

利用Amazon CloudWatch 搭建无人值守的监控预警平台

by AWS Team | on 16 NOV 2016 | [Permalink](#) | [Share](#)

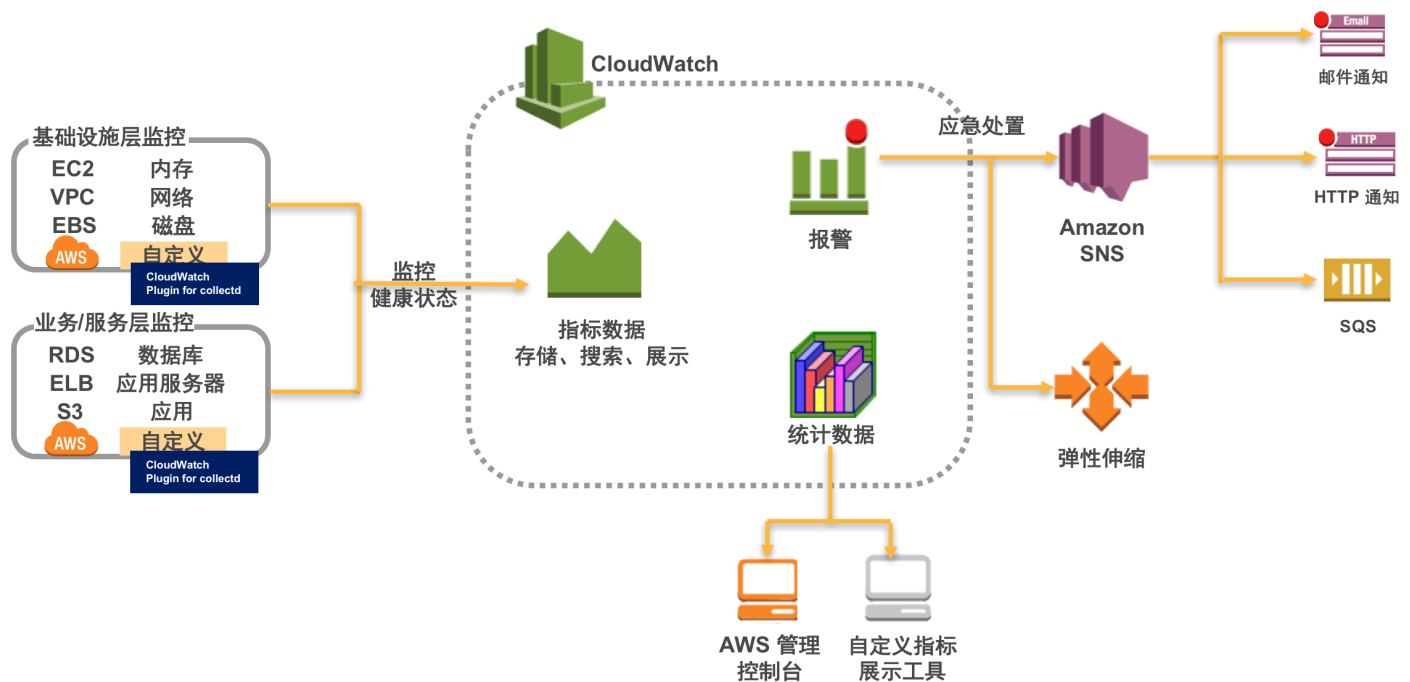
资源与应用服务层监控

Amazon CloudWatch 监控和预警平台可以帮助客户统一管理和运维AWS云端和本地资源、服务和业务系统；使用Amazon CloudWatch 可以收集和跟踪指标，收集和监控日志文件，设置警报。您可通过使用 Amazon CloudWatch 全面地了解资源使用率、应用程序性能和运行状况。使用这些分析结果，您可以及时做出反应，保证应用程序顺畅运行。

Amazon CloudWatch 的基本概念

请参考[AWS 官方文档](#)了解 Amazon CloudWatch的核心概念和术语，比如指标、命名空间、维度、时间戳、单位、统计数据、时间段、聚合、警报等。

