

Detecting Workload Anomalies with Prometheus and Machine Learning

Anthony E. Nocentino

@nocentino

If you want to follow along: <https://github.com/nocentino/MetricsML/>

April 8-11, 2024





PURE STORAGE®



ScriptRunner®

The #1 for PowerShell Management



PDQ DataOn®



PATCH
MY PC



MANNING



Anthony E. Nocentino

Principal Field Solution Architect @ Pure Storage

- Specialize in system architecture, performance, SQL Server, Kubernetes, Containers, Microsoft Azure and VMware
- Masters Computer Science

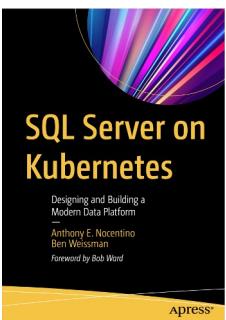
email: anocentino@purestorage.com

Blog: www.nocentino.com

Twitter: @nocentino

GitHub: <https://github.com/nocentino/>

Pluralsight Author: www.pluralsight.com



EIGHTKB



Agenda

- Introduction to Telegraf, Prometheus and Grafana
- Building an Anomaly Detection Model
- Practical Use Cases for Anomaly Detection

If you want to follow along: <https://github.com/nocentino/MetricsML>

Why Did I Build This?

Customers often have no monitoring platforms 😞

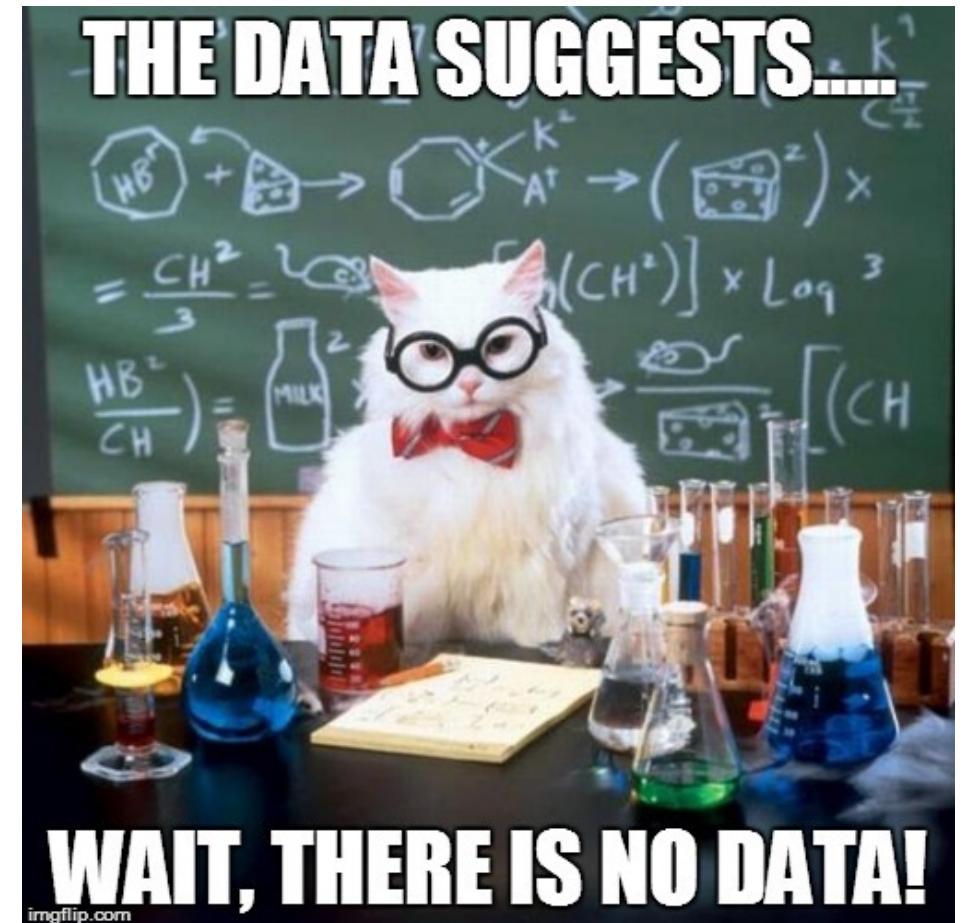
Defining what's normal and what's not normal

