文章
Michael Lei · 八月 26, 2021 阅读大约需 9 分钟

SAM - 设置和添加非 IRIS 目标指标的技巧和提示

SAM - 设置和添加非 IRIS 目标指标的技巧和提示

SAM（系统警报和监视）以“ 功能齐全 ”的 docker-compose
容器集的形式提供，只要启动就可以开始以默认的仪表板监视 IRIS 实例。 使用初始配置就可以很好地了解 SAM
功能并开始对 IRIS 系统进行基本监视。
但是，当开始监视多个系统并收集大量指标数据时，需要更改一些默认设置。 为了从 SAM
获取更多价值，您还会想要添加来自其他数据源（目标）的指标。 以下技巧将帮助您在生产环境中部署
SAM，从多个目标收集指标并将这些指标组合到您自己的仪表板和图表中。 此外，您还将看到一些可能有助于探索
SAM 容器和应用程序的命令。

> *注意：*我应该指出，其中一些技巧和提示可能不是最佳做法；这更像是一个日志，记录了我第一次如何配置
> SAM 来监视相同系统上的多个服务器和非 IRIS 目标的基准。 如果您有建议，请在评论中指教 ;)
> 所以，记住本帖可能会随着时间的推移而有所变化，让我们开始吧；

在下面的技巧中，有重启 docker 以及启动和停止 SAM 的操作。
请通读这些技巧，确定哪些适合您，然后按照下面的相同顺序执行。

1. 确保有足够的空间用于 SAM 数据库

默认情况下，docker 容器将文件存储在根 (/) 文件系统。 SAM 不需要很多 CPU
或内存资源；但是，指标收集将占用空间。

> 指标需要的存储容量“ 视情况而定 ”。
> 虽然在开始监视之前可能无法明确数据库大小，但大致上，在抓取周期为 15 秒的情况下，监视 10 个 IRIS
> 虚拟机以及操作系统指标大约消耗 50GB 存储。

策略包括：增加监视实例的根存储，或更改数据库的卷位置。 我使用了以下命令来更改运行 docker 的虚拟机上的
docker 目录。 也许是杀鸡用了牛刀，但很管用。

- 停止 docker 并将 docker 文件复制到空间充足的文件系统（本例中为 /data/docker/data）。 见下面的示例：

```
[root@mysamserver lib]# sudo systemctl stop docker
[root@mysamserver lib]# pwd
/var/lib
[root@mysamserver lib]# cp -rp docker /data/docker/data
[root@mysamserver lib]#
[root@mysamserver lib]# rm -rf docker
```

- 在 docker 配置文件中更新卷路径。 注意此文件还有一个网络设置“ bip ”（请参见注释：...）

```
cat /etc/docker/daemon.json
```

```
{
        "data-root": "/data/docker/data",
        "bip": "192.168.0.1/24"
}
```

- 重启 docker

```
sudo systemctl daemon-reload
sudo systemctl restart docker

systemctl status docker.service
```

2. 设置 SAM

我假定您已在测试系统上设置 SAM，并熟悉它的基本操作：添加集群和实例，以及查看系统指标。 建议您花 20 分钟时间查看我在虚拟全球峰会 2020 上的展示，以了解安装步骤的概述，以及添加多个目标的运行指标时 SAM 的外观。 要观看此会议，请使用以下链接（您需要使用您的电子邮件注册）：

DEV007 系统警报和监视

登录到 SAM 门户并配置一些 IRIS 实例。 这会填充配置文件并提供向导。

http://mysamserver:8080/api/sam/app/index.csp#/

> 注意：如果要添加多个实例，或者希望用脚本执行此步骤，可以通过 API 添加实例。 请参见文档。

3. 升级到生产许可证

SAM 随附了一个 IRIS 社区版许可证。 有几个限制，包括 IRIS.DAT 限制为 10GB。 10GB 不足以长时间从多个目标收集数据。 请联系您的 InterSystems 联系人以获取生产许可证。 在没有编辑器的情况下，在精简的容器中更新许可证可能很棘手，我只是在 IRIS 容器上登录一个交互式会话，然后使用以下命令更新许可证密钥：

