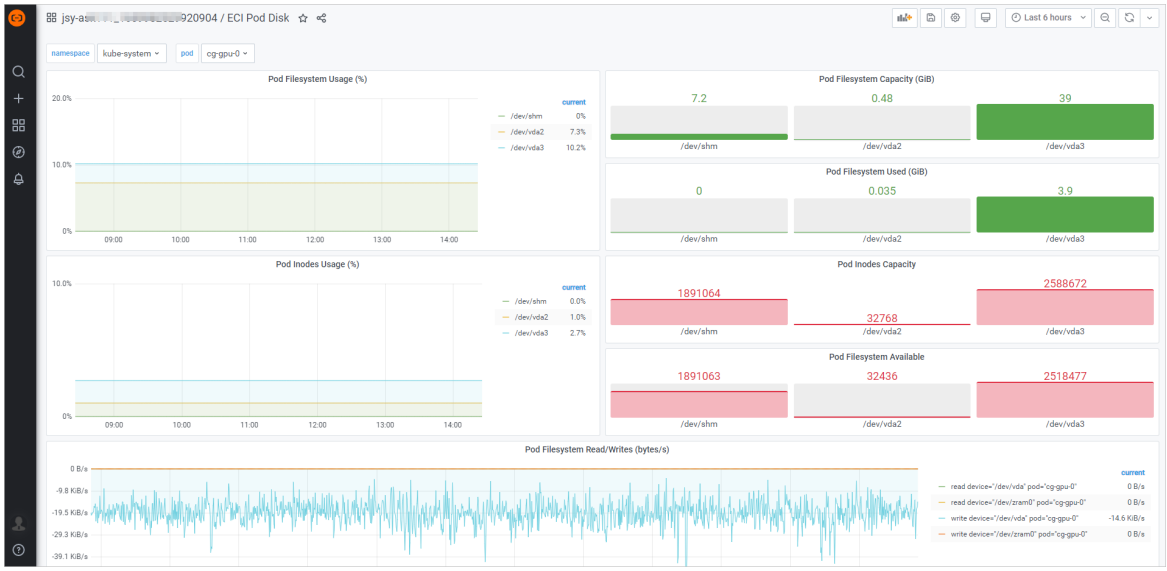
1. 登录容器服务管理控制台。
2. 打开Grafana页面。
   i. 在**集群**页面，找到目标集群，单击集群名称。
   ii. 在**集群信息**页面，单击右上角的**Prometheus监控**。
   iii. 在**Prometheus监控**页面，单击右上角的**在新页面打开**。
3. 在Grafana页面的左侧导航栏，单击图标，选择**Import**。
4. 单击**Upload JSON file**，然后上传JSON文件。

   JSON文件包含大盘相关配置，示例文件请下载ECI Pod Disk。
5. 修改大盘文件夹和数据源为目标集群。



6. 单击**Import**。

   导入成功后，您可以查看对应ECI Pod的磁盘相关监控信息。



在ARMS控制台的**Prometheus监控**页面，您可以看到对应集群的大盘列表中已添加新的自定义磁盘监控大盘。

7. （可选）如果需要筛选查看Deployment级别或者StatefulSet级别的监控，您可以将当前磁盘监控大盘中的对应Panel复制到Deployment或者StatefulSet的大盘中，或者直接在该大盘中修改Dashboard settings。

   在磁盘监控大盘中修改Dashboard settings，添加筛选变量的操作如下：

   i. 在磁盘监控大盘的Grafana页面，单击右上角的图标。

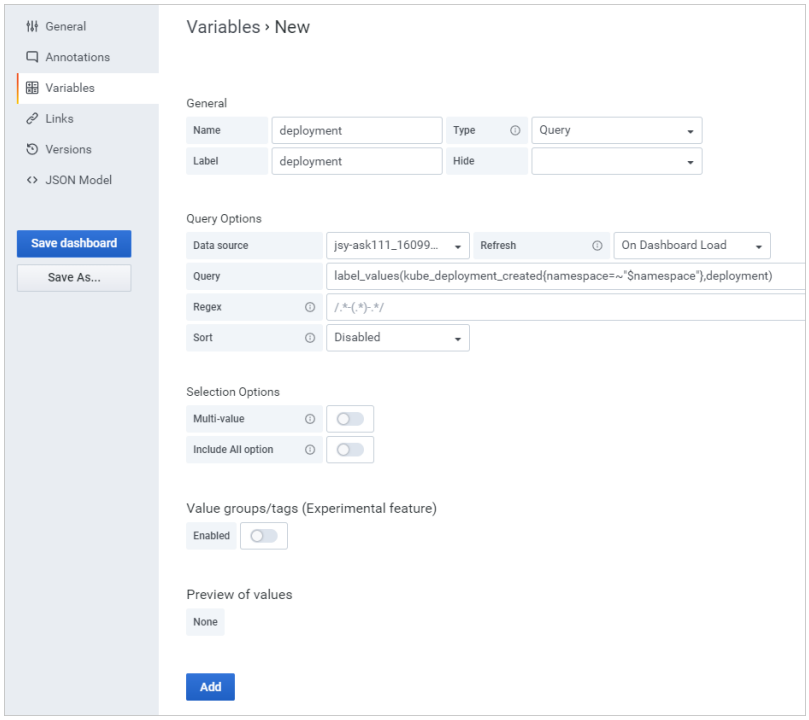
   ii. 在**Settings**页面的左侧导航栏，选择**Variables**。