基于CloudWatch 的监控预警平台架构



CloudWatch 提供了一套标准的API接口，用户可以利用该平台发布自定义应用、业务或者更加详细的系统指标。用户发布到Amazon CloudWatch 的指标是按时间排序的数据点集合，数据点本身可以来自于任何应用程序或者业务活动；指标通过名称、命名空间和维度进行唯一定义；维度可以帮助你设计数据点的分组特征或者类别，发布指标数据点时必须指定维度，比如虚机的CPU使用率，用户可以查看单独某个虚机的监控指标也可以按AutoScaling组来查看，这里的单个虚机或者AutoScaling组就是同一数据点的不同的维度。用户可以使用秒级甚至千分之一秒的频率发布自定义指标，但是Amazon CloudWatch 还是会将数据聚合到1分钟为最小粒度。

基于指标数据，用户可以翻译业务的波动异常到相应的指标，从而创建警报来和相应的操作来自动化应对各种异常情况，操作包括弹性伸缩（Auto Scaling）机制来应对访问流量变化或者Amazon SNS 主题订阅绑定的邮件通知、HTTP请求的调用和消息队列异步处理。

指标数据用户可以直接通过AWS 控制台进行的图形化按时间筛选、查看和分享；同时，用户也可以通过API接口获取指标数据进行第三方的处理和展示。CloudWatch默认保存两周的指标数据（海外区域部分可以支持免费存储最多15个月的统计数据，[详情请查看AWS CloudWatch文档](#)）。

本文的架构中，自定义指标收集不需要自己编程而是利用collectd守护进程进行监控和获取，同时利用CloudWatch Plugin for collectd直接将自定义指标发布和存储到CloudWatch中，用户随后可以基于自定义指标的进行自动化警报处理从而实现无人值守的统一监控平台。

什么是CloudWatch Plugin for collectd

CloudWatch一直支持用户发布自定义指标来存储、监控自己关心的业务、应用和系统健康状况；AWS最新发布了CloudWatch Plugin for collectd开源项目，该插件整合了collectd强大的收集各种类型统计数据的能力，帮助客户简化了开发收集自定义指标的相关工作，开箱即用地支持发布Apache、Nginx Web服务器应用指标，内存监控指标等监控数据到CloudWatch进行统一存储、展示和预警。

什么是collectd

collectd是一个基于C语言的守护进程，主要任务就是用来收集统计信息，它提供各种存储方式来存储不同值的机制。它支持超过100种各类插件，下面大概列出一些比较常见的插件类型，具体的请参考[collectd官方网站](#)。

- Web应用：Apache、nginx
- 数据库：MySQL、Oracle、PostgreSQL、memcached
- 网络：OpenVPN、Ping、TCPConns、
- 系统：Memory、Disk、FileCount、vmem、uptime、df

安装配置CloudWatch Plugin for collectd

下面的步骤以BJS区域的EC2为例来说明如何安装配置和使用CloudWatch Plugin for collectd：

用户授权

该插件支持IAM Role或者 IAM User两种方式的授权，按照AWS最佳实践，我们推荐使用IAM Role的方式进行授权。所以，开始之前，我们先创建一个IAM Role并赋予相应的权限。

创建一个角色名字为role4collectd

IAM > 角色 > role4collectd

▼ 摘要

角色 ARN	arn:aws-cn:iam: [redacted] :role/role4collectd
实例配置文件 ARN	arn:aws-cn:iam: [redacted] :instance-profile/role4collectd
路径	/
创建时间	2016-11-07 13:50 UTC+0800

然后在该角色的权限页面，创建角色策略，赋予该角色拥有发布指标的权限。

Review Policy

Customize permissions by editing the following policy document. For more information about the access policy language, see [Overview of Policies](#) in the *Using IAM* guide.

