# K8S中的监控和日志

**Monitoring and Logging in Kubernetes**

[3 K8S中的监控和日志 1](#__RefHeading___Toc1050_758888336)

[3.1 指标与日志 1](#__RefHeading___Toc1030_849562681)

[3.2 监控技术 1](#__RefHeading___Toc1147_849562681)

[3.3 监控模式 1](#__RefHeading___Toc1149_849562681)

[3.4 指 3](#__RefHeading___Toc1151_849562681)

[3.5 指 3](#__RefHeading___Toc1153_849562681)

[3.6 指 3](#__RefHeading___Toc1155_849562681)

[3.7 指 3](#__RefHeading___Toc1157_849562681)

[3.8 指 3](#__RefHeading___Toc1159_849562681)

[3.9 指 3](#__RefHeading___Toc1161_849562681)

在本章中，我们讨论了在Kubernetes中进行监控和日志的最佳实践。 我们将深入探讨各种监控模式，要收集的重要指标以及根据这些原始指标构建仪表板的细节。 然后，我们总结一些为Kubernetes集群实施监控的示例。

## 指标与日志

您首先需要了解日志收集和指标收集之间的区别。 它们彼此互补，但具有不同的目的。

* 指标：一段时间内测得的一系列数字
* 日志：用于系统的探索性分析

当应用程序性能不佳时，需要同时使用指标和日志记录的一个示例。 我们对该问题的第一个指示可能是在托管应用程序的Pod上出现高延迟的警报，但是这些指标可能无法很好地指示该问题。 然后，我们可以查看日志以调查从应用程序发出的错误。

## 监控技术

黑匣子监控着重于从应用程序外部进行监控，这是监控系统中的CPU，内存，存储等组件的传统方法。 黑盒监控对于在基础架构级别进行监控仍然很有用，但是缺乏对应用程序运行方式的见识和上下文。 例如，要测试集群是否健康，我们可以安排一个Pod，如果成功，我们知道调度程序和服务发现在集群中是健康的，因此我们可以假设集群组件是健康的。

白盒监控侧重于应用程序状态上下文中的详细信息，例如HTTP请求总数，500错误的数量，请求的延迟等等。 通过白盒监控，我们可以开始了解系统状态的“原因”。 它允许我们问：“为什么磁盘已满？” 而不仅仅是“磁盘已满”。

## 监控模式

您可能会看着监控说：“这有多难？我们一直在监控我们的系统。”是的，您今天使用的一些典型监控模式也适合您监控Kubernetes的方式。区别在于，像Kubernetes这样的平台具有更大的动态性和瞬态性，您需要改变如何监控这些环境的想法。例如，在监控虚拟机（VM）时，您希望VM处于24/7状态，并保留其所有状态。在Kubernetes中，Pod可能是非常动态且短暂的，因此您需要进行适当的监控以处理这种动态和短暂的特性。

监控分布式系统时，有两种不同的监控模式需要关注。

由Brendan Gregg推广的USE方法着重于以下方面：

* U—利用率（Utilization）
* S—饱和度 （Saturation）
* E-错误 （Errors）

此方法专注于基础结构监控，因为将其用于应用程序级监控存在局限性。 USE方法描述为“对于每种资源，请检查利用率，饱和度和错误率。”此方法使您可以快速确定系统的资源限制和错误率。例如，要检查群集中节点的网络运行状况，您将需要监控利用率，饱和度和错误率，以便能够轻松识别网络堆栈中的任何网络瓶颈或错误。 USE方法是较大工具箱中的工具，不是您将用来监控系统的唯一方法。

Tom Willke推广了另一种称为RED方法的监控方法。 RED方法的方法集中在以下方面：

* R-费率 （Rate）
* E-错误（Errors）
* D—持续时间（Duration）

该理念取自Google的四个黄金信号：

* 延迟Latency（处理请求需要多长时间）
* 流量Traffic（系统上有多少需求）
* 错误Errors（失败的请求率）
* 饱和度Saturation（如何利用您的服务）

