Prometheus
Monitoring system

B. VAN WALSTIJN, R. STAUTTENER, R. BOLDING
Amsterdam University of Applied Sciences
January 30, 2019

-Overview

Prometheus is an open-source systems monitoring and alerting toolkit. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community. It is now a standalone open source project and maintained independently of any company. Prometheus joined the Cloud Native Computing Foundation in 2016 as the second hosted project, after Kubernetes.

Prometheus works well for recording any purely numeric time series. It fits both machine-centric monitoring as well as monitoring of highly dynamic service-oriented architectures. In the cause of the power monitoring project we use it for the machine-centric monitoring.

Prometheus values reliability. You can always view what statistics are available about your system, even under failure conditions. If you need 100% accuracy, such as for per-request billing, Prometheus is not a good choice as the collected data will likely not be detailed and complete enough. In such a case you would be best off using some other system to collect and analyze the data for billing, and Prometheus for the rest of your monitoring.[1]

Features

- A multi-dimensional data model with time series data identified by metric name and key/value pairs
- PromQL, a flexible query language to leverage this dimensionality
- No reliance on distributed storage; single server nodes are autonomous
- Time series collection happens via a pull model over HTTP
- Pushing time series is supported via an intermediary gateway
- Targets are discovered via service discovery or static configuration
- Multiple modes of graphing and dashboarding support
- Alarm manager that is able to notify you when something is wrong

In the case of the power monitoring project we mainly use the time series collection, multiple nodes of graphing and dashboard support, targets discovery and no reliance on distributed storage features. We don't use the the alarm manager of Prometheus because Grafana already has an alarm manager and for organizing reasons we want to use the system that visualizes the data to also give you alarm alert of the data, so that it's all on one system.[1]

Architecture

Figure 1 illustrates the architecture of Prometheus and some of its ecosystem components: Prometheus scrapes metrics from instrumented jobs, either directly or via an intermediary push gateway for short-lived jobs. It stores all scraped samples locally and runs rules over this data to either aggregate and record new time series from existing data or generate alerts. Grafana or other API consumers can be used to visualize the collected data.[1]

Prometheus for power monitoring

This section is about how Prometheus is implemented in the power monitoring project. Here will be explained how to download Prometheus, how to configure Prometheus to get the metrics from the right and how to upload metrics to the right place.

Downloading Prometheus

Make sure you download the latest release of Prometheus. the latest version of Prometheus can be found on this url:

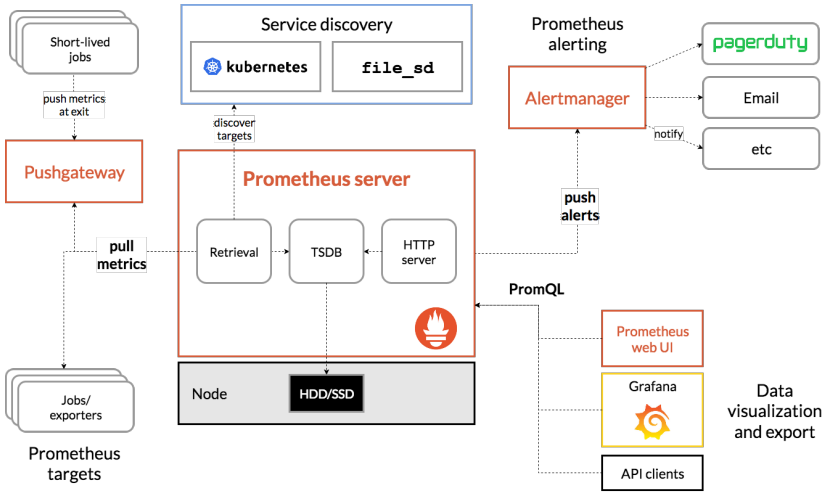

Figure 1: the architecture of Prometheus

`https://prometheus.io/download/`

Change "Operating system" from popular to all and architecture" from popular to all. pick the release that's suited for your computer. In the case of

