SolidFire Stats Collection with Grafana and the Netapp Docker Volume Plugin

All Your Graphs Are Belong to Us

High level steps

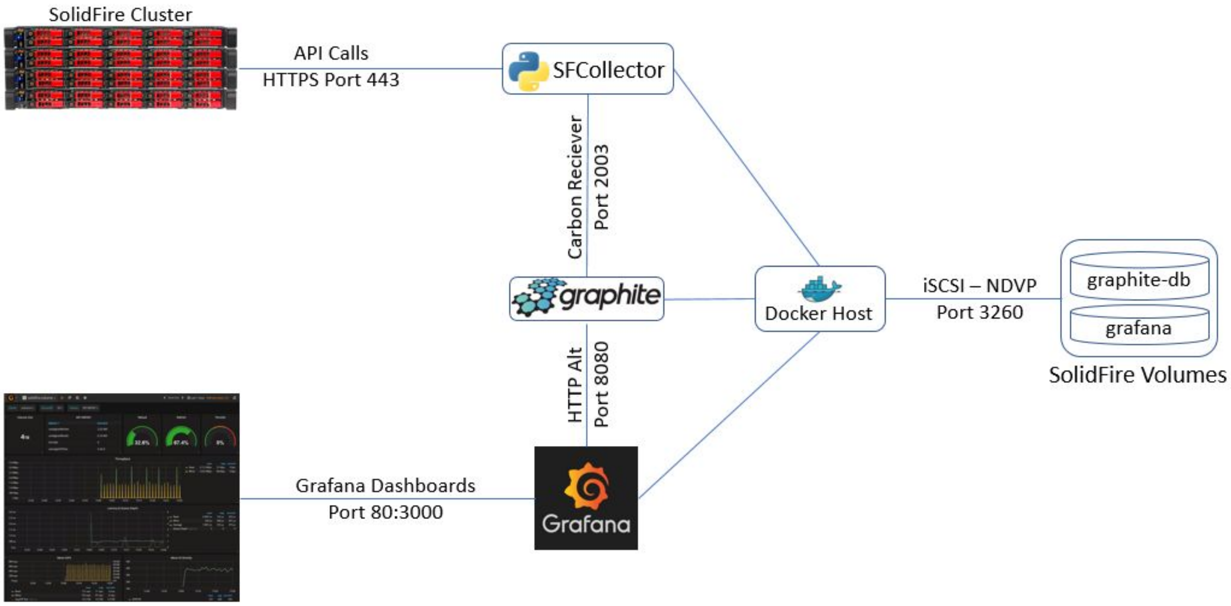
1. (optional) Install and configure NDVP for persistent storage
2. Clone the jedimt/sfcollector repository
3. Setup and run the collector

Overview

The SolidFire collector is a fully packaged metrics collection and graphing solution for Element OS 8+ and is based on three Docker containers.

- SFCollector-> runs a python script to scrape results from SolidFire clusters
- Graphite database -> keeps all time series data from the SFCollector
- Grafana -> Graphing engine

The collector stores metrics in graphite and presents those metrics through a set of pre-configured Grafana dashboards. Optionally, the Netapp Docker Volume Plugin (NDVP) can be used for persistent storage of metrics on a NetApp system.



NetApp Docker Volume Plugin

Prerequisites

Operating System	● Debian ● Ubuntu 14.04+ if not using iSCSI multipathing ● Ubuntu 15.10+ with iSCSI multipathing. ● CentOS, 7.0+ ● RHEL, 7.0+
Element OS Version	7.0+
Docker Version	17.03+

Note: Ubuntu 16.04.1 LTS is used in a multipath configuration for this example.

