AIOps: Anomaly detection with Prometheus

Spice up your Monitoring with AI

Marcel Hild
Principal Software Engineer @ Red Hat AI CoE / Office of the CTO
Marcel Hild

old school opensource hacker and daemon zombie slayer at @b4mad and Red Hat’s @AlCoE CTO Office

Red Hat
Kiel, Germany
hild@b4mad.net
http://durandom.de

Organizations
HOW RED HAT SEES AI

Represents a workload requirement for our **platforms** across the hybrid cloud.

Applicable to Red Hat’s existing core business in order to increase **Open Source** development and production **efficiency**.

Valuable to our customers as specific services and product capabilities, providing an **Intelligent Platform** experience.

Enable customers to build **Intelligent Apps** using Red Hat products as well as our broader partner ecosystem.

**Data as the Foundation**
HOW RED HAT SEES AI

Represents a workload requirement for our platforms across the hybrid cloud.

Applicable to Red Hat’s existing core business in order to increase Open Source development and production efficiency.

Valuable to our customers as specific services and product capabilities, providing an Intelligent Platform experience.

Enable customers to build Intelligent Apps using Red Hat products as well as our broader partner ecosystem.

Data as the Foundation

Project Thoth and Bots http://bit.ly/2zYfb6h
HOW RED HAT SEES AI

Represented as a workload requirement for our platforms across the hybrid cloud.

Applicable to Red Hat’s existing core business in order to increase Open Source development and production efficiency.

Valuable to our customers as specific services and product capabilities, providing an Intelligent Platform experience.

Enable customers to build Intelligent Apps using Red Hat products as well as our broader partner ecosystem.

Data as the Foundation

Project Thoth and Bots
http://bit.ly/2zYfb6h

OpenDataHub
http://bit.ly/2y6Nh6m
HOW RED HAT SEES AI

Represents a workload requirement for our platforms across the hybrid cloud.

Applicable to Red Hat’s existing core business in order to increase Open Source development and production efficiency.

Valuable to our customers as specific services and product capabilities, providing an Intelligent Platform experience.

Enable customers to build Intelligent Apps using Red Hat products as well as our broader partner ecosystem.

Data as the Foundation

Project Thoth and Bots
http://bit.ly/2zYfb6h

OpenDataHub
http://bit.ly/2y6Nh6m

This Talk
Overview

Prometheus

Long term storage

Atonomy of an Anømål¥

Integration into monitoring setup
What's **not** in this talk

- shiny product and the holy grail of monitoring
- ready solution to turn your monitoring setup into spider demon
- success story how we turned our messy monitoring into an advance ai monitoring
What **is** in this talk

- tools and scripts to get you started
- Q&A to problems
- all OSS
What is Prometheus?
Prometheus architecture

- Service discovery
  - kubernetes
  - file_sd
- Prometheus server
  - Retrieval
  - TSDB
  - HTTP server
- Prometheus targets
- Short-lived jobs
- Jobs/exporters
- Pushgateway
  - pull metrics
- Node
  - HDD/SSD
- Alertmanager
  - Prometheus alerting
- Pagerduty
  - notify
  - Email
  - etc
- Prometheus web UI
  - Data visualization and export
  - Grafana
  - API clients

Everybody loves architecture slides
Prometheus architecture

Simplistic world view
Prometheus architecture

Simplistic world view
Prometheus architecture

Simplistic world view
Prometheus architecture

Simplistic world view
Prometheus architecture

Simplistic world view
Prometheus is made for

MONITORING

ALERTING

SHORT TERM TIME SERIES DB
What do we need for machine learning?

--->

DATA DATA DATA DATA
Long term storage of Prometheus data
Too good to be true...

- Prometheus at scale
- Global query view
- Reliable historical data storage
- Unlimited retention
- Downsampling

thanos is in the making, but until then?
Works great, but...

- easily hooked into prometheus with write and read endpoint
- Reliable historical data storage
- Great for data science
  - Pandas integration

Eats RAM for breakfast

gh/AlCoE/p-influx
Let’s just store it...

