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

durandom

old school opensource hacker and
demon zombie slayer at @b4mad
and Red Hat's @AICoE 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

Data as the Foundation

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

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.

OpenDataHub
http://bit.ly/2y6Nh6m

Marcel Hild, Red Hat
HOW RED HAT SEES AI

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

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.

**OpenDataHub**
http://bit.ly/2y6Nh6m

Data as the Foundation

This Talk
Overview

Prometheus

Long term storage

Atonomy of an Anomaly

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

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/AICoE/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

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

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

    # calculate variance
    var = df.agg(F.variance(F.col("values"))).head()[0]

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

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

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

Success on OpenShift

Blog Post https://red.ht/2HpB8Az
What do we need for machine learning?

---＞ CONSISTENT DATA
Prometheus Metric Types

- **Gauge**: A time series, monotonically increasing
- **Counter**: A time series, cumulative
- **Histogram**: A cumulative histogram of values
- **Summary**: A snapshot of values in a time window
Prometheus Metric Types

Gauge

Counter
Prometheus Metric Types

Histogram
Cumulative
Summary
Time Window
Anatomy of a metric
E.g. docker_latency

Kubelet
Docker
Operations
Latency

Hostname
Operation
Type
Clam
Controller
Enabled

Time
Value

Marcel Hild, Red Hat
E.g. `docker_latency`

Every unique combination of labels makes up a Time Series

- **Hostname**: `free-stg-master-0-3fb6`
- **Operation Type**: `List Images`
- **Clam Controller Enabled**: `True`
Monitoring is hard

GET /metrics

- prometheus doesn't enforce a schema
  - /metrics can expose anything it wants
  - no control over what is being exposed by endpoints or targets
  - it can change if your endpoints change versions
- # of metrics to choose from
  - 1000+ for OpenShift
- State of the Art is Dashboards and Alerting
  - Dashboards and Alerting need domain knowledge
- No tools to explore meta-information in metrics

```bash
# 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-05
  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 th
# TYPE go_goroutines gauge
  go_goroutines 144
# HELP go_memstats_alloc_bytes Number of byte
# TYPE go_memstats_alloc_bytes gauge
  go_memstats_alloc_bytes 4.5694928e+07
# HELP go_memstats_alloc_bytes_total Total r
# 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
Components of Time Series

<table>
<thead>
<tr>
<th>Trend</th>
<th>Seasonality</th>
<th>Cyclicity</th>
<th>Irregularity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increase or decrease in the series over a period of time.</td>
<td>Regular pattern of up and down fluctuations. It is a short-term variation occurring due to seasonal factors.</td>
<td>It is a medium-term variation caused by circumstances, which repeat in irregular intervals.</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 components of time series](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
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
  - Target 1
  - Target 2
  - Target N

- Prometheus
  - 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-templa...</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
Real vs Prophet

kubelet_docker_operations_latency_microseconds(beta_kubernetes_io_arch="amd64", beta_kubernetes_io_os="linux", instance="cpt-0001.ocp.prod.upshift.eng.rdu2")
predicted_kubelet_docker_operations_latency_microseconds_prophet_yhat_upper(beta_kubernetes_io_arch="amd64", beta_kubernetes_io_os="linux", exported_instance="c")
predicted_kubelet_docker_operations_latency_microseconds_prophet_yhat_lower(beta_kubernetes_io_arch="amd64", beta_kubernetes_io_os="linux", exported_instance="c")
predicted_kubelet_docker_operations_latency_microseconds_prophet(beta_kubernetes_io_arch="amd64", beta_kubernetes_io_os="linux", exported_instance="c")
Alerting Rules

```yaml
groups:
  name: Testing alert
  rules:

  - alert: MetricOutOfRangeBounds
    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: MetricOutOfRangeFourierBounds
    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"
```

---

Marcel Hild, Red Hat