# 阿里云 应用实时监控服务

# Prometheus 监控

文档版本: 20190920

为了无法计算的价值 | [-] 阿里云

### <u>法律声明</u>

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。

- 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。
- 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。
A	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。	道 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 listinstanceid Instance_ID
[]或者[a b ]	表示可选项,至多选择一个。	ipconfig [-all -t]
	表示必选项,至多选择一个。	<pre>swich {stand   slave}</pre>

### 目录

法律声明	I
通用约定	I
1 Prometheus 监控概述	1
2 开始使用 Prometheus 监控	6
3 查看 Prometheus 监控指标	
4 使用教程	
4.1 通过 ARMS Prometheus 监控 JVM	

### 1 Prometheus 监控概述

ARMS Prometheus 监控全面对接开源 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,且提供全面托管的 Prometheus 服务。

### 什么是 Prometheus?

Prometheus 是一套开源的系统监控和报警框架,灵感源自 Google 的 Borgmon 监控系统。2012 年,SoundCloud 的 Google 前员工创造了 Prometheus,并作为社区开源项目进行开发。2015 年,该项目正式发布。2016 年,Prometheus 加入云原生计算基金会(Cloud Native Computing Foundation),成为受欢迎度仅次于 Kubernetes 的项目。

### Prometheus 具有以下特性:

- · 多维的数据模型(基于时间序列的 Key/Value 键值对)
- ・灵活的查询和聚合语言 PromQL
- ·提供本地存储和分布式存储
- · 通过基于 HTTP 的 Pull 模型采集时间序列数据
- · 可利用 Pushgateway (Prometheus 的可选中间件) 实现 Push 模式
- ·可通过动态服务发现或静态配置发现目标机器
- · 支持多种图表和数据大盘

### 为什么要使用 ARMS Prometheus 监控?

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。

ARMS Prometheus 监控具有以下特性:

- · 类型丰富的组件监控
- · 开箱即用的监控大盘
- · 全面托管的 Prometheus 服务

### 类型丰富的组件监控

ARMS Prometheus 监控全面对接 Prometheus 生态,支持数据库、消息、HTTP 等多种类型 组件的监控。



开箱即用的监控大盘

ARMS Prometheus 监控提供若干预先配置的监控大盘,可展示丰富的关键性能指标,包括但不限于:

- · Kubernetes 集群监控
  - Kubelet
  - kube-state-metric
  - api-server

- ・主机监控
  - CPU 使用率
  - 内存使用率
  - 系统负载
  - 磁盘使用量

### - 网络流量

### 图 1-1: 预置监控大盘: 主机详情



### 图 1-2: 预置监控大盘: Kubernetes 概览



### 全面托管的 Prometheus 服务

ARMS Prometheus 监控为您提供全面托管的 Prometheus 服务,成本比自建 Prometheus 监 控系统更低。该服务的特性包括:

- · 提供高可用、可扩展的 Prometheus Server
- · 与阿里云容器服务深度集成
- · 提供监控数据无限存储能力
- · 提供 Prometheus on Prometheus 能力

### 2 开始使用 Prometheus 监控

对于部署在阿里云容器服务 Kubernetes 版中的 Kubernetes 集群,您可以在 ARMS 中为其一键 安装 Prometheus 监控插件,此后即可通过 ARMS 预定义的仪表板监控主机和 Kubernetes 集 群的众多性能指标。

前提条件

- · #unique\_5: 在阿里云容器服务 Kubernetes 版中创建 Kubernetes 集群。
- **#unique\_6**

### 安装 Prometheus 监控插件

- 1. 登录 ARMS 控制台,并在页面左上角选择所需地域。
- 2. 在左侧导航栏中单击 Prometheus 监控。

Prometheus 监控页面会列出您的阿里云账号下在阿里云容器服务 Kubernetes 版中的全部 Kubernetes 集群。

3. 在 Prometheus 监控页面上,单击操作列中的安装,并在提示对话框中单击确认。

安装插件过程中,操作列将以进度条和百分比显示安装进度。

安装插件完毕后,已安装插件列中将显示全部已安装的插件。

### 打开 Prometheus 监控仪表板

- 1. 在左侧导航栏中单击 Prometheus 监控。
- 2. 在 Prometheus 监控页面上,单击已安装插件列中的链接,即可在浏览器新窗口中打开对应的 监控仪表板。