Policy Name

policygen-role4collectd-201611071351

Policy Document

```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Sid": "Stmt1478497860000",  
6       "Effect": "Allow",  
7       "Action": [  
8         "cloudwatch:PutMetricData"  
9       ],  
10      "Resource": [  
11        "*"   
12      ]  
13    }  
14  ]  
15 }
```

☒ Use autoformatting for policy editing

Cancel

Validate Policy

Apply Policy

启动实例并绑定角色

如果你想在已经启动的EC2或者你自己的机器上启用该插件，可以忽略该步骤并创建一个具备CloudWatch发布指标权限的IAM User。

本文启动一个新的EC2实例并绑定上一步创建好的角色：

1. 选择 AMI 2. 选择实例类型 3. 配置实例 4. 添加存储 5. 标签实例 6. 配置安全组 7. 审核

步骤 3: 配置实例详细信息

配置实例以便满足您的需求。您可以从同一 AMI 上启动多个实例，请求竞价型实例以利用其低价优势，向实例分配访问管理角色等等。

实例的数量	<input type="text" value="1"/>
网络	vpc-137a6676 (10.0.0.0/16) jvpc  新建 VPC
子网	subnet-65956612(10.0.0.0/24) jpublic cn-north-1b  新建子网 245 个可用 IP 地址
自动分配公有 IP	<input type="text" value="启用"/>
IAM 角色	role4collectd  创建新的 IAM 角色

安装插件

登陆到EC2并更新系统。

```
[ec2-user@ip-10-0-0-6 ~]$ sudo yum -y install collectd  
已加载插件: priorities, update-motd, upgrade-helper  
正在解决依赖关系
```

```
--> 正在检查事务
---> 软件包 collectd.x86_64.0.5.4.1-1.11.amzn1 将被 安装
--> 解决依赖关系完成
```

下载CloudWatch Plugin for collectd 安装文件并执行，安装脚本会自动读取EC2的Meta Data，因此选项都可以默认选择，比如区域，EC2绑定的IAM Role等等。

```
[ec2-user@ip-10-0-0-6 ~]$ sudo -s
[root@ip-10-0-0-6 ec2-user]# wget
https://raw.githubusercontent.com/aws-labs/collectd-cloudwatch/master/src/setup.py
[root@ip-10-0-0-6 ec2-user]# chmod +x setup.py
[root@ip-10-0-0-6 ec2-user]# ./setup.py
[root@ip-10-0-0-6 ec2-user]# ./setup.py
Installing dependencies ... OK
Installing python dependencies ... OK
Downloading plugin ... OK
Extracting plugin ... OK
Moving to collectd plugins directory ... OK
Copying CloudWatch plugin include file ... OK
```

Choose AWS region for published metrics:

1. Automatic [cn-north-1]

2. Custom

Enter choice [1]:

Choose hostname for published metrics:

1. EC2 instance id [i-cc97bc74]

2. Custom

Enter choice [1]:

Choose authentication method:

1. IAM Role [role4collectd]

2. IAM User

Enter choice [1]:

Choose how to install CloudWatch plugin in collectd:

1. Do not modify existing collectd configuration

2. Add plugin to the existing configuration

Enter choice [2]:

```
Plugin configuration written successfully.  
Stopping collectd process ... NOT OK  
Starting collectd process ... OK
```

collectd和CloudWatch plugin for collectd到此就已经安装完成了。

定义发布到CloudWatch的指标

要在CloudWatch中存储和查看collectd收集的指标数据，需要完成两件事情（1）安装相应的collectd插件（2）在CloudWatch plugin配置文件中添加指标白名单。

查看CloudWatch plugin的指标白名单，版本collectd 5.5以下默认CloudWatch白名单指标是空的，也就是默认不会发布任何collectd的指标到CloudWatch：

```
[ec2-user@ip-10-0-0-6 ~]$ sudo cat /opt/collectd-  
plugins/cloudwatch/config/whitelist.conf
```

对于可用的collectd的指标数据类型，我们可以查看以下文件来确认，该文件包含没有发布到CloudWatch的指标类型，该文件是系统自动维护，不要人为进行修改：

```
blocked_metrics  plugin.conf      whitelist.conf  
[ec2-user@ip-10-0-0-6 ~]$ sudo cat /opt/collectd-  
plugins/cloudwatch/config/blocked_metrics  
# This file is automatically generated - do not modify  
this file.  
# Use this file to find metrics to be added to the  
whitelist file instead.  
cpu-0-cpu-user  
cpu-0-cpu-nice  
cpu-0-cpu-system  
cpu-0-cpu-idle  
cpu-0-cpu-wait  
cpu-0-cpu-interrupt  
cpu-0-cpu-softirq  
cpu-0-cpu-steal  
interface-lo-if_octets-  
interface-lo-if_packets-  
interface-lo-if_errors-  
interface-eth0-if_octets-  
interface-eth0-if_packets-  
interface-eth0-if_errors-  
load--load-
```

```
memory--memory-used
memory--memory-buffered
memory--memory-cached
memory--memory-free
```