- 打开 shell，切换目录至 mgr 文件夹（iris.key 文件的默认位置）

```
docker exec -it sam_iris_1 bash
cd /dur/iconfig/mgr
```

- 使用 unix “ here 文档 ” 更新密钥。 在 “ > ” 后面粘贴密钥文本。 在密钥文本后面，输入 “ >EOF ” 以提交命令。 然后输入 “ exit ” 退出 shell。

```
cat <<EOF >iris.key
>
[ConfigFile]
FileType=InterSystems License Rev-A.1
LicenseID=999999

[License]
LicenseCapacity=InterSystems IRIS 2020.2 Server for SAM:etc etc, the key you were sen
t by your InterSystems contact.
>EOF
```

```
exit
```

- 然后使用提供的 docker-compose shell 脚本停止再启动 SAM：

```
./stop.sh
./start.sh
```

- 可以通过访问系统管理门户或登录到 iris 实例并检查 messages.log 来检查许可证是否一切正常。

```
docker exec -it sam_iris_1 bash
cd /dur/iconfig/mgr
cat messages.log
```

4. 在目标上安装其他 prometheus 导出程序

例如，prometheus 节点导出程序可显示各种硬件和内核相关的指标。

节点导出程序文档

通过请求（抓取）实例端点的指标，测试节点导出程序是否正常工作：

```
curl my_target_server_name:9100/metrics
```

您应该看到类似下面的信息：

```
mylaptop:~ mo$ my_target_server_name:9100/metrics | more
HELP go_gc_duration_seconds A summary of the GC invocation durations.
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 4.8862e-05
go_gc_duration_seconds{quantile="0.25"} 7.5898e-05
go_gc_duration_seconds{quantile="0.5"} 9.2974e-05
go_gc_duration_seconds{quantile="0.75"} 0.000130664
go_gc_duration_seconds{quantile="1"} 0.000358762
go_gc_duration_seconds_sum 303.291715258
go_gc_duration_seconds_count 2.572586e+06
# HELP go_goroutines Number of goroutines that currently exist.
# TYPE go_goroutines gauge
go_goroutines 9
:
: many many metrics will be displayed
```

注意您可以对您的 IRIS 实例执行同样操作：

```
mylaptop:~ mo$ curl my_target_server_name:52776/api/monitor/metrics | more
iris_cpu_pct{id="AUXWD"} 0
iris_cpu_pct{id="CSPDMN"} 0
iris_cpu_pct{id="CSPSRV"} 0
iris_cpu_pct{id="ECPCliR"} 0
iris_cpu_pct{id="ECPCliW"} 0
iris_cpu_pct{id="ECPSrvR"} 0
iris_cpu_pct{id="ECPSrvW"} 0
```

```
:
: many many metrics will be displayed
```

5. 编辑配置文件以添加对新目标的抓取

例如，前一个技巧中的 node-exporter 实例。 配置文件将位于 SAM 的安装位置。 如下所示，可以看到 grafana 和 prometheus yml 配置文件。

```
[root@mysamserver sam-1.0.0.115-unix]# ls
config  docker-compose.yml  readme.txt  start.sh  stop.sh
[root@mysamserver sam-1.0.0.115-unix]# tree -x config
config
??? alertmanager
?   ??? isc_alertmanager.yml
??? grafana
?   ??? dashboard.json
?   ??? dashboard-provider.yml
?   ??? datasource.yml
?   ??? grafana.ini
??? nginx
?   ??? nginx.conf
??? prometheus
    ??? isc_alert_rules.yml
    ??? isc_prometheus.yml

4 directories, 8 files
```

5.1 添加目标到 prometheus

以下示例是使用 SAM 中的配置 GUI 屏幕创建的 iscprometheus.yml 文件。 该文件包含两个集群。 一个集群监视 sam 实例本身，另一个集群监视五个 IRIS 实例。

```
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      - alertmanager:9093
global:
  evaluation_interval: 15s
  scrape_interval: 15s
remote_read:
- url: http://iris:52773/api/sam/private/db/read
remote_write:
- url: http://iris:52773/api/sam/private/db/write
rule_files:
- ./isc_alert_rules.yml
scrape_configs:
- job_name: SAM
  metrics_path: /api/monitor/metrics
  scheme: http
  static_configs:
  - labels:
      cluster: "1"
    targets:
```

```
    - mysaminstance.mycompany.com:8080
  - labels:
      cluster: "2"
    targets:
    - myiristarget1:52776
    - myiristarget2:52776
    - myiristarget3:52776
    - myiristarget4:52776
    - myiristarget5:52776
```