例如，您可以使用此方法监控在Kubernetes中运行的前端服务，以计算以下内容：

* 我的前端服务正在处理多少个请求？
* 服务的用户收到多少个500错误？
* 服务是否被请求过度使用？

从前面的示例中可以看到，此方法更着重于用户的体验以及他们对服务的体验。

假设USE方法关注基础结构组件，而RED方法关注监控应用程序的最终用户体验，则USE和RED方法是互补的。

## Kubernetes指标概述

现在我们知道了不同的监控技术和模式，下面让我们看看您应该在Kubernetes集群中监控哪些组件。 Kubernetes集群由控制平面组件和工作节点组件组成。 控制平面组件由API服务器，etcd，调度程序和控制器管理器组成。 工作节点由kubelet，容器运行时，kube-proxy，kube-dns和pod组成。 您需要监控所有这些组件，以确保群集和应用程序正常运行。

Kubernetes以多种方式公开这些指标，因此让我们看一下可用于在集群中收集指标的不同组件。

### cadvisor

Container Advisor或cAdvisor是一个开源项目，它为节点上运行的容器收集资源和指标。 cAdvisor内置在Kubernetes kubelet中，该kubelet在集群中的每个节点上运行。 它通过Linux控制组（cgroup）树收集内存和CPU指标。 如果您不熟悉cgroup，则它是Linux内核功能，可隔离CPU，磁盘I / O或网络I / O的资源。 cAdvisor还将通过内置于Linux内核中的statfs收集磁盘指标。 这些是您无需真正担心的实施细节，但是您应该了解这些指标的显示方式以及可以收集的信息类型。 您应该将cAdvisor视为所有容器指标的真实来源。

### Metrics Server

Kubernetes指标服务器和Metrics Server API替代了已弃用的Heapster。 Heapster在实现数据接收器方面存在一些体系结构方面的弊端，这在Heapster核心代码库中引起了许多供应商解决方案。通过在Kubernetes中实现资源和自定义指标API作为聚合API来解决此问题。这允许在不更改API的情况下切换实现。

在Metrics Server API和Metrics服务器中需要了解两个方面。

首先，Resource Metrics API的规范实现是指标服务器。指标服务器收集诸如CPU和内存之类的资源指标。它从kubelet的API收集这些指标，然后将它们存储在内存中。 Kubernetes在调度程序，Horizo​​ntal Pod Autoscaler（HPA）和Vertical Pod Autoscaler（VPA）中使用这些资源指标。

其次，自定义指标API允许监视系统收集任意指标。这允许监视解决方案构建自定义适配器，从而允许扩展到核心资源指标之外。例如，Prometheus构建了第一个自定义指标适配器，它使您可以基于自定义指标使用HPA。这可以根据您的用例提供更好的扩展，因为现在您可以基于Kubernetes外部的度量引入诸如队列大小和扩展之类的度量。

现在有了标准化的Metrics API，这为扩展到普通的旧CPU和内存指标之外提供了许多可能性。

### kube-state-metrics

kube-state-metrics是一个Kubernetes插件，用于监视存储在Kubernetes中的对象。 在使用cAdvisor和指标服务器提供有关资源使用情况的详细指标的情况下，kube-state-metrics专注于确定部署到集群的Kubernetes对象的条件。

以下是kube-state-metrics可以为您解答的一些问题：

* Pods
  + 集群中部署了多少个Pod？
  + 有多少个Pod处于待处理状态？
  + 是否有足够的资源来满足Pod请求？
* Deployments
  + 相对于期望状态，有多少个Pod处于运行状态？
  + 有多少个副本可用？
  + 哪些Deployments已更新？
* Nodes
  + 我的worker节点的状态如何？
  + 群集中可分配的CPU core是什么？
  + 是否有不可调度的节点？
* Jobs
  + 什么时候开始工作？
  + 工作何时完成？
  + 有多少工作失败了？

在撰写本文时，kube-state-metrics跟踪了22种对象类型。 这些总是在扩展，您可以在Github存储库中找到文档。

## 要监控哪些指标？

一个简单的答案是“一切”，但是如果您尝试监视太多，则会产生过多的噪声，从而滤除您需要深入洞察的真实信号。 在考虑使用Kubernetes进行监视时，我们希望采取一种考虑以下因素的分层方法：