假定我们对于内存相关指标比较感兴趣，可以将memory开头的指标类型添加到CloudWatch白名单：

```
[ec2-user@ip-10-0-0-6 ~]$ sudo vim /opt/collectd-plugins/cloudwatch/config/whitelist.conf
```

输入如下信息：

```
memory--memory-.*
```

重启collectd服务：

```
[ec2-user@ip-10-0-0-6 ~]$ sudo service collectd restart
Stopping collectd: [ OK ]
Starting collectd: [ OK ]
```

注：在BJS区域的EC2上默认安装和配置后，我们从日志会发现报错，主要原因是默认安装后的脚本不支持BJS区域，详细的错误识别请参考以下步骤。

打开debug模式：

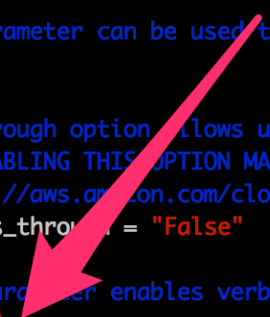
```
[root@ip-10-0-0-6 ~]# vim /opt/collectd-plugins/cloudwatch/config/plugin.conf
```

```
# The target region which will be used to publish metric data
# For list of valid regions visit: http://docs.aws.amazon.com/general/latest/gr/rande.html#cw_region
# region =

# The host parameter can be used to override instance-id or host information published with every metric
# host =

# The pass through option allows unsafe regular expressions such as '.*' or '.*+'.
# WARNING: ENABLING THIS OPTION MAY LEAD TO PUBLISHING A LARGE NUMBER OF METRICS
# SEE https://aws.amazon.com/cloudwatch/pricing/ TO UNDERSTAND HOW TO ESTIMATE YOUR BILL.
whitelist_pass_through = "False"

# The debug parameter enables verbose logging of published metrics
debug = "True"
```



重启collectd服务：

```
[root@ip-10-0-0-6 ~]# sudo service collectd restart
Stopping collectd: [ OK ]
Starting collectd: [ OK ]
```

查看日志：

```
[root@ip-10-0-0-6 ~]# sudo tail -f /var/log/messages
```

从下面的错误信息可以看到，脚本默认发布指标到如下的endpoint：<https://monitoring.cn-north-1.amazonaws.com/>，但BJS区域的CloudWatch endpoint和海外区域的命名规则有差异，详细的AWS BJS区域服

务的终端节点可以参考：[中国（北京）区域](#)；

为了解决BJS区域终端节点的支持问题，我们需要更新插件的Python脚本文件使其支持BJS区域：

1. `confighelper.py`文件：该文件默认安装后的目录是 `/opt/collectd-plugins/cloudwatch/modules/configuration`

Branch: master ▼

[collectd-cloudwatch](#) / [src](#) / [cloudwatch](#) / [modules](#) / [configuration](#) /

更新如下函数，

```
122     def _set_endpoint(self):
123         """ Creates endpoint from region information """
124         if self.region is "localhost":
125             self.endpoint = "http://" + self.region + "/"
126         else:
127             self.endpoint = "https://monitoring." + self.region + ".amazonaws.com/"
```

参考修改后的代码如下：

```
122     def _set_endpoint(self):
123         """ Creates endpoint from region information """
124         if self.region is "localhost":
125             self.endpoint = "http://" + self.region + "/"
126         elif self.region is "cn-north-1":
127             self.endpoint = "https://monitoring." + self.region + ".amazonaws.com.cn/"
128         else :
129             self.endpoint = "https://monitoring."+self.region+".amazonaws.com.cn/"
```

