Question
Murray Oldfield · Mar 19, 2021

SAM - Hacks and Tips for set up and adding metrics from non-IRIS targets

SAM - Hacks and Tips for set up and adding metrics from non-IRIS targets

SAM (System Altering and Monitoring) comes with as a 'batteries included' docker-compose container set that is ready to start monitoring IRIS instances with a default dashboard as soon as it starts up. The initial configuration is good to understand SAM functionality and start basic monitoring of your IRIS systems. However, out of the box, there are some setting s that you will need to change when you start to monitor many systems and collect a lot of metric data. To get more value out of SAM, you will also want to add metrics from additional data sources (targets). The following tips will assist you on the path to deploying SAM in a production environment and collecting metrics from multiple targets and combining them in your own dashboards and charts. Also, you will see some of the commands might be useful as you explore the SAM containers and applications.

> *Caveat emptor:* I should point out that some of these hacks and tips are probably not best practice; it is more a log of how I configured SAM the first time to monitor a benchmark with multiple servers and non-IRIS targets on the same systems. If you have suggestions, please educate me in the comments ;) So, remembering that this post may change over time, let's start;

In this tips below there are restarts of docker and starts and stops of SAM. Read through the tips, decide which ones apply to you, then do them in the same order as below.

1. Ensure you have enough space for SAM databases

By default docker containers store files in root (/) filesystem. SAM does not need much CPU or memory resources; however, metrics collection will take up space.

> How much storage required for metrics 'depends'. Although it may not be clear how big your databases will be before you start monitoring, on the back of an envelope; monitoring 10 VMs for IRIS and operating system metrics on a 15-second scrape cycle consumed approximately 50GB of storage.

Strategies include; increasing the root storage of your monitoring instance, or changing the volume location for the databases. I used the following commands to change the docker directory on my VM running docker. Perhaps cracking a nut with a sledgehammer, but it works.

- Stop docker and copy the docker files to a filesystem with plenty of space (in this case /data/docker/data). See the following example:

```
[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
```

- Update the volume path in the docker configuration file. Note this file also has a network setting 'bip' (see note:...)

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

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

- Restart docker

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

systemctl status docker.service
```

2. Set up SAM

I assume you have set up SAM on a test system, and a familiar with its basic operations; adding clusters and instances, and viewing system metrics. I suggest you take 20 minutes to view my Virtual Global Summit 2020 presentation for an overview of the install steps, and also how SAM looks when operating metrics from multiple targets have been added. To view the session use the following links (you will need to register with your email :

DEV007 System Alerting & Monitoring

Log on to the SAM portal and configure some IRIS instances. This populates the configuration files and give you a guide.

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

> Note: If you have many instances to add, or you wish to script this step, it is possible to add instances via an API. See the documentation.

3. Upgrade to a production licence

Out of the box SAM ships with an IRIS Community Edition license. There are several limitations, including an IRIS.DAT is limited to 10GB. 10GB is not big enough to collect data from many targets over a long period of time. Follow up with your InterSystems contacts for a production licence. Updating the licence in a stripped down container without an editor can be tricky, I simply logged into an interactive session on the IRIS container and updated the licence key using the following commands;

- open a shell and change directory to the mgr folder (the default location of the iris.key file)

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

- Update the key with a unix 'here document'. After the '>' paste the key text. After the the key text, commit

the command by typing '>EOF'. Then 'exit' the 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
```

- Then stop and start SAM using the supplied docker-compose shell scripts;

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

- You can check everything is OK with the licence by accessing the System management portal, or by logging into the iris instance and checking the messages.log.

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

4. Install additional prometheus exporters on targets

For example; The prometheus Node Exporter exposes a wide variety of hardware- and kernel-related metrics.

Node exporter documentation

Test that node exporter is working by requesting (scraping) the instance endpoint for metrics;

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

You should see something like:

```
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
```

Note you can do the same with your IRIS instances:

```
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. Edit configuration files to add scrape of new targets

For example, the node-exporter instance from the previous tip. Configuration files will be where you installed SAM. As shown below you can see grafana and prometheus yml configuration files.

```
[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 Add targets to prometheus

The following example an iscprometheus.yml file created using the configuration GUI screens in SAM. The file shows two clusters. One cluster is monitoring the sam instance itself, the other cluster is monitoring five IRIS instances.

```
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
```

- To add scarping of additional targets running node-exporter the following is added to the bottom of the iscprometheus.yml file. Note the API metricspath is different to IRIS.

```
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
```

- Then stop and start SAM using the supplied docker-compose shell scripts;

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

SAM is now collecting metrics from the IRIS instances you added through the GUI and from node-exporter on the same instances.

6. Increase the number of days prometheus collects metrics

In the first-release of SAM you can change the number of days to collect metrics in the GUI. However, I had an issue displaying all the metrics. Until I figure whats happening I changed the retention days in Prometheus, otherwise SAM will collect data; but you will not see metrics in Prometheus queries in Grafana.