1. Install packages

```
 sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools scsitools
```

2. Enable multipathing

```
sudo tee /etc/multipath.conf <<-'EOF'
defaults {
    user_friendly_names yes
    find_multipaths yes
}
EOF

sudo service multipath-tools restart
```

3. Start services

```
sudo service open-iscsi start
sudo service multipath-tools start
```

4. Create a location to store the NDVP configuration files

```
sudo mkdir -p /etc/netappdvp
```

5. Create the configuration file for SolidFire

```
cat << EOF > /etc/netappdvp/config.json
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://admin:solidfire@172.27.40.200/json-rpc/9.0",
    "SVIP": "172.27.41.200:3260",
    "TenantName": "docker",
    "InitiatorIFace": "default",
    "Types": [
        {
            "Type": "docker-default",
            "Qos": {
                "minIOPS": 1000,
                "maxIOPS": 2000,
                "burstIOPS": 4000
            }
        },
        {
            "Type": "docker-app",
            "Qos": {
                "minIOPS": 4000,
                "maxIOPS": 6000,
                "burstIOPS": 8000
            }
        },
        {
            "Type": "docker-db",
            "Qos": {
                "minIOPS": 6000,
                "maxIOPS": 8000,
                "burstIOPS": 10000
            }
        }
    ]
}
EOF
```

6. Install the NDVP plugin

```
docker plugin install netapp/ndvp-plugin:17.04 --alias netapp
--grant-all-permissions
```

7. Create docker volumes to be used for Graphite and Grafana persistent storage. This example uses 100GB containers which should be more than sufficient for collecting long term stats for multiple clusters

```
# Create Graphite docker volume
docker volume create -d netapp --name graphite-db -o type=docker-db
-o size=100G

#Create Grafana docker volume
docker volume create -d netapp --name grafana -o type=docker-app -o
size=100G

#show volume
```

```
docker volume list
DRIVER              VOLUME NAME
netapp:latest       grafana
netapp:latest       graphite-db
```

Graphite and Collector Setup

1. Install docker-compose if not installed

```
apt install docker-compose
```

2. Clone the jedimt/sfcollector repo from git

```
mkdir -p /opt/github/sfcollector
git clone https://github.com/jedimt/sfcollector
/opt/github/sfcollector
```

3. Run the bootstrap.sh file to create the password for basic_auth. This will be used to secure the nginx web component

```
/opt/github/sfcollector/bootstrap.sh
```

4. Edit the docker-compose.yml file and specify the persistent data volumes to use for graphite and grafana as well as the password to secure the Grafana web interface

```
graphite:
    build: ./graphiteconfig
    restart: always
    ports:
        - "8080:80"
        - "8125:8125/udp"
        - "8126:8126"
        - "2003:2003"
        - "2004:2004"
    volumes: #Point to NDVP or local volumes for persistent storage
        - "graphite-db:/opt/graphite/storage/whisper"

grafana:
    image: grafana/grafana
    restart: always
    ports:
        - "80:3000"
    volumes: #Point to NDVP or local volumes for persistent storage
        - "grafana:/var/lib/grafana"
    links:
        - graphite
    environment: #Set password for Grafana web interface
        - GF_SECURITY_ADMIN_PASSWORD=P@ssw0rd

sfcollector:
    build: ./collector
    restart: always
    links:
        - graphite
```

5. Modify the `/opt/github/sfcollector/collector/wrapper.sh` script with the appropriate SolidFire cluster MVIP, username and password. If the graphite container name was changed, also specify the new hostname