2. `requestbuilder.py` 文件：该文件默认安装后的目录为 `/opt/collectd-plugins/cloudwatch/modules/client`

该类的主要功能是构建签名版本4的 `PutMetricData` API 请求，因此会使用到 `endpoint` 信息，具体请参考[在线文档](#)：

Branch: master ▼

[collectd-cloudwatch](#) / [src](#) / [cloudwatch](#) / [modules](#) / [client](#) / [requestbuilder.py](#)

更新如下函数，

```
84     def _get_host(self):
85         """ Returns the endpoint's hostname derived from the region """
86         if self.region == "localhost":
87             return "localhost"
88         return "monitoring." + self.region + ".amazonaws.com"
```

参考修改后的代码如下：

```

84     def _get_host(self):
85         """ Returns the endpoint's hostname derived from the region """
86         if self.region == "localhost":
87             return "localhost"
88         elif self.region == "cn-north-1":
89             return "monitoring." + self.region + ".amazonaws.com.cn"
90         return "monitoring." + self.region + ".amazonaws.com"

```

重启collectd服务：

```

[root@ip-10-0-0-6 ~]# sudo service collectd restart
Stopping collectd: [ OK ]
Starting collectd: [ OK ]

```

查看以 [AmazonCloudWatchPlugin] 开头的日志内容，从日志可以看出，我们添加到白名单的memory相关的日志包含四个不同的指标，每分钟发布一次：

```

[root@ip-10-0-0-6 ~]# sudo tail -f /var/log/messages

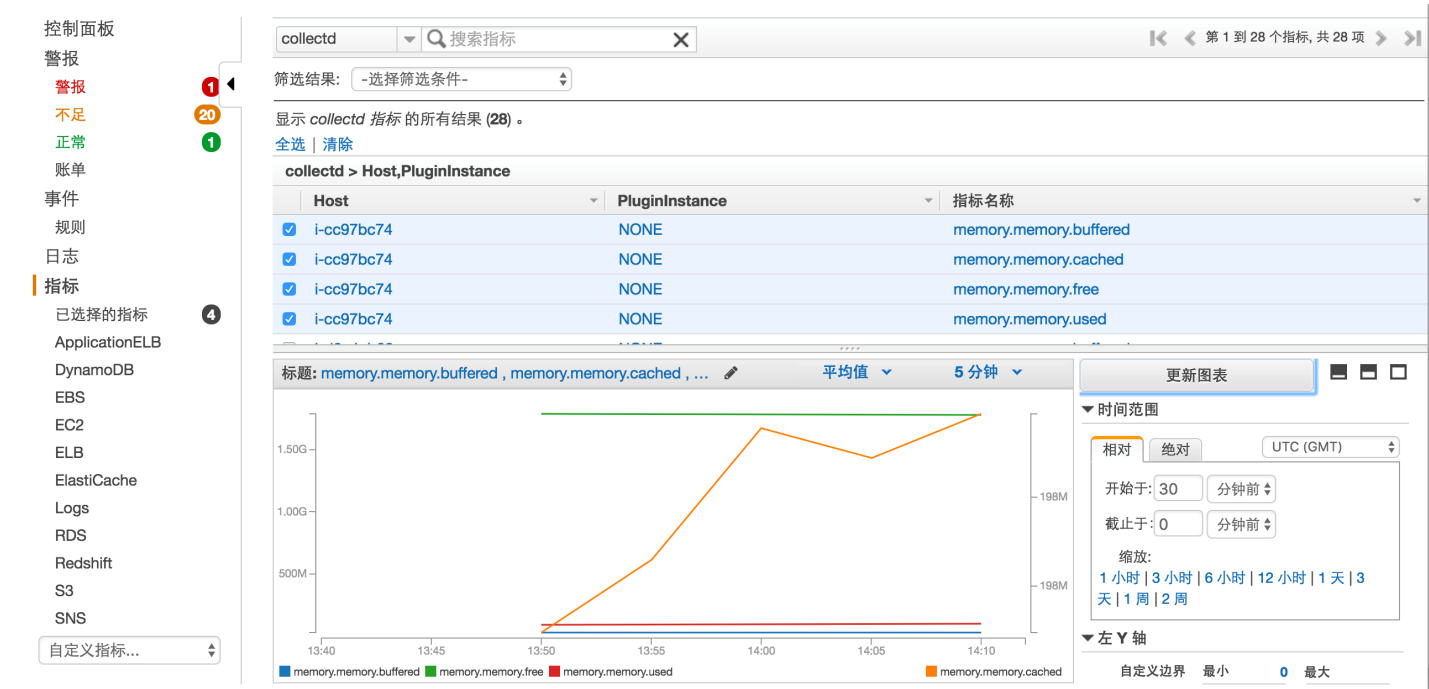
```

```

Nov 7 14:01:00 ip-10-0-0-6 collectd[24595]: [AmazonCloudWatchPlugin][cloudwatch.modules.flusher] [debug] flushing metrics memory--memory-cached[6] memory--memory-buffered[6] memory--memory-free[6] memory--memory-used[6]
Nov 7 14:02:00 ip-10-0-0-6 collectd[24595]: [AmazonCloudWatchPlugin][cloudwatch.modules.flusher] [debug] flushing metrics memory--memory-cached[6] memory--memory-buffered[6] memory--memory-free[6] memory--memory-used[6]
Nov 7 14:02:15 ip-10-0-0-6 dhclient[2144]: XMT: Solicit on eth0, interval 118450ms.
Nov 7 14:03:00 ip-10-0-0-6 collectd[24595]: [AmazonCloudWatchPlugin][cloudwatch.modules.flusher] [debug] flushing metrics memory--memory-cached[6] memory--memory-buffered[6] memory--memory-free[6] memory--memory-used[6]
Nov 7 14:04:00 ip-10-0-0-6 collectd[24595]: [AmazonCloudWatchPlugin][cloudwatch.modules.flusher] [debug] flushing metrics memory--memory-cached[6] memory--memory-buffered[6] memory--memory-free[6] memory--memory-used[6]
Nov 7 14:04:14 ip-10-0-0-6 dhclient[2144]: XMT: Solicit on eth0, interval 123270ms.
Nov 7 14:05:00 ip-10-0-0-6 collectd[24595]: [AmazonCloudWatchPlugin][cloudwatch.modules.flusher] [debug] flushing metrics memory--memory-cached[6] memory--memory-buffered[6] memory--memory-free[6] memory--memory-used[6]