#### 卸载监控插件

如需停止对 Kubernetes 集群的 Prometheus 监控,请按照以下步骤卸载 Prometheus 监控插件。

- 1. 在左侧导航栏中单击 Prometheus 监控。
- 2. 在 Prometheus 监控页面上,单击操作列中的卸载,并在提示对话框中单击确认。

卸载插件完毕后,已安装插件列中的插件将会消失。

#### 后续步骤

#unique\_7

### 更多信息

#unique\_8

## 3 查看 Prometheus 监控指标

ARMS Prometheus 监控提供的预置监控仪表板包括 K8s 集群概览、K8s 部署、Pod 和主机详

- 情,您可以通过这些仪表板查看丰富的 Prometheus 监控指标,并按需更改仪表板数据的时间区
- 间、刷新频率等属性。

前提条件

#unique\_10/unique\_10\_Connect\_42\_section\_cpd\_cg9\_vul

### 打开监控仪表板

安装 Prometheus 监控插件后,即可在 Prometheus 监控页面打开预置的监控仪表板。

- 1. 登录 ARMS 控制台。
- 在左侧导航栏中单击 Prometheus 监控。
   Prometheus 监控页面会列出您的阿里云账号下在阿里云容器服务 Kubernetes 版中的全部 Kubernetes 集群。
- 3. 在 Prometheus 监控页面上,单击已安装插件列中的链接,即可在浏览器新窗口中打开对应的 监控仪表板。

### 查看监控指标

您可以通过以下预置监控仪表板查看监控指标。

### ・ K8s 集群概览仪表板

■ Kubernetes概览 -				() Last	5 minutes Refresh every 30s Q 2
Hostip All 🕶					≡ K8S-Dashboards
√ Total usage					
Cluster men	nory usage	Cluster CPU I	usage (1m avg)	Cluster files	ystem usage
Used	Total	Used	Total	Used	Total
17.58 GiB	45.83 GiB	1.06 cores	24.00 cores	26.58 GiB	707.95 GiB
<ul> <li>Network I/O pressure</li> </ul>					
4 MB/s		Network I	/O pressure		
2 MB/s					
0 B/s					
> Pods CPU usage (/ panel)					
> All processes CPU usage (1 panel)					
> Pods memory usage (1 panel)					
> All processes memory usage (1 panel)					

该仪表板展示的监控指标主要包括:

- 总体使用量信息:例如集群 CPU 使用率、集群内存使用率、集群文件系统使用率。
- CPU 信息:例如 Pod CPU 使用率、全部进程 CPU 使用率。
- 内存信息:例如 Pod 内存使用率、全部进程内存使用率。
- 网络信息:例如网络 I/O 压力、Pod 网络 I/O、所有进程网络 I/O。

### ・K8s 部署仪表板

■ Kubernetes部署 -				0	Last 1 hour Refresh every 30s Q 2
Deployment All  Statefulset All	Daemonset All      Pod flexw	Pod_ip 192.168.0.122	•		≡ K8S-Dashboards
~ Overview					
Deployment m	emory usage	Deploymer	1 CPU usage	Unavail	0%
Used	Total	Used	Total	Available (cluster)	Total (cluster)
56.7 MiB	7.64 GiB	0.000014 cores	0.3 cores	42	42
✓ Detail					
0.13		CPU	usage		
0.12					rqst: flexvolume-qix9w 0.100 0.100
0.11					
8 0.10					
0.09					
0.08					

该仪表板展示的监控指标主要包括:

- 概览: 部署 CPU 使用率、部署内存使用率、不可用副本数。
- 详情: CPU 使用率、内存使用率、全部进程网络 I/O。

### ・ Pod 仪表板

🗱 Kubernetes容器副本 🗸			🕑 Last 30 minutes Refresh every 30s 🛛 Q 🛛 🎜		
Pod kube-apiserver-cn-hangzhou.192.168.0.119 • Pod	lp 192.168.0.119 ▼		$\equiv$ K8S-Dashboards		
✓ Pod Info					
Pod IP Address 192.168.0.119	Pod Status Running	Pod Container kube-apiserver	Container restarts O		
✓ Network I/O pressure					
122.02/-	Network I	I/O pressure			
50.8/s 0 0 s - 0.0 kB/s - 10 k					
Pod memory usage Pod CPU usage					
Pod Mem Used	Machine Mem Total	Pod CPU Usage Secs	Machine CPU Usage Secs		
2.307 GiB	7.64 GiB	176 ms	1.4 s		

该仪表板展示的监控指标主要包括:

- Pod 基本信息: Pod IP 地址、Pod 状态、Pod 容器、容器重启次数。
- 总体使用量信息:例如 Pod CPU 使用率、Pod 内存使用率。
- CPU 信息:例如 Pod CPU 使用率、全部进程 CPU 使用率。
- 内存信息:例如 Pod 内存使用率、全部进程内存使用率。
- 网络信息:例如网络 I/O 压力、Pod 网络 I/O、所有进程网络 I/O。

### ・主机详情仪表板

👪 主机详情 🛛									O Last 5 minutes	ବ ଅ
interval auto 🕶	节点 192.168.0.112:9100 ▼									
				192.168.0.112:9100						
<sup>系统运行时间</sup> 1.3 day	CPU 根數 4 内存总量 7.64 GiB	(5m) CPU low	ait (5m)	内存使用案	当前打开的文件:	描述符	R)	区使用率 3%	t 最大分区(/etc/hostnam	e)使用率
	系统平均	的负载		R	盘总空间			各分区可	用空间	
0.40			max avg current			c	文件系统	分区	可用空间	使用率
0.30		= 1m 0.	3400 0.1725 0.0300 3700 0.3042 0.2100		<ul> <li>/etc/hostname</li> <li>/etc/hosts</li> </ul>	118	ext4	/host/root	108.32 GiB	3 95%
0.20		_ 15m 0.	3900 0.3675 0.3300		<ul> <li>/etc/resolv.conf</li> </ul>	118				
0.20 /					- ///ost//oot	110	ext4	/etc/resolv.conf	108.32 GiB	3.95%
0.10		_					ext4	/etc/hosts	108.32 GiB	3.95%
0 19:06	19:07 19:08 19:09	19:10					ext4	/etc/hostname	108.32 GiB	3.95%
i		CPU使用率、磁盘每秒的I/O	操作耗费时间(%)					内存自	言息	
6.00%					max avg	current	9 GIB			
5.00%				- System	2.88% 2.73%	2.82%	7 GiB			
4.00%				<ul> <li>Oser</li> <li>Iowait</li> </ul>	0.35% 0.27%	0.33%				
3.00%				vda_每秒I/O操作%	1.30% 1.06%	1.30%	5 GiB			
2.00%							2 GIB			
1.00%										
							0 B 19:06	19:07 19	19:09	19:10
19:06:0	0 19:06:30 19:07:00 19:07:30 19:	08:00 19:08:30 19:09:00	19:09:30 19:10:00 19:10:	30			— 总内存 Current	: 7.64 GIB — 已用 Current: 1.	.41 GIB — 町用 Current: 6.23 GIB	3

该仪表板展示的监控指标主要包括:

- CPU 信息:例如 CPU 核数、CPU 使用率、CPU I/O 等待时间。
- 内存信息:例如内存总量、内存使用率。
- 磁盘信息:例如磁盘总空间、磁盘 IO 读写时间、磁盘读写速率。
- 网络信息:例如网络流量、TCP 连接情况。

#### 仪表板常用操作

对仪表板的常用操作如下所示。



### 1. 切换仪表板

该下拉菜单显示当前查看的仪表板名称,并可用于切换至下拉菜单中的其他仪表板。您可以通过 在顶部搜索栏中输入名称来查找仪表板,或者使用 Filter(筛选条件)在下拉菜单中筛选出带有 指定 Tag(标签)的仪表板。