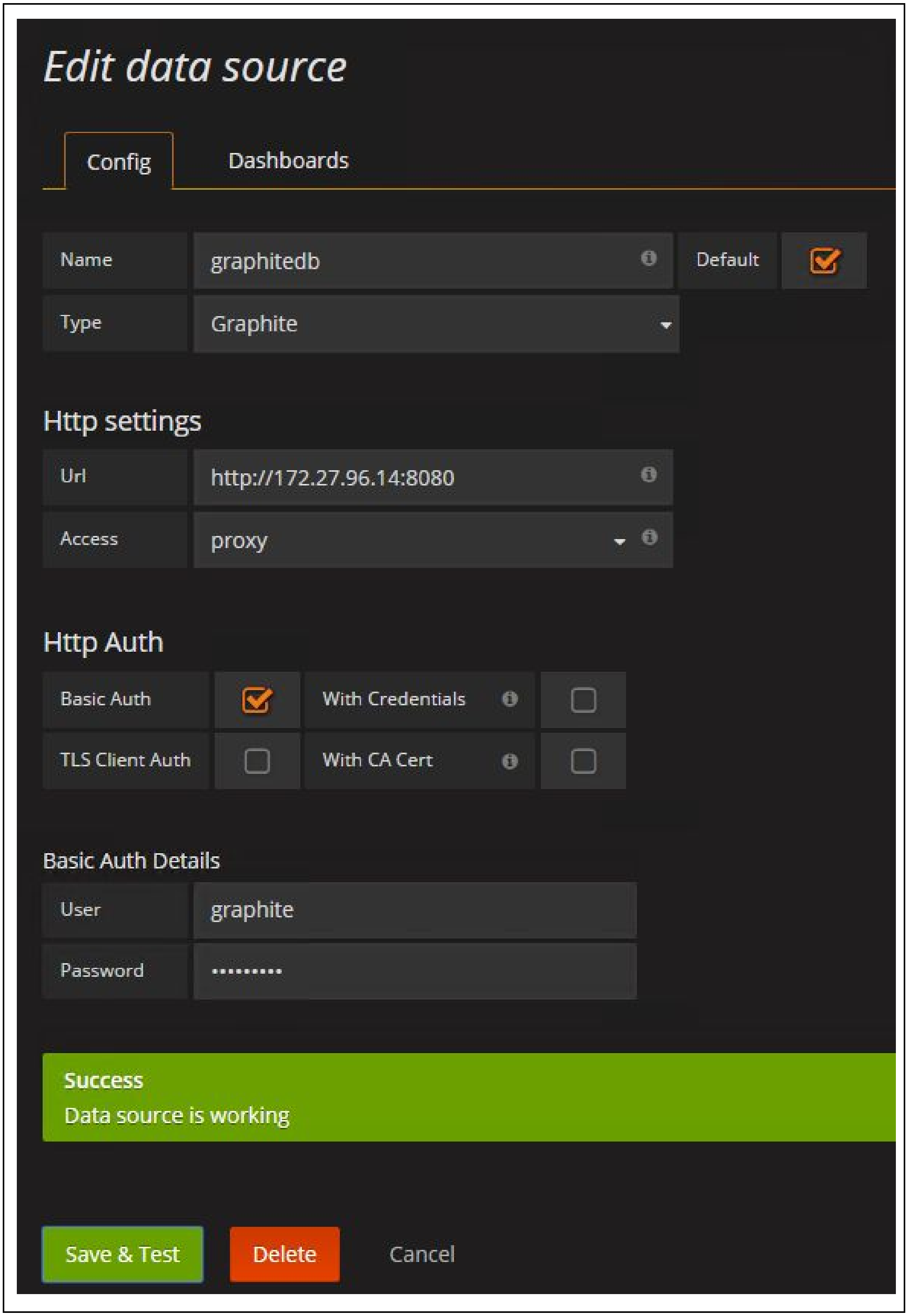
```
#!/usr/bin/env bash
while true
do
/usr/bin/python /solidfire_graphite_collector_v2.py -s 172.27.40.200
-u admin -p solidfire -g graphite

#To monitor additional clusters add another line
# /usr/bin/python /solidfire_graphite_collector_v2.py -s
172.27.40.205 -u admin -p solidfire -g graphite
sleep 60
done
```

6. Bring up the containers using docker-compose. This will take several moments to complete.
   TIP: The containers can be brought up the first time with the (-d) flag ommited so the logs can be viewed.

```
docker-compose -f /opt/github/sfcollector/docker-compose.yml up -d
```

7. When the compose process finishes, launch a web browser to http://<VM IP Addr>
8. The Grafana web interface should appear. Log in with the admin account and password configured in the docker-compose.yml file
9. From the home dashboard, add the Graphite database as a datastore for Grafana

   A. Click the **Add data source** button
   B. In the Name field, create a name for the data source
   C. In the Type field, select Graphite
   D. In the URL field, enter “http://<VM IP>:8080”
   E. In the Access field. select proxy
   F. Check the Basic Auth checkbox
   G. In the User field, enter ‘graphite’
   H. In the Password field, enter the configured password from the basic_auth file



