Third Party [FOSS] Data Collectors

Book for third party / free open source software data collectors.

Varken

Varken is a utility application that can gather data from Sonarr, Radarr, Lidarr, Tautulli, and Ombi to monitor your Plex Media Server setup. This info is then sent to Influx and graphed by Grafana.

Information

Varken Github: https://github.com/boerderij/Varken

Varken F.A.Q. Wiki: https://wiki.cajun.pro/books/varken

Varken Support: https://ko-fi.com/varken

Varken can monitor data from the following modules to function. Not all of them are required to operate Varken though!

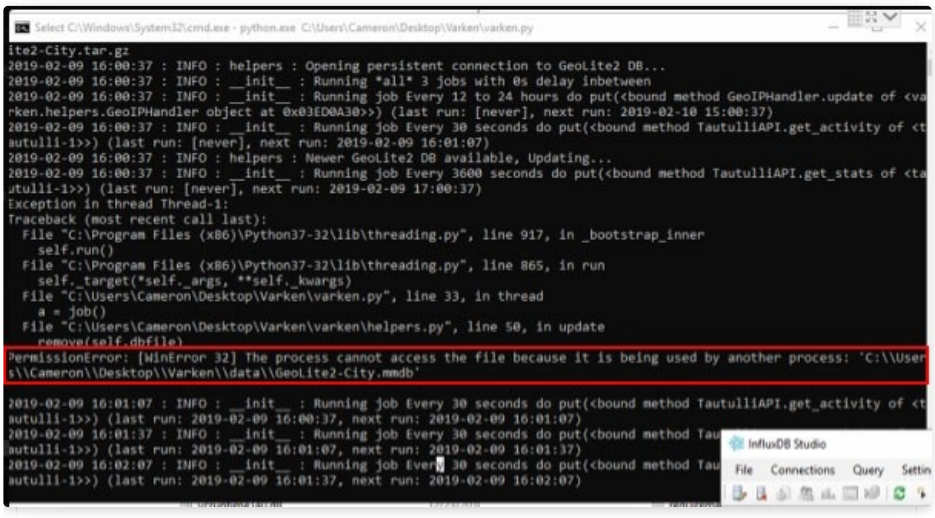
Service	URL	Purpose
Unifi Controller	https://unifi-sdn.ubnt.com	Single Pane of Glass to monitor and configure your Ubiquiti Unifi Network devices.
Sonarr	https://sonarr.tv	TV show handler/PVR. Able to read RSS feeds and import TV shows for your media server.
Radarr	https://radarr.video	Same as Sonarr but for movies.
Lidarr	https://lidarr.audio	Same as Sonarr/Radarr but for music.
SickChill	https://sickchill.github.io	Automatic video library manager for TV shows.
Tautulli	https://tautulli.com	Plex Media Server monitoring application written in Python.
Ombi	https://ombi.io	Media Requesting service for your Media server. Integrates with Sonarr, Radarr, Lidarr, etc.

PermissionError: [WinError 32] The process cannot access the file because...

> This is a Windows specific tutorial.

I encountered this error the first time I installed Varken. I'm not 100% sure if the developer has addressed it yet, but read on to see the workaround. The developer thinks it's a Windows specific error so it's unlikely other platforms will experience this.

> PermissionError: [WinError 32] The process cannot access the file because it is being used by another process 'C:\\Users\\USER_NAME\\Desktop\\Varken\\data\\GeoLite2-City.mmdb'



At this point Varken would would not talk to Influx, so I headed over to the Varken Discord channel where I spoke with DirtyCajunRice who was kind enough to help me with this.

This is an exact quote from the developer:

> in helpers.py
> comment out self.update() in init
> in varken.py
> comment out “schedule every 12-24 hours update db” line in the tautulli section

After editing the helpers.py and varken.py according to the instructions above, Varken started moving data into Influx as expected.

Standalone Scripts

A section for standalone data collection scripts.

Pi-hole InfluxDB Data Collector

⚠ This script is written in Python and thus needs Python to be installed on the host you wish to run it on.

Running the script on the Pi-hole host.

