AIOps: Anomaly detection with Prometheus

Marcel Hild, Red Hat
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
daemon zombie slayer at @b4mad
and Red Hat's @AICoE CTO Office

Red Hat

Kiel, Germany

hild@b4mad.net

http://durandom.de

Organizations

#b4MAD
Office of the CTO
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

Represent 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.

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

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.

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

OpenDataHub http://bit.ly/2y6Nh6m

Data as the Foundation
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
Agenda

Prometheus
Long term storage
Anatomy 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

- Prometheus server
  - Retrieval
  - TSDB
  - HTTP server
- Prometheus targets
- Pushgateway
  - pull metrics
- Jobs/exporters
- Short-lived jobs
  - push metrics at exit
- Service discovery
  - kubernetes
  - file_sd
- Prometheus alerting
  - pagerduty
    - Email
    - etc
  - notify
  - push alerts
  - Prometheus web UI
  - Grafana
    - API clients
- PromQL
- Data visualization and export

Everybody 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?
Show me your 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

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 REALLY 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
E.g. docker_latency
E.g. docker_latency

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

GET /metrics

```plaintext
# HELP go_gc_duration_seconds A summary of the
time spent in the GC
# TYPE go_gc_duration_seconds summary
go_gc_duration_seconds{quantile="0"} 9.7014e-05
go_gc_duration_seconds{quantile="0.25"} 0.0001
go_gc_duration_seconds{quantile="0.5"} 0.0001
go_gc_duration_seconds{quantile="0.75"} 0.0001
go_gc_duration_seconds{quantile="1"} 0.0001
go_gc_duration_seconds_sum 0.239829369
# HELP go_goroutines Number of goroutines that
are running
# TYPE go_goroutines gauge
go_goroutines 144
# HELP go_memstats_alloc_bytes Number of bytes
that have been allocated
# TYPE go_memstats_alloc_bytes gauge
go_memstats_alloc_bytes 4.5694928e+07
# HELP go_memstats_alloc_bytes_total Total amount
of bytes allocated
# TYPE go_memstats_alloc_bytes_total counter
go_memstats_alloc_bytes_total 4.19435624e+09
```

- 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
analysis of metrics meta data
Marcel Hild, Red Hat

analysis of metrics meta data

Meta-data tooling
Anomaly Types
Components of Time Series

**Trend**
Increase or decrease in the series over a period of time.

**Seasonality**
Regular pattern of up and down fluctuations. It is a short-term variation occurring due to seasonal factors.
Anomaly Types

- Point-wise
- 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

CoE/decision rules
The Tail Probability
Combined

CoE/decision rules
architecture setup so far
Now what? I want to
Prometheus Training Pipeline

- Targets Providing metric data to Prometheus host
- Live metrics data from Prometheus to train prediction models

Prometheus

- Prophet Forecasting
- Fourier Extrapolation

Target N

Metric Data

Predicted values served as metric data

Flask Server Serving Predicted Metrics

Target N+1
Prometheus

API /query

Forecaster / Prophet
store forecasted values

Scrape /metrics

Exporter
read

PVS
- 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-templates</td>
<td>Add deployment templates</td>
</tr>
</tbody>
</table>
Expose predictions via /metrics endpoint

# 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" 0001.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" 0001.ocp.prod.upshift.eng.rdu2.redhat.com",operation_type="version",provider="rhos",quantile="0.5",region="compute",size="small"} 0.0
Alerting Rules

```yaml
groups:
  - name: Testing alert
    rules:
      - alert: MetricOutofProphetBounds
        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: MetricOutofFourierBounds
        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
Demo
Time
I like what you got
QUESTIONS?

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

- Meta-data tooling

- CoE/prophet

- CoE/decision rules

- CoE/prom-ad

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

- Thanos Blog Post
  https://red.ht/2HpB8Az

- notebooks

- gh/AICoE/p-influx

- CoE/prom-ad
  http://bit.ly/2Qw9pho

- CoE/decision rules

- Meta-data tooling