* 物理或虚拟节点
* 集群组件
* 集群附加组件
* 最终用户应用

使用这种分层的监视方法，您可以更轻松地在监视系统中识别正确的信号。 它使您可以使用更有针对性的方法来解决问题。 例如，如果您的Pod处于挂起状态，则可以从节点的资源利用率开始，如果一切正常，则可以将群集级别的组件作为目标。

以下是您要在系统中定位的指标：

* 物理或虚拟节点
  + CPU利用率
  + 内存利用率
  + 网络利用率
  + 磁盘利用率
* 集群组件
  + etcd延迟
  + 集群附加组件
  + 集群自动缩放器
  + 入口控制器
* 应用
  + 容器内存利用率和饱和度
  + 容器CPU利用率
  + 容器网络利用率和错误率
  + 特定于应用程序框架的指标

## 监控工具

有许多可以与Kubernetes集成的监视工具，并且每天都有更多的监视工具，这些工具基于它们的功能集可以更好地与Kubernetes集成。 以下是一些与Kubernetes集成的流行工具：

* Prometheus

Prometheus是最初在SoundCloud上构建的开源系统监视和警报工具包。自2012年成立以来，许多公司和组织都采用了Prometheus，该项目拥有非常活跃的开发人员和用户社区。现在，它是一个独立的开源项目，并且独立于任何公司进行维护。为了强调这一点并阐明项目的治理结构，Prometheus于2016年加入了Cloud Native Computing Foundation（CNCF），这是继Kubernetes之后的第二个托管项目。

* InfluxDB

InfluxDB是一个时序数据库，旨在处理较高的写入和查询负载。它是TICK（Telegraf，InfluxDB，Chronograf和Kapacitor）堆栈的组成部分。 InfluxDB旨在用作涉及大量时间戳数据的任何用例的后备存储，包括DevOps监控，应用程序指标，IoT传感器数据和实时分析。

* Datadog

Datadog为云级应用程序提供监视服务，并通过基于SaaS的数据分析平台监视服务器，数据库，工具和服务。

* Sysdig

Sysdig Monitor是一个商业工具，可为容器本机应用程序提供Docker监视和Kubernetes监视。 Sysdig还允许您通过直接Kubernetes集成来收集，关联和查询Prometheus指标。

云提供商工具

* GCP Stackdriver

Stackdriver Kubernetes Engine监视旨在监视Google Kubernetes Engine（GKE）群集。它一起管理监视和日志记录服务，并具有一个界面，该界面提供了针对GKE集群定制的仪表板。 Stackdriver Monitoring提供了对基于云的应用程序的性能，正常运行时间和整体运行状况的可见性。它从Google Cloud Platform（GCP），Amazon Web Services（AWS），托管正常运行时间探测和应用程序工具收集指标，事件和元数据。

* Microsoft Azure Monitor for containers

用于容器的Azure Monitor是一项功能，旨在监视部署到Azure容器实例或托管在Azure Kubernetes Service上的托管Kubernetes群集的容器工作负载的性能。监视容器至关重要，尤其是在使用多个应用程序大规模运行生产集群时。通过容器的Azure Monitor，可以通过Metrics API从Kubernetes中可用的控制器，节点和容器中收集内存和处理器指标，从而使您可以查看性能。还将收集容器日志。从Kubernetes群集启用监视后，将通过适用于Linux的Log Analytics代理的容器化版本为您自动收集指标和日志。

* AWS Container Insights

如果您在Amazon EC2上使用Amazon Elastic Container Service（ECS），Amazon Elastic Kubernetes Service或其他Kubernetes平台，则可以使用CloudWatch Container Insights从容器化的应用程序和微服务中收集，汇总和汇总指标和日志。这些指标包括对CPU，内存，磁盘和网络等资源的利用率。 Container Insights还提供诊断信息，例如容器重新启动失败，以帮助您隔离问题并快速解决它们。

查看实施监视指标的工具时，一个重要方面是查看如何存储指标。 为时序数据库提供键/值对的工具将为您提供度量标准的更高级别的属性。

始终评估您已经拥有的监视工具，因为使用新的监视工具会产生学习曲线，并且由于该工具的操作实现而导致成本。 现在，许多监视工具已集成到Kubernetes中，因此请评估您今天拥有的工具以及它们是否会满足您的要求。