1. Install Python pip

```
sudo apt-get install python-pip -y
```

2. Create a new directory for the script to live in.

```
mkdir /opt/pihole-influx
```

3. Clone the Pi-hole script repo.

```
git clone https://github.com/janw/pi-hole-influx.git /opt/pihole-influx
```

4. Once that finishes, cd to /pihole-influx and run:

```
pip install -r requirements.txt
```

5. Now clone the config.example.ini to config.ini.

```
cp config.example.ini config.ini
```

6. Edit the config.ini file to match your environment.

```
nano config.ini
```

[InfluxDB]	
port	<influxdb port>
hostname	<ip.of.influx.db>
username	<influxdb username for pihole db>
password	<influxdb password for pihole db>
database	<db name for pihole in influxdb>
[pihole]	
api_location	address of the /admin/api.php of your pi-hole instance
instance_name	<hostname>

ℹ You can scrape multiple pi-hole instances if you run more than 1 by adding a second config block called [pihole_2].

⚠ You'll need to create the database, in InfluxDB, for the pi-hole stats. See here for how to create a new database.

✓ You can start the script by running `./piholeinflux.py` in your script directory.

Running the script as a service

1. Create the service file

```
nano piholeinflux.service
```

2. Add the following to the .service file

```
User=pi #YOUR USERNAME
ExecStart=/usr/bin/python /path/to/pihole-influx/piholeinflux.py
```

3. Create the service link

```
sudo ln -s /opt/pihole-influx/piholeinflux.service /etc/systemd/system
```

4. Enable the service and reload the service daemon

```
sudo systemctl enable piholeinflux.service && sudo systemctl daemon-reload
```

5. Start the piholeinflux service

```
sudo systemctl start piholeinflux.service
```

> ⚠ If you get an error while running make sure you can A: communicate to InfluxDB and B: the “USER=” in the .service file is set to a user that can run it (i.e. root or you).

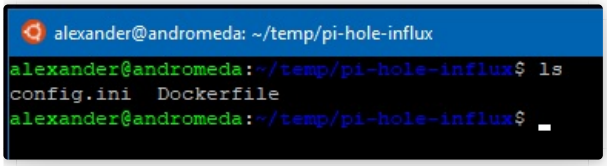
> ℹ You can check the service status by running: `sudo systemctl status piholeinflux.service`

Running the script as a Docker container

1. Copy the Dockerfile and config.ini into the same folder

```
FROM alpine as builder
RUN apk add --no-cache git
WORKDIR /app
RUN git clone https://github.com/janw/pi-hole-influx.git
FROM python:3-alpine
WORKDIR /usr/src/app
COPY --from=builder /app/pi-hole-influx/requirements.txt /usr/src/app
RUN pip install --no-cache-dir -r requirements.txt
COPY --from=builder /app/pi-hole-influx/piholeinflux.py /usr/src/app
COPY config.ini .
CMD [ "python", "./piholeinflux.py" ]
```

```
[influxdb]
port = 8086
hostname = 10.9.9.120
username = pihole
password = allthosesweetstatistics
database = pihole
# Time between reports to InfluxDB (in seconds)
reporting_interval = 10
[pihole]
api_location = http://10.9.9.120/admin/api.php
instance_name = pihole
timeout = 10
```



2. Edit the config.ini to match your environment.

(Can use above steps to see what needs changed)

3. Build the container.

> Note: You will have to be in the same subdirectory as the Dockerfile in order for it to build using the below command!

```
docker build -t your-name/of-image .
```

4. Deploy the container

```
docker run --name Pi-Hole_Influx your-name/of-image -d
```

> You can combine the docker run and build commands but pointing docker run to the Dockerfile. This will build and deploy the container!

Useful InfluxDB Collectors

Speedtest data script

Github: https://github.com/barrycarey/Speedtest-for-InfluxDB-and-Grafana

Developer: https://github.com/barrycarey

Use Case: Run periodic speedtests and output the data to InfluxDB which can then be graphed.

Comcast account scraper script

Github: https://github.com/billimek/comcastUsage-for-influxdb

Developer: https://github.com/billimek

Use Case: Periodically scrapes your comcast account for bandwidth usage and sends it to Influx to be graphed by Grafana. Can be useful to people that have data caps and want to see how fast they are using their data allowance.