```

登录AWS 控制台，打开到CloudWatch页面，左侧导航最底端有个自定义指标的选择框，下拉就可以选择collectd来查看刚刚发布的指标内容：



安装启用收集Apache监控数据插件

如果未安装配置好Apache Web服务器，可以参考 [教程：在 Amazon Linux 上安装 LAMP Web 服务器](#) 来搭建好该Web服务器环境。

通过以下命令可以查询 Amazon Linux AMI带有的collectd的插件列表：

```
[ec2-user@ip-10-0-0-6 ~]$ sudo yum list | grep collectd
collectd.x86_64                               5.4.1-1.11.amzn1      @amzn-main
collectd-amqp.x86_64                          5.4.1-1.11.amzn1      amzn-main
collectd-apache.x86_64                       5.4.1-1.11.amzn1      amzn-main
collectd-bind.x86_64                         5.4.1-1.11.amzn1      amzn-main
collectd-curl.x86_64                         5.4.1-1.11.amzn1      amzn-main
collectd-curl_xml.x86_64                     5.4.1-1.11.amzn1      amzn-main
collectd-dbi.x86_64                          5.4.1-1.11.amzn1      amzn-main
collectd-dns.x86_64                          5.4.1-1.11.amzn1      amzn-main
collectd-email.x86_64                       5.4.1-1.11.amzn1      amzn-main
collectd-generic-jmx.x86_64                  5.4.1-1.11.amzn1      amzn-main
collectd-gmond.x86_64                       5.4.1-1.11.amzn1      amzn-main
collectd-ipmi.x86_64                        5.4.1-1.11.amzn1      amzn-main
collectd-iptables.x86_64                    5.4.1-1.11.amzn1      amzn-main
collectd-ipvs.x86_64                        5.4.1-1.11.amzn1      amzn-main
collectd-java.x86_64                        5.4.1-1.11.amzn1      amzn-main
collectd-lvm.x86_64                         5.4.1-1.11.amzn1      amzn-main
collectd-memcached.x86_64                   5.4.1-1.11.amzn1      amzn-main
collectd-mysql.x86_64                       5.4.1-1.11.amzn1      amzn-main
collectd-netlink.x86_64                     5.4.1-1.11.amzn1      amzn-main
collectd-nginx.x86_64                       5.4.1-1.11.amzn1      amzn-main
collectd-notify_email.x86_64                 5.4.1-1.11.amzn1      amzn-main
collectd-postgresql.x86_64                  5.4.1-1.11.amzn1      amzn-main
collectd-rrdcached.x86_64                   5.4.1-1.11.amzn1      amzn-main
collectd-rrdtool.x86_64                     5.4.1-1.11.amzn1      amzn-main
collectd-snmp.x86_64                        5.4.1-1.11.amzn1      amzn-main
collectd-varnish.x86_64                     5.4.1-1.11.amzn1      amzn-main
collectd-web.x86_64                         5.4.1-1.11.amzn1      amzn-main
```