In the file the docker-compose.yml file at the level where you installed SAM change the retention days which is set when prometheus starts, for example in the prometheus stanza update the storage.tsdb.retention.time parameter to match the retention days you want:

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

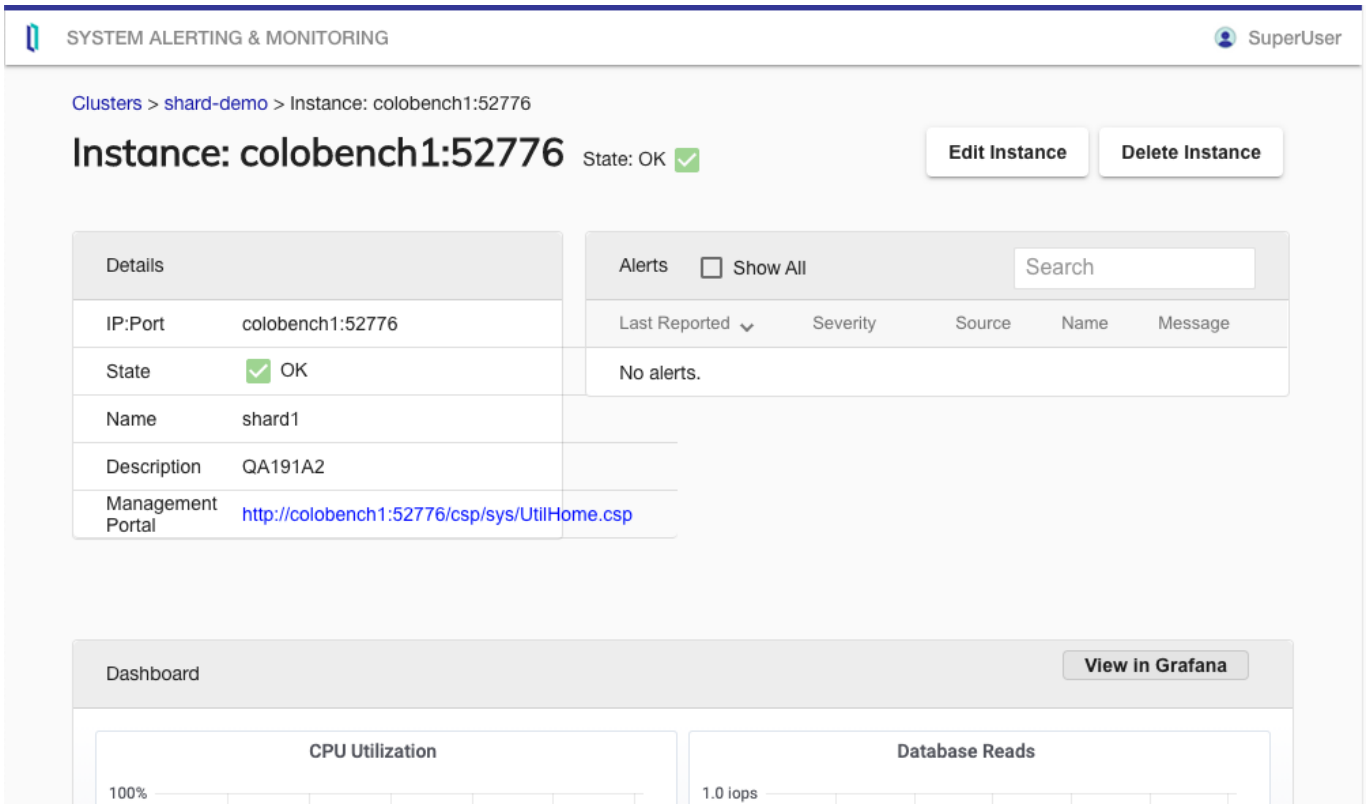
Note: In version 1 of SAM the maximum region days is 30.
Then stop and start SAM using the supplied docker-compose shell scripts;

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

7. Create your own dashboards

You can add panels to existing dashboards, or create new dashboards to suit your monitoring needs. This is a big subject, so I will leave that for the next post. However, to help you on your way.

To switch in to Grafana use the View in Grafana button on the SAM screen.



Once in Grafana you can create or edit dashboards;

There are many examples on the web for querying exporters such as node-exporter.

> The real power comes when you can display your IRIS system metrics (the defaults in SAM), your IRIS application metrics (you need to build these into your applications), and other metrics such as node-exporter or any number of others created by vendors. e.g. Monitor Docker containers using SAM and cAdvisor

#DevOps #Monitoring #System Alerting and Monitoring (SAM) #InterSystems IRIS
Product version: IRIS 2020.4

Source URL:https://community.intersystems.com/post/sam-hacks-and-tips-set-and-adding-metrics-non-iris-targets