A couple disclaimers

- I'm not a data scientist
- Also, not a professional developer

But I had a problem I wanted to solve

<https://github.com/nocentino/MetricsML>



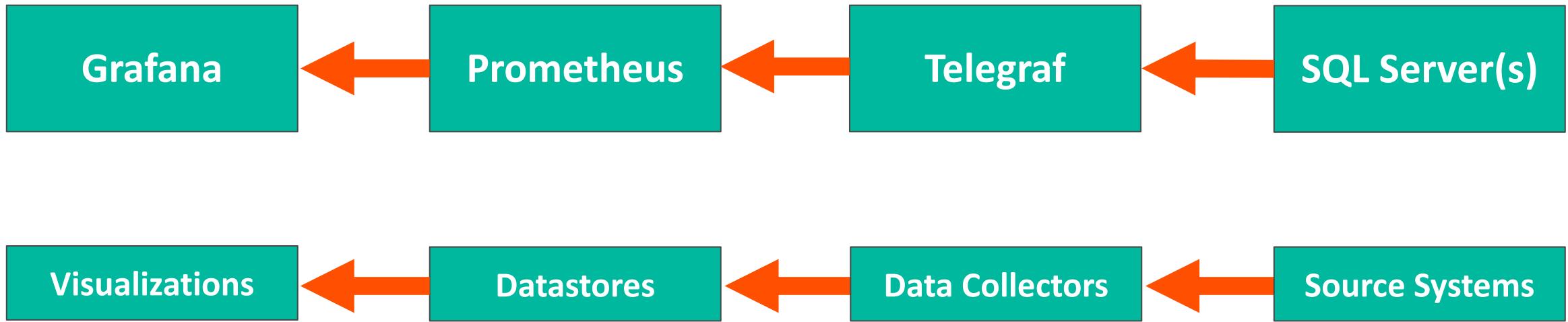
What's Observability?

Understanding what's happening and how it impacts the health of the system



Monitoring Architecture

Keeping an eye on complex systems



Container Based Deployment

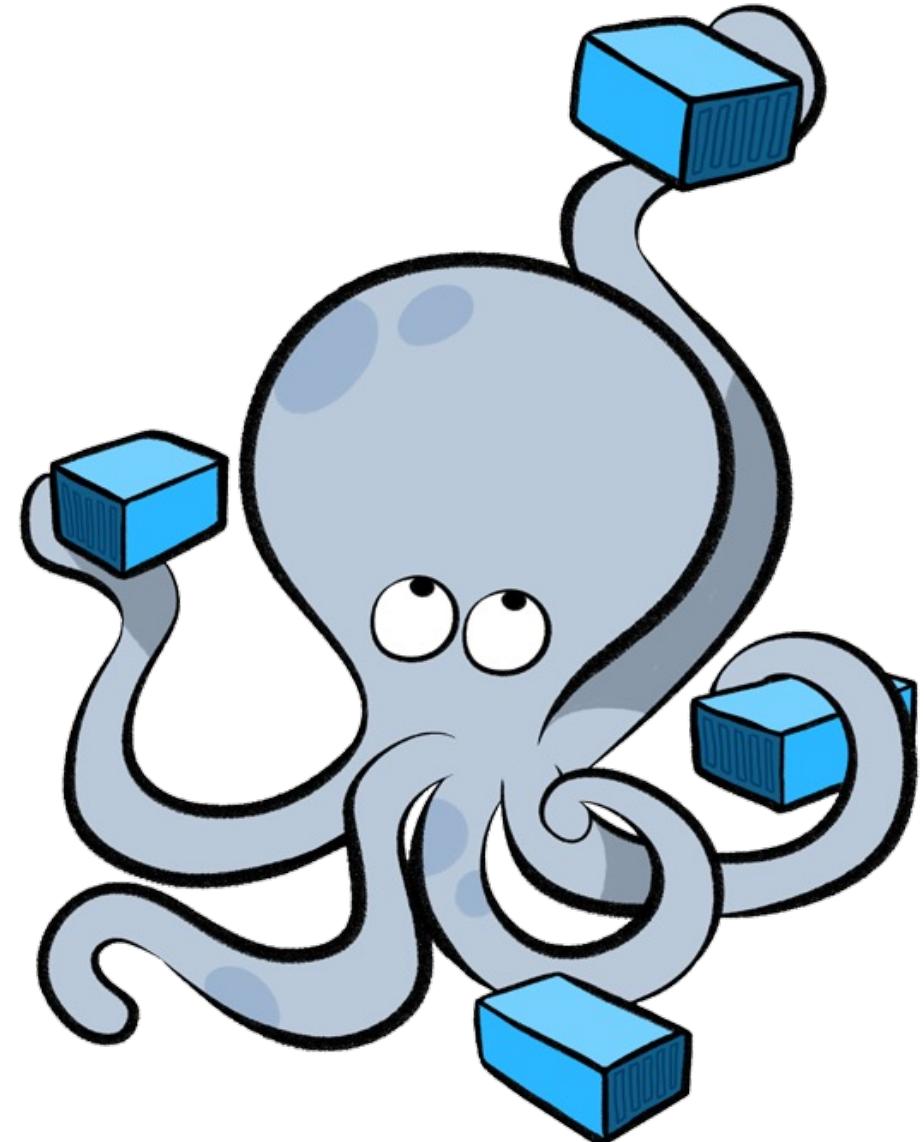
Container is a self-contained application