apache的状态信息来自于自身的mod_status模块，collectd解析出例如传输的bytes大小，接受到的请求数量等指标；详情请参考该[插件介绍页面](#)；安装collectd apache 监控插件：

```
[ec2-user@ip-10-0-0-6 ~]$ sudo yum install -y collectd-apache
已加载插件： priorities, update-motd, upgrade-helper
amzn-main/latest                               | 2.1 kB    00:00
amzn-updates/latest                            | 2.3 kB    00:00
正在解决依赖关系
--> 正在检查事务
---> 软件包 collectd-apache.x86_64.0.5.4.1-1.11.amzn1 将被 安装
--> 解决依赖关系完成
```

修改collectd的配置，默认安装的文件位置 /etc/collectd.conf：

```
LoadPlugin apache
<Plugin apache>
  <Instance "local">
    URL "http://localhost/server-status?auto"
  </Instance>
</Plugin>
```

以下是针对httpd-2.2版本的参考配置，默认的配置文件位于/etc/httpd/conf/httpd.conf：

```
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from localhost
</Location>
```

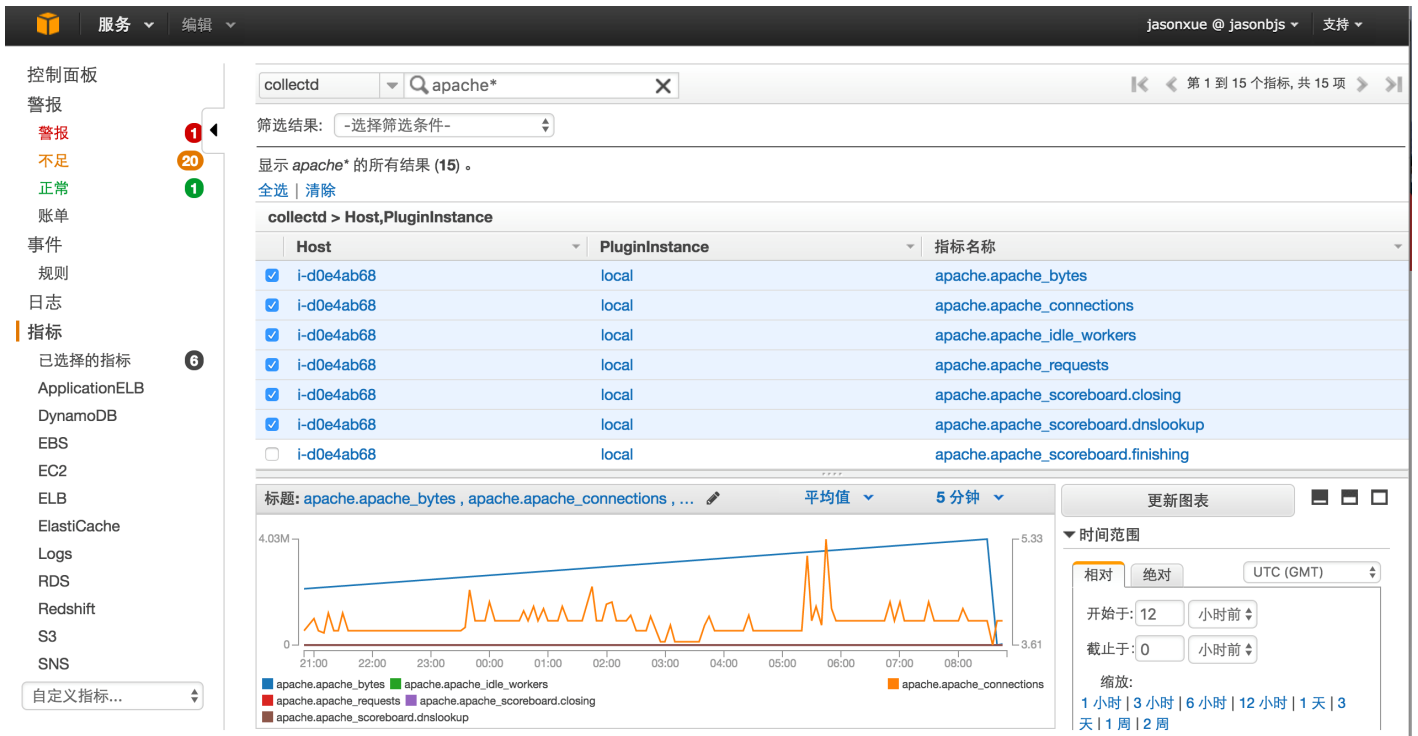
将apache相关状态指标加入到CloudWatch collectd插件配置在白名单列表，更新文件 /opt/collectd-plugins/cloudwatch/config/whitelist.conf：

```
memory--memory-.*
apache-local-apache_.*
```