- 要添加对其他运行 node-exporter 的目标的抓取，请将以下内容添加到 iscprometheus.yml 文件的底部。注意 API metricspath 与 IRIS 不同。

```
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      - alertmanager:9093
global:
  evaluation_interval: 15s
  scrape_interval: 15s
remote_read:
- url: http://iris:52773/api/sam/private/db/read
remote_write:
- url: http://iris:52773/api/sam/private/db/write
rule_files:
- ./isc_alert_rules.yml
scrape_configs:
- job_name: SAM
  metrics_path: /api/monitor/metrics
  scheme: http
  static_configs:
  - labels:
      cluster: "1"
    targets:
    - iscsydsam.iscinternal.com:8080
  - labels:
      cluster: "2"
    targets:
    - myiristarget1:52776
    - myiristarget2:52776
    - myiristarget3:52776
    - myiristarget4:52776
    - myiristarget5:52776
- job_name: node_shard1
  metrics_path: /metrics
  scheme: http
  static_configs:
  - labels:
      cluster: "2"
      group: node
    targets:
    - myiristarget1:9100
- job_name: node_shard2
  metrics_path: /metrics
  scheme: http
  static_configs:
  - labels:
```

```
      cluster: "2"
      group: node
    targets:
    - myiristarget2:9100
- job_name: node_shard3
  metrics_path: /metrics
  scheme: http
  static_configs:
  - labels:
      cluster: "2"
      group: node
    targets:
    - myiristarget3:9100
- job_name: node_shard4
  metrics_path: /metrics
  scheme: http
  static_configs:
  - labels:
      cluster: "2"
      group: node
    targets:
    - myiristarget4:9100
- job_name: node_shard5
  metrics_path: /metrics
  scheme: http
  static_configs:
  - labels:
      cluster: "2"
      group: node
    targets:
    - myiristarget5:9100
```

- 然后使用提供的 docker-compose shell 脚本停止再启动 SAM：

```
./stop.sh
./start.sh
```

SAM 现在从您通过 GUI 添加的 IRIS 实例以及相同实例上的 node-exporter 中收集指标。

6. 增加 Prometheus 收集指标的天数

在 SAM 的第一版中，您可以在 GUI 中更改收集指标的天数。 但是，我在显示所有指标时遇到一个问题。
在我弄清楚发生了什么之前，我更改了 Prometheus 中的保留天数，否则 SAM 将收集数据，但是您在 Grafana 的 Prometheus 查询中看不到指标。

在文件 docker-compose.yml 中用来安装 SAM 的层，更改 Prometheus 启动时设置的保留天数，例如在 prometheus 部分中，更新 storage.tsdb.retention.time 参数以匹配所需的保留天数：

```
prometheus:
    command:
      - --web.enable-lifecycle
      - --config.file=/config/isc_prometheus.yml
      - --storage.tsdb.retention.time=30d
```