## 使用Prometheus监控Kubernetes

在本节中，我们重点介绍通过Prometheus监控指标，该指标与Kubernetes标签，服务发现和元数据提供了良好的集成。 我们在本章中贯彻的高级概念也将适用于其他监视系统。

Prometheus是由CNCF托管的开源项目。 它最初是由SoundCloud开发的，其许多概念都基于Google的内部监控系统BorgMon。 它使用密钥对实现了多维数据模型，该密钥对的工作方式与Kubernetes标签系统的工作方式非常相似。 Prometheus以人类可读的格式公开指标，如以下示例所示：

*# HELP node\_cpu\_seconds\_total Seconds the CPU is spent in each mode.*

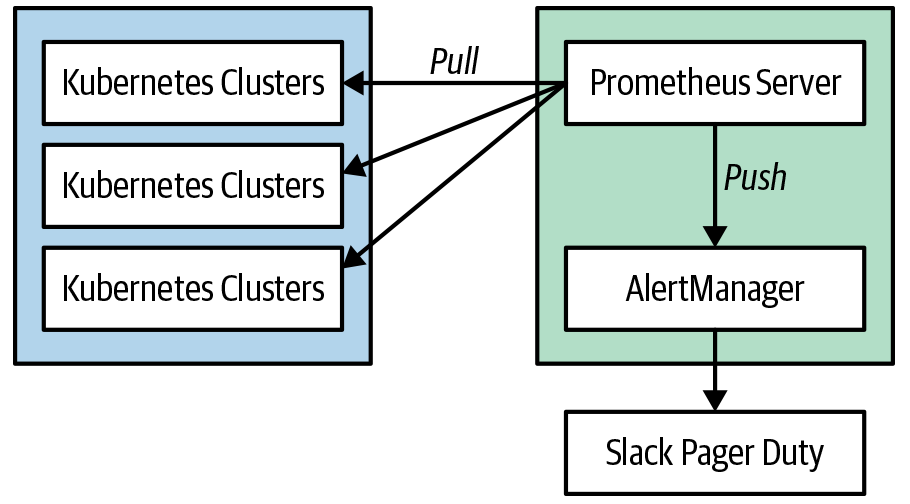
*# TYPE node\_cpu\_seconds\_total counter*

*node\_cpu\_seconds\_total{cpu="0",mode="idle"} 5144.64*

*node\_cpu\_seconds\_total{cpu="0",mode="iowait"} 117.98*

为了收集度量标准，Prometheus使用pull模型，在该模型中，它会刮取度量标准端点以收集度量并将其吸收到Prometheus服务器中。 像Kubernetes这样的系统已经以Prometheus格式公开了它们的指标，从而使收集指标变得很容易。 许多其他Kubernetes生态系统项目（NGINX，Traefik，Istio，LinkerD等）也以Prometheus格式公开了其指标。 Prometheus还可以使用exporters，该exporters允许您从服务中获取发出的指标并将其转换为Prometheus格式的指标。

Prometheus的架构非常简化，如图3-1所示。



您可以在群集内或群集外安装Prometheus。 从“utility cluster”监视群集是一个很好的做法，以避免生产问题也影响监视系统。 有诸如Thanos之类的工具可为Prometheus提供高可用性，并允许您将指标导出到外部存储系统。

对Prometheus体系结构的深入探讨超出了本书的范围，您应该参考另一本有关该主题的专门书籍。 Prometheus：Up＆Running（O’Reilly）是一本很好的入门书籍。

因此，让我们深入研究并在我们的Kubernetes集群上设置Prometheus。 有许多不同的方法可以执行此操作，并且部署将取决于您的特定实现。 在本章中，我们将安装Prometheus Operator：

Prometheus Server

拉并存储从系统收集的指标。

Prometheus Operator

使Prometheus配置Kubernetes本地化，并管理和操作Prometheus和Alertmanager集群。 允许您通过本机Kubernetes资源定义创建，销毁和配置Prometheus资源。

节点导出器

从集群中的Kubernetes节点导出主机指标。

库伯状态量度

收集特定于Kubernetes的指标。

警报经理

允许您配置警报并将警报转发到外部系统。

格拉法纳

提供Prometheus仪表板功能的可视化。

## 指

## 指

*TODO*