Docker Compose

- Starts up the containers
- Configures the applications
- Networking connecting the applications
- Expose Grafana to the network

Orchestrated solution defined in code

Can run anywhere you have Docker



Telegraf – What's a Metric?

Performance data

Plugins

Collecting metrics

Exposes a metrics HTTP/S endpoint



```
# HELP sqlserver_cpu_other_process_cpu Telegraf collected metric
# TYPE sqlserver_cpu_other_process_cpu untyped
sqlserver_cpu_other_process_cpu{host="telegraf",measurement_db_type="SQLServer",sql_instance="sql1"} 100
sqlserver_cpu_other_process_cpu{host="telegraf",measurement_db_type="SQLServer",sql_instance="sql2"} 100
# HELP sqlserver_cpu_sqlserver_process_cpu Telegraf collected metric
# TYPE sqlserver_cpu_sqlserver_process_cpu untyped
sqlserver_cpu_sqlserver_process_cpu host="telegraf",measurement_db_type="SQLServer",sql_instance="sql1" 0
sqlserver_cpu_sqlserver_process_cpu host="telegraf",measurement_db_type="SQLServer",sql_instance="sql2" 0
# HELP sqlserver_cpu_system_idle_cpu Telegraf collected metric
# TYPE sqlserver_cpu_system_idle_cpu untyped
sqlserver_cpu_system_idle_cpu{host="telegraf",measurement_db_type="SQLServer",sql_instance="sql1"} 0
sqlserver_cpu_system_idle_cpu{host="telegraf",measurement_db_type="SQLServer",sql_instance="sql2"} 0
```

Other Telegraf Plugins

 Windows Server 2022



vmware[®]



<https://github.com/influxdata/telegraf>

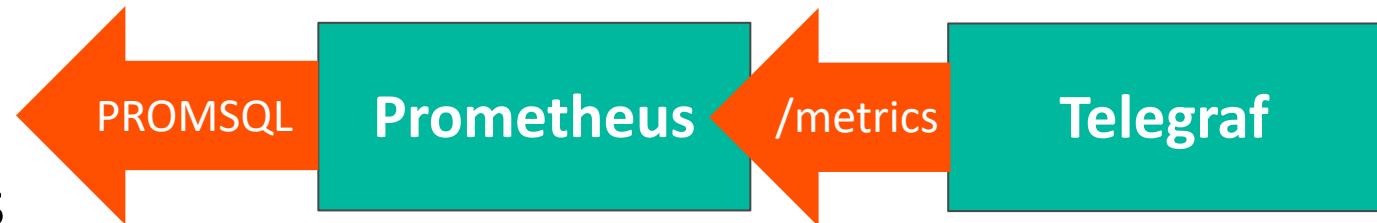
Prometheus – Time Series Database to Store Metrics