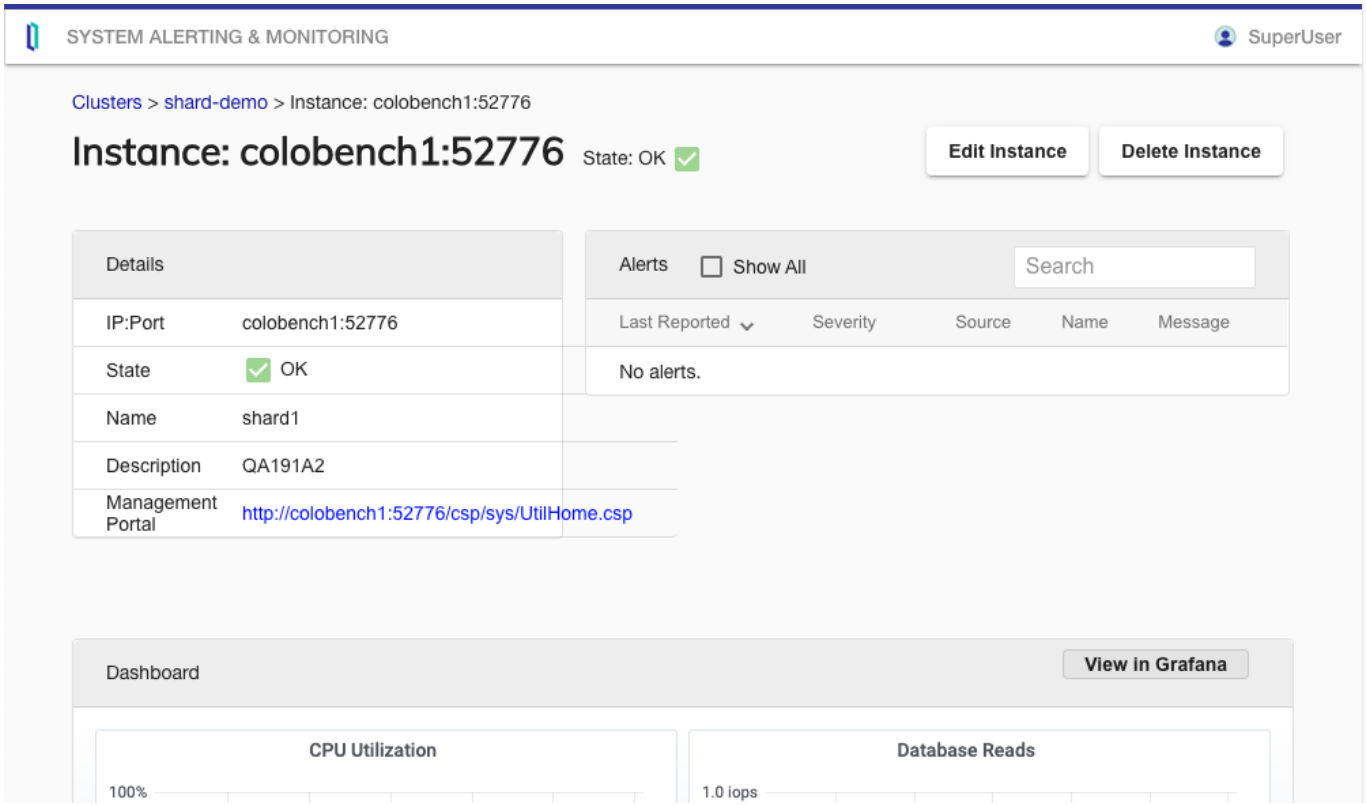
> 注意：在 SAM 的第一版中，最长保留天数为 30 天。 然后使用提供的 docker-compose shell 脚本停止再启动 SAM：

```
./stop.sh
./start.sh
```

7. 创建您自己的仪表板

您可以向现有仪表板添加面板，也可以创建新的仪表板以满足您的监视需求。
这是一个很大的主题，所以留到下一个帖子再说。 不过，这里先帮助您上手。

在 SAM 屏幕上使用 View in Grafana（在 Grafana 中查看）按钮切换到 Grafana。



打开 Grafana 后，可以创建或编辑仪表板；

网上有许多查询导出程序（如 node-exporter）的示例。

> 当可以显示 IRIS 系统指标（SAM 中的默认设置）、IRIS 应用程序指标（您需要将这些指标构建到应用程序中）以及其他指标（如 node-exporter 或由供应商创建的任何数量的指标）时，真正强大的功能才体现出来。 例如，使用 SAM 和 cAdvisor 监视 Docker 容器

#开发运维 #性能 #文档

源 URL: https://cn.community.intersystems.com/post/sam-%E8%AE%BE%E7%BD%AE%E5%92%8C%E6%B7%BB%E5%8A%A0%E9%9D%9E-iris-%E7%9B%AE%E6%A0%87%E6%8C%87%E6%A0%87%E7%9A%84%E6%8A%80%E

5%B7%A7%E5%92%8C%E6%8F%90%E7%A4%BA