Edit data source

Name: graphitedb

Default: ☑

Type: Graphite

Http settings

Url: http://172.27.96.14:8080

Access: proxy

Http Auth

Basic Auth: ☑

With Credentials: □

TLS Client Auth: □

With CA Cert: □

Basic Auth Details

User: graphite

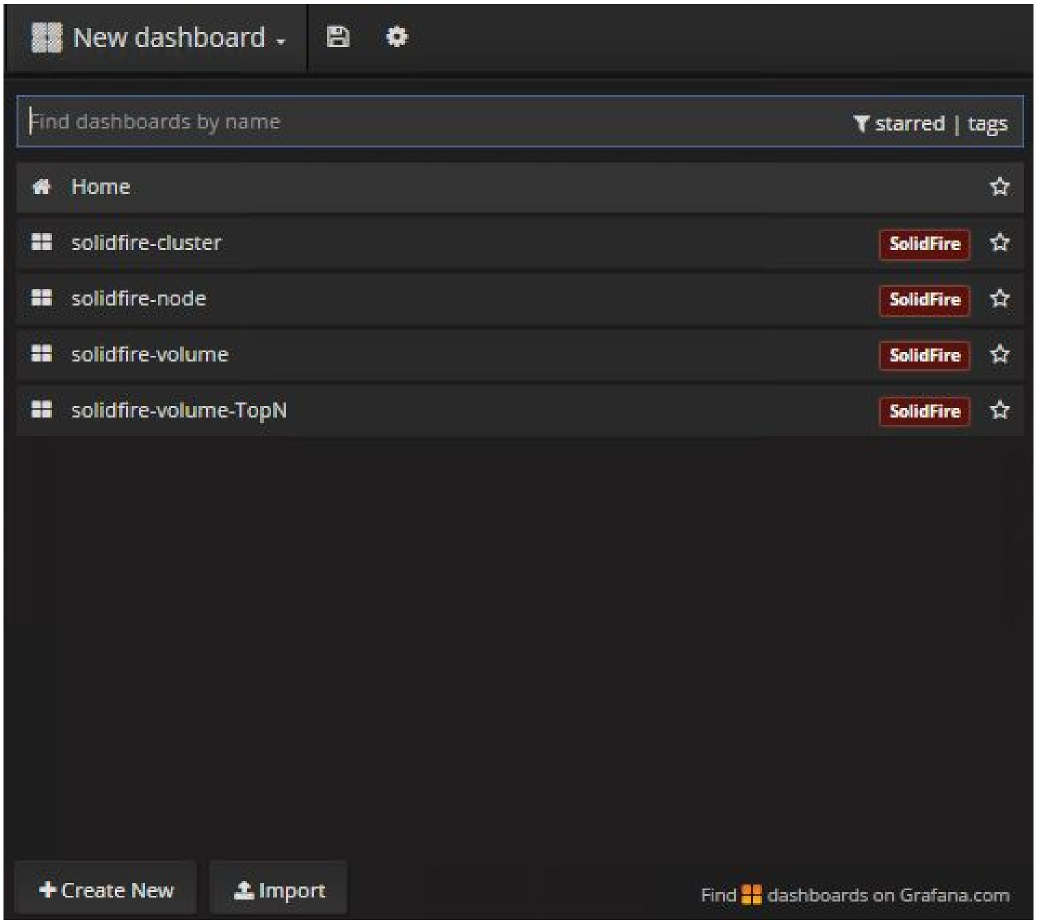
Password: •••••••••

Success
Data source is working

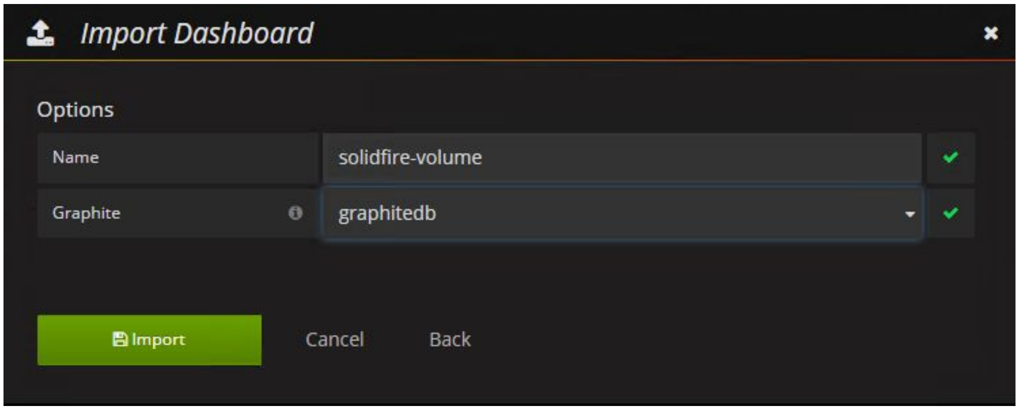
10. Import the included Grafana example dashboards.