Time Series Database

Stores collected metrics

Jobs scrape the metrics endpoints



The screenshot shows the Prometheus web interface with the following elements:

- Header:** Shows the URL `localhost:9090/graph?g0.expr=&g0.tab=1&g0.display_mode=lines&g0.show_exemplars=0&g0.range_input=1h` with the port number highlighted.
- Toolbar:** Includes links for Prometheus, Alerts, Graph, Status, Help, and various configuration icons.
- Query Settings:** Options for "Use local time", "Enable query history", "Enable autocomplete" (checked), "Enable highlighting" (checked), and "Enable linter" (checked).
- Query Input:** A search bar containing the query `sqlserver_cpu_sqlserver_process_cpu sql_instance='sql1' [1d]`, with the instance part highlighted.
- Result Preview:** A table showing the results of the query. The first row is highlighted with an orange box and contains the metric name `sqlserver_cpu_sqlserver_process_cpu{host="telegraf", instance="telegraf:9273", job="telegraf", measurement_db_type="SQLServer", sql_instance="sql1"}`.
- Result Data:** The rightmost column shows two data points:
 - 0 @1712069022.464
 - 0 @1712069052.455

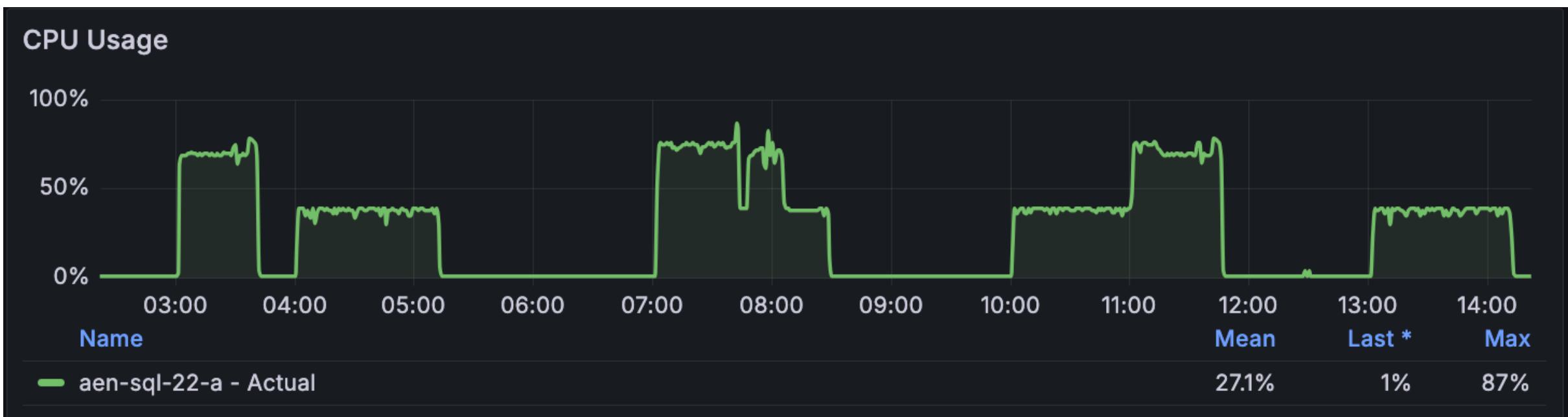
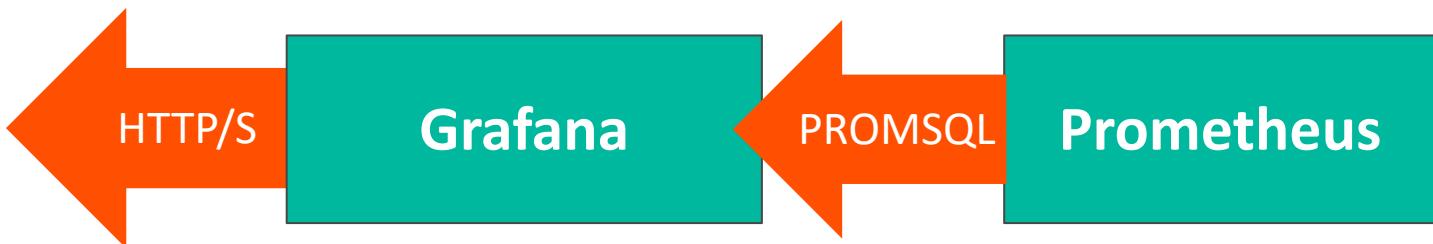
Grafana