#### 2. 设置时间区间和刷新频率

单击此图标后,您可以在浮层中选择预定义的监控数据相对时间区间,例如过去5分钟、过去 12小时、过去30天等,也可以通过设置时间起点和终点来设置自定义的绝对时间区间。此 外,您可以在浮层中设置仪表板的刷新频率。

				O Last !	5 minutes Q	<b>с</b>
	Quick ranges					
	Last 2 days Last 7 days Last 30 days Last 90 days Last 6 months Last 1 year Last 2 years Last 5 years	Yesterday Day before yesterday This day last week Previous week Previous month Previous year	Today Today so fa This week This week s This month This month This year This year so	r sofar sofar ofar	Last 5 minutes Last 15 minutes Last 30 minutes Last 1 hour Last 3 hours Last 6 hours Last 12 hours Last 24 hours	
	Custom range					
系	From: now-5m				Ê	
	now					
	Refreshing every:			¥	Apply	

3. 扩大时间区间

每单击一次该扩大按钮,时间区间就会扩大为当前的两倍,且时间起点迁移和终点后移的幅度相等。例如,假设当前选择的时间区间为过去 10 分钟,则单击一次该过大按钮后,时间区间的前 面和后面将会各延长 5 分钟。

4. 手动刷新

单击此按钮将会刷新当前仪表板中所有面板的监控数据。

5. 筛选监控数据

选择此下拉菜单中的选项即可筛选当前仪表板显示的监控数据。

仪表板面板常用操作

单击面板顶部的下拉菜单后,可进行以下操作:

CPU usage 🔻	
View v	
产 Share 🔤 p s	
😥 More 🕨	Panel JSON
	Export CSV
	Toggle legend 🛛 📼 p l

- ・全屏查看当前面板: 单击 View, 或按快捷键 V。再次按快捷键 V 或 Esc 即可退出全屏模式。
- · 分享当前面板:单击 Share,或依次按下 P 和 S 打开分享对话框,获得当前面板的分享链接、 嵌入链接或快照链接。
- · 获得当前面板的 JSON 代码:选择 More > Panel JSON,然后在 JSON 对话框中拷贝 JSON 代码。
- · 将当前面板的数据导出为 CSV 文件:选择 More > Export CSV,然后在 Export CSV 对话框 中设置导出格式并导出。
- · 打开或关闭图例:选择 More > Toggle legend,或依次按下 P 和 L 即可切换图例的可见性。

#unique\_8

#unique\_10/unique\_10\_Connect\_42\_section\_cpd\_cg9\_vul

### 4 使用教程

### 4.1 通过 ARMS Prometheus 监控 JVM

通过在应用中埋点来暴露 JVM 数据,使用 ARMS Prometheus 监控抓取 JVM 数据,并借助 ARMS Prometheus Grafana 大盘来展示 JVM 数据,即可实现通过 ARMS Prometheus 监控 JVM 的目的。

前提条件

- ・ 下载 Demo 工程
- **#unique\_13**
- ・ 创建容器镜像

#### 背景信息

### 本教程的操作流程如图所示。



### 步骤一:通过埋点暴露 JVM 数据

首先需要在应用中使用 JVM Exporter 暴露 JVM 数据。

1. 在 pom.xml 文件中添加以下依赖。

```
<dependency>
    <groupId>io.prometheus</groupId>
    <artifactId>simpleclient_hotspot</artifactId>
    <version>0.6.0</version>
```

</dependency>

2. 在可以执行初始化的位置添加对 JVM Exporter 的依赖。

例如 Demo 工程的 \src\main\java\com\monitise\prometheus\_demo\DemoContro

ller.java 文件。

```
@PostConstruct
    public void initJvmExporter() {
        io.prometheus.client.hotspot.DefaultExports.initialize();
    }
```