A. Log into Grafana

B. From the Home screen, click the Import button



C. Select the Graphite database instance and click import



Import Dashboard

Options

Name: solidfire-volume

Graphite: graphitedb

D. Repeat for the other included example dashboards

Troubleshooting

Graphing a datapoint from the Graphite database to validate metrics

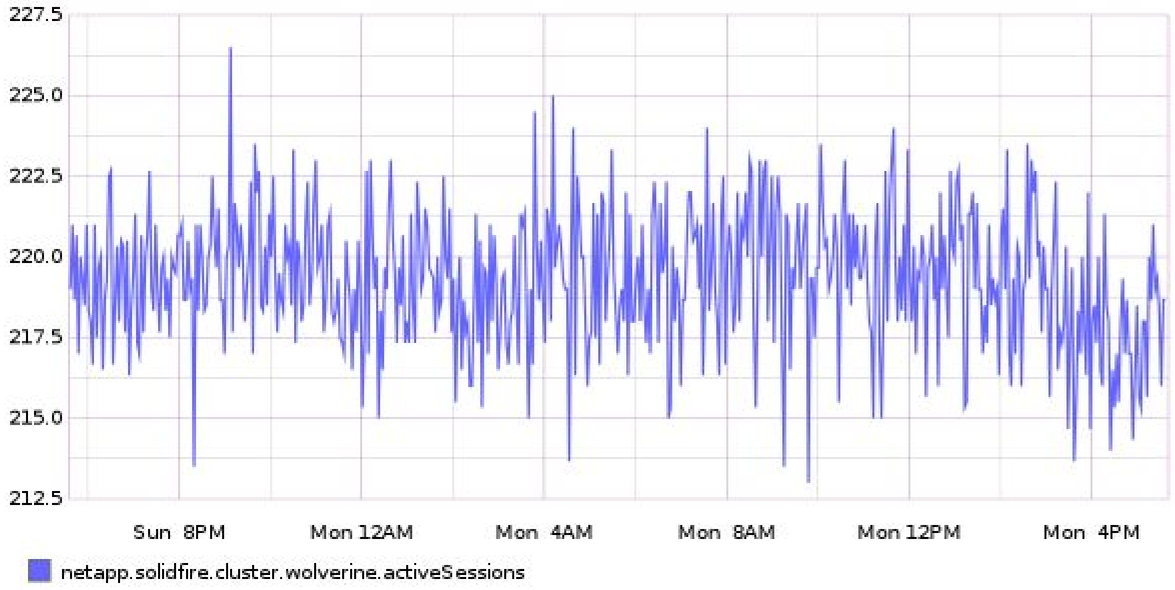
The Graphite API used in this project does not include the graphical front end for Graphite so the render API for Graphite can be used to verify that metrics are being pushed into the graphite database.

Validating SolidFire metrics are being pushed to Graphite

The format for displaying cluster metrics is:

```
http://<docker VM IP>:8080/render?target=netapp.solidfire.cluster.<cluster name>.<metric>
```