重新启动相关服务：

```
[ec2-user@ip-172-31-12-60 ~]$ sudo service httpd restart
Stopping httpd: [ OK ]
Starting httpd: [ OK ]
[ec2-user@ip-172-31-12-60 ~]$ sudo service collectd restart
Stopping collectd: [ OK ]
Starting collectd: [ OK ]
```

这样我们就完成了安装配置新的collectd的apache插件，同时将apache的相关监控指标添加到CloudWatch的白名单，这样我们就可以无缝整合collectd的收集数据能力和CloudWatch统一存储、展示和预警能力，下图就是apache web应用相关的指标在CloudWatch中展示的结果：



基于CloudWatch指标创建警报

既然我们通过插件收集了很多系统和应用的指标，那如果发生一些异常或者超过预先定义的阈值的时候，我们如何去应对和处理呢？理想情况我们能够尽可能自动化来应对这些警报，即搭建“无人”值守的监控预警平台。下图是基于某一个采集指标定义警报及处理警报的操作的截图；当一个警报发生时，你可以定义一个或多个操作来应对和处理；你可以采取的操作类型有，发送邮件通知，发布消息到SNS服务，添加Auto Scaling操作，以及EC2实例操作；这里的EC2实例操作包含停止、终止、重启或恢复，任何EC2实例指标（在 AWS/EC2 命名空间中）或者包含“InstanceId=”维度的任何自定义指标都可以在警报中触发EC2实例操作。



一些限制

很多用户开始接触CloudWatch的时候就非常关心性能问题，目前发布指标API [PutMetricData](#) 每秒可处理 150 个事务 (TPS)，这是您每秒可以发出而不会受到限制的操作请求的最大数量，但可以在线请求提高限制；[PutMetricData](#) 同时支持GET和POST操作，请求最大大小分别为8KB和40KB。每个指标最多可以有10个维度；

总结

本文和大家一起学习了如何基于CloudWatch及collectd相关插件构建无人值守的监控预警平台；CloudWatch默认提供了AWS资源的基本监控数据，同时提供了CLI和REST API的方式供用户自行扩展自定义业务和系统指标数据；[CloudWatch plugin for collectd](#) 是AWS最新发布的一个开源插件，大家可以进一步通过学习该项目的源代码掌握如何基于Python扩展和集成AWS的服务。