Visualization tool

Define dashboards

Queries a datastore

See trends over time



Let's hop into a demo

Start up the monitoring stack

Look at how metrics are exposed and collected

Check out how Grafana visualizes metrics

Machine Learning

What is Machine Learning (ML)?

Predictions based on past performance

Take a sample of data

Fit a model to data points

Parameters are used to tune the model

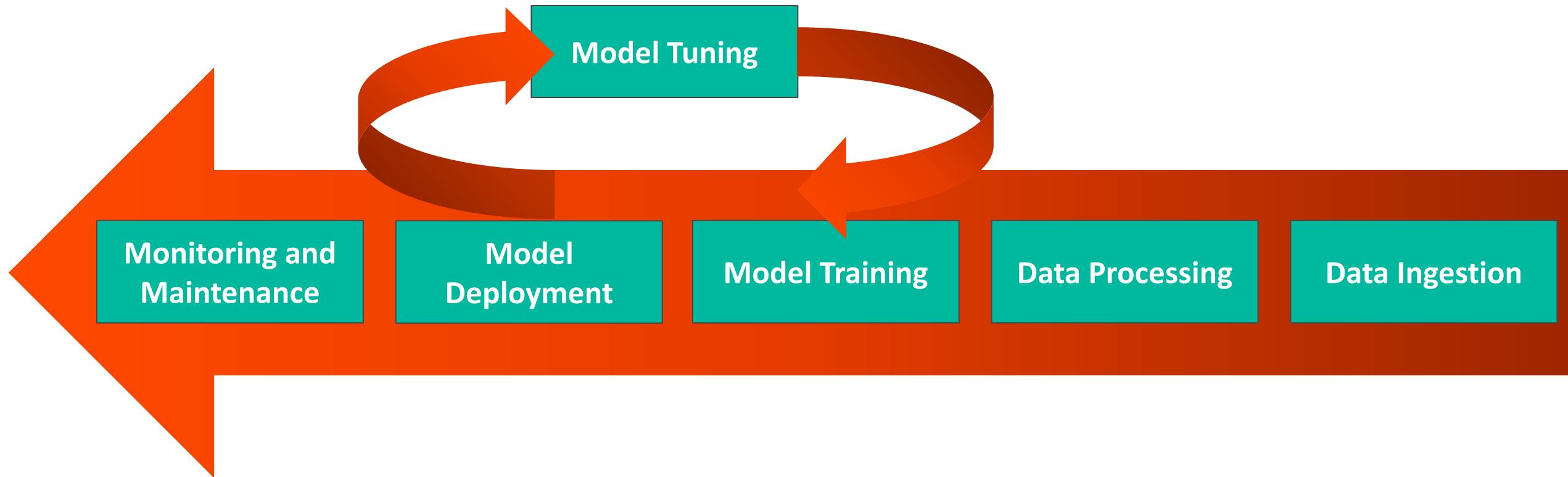
Python and R

I don't know Python

So, I had to use Copilot for this



A Machine Learning Pipeline



Using Prophet

I not an expert at Machine Learning and I'm not a data scientist

Find something off the shelf

Time series machine learning model

Use for forecasting based on seasonality

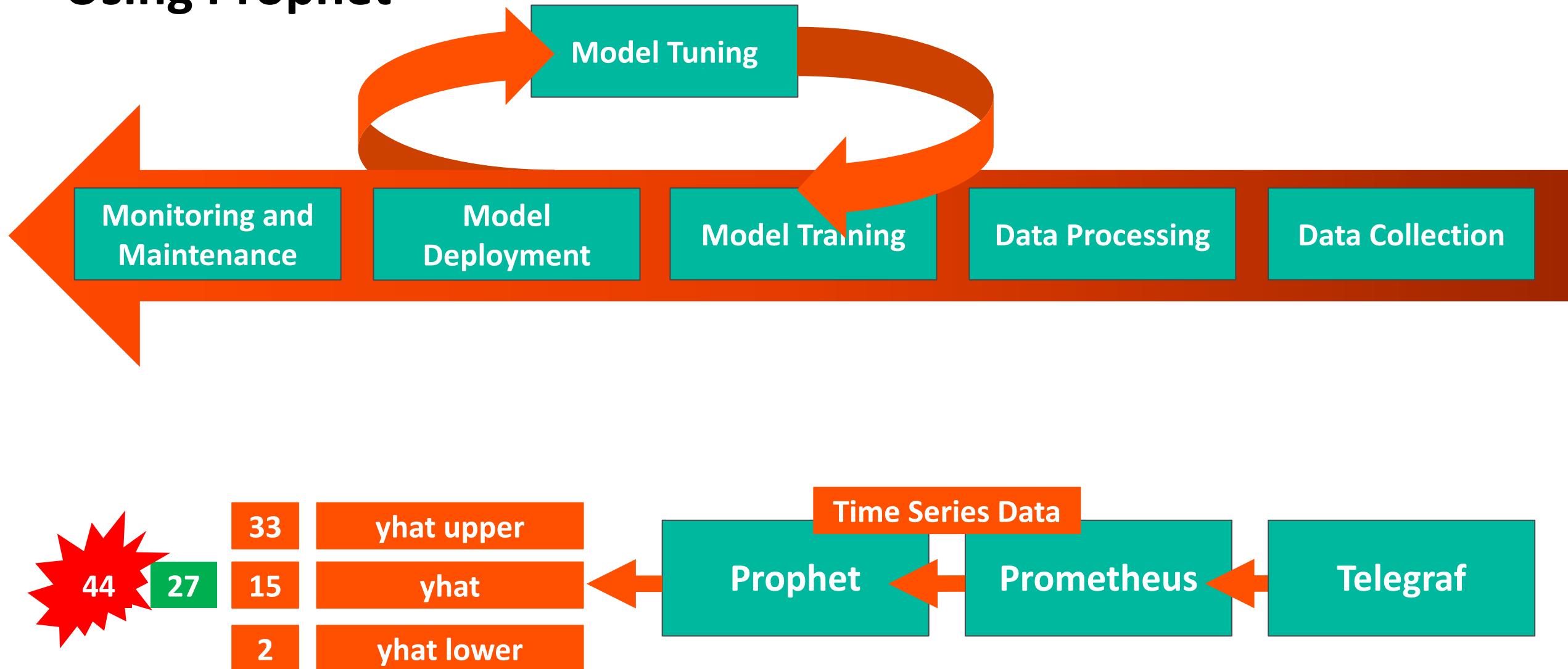