3. 在 \src\main\resources\application.properties 文件中暴露用于 Prometheus 监

```
控的端口 (Port) 和路径 (Path)。
```

management.port: 8081
endpoints.prometheus.path: prometheus-metrics

步骤二:将应用部署至阿里云容器服务 K8s 集群

其次需要将应用部署至容器服务 K8s 集群,以便 ARMS Prometheus 监控抓取 JVM 数据。

1. 逐行运行 buildDockerImage.sh 中的以下命令。

```
mvn clean install -DskipTests
docker build -t <本地临时Docker镜像名称>:<本地临时Docker镜像版本号> . --no
-cache
sudo docker tag <本地临时Docker镜像名称>:<本地临时Docker镜像版本号> <
Registry域名>/<命名空间>/<镜像名称>:<镜像版本号>
sudo docker push <Registry域名>/<命名空间>/<镜像名称>:<镜像版本号>
```

例如:

mvn clean install -DskipTests
docker build -t promethues-demo:v0 . --no-cache
sudo docker tag promethues-demo:v0 registry.cn-hangzhou.aliyuncs.com
/fuling/promethues-demo:v0
sudo docker push registry.cn-hangzhou.aliyuncs.com/fuling/promethues
-demo:v0

此步骤会构建 Docker 镜像,并将镜像推送至阿里云 Docker Registry。

2. 登录容器服务 Kubernetes 版控制台。

3. 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控制台。

■ (-)阿里云	账号全部资源 🖌 🛞 全球	Q 搜索		费用 工单	备案 企业 支	持与服务 🖸 🗳	© #	🏫 简体中文 📀
容器服务 - Kubernetes	• • • • • • • • • • • • • • • • • • •	87	1	G-10120084-1910.				不再显示
概范	▲ 集群列表					查看当前集群与节点配额。	RIST	创建 Kubernetes 集群
<ul> <li>◆ 集群</li> <li>集群</li> </ul>	② 如何创建集群 ③ 创建 GPU 集群 ③ 如何接入 Kubernetes 集群 ④ 御	◎ 扩容和缩容集群 ◎ 通过 kubecti 连接 Kub を工单	ernetes 集群 🔗 通过命令管理应用 🔗 VPC 🏾	F Kubernetes 的网络地址段规划		敗 Ø 授权管理 Ø 敗集 K	.bernetes 诊断信	息
节点	名称 ▼ demo	标签						
存储卷	集群名称/ID	标签 集群类型	地域(全部)▼ 网络类型	集群状态 节点	i个数 创建时间	版本		操作
命名空间 授权管理	arms-demo-	Kubernetes	华北2 虚拟专有网络 vpc-	●运行中 5	2019-07-30 16:44:35	) 1.12.6-aliyun.1	管理	至 2 控制台 来#授 容   更多 <del>-</del>

 4. 在左侧导航栏选择工作负载 > 部署,在页面右上角单击创建,并在使用文本创建页签上填写以下 内容。

📕 说明:

以下配置文件中的 prometheus.io/port 和 prometheus.io/path 的值分别为步骤一:通

过埋点暴露 JVM 数据中暴露的 Prometheus 监控端口和路径。

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: prometheus-demo
spec:
  replicas: 2
  template:
    metadata:
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/path: '/prometheus-metrics'
        prometheus.io/port: '8081'
      labels:
        app: tomcat
    spec:
      containers:
       name: tomcat
        imagePullPolicy: Always
        image: registry.cn-hangzhou.aliyuncs.com/fuling/promethues-
demo:v0
        ports:
        - containerPort: 8080
          name: tomcat-normal
        - containerPort: 8081
          name: tomcat-monitor
```

此步骤将步骤1中的 Docker 镜像部署至容器服务 K8s 集群中。

5. 在左侧导航栏选择服务发现与负载均衡 > 服务,在页面右上角单击创建,并在使用文本创建页签

上填写以下内容。