**prometheus scraper**

- container can be configured to scrape any prometheus server
- can scrape all or a subset of the metrics
- stores data in ceph or S3 compliant storage
- can be queried with spark sql
- Future Proof: path to Thanos

[GitHub link](gh/AICoE/p-lts)

[Website link](http://bit.ly/2Qw9pho)
Harness the power of spark to
- Query stored JSON files
- Distribute the workload
- Use spark library

**notebook**

```python
def get_stats(df):
    # calculate mean
    mean = df.agg(F.avg(F.col("values"))).drop(0)

    # calculate variance
    var = df.agg(F.variance(F.col("values"))).drop(0)

    # calculate standard deviation
    stddev = df.agg(F.stddev(F.col("values"))).drop(0)

    # calculate median
    median = float(df.approxQuantile("values", [0.5], 0.25).drop(0))

    return mean, var, stddev, median

mean, var, stddev, median = get_stats(data)

print("\tMean(values): ", mean)
print("\tVariance(values): ", var)
print("\tStddev(values): ", stddev)
print("\tMedian(values): ", median)
```

---

Mean(values): 67087.9063346175
Variance(values): 56691431555.4375
Stddev(values): 238099.62527361838
Median(values): 628.0
What do we need for machine learning?

--->

CONSISTENT DATA
Prometheus Metric Types

- **Gauge**: A Time Series
- **Counter**: Monotonically Increasing
- **Histogram**: Cumulative Histogram of Values
- **Summary**: Snapshot of Values in a Time Window
Prometheus Metric Types

- Gauge
- Counter
Prometheus Metric Types

- Histogram
- Cumulative
- Summary
- Time Window
Anatomy of a metric
Kubelet Docker Operations Latency

E.g. docker_latency

Hostname
Operation Type
Clam Controller Enabled

Time Value

Time Value

Time Value

Marcel Hild, Red Hat
E.g. docker_latency

Every unique combination of labels makes up a Time Series
Monitoring is hard

GET /metrics

```javascript
# HELP go_gc_duration_seconds A summary of the GC duration
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 9.7014e-07
  go_gc_duration_seconds{quantile="0.25"} 0.00
  go_gc_duration_seconds{quantile="0.5"} 0.00
  go_gc_duration_seconds{quantile="0.75"} 0.00
  go_gc_duration_seconds{quantile="1"} 0.1029
  go_gc_duration_seconds_sum 0.239829369
  go_gc_duration_seconds_count 196
# HELP go_goroutines Number of goroutines the runtime is tracking.
# TYPE go_goroutines gauge
  go_goroutines 144
# HELP go_memstats_alloc_bytes Number of bytes allocated and still
# TYPE go_memstats_alloc_bytes gauge
  go_memstats_alloc_bytes 4.5694928e+07
# HELP go_memstats_alloc_bytes_total Total number of bytes allocated
# TYPE go_memstats_alloc_bytes_total counter
  go_memstats_alloc_bytes_total 4.19435624e+09
```
analysis of metrics meta data
analysis of metrics meta data

Meta-data tooling
Anomaly Types
<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Trend</strong></td>
<td>Increase or decrease in the series over a period of time.</td>
</tr>
<tr>
<td><strong>Seasonality</strong></td>
<td>Regular pattern of up and down fluctuations. It is a short-term variation occurring due to seasonal factors.</td>
</tr>
<tr>
<td><strong>Cyclicity</strong></td>
<td>It is a medium-term variation caused by circumstances, which repeat in irregular intervals.</td>
</tr>
<tr>
<td><strong>Irregularity</strong></td>
<td>It refers to variations which occur due to unpredictable factors and also do not repeat in particular patterns.</td>
</tr>
</tbody>
</table>

![Graph showing Trend, Seasonality, and Irregularity](image-url)
Anomaly Types

Point-wise

Seasonal

Seasonal
Anomaly Detection with Prophet

Predicting future data and dynamic thresholds

- list_images operation
- on OpenShift
- monitored by prometheus
- detecting outliers
- upper and lower bands
Anomaly Detection with Prophet
Extracting trends and seasonality