Tunable and Automated

Available in R or Python

Returns a **predicted value**, and an error range **high** and **low**

<https://facebook.github.io/prophet/>

Using Prophet



Let's hop into a demo

Using Prophet at the CLI to generate predicted values

Building your own Metrics Exporter

Prometheus Client

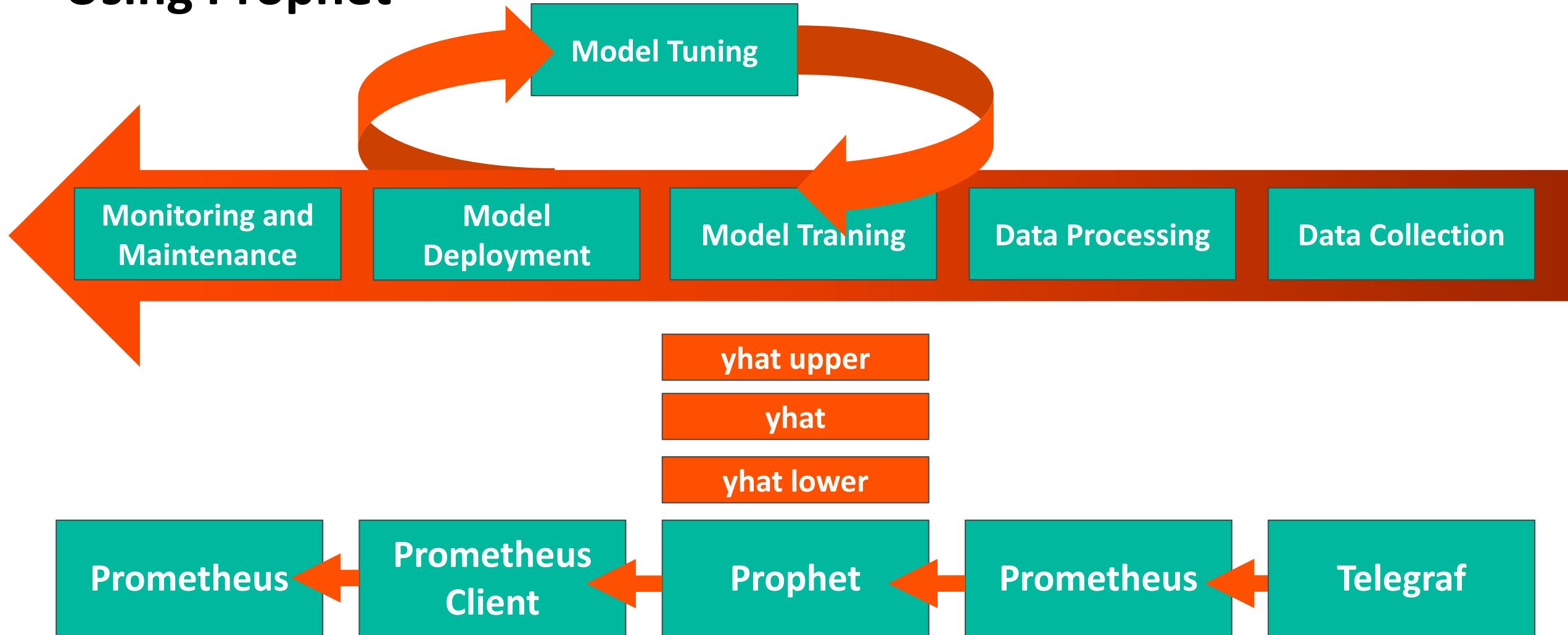
Defines an interface for the metric output