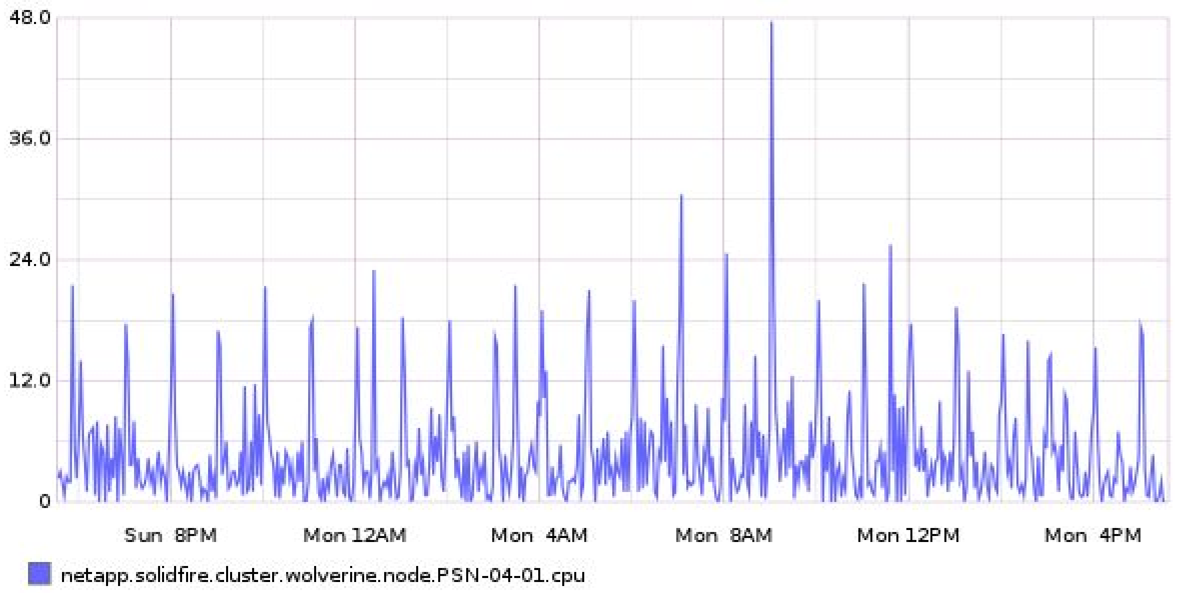
For example, to see cluster activeSessions:
http://172.27.96.14:8080/render?target=netapp.solidfire.cluster.wolverine.activeSessions



To display node metrics:

```
http://<docker VM IP>:8080/render?target=netapp.solidfire.cluster.<cluster name>.node.<node name>.<metric>
```

For example, to see a cluster node CPU usage:
http://172.27.96.14:8080/render?target=netapp.solidfire.cluster.wolverine.node.PSN-04-01.cpu



Checking sfcollector logs

If it becomes necessary to connect to the sfcollector to troubleshoot the ENTRYPOINT for the container will need to be overriden

```
docker run --entrypoint "/bin/bash" -it sfcollector_sfcollector
```

Logs for the collector are stored in the /tmp directory.

Appendix A - Source Material

#Install Docker
https://docs.docker.com/engine/installation/linux/ubuntu

#NDVP Quick Start
http://netappdvp.readthedocs.io/en/latest/quick_start.html

#Composing Graphite server w/Docker
https://thepracticalsysadmin.com/composing-a-graphite-server-with-docker/

#SolidFire Collector for Graphite
https://github.com/cbiebers/solidfire-graphite-collector

Appendix B - Docker Information

Graphite-docker container apps

carbon - a Twisted daemon that listens for time-series data
whisper - a simple database library for storing time-series data (similar in design to RRD)
graphite webapp - A Django webapp that renders graphs on-demand using Cairo
grafana - tool for graphing time series data

#Configuration files:

```
-> carbon.conf
-> nginx.conf
        --> uwsgi.conf
-> statsd.conf
-> graphite
        --> storage-aggregation.conf
        --> storage-schemas.conf
        --> supervisord.conf
-> grafana
```

#Docker Compose files

```
        graphite:
                build: ./config
                restart: always
                ports:
                        - "8080:80" #nginx web port
                        - "8125:8125/udp" #statsd
                        - "8126:8126"
                        - "2003:2003" #carbon reciever port  <- SF collector pushes here
                        - "2004:2004" #carbon pickle reciever port
                volumes:
                        - "graphite-db:/opt/graphite/storage/whisper"

        grafana:
                image: grafana/grafana
                restart: always
                ports:
                        - "80:3000"
                volumes:
                        - "grafana:/var/lib/grafana"
                links:
```

```
            - graphite
        environment:
            - GF_SECURITY_ADMIN_PASSWORD=password
```

Dockerfile for the collector container

```
# Docker container for sfcollector
# aaron.patten@netapp.com | @jedimt
FROM ubuntu:16.04
MAINTAINER aaron.patten@netapp.com
RUN apt update \
        && apt install -y python-pip git \
        && apt clean
RUN pip install solidfire-sdk-python==1.2.0.108 \
        requests \
        graphyte \
        logging
ADD solidfire_graphite_collector_v2.py /solidfire_graphite_collector_v2.py
ADD wrapper.sh /wrapper.sh
ENTRYPOINT /wrapper.sh
```