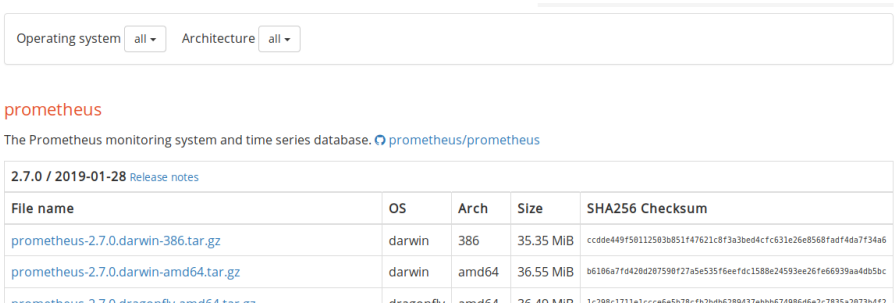

Figure 2: View of the download page of Prometheus[3]

the power monitoring project we use RaspberryPi's to simulate a computer cluster. The RPI model B+ (the one used for the power) has an ARMv8 chip, which is compatible with ARMv7 packages and that's happens to be the latest for Linux operating systems (prometheus-2.7.0.linux-armv7.tar.gz). To get this version on the RPI, use the command:

wget https://github.com/prometheus/prometheus/releases/download/v2.7.0/prometheus-2.7.0.linux-armv7.tar.gz

Extract this package to the directory you want to work in and then Prometheus is downloaded. In case of the power monitoring project Prometheus is located on the power monitoring git and with the ansible install/update script (see the section Ansible) not only prometheus but also all the other dependencies for this project will be installed on the RPI on the right location. [2]

Configuring Prometheus

Prometheus is configured with the file *prometheus.yml*. Configuring prometheus

```
# my global config
global:
  scrape_interval:     1s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
    - targets: ['localhost:8000']
```

Figure 3: The part of metrics_INA219.py that uploads metrics

to pull metrics from the right place happens under *scrape_configs*. The metrics gatherer uploads the metrics on port 8000 on a localhost. As said before: we are going to use the alarm manager of Grafana, but its implemented anyway in case we change directions in the future and want to use the Prometheus alarm manager.

Uploading metrics

Uploading metrics happens in de *metrics_INA219.py* file located in the *CurrentMeasurement* directory (see GIT). First it starts a Prometheus http sever

```
collect_time = 1


inamodule = INA219(0.1, 2.0)
inamodule.configure(inamodule.RANGE_16V, inamodule.GAIN_8_320MV, inamodule.ADC_12BIT, inamodule.ADC_12BIT)


REQUEST_TIME = Summary('request_processing_seconds',
                       'Time spent processing request')

prometheus_current = Gauge('energy_current', 'Filtered current of the system')
prometheus_voltage = Gauge('energy_voltage', 'Filtered voltage of the system')
prometheus_power = Gauge('energy_power', 'Filtered power of the system')


filtered_current = Filtervalue(inamodule.current())
filtered_voltage = Filtervalue(inamodule.voltage() * 1000)
filtered_power = 0.0


if __name__ == '__main__':
        # Start server for Prometheus to listen to
        start_http_server(8000)

        while True:

                # Filter every /interval/ time
                start_time = time.time()
                while time.time() < start_time + collect_time:
                        filtered_current.updatevalue(inamodule.current()               , 0.1)
                        filtered_voltage.updatevalue(inamodule.voltage() * 1000, 0.1)

                filtered_power = filtered_current.averagevalue * filtered_voltage.averagevalue / 1000

                # post metrics
                prometheus_current.set(filtered_current.averagevalue)
                prometheus_voltage.set(filtered_voltage.averagevalue)
                prometheus_power.set(filtered_power)
```

Figure 4: The part of metrics_INA219.py that uploads metrics

on port 8000, which is the port Prometheus will scrape it's data from. Before the part of *metrics_INA219.py* that uploads the metrics, is a part that filters all the power and CPU measurements and makes them ready to be a metric. Once the measurements have been made ready to be turned into metrics they will be turned into readable metrics for Prometheus by the Gauge constructor.

References

1. Prometheus Authors 2014-2019. 2019 [Accessed on Oct 2018]. [Online]. Available: `https://prometheus.io/docs/introduction/overview/`
2. Prometheus Authors 2014-2019. 2019 [Accessed on Oct 2018]. [Online]. Available: `https://prometheus.io/docs/introduction/first_steps/`
3. Prometheus Authors 2014-2019. 2019 [Accessed on Oct 2018]. [Online]. Available: `https://prometheus.io/download/`