iii. 单击**New**，编辑General和Query Options相关设置，然后单击**Add**。

添加变量时，请根据需要变量类型设置Query。示例如下表所示。

类型	Query
deployment	label_values(kube_deployment_created{namespace=~"$namespace"},deployment)
statefulset	label_values(kube_statefulset_created{namespace=~"$namespace"},statefulset)

下图以添加deployment变量为例：



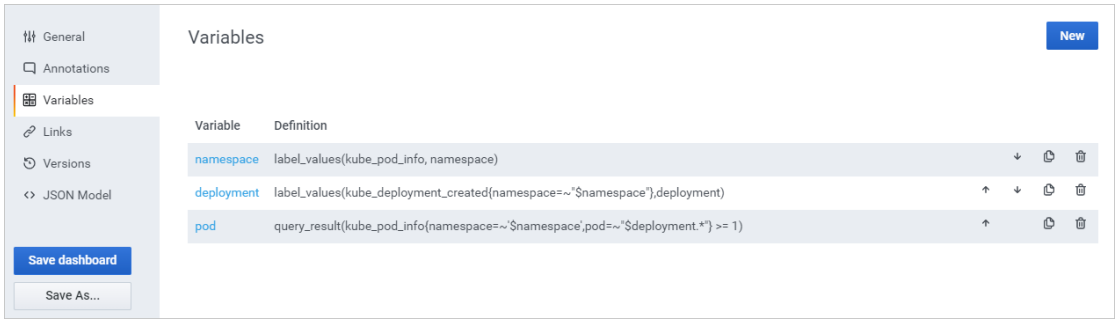
iv. 单击pod变量，在**Edit**页面修改Query，然后单击**Update**。

请根据上一步添加的变量类型修改Query。示例如下表所示。

类型	Pod对应的Query
deployment	query_result(kube_pod_info{namespace=~'$namespace',pod=~"$deployment.*"} >= 1)
statefulset	query_result(kube_pod_info{namespace=~'$namespace',pod=~"$statefulset.*"} >= 1)

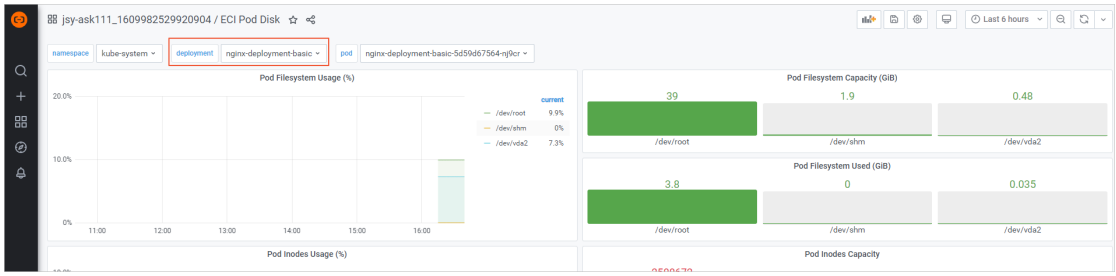
v. 调整deployment变量（或者statefulset变量）与pod变量的顺序，使deployment变量（或者statefulset变量）在pod变量之上。

下图以deployment变量为例：



vi. 返回磁盘监控大盘页面，查看筛选功能是否生效。

下图以生效Deployment级别的筛选功能为例：



3.查看ECI实例监控指标

在弹性容器实例控制台上，您可以查看ECI实例的监控数据，包括CPU、内存和网络等相关指标。本文介绍ECI实例监控数据的含义和计算方式，方便您了解各项指标的具体作用，以便进行二次计算开发。