Gets metrics from the target via the network

Generally, HTTP/HTTPS

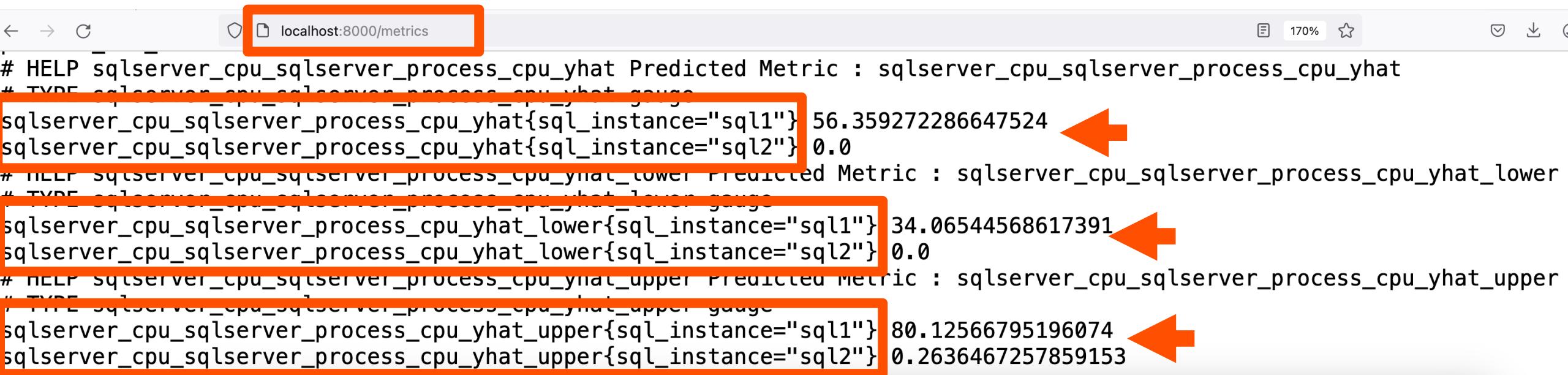


Using Prophet



MetricsML – Bringing it all together

Use the Prometheus Client to expose our predicted values back



A screenshot of a web browser window displaying Prometheus metrics. The address bar shows "localhost:8000/metrics". The page content lists several Prometheus metrics, each with an orange box highlighting specific parts of the metric name and its value. Three orange arrows point from the bottom right towards these highlighted areas.

```
# HELP sqlserver_cpu_sqlserver_process_cpu_yhat Predicted Metric : sqlserver_cpu_sqlserver_process_cpu_yhat
# TYPE sqlserver_cpu_sqlserver_process_cpu_yhat gauge
sqlserver_cpu_sqlserver_process_cpu_yhat{sql_instance="sql1"} 56.359272286647524 ←
sqlserver_cpu_sqlserver_process_cpu_yhat{sql_instance="sql2"} 0.0 ←

# HELP sqlserver_cpu_sqlserver_process_cpu_yhat_lower Predicted Metric : sqlserver_cpu_sqlserver_process_cpu_yhat_lower
# TYPE sqlserver_cpu_sqlserver_process_cpu_yhat_lower gauge
sqlserver_cpu_sqlserver_process_cpu_yhat_lower{sql_instance="sql1"} 34.06544568617391 ←
sqlserver_cpu_sqlserver_process_cpu_yhat_lower{sql_instance="sql2"} 0.0 ←

# HELP sqlserver_cpu_sqlserver_process_cpu_yhat_upper Predicted Metric : sqlserver_cpu_sqlserver_process_cpu_yhat_upper
# TYPE sqlserver_cpu_sqlserver_process_cpu_yhat_upper gauge
sqlserver_cpu_sqlserver_process_cpu_yhat_upper{sql_instance="sql1"} 80.12566795196074 ←
sqlserver_cpu_sqlserver_process_cpu_yhat_upper{sql_instance="sql2"} 0.2636467257859153 ←
```

Let's hop into a demo

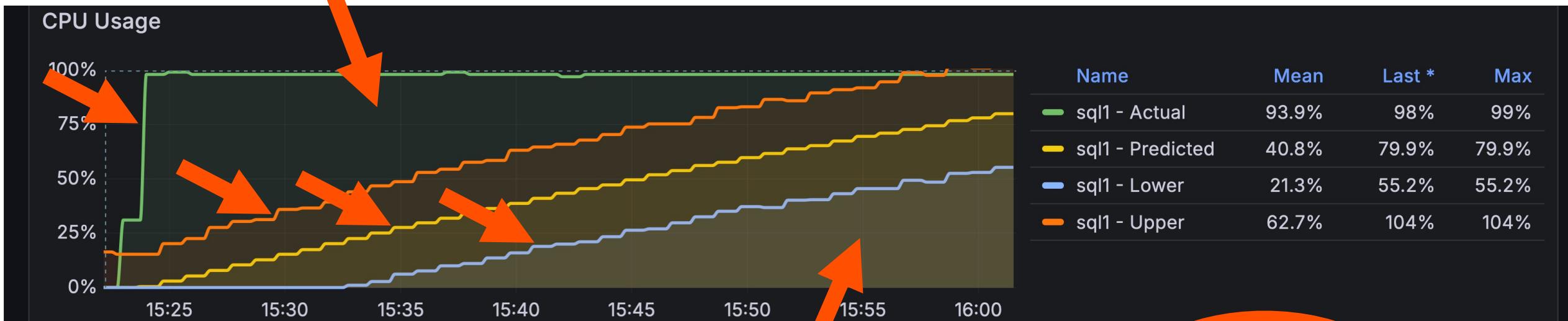
Getting data into Prophet from Prometheus using Python

Build some visualizations in Grafana

What's happening here?

Too high

When is
too high
an issue?



Where are the anomalies?

Too low

When is
too low an
issue?

Example Chart

A little more tuning is needed

Delayed alerting can help