```
apiVersion: v1
kind: Service
metadata:
labels:
name: tomcat
```

```
name: tomcat
namespace: default
spec:
ports:
- name: tomcat-normal
port: 8080
protocol: TCP
targetPort: 8080
- name: tomcat-monitor
port: 8081
protocol: TCP
targetPort: 8081
type: NodePort
selector:
app: tomcat
```

步骤三:配置 ARMS Prometheus 监控以抓取 JVM 数据

接下来需要在 ARMS 控制台配置 ARMS Prometheus 监控以抓取 JVM 数据。

- 1. 登录 ARMS 控制台。
- 2. 在左侧导航栏中单击 Prometheus监控。
- 3. 在 Prometheus监控页面顶部选择容器服务 K8s 集群所在的地域,并在目标集群右侧的操作列 单击安装。
- 4. ARMS Prometheus Agent 安装完毕后,在目标集群右侧的操作列单击设置。
- 5. 在配置详情页签上单击添加ServiceMonitor,在新增ServiceMonitor对话框中填写以下内容。

```
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:
     app: tomcat
```

步骤四:通过 Grafana 大盘展示 JVM 数据

最后需要在 ARMS 控制台导入 Grafana 大盘模板并指定 Prometheus 数据源所在的容器服务 K8s 集群。

1. 打开 ARMS Prometheus Grafana 大盘概览页。

 在左侧导航栏中选择 + > Import,并在 Grafana.com Dashboard 文本框输入 10877,然后 单击 Load。

Import Import dashboard from file or Grafana.com	
Grafana.com Dashboard	Upload .json file
Paste Grafana.com dashboard url or id Or paste JSON	
ELoad	

3. 在 Import 页面输入以下信息,然后单击 Import。

Import Import dashboard fr	rom file or Grafana.com
Importing Dashboard from <u>Graf</u>	fana.com
Published by	liguozhong
Updated on	2019-09-18 16:12:38
Options	
Name 1	Prometheus JVM Overview ARMS
Folder	arms-demo
Unique identifier (uid)	GHxpl 🗸
arms-demo	arms-demo-
	3
Import	Cancel

- a) 在 Name 文本框中输入自定义的大盘名称。
- b) 在 Folder 下拉框中选择您的阿里云容器服务 K8s 集群。
- c) 在 Select a Prometheus data source 下拉框中选择您的阿里云容器服务 K8s 集群。

#### 预期结果

Ø	🖁 arms-demo-lange and the second state of the	📫 슈 岱 啓 拳  Olasi3 hours - Q 💭 305 -
+	Namespace default • app tomcat • pod promethues-tomcat-demo- • • • 内存部分	
#	每分钟OC的时间 10 s 500 ms 0 ns -1.0 s -1.0 s -1.3 0 13.0 14.00 14.30 15.00 15.00 16.00 - PS MarkSveep [ 931] - PS Sciencede [ 1 10051]	Memory used min max mg current — heap 36 MB 535 MB 333 MB 60 MB — nonheap 63 MB 63 MB 63 MB 63 MB 1330 1400 1430 1500 1530 1600
	JVVM memory used by pool 600 Mi 200 Mi 201 0 113.00 14.00 14.30 15.00 15.30 - (pool=*D6 Sarrior Space*) - (pool=*D6 Sarrior Space*) - (pool=*D6 Sarrior Space*) - (pool=*D6 Sarrior Space*) - (pool=*D6 Sarrior Space*)	JVM memory committed by pool           400 Mit           400 Mit           200 Mit           0           15:00           0           15:00           0           15:00           0           0           13:30           14:00           16:00           0           15:00           15:00           15:00           15:00           0           0           0           10:00*PB Store Space*)           (pool*PB Store Space*)
-	> 鉄程运行部分 (0 panels)	Threads 1400 1430 1500 1530 1600 - deadock [[contat]]
?	→ <b>GC</b> 和文件操作网络部分	Perkana adhada ku ga

### 配置完毕后的 ARMS Prometheus Grafana JVM 大盘如图所示。

#### 后续步骤

ARMS Prometheus Grafana JVM 大盘配置完毕后,您可以查看 Prometheus 监控指标和进一 步自定义大盘,详见相关文档。 相关文档 #unique\_15