监控指标概述

在弹性容器实例控制台上查看ECI实例（即容器组）的监控数据时，您可以筛选时间段查看某一小时的数据，或者查看近5分钟的实时数据，支持查看的监控指标如下：

- CPU

  显示CPU核数利用率，对应多核CPU的监控数据总和，上限为ECI实例的CPU核心数。例如：如果CPU核数利用率为150%，表示使用了1.5 vCPU。

- 内存

  显示内存利用率，即实例的内存使用率，上限为100%。

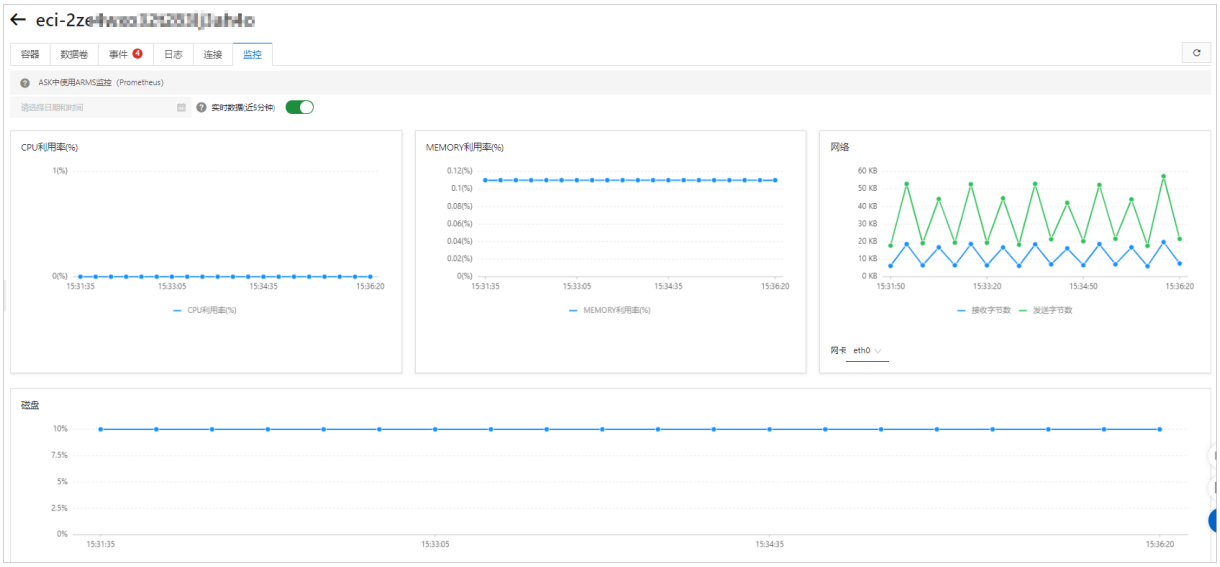
- 网络

  显示接收字节数和发送字节数，即在对应时间窗内累计发送字节数和累计接收字节数。

- 磁盘

  显示磁盘分区和空间数据。具体如下：

  - 磁盘分区数据：包括系统分区和数据分区，其中数据分区为挂载作为数据盘的云盘分区。
  - 磁盘空间数据：包括磁盘的总空间、已使用空间、剩余可用空间和使用率。



对于ECI实例的监控数据，您可以通过DescribeContainerGroupMetric和DescribeMultiContainerGroupMetric接口进行查询，然后进行二次开发计算。查询监控数据时，系统将同时返回容器组，以及容器组内容器的监控数据：

- 返回结构体的根节点Records中包含容器组整体的监控数据（CPU、内存、网络和磁盘数据）。
- 返回结构体的子节点Containers中包含各个容器的监控数据（CPU和内存数据）。

更多信息，请参见DescribeContainerGroupMetric和DescribeMultiContainerGroupMetric。

CPU指标计算方式

调用openAPI接口可以获取的CPU原始数据如下：

名称	类型	示例值	描述
UsageNanoCores	Long	0	CPU在采样窗口内的使用量（纳秒）。
UsageCoreNanoSeconds	Long	70769883	CPU历史使用总量。
Load	Long	0	最近10秒的平均负载情况。
Limit	Long	2000	CPU使用上限（CPU核数*1000）。

CPU相关指标计算方式如下：

- CPU核数利用率=UsageNanoCores/10^9
- CPU利用率=UsageNanoCores/Limit/10^6

内存指标计算方式

调用openAPI接口可以获取的内存原始数据如下：

名称	类型	示例值	描述
AvailableBytes	Long	4289445888	可用内存。
UsageBytes	Long	11153408	已使用内存。
Cache	Long	7028736	缓存。
WorkingSet	Long	5521408	当前内存工作集使用量。
Rss	Long	1593344	常驻内存集，即实际使用的物理内存。

内存相关指标计算方式如下：

内存利用率=WorkingSet/（WorkingSet + AvailableBytes）

网络指标计算方式

调用openAPI接口可以获取的网络原始数据如下：

名称	类型	示例值	描述
TxBytes	Long	1381805699	累计发送字节数。
RxBytes	Long	505001954	累计接收字节数。
TxErrors	Long	0	累计发送错误数。
RxErrors	Long	0	累计接收错误数。
TxPackets	Long	5158427	累计发送包数量。
RxPackets	Long	4800583	累计接收包数量。
TxDrops	Long	0	累计发送丢包数。
RxDrops	Long	0	累计接收丢包数。
Name	String	eth0	网卡名称。

网络相关指标计算方式如下：

- 网络带宽速率（每秒发送比特数，单位为bps）

  网络带宽速率=（B时刻的累计发送字节数-A时刻的累计发送字节数）/A时刻和B时刻之间的秒数*8
- 网络吞吐率（每秒发送包数量，单位为pps）

  网络吞吐率=（B时刻的累计发送包数量-A时刻的累计发送包数量）/A时刻和B时刻之间的秒数