- list_images operation
- on OpenShift
- monitored by prometheus
- upward trends
- intraday seasonality

CoE/prophet
The Accumulator

Anomaly Detection and Forecast for HTTP request duration microseconds

Value

0 30000 60000 90000 120000 150000 180000 210000 240000

Timestamp

0 10000 20000 30000 40000 50000 60000 70000 80000

train

test

forecast
The Tail Probability
Combined
architecture setup so far
Research Setup

100% OpenSource Tooling
Now what? I want to <insert installer img>
Prometheus Training Pipeline

Targets Providing metric data to Prometheus host

Live metrics data from Prometheus to train prediction models

Prophet Forecasting
Fourier Extrapolation

Target N+1

Flask Server Serving Predicted Metrics

Predicted values served as metric data
- Ready to use container
  - Local deployment
  - Kubernetes
  - OpenShift build config

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dockerfile</td>
<td>Update Dockerfile</td>
</tr>
<tr>
<td>Makefile</td>
<td>Add Makefile for ease of</td>
</tr>
<tr>
<td>README.md</td>
<td>Update README.md</td>
</tr>
<tr>
<td>app.py</td>
<td>Add more comments for</td>
</tr>
<tr>
<td>ceph.py</td>
<td>Add functionality to retain</td>
</tr>
<tr>
<td>model.py</td>
<td>Make the live data query</td>
</tr>
<tr>
<td>prometheus.py</td>
<td>Make the live data query</td>
</tr>
<tr>
<td>requirements.txt</td>
<td>Update requirements.txt</td>
</tr>
<tr>
<td>train-prophet-deployment-template</td>
<td>Add deployment template</td>
</tr>
</tbody>
</table>
Runtime configuration

```python
# Specific metric to run the model on
metric_name = os.getenv('METRIC_NAME','kubelet_docker_operations_latency_microseconds')
```

Expose predictions via `/metrics` endpoint

```plaintext
# HELP predicted_kubelet_docker_operations_latency_microseconds_prophet_anomaly Detected Anomaly using the Prophet model
# TYPE predicted_kubelet_docker_operations_latency_microseconds_prophet_anomaly gauge
predicted_kubelet_docker_operations_latency_microseconds_prophet_anomaly
  {beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",ocp_prod_upshift.eng.rdu2.redhat.com",operation_type="version",provider="rhos",quantile="0.5",region="compute",size="small"} 0.0
# HELP predicted_kubelet_docker_operations_latency_microseconds_fourier_anomaly Detected Anomaly using the Fourier model
# TYPE predicted_kubelet_docker_operations_latency_microseconds_fourier_anomaly gauge
predicted_kubelet_docker_operations_latency_microseconds_fourier_anomaly
  {beta_kubernetes_io_arch="amd64",beta_kubernetes_io_os="linux",ocp_prod_upshift.eng.rdu2.redhat.com",operation_type="version",provider="rhos",quantile="0.5",region="compute",size="small"} 0.0
```
Marcel Hild, Red Hat
Alerting Rules

groups:
- name: Testing alert
rules:

- alert: MetricOutOfRange
  expr: kubelet_docker < ignoring(job, instance) predicted_values_prophet_yhat_lower or kubelet_docker > ignoring(job, instance) predicted_values_prophet_yhat_upper
  for: 5m
  annotations:
    summary: "Metric out of bounds"
    description: "Metric is out of range of the predicted Prophet values"

- alert: MetricOutOfRange
  expr: kubelet_docker < ignoring(job, instance) predicted_values_fourier_yhat_lower or kubelet_docker > ignoring(job, instance) predicted_values_fourier_yhat_upper
  annotations:
    summary: "Metric out of bounds"
    description: "Metric is out of range of the predicted Fourier values"
QUESTIONS?

- Project Thoth and Bots
- OpenDataHub
  - [https://bit.ly/2y6Nh6m](https://bit.ly/2y6Nh6m)
- Meta-data tooling
- CoE/prom-ad
- CoE/prophet
- gh/AICoE/p-influx
- gh/AICoE/p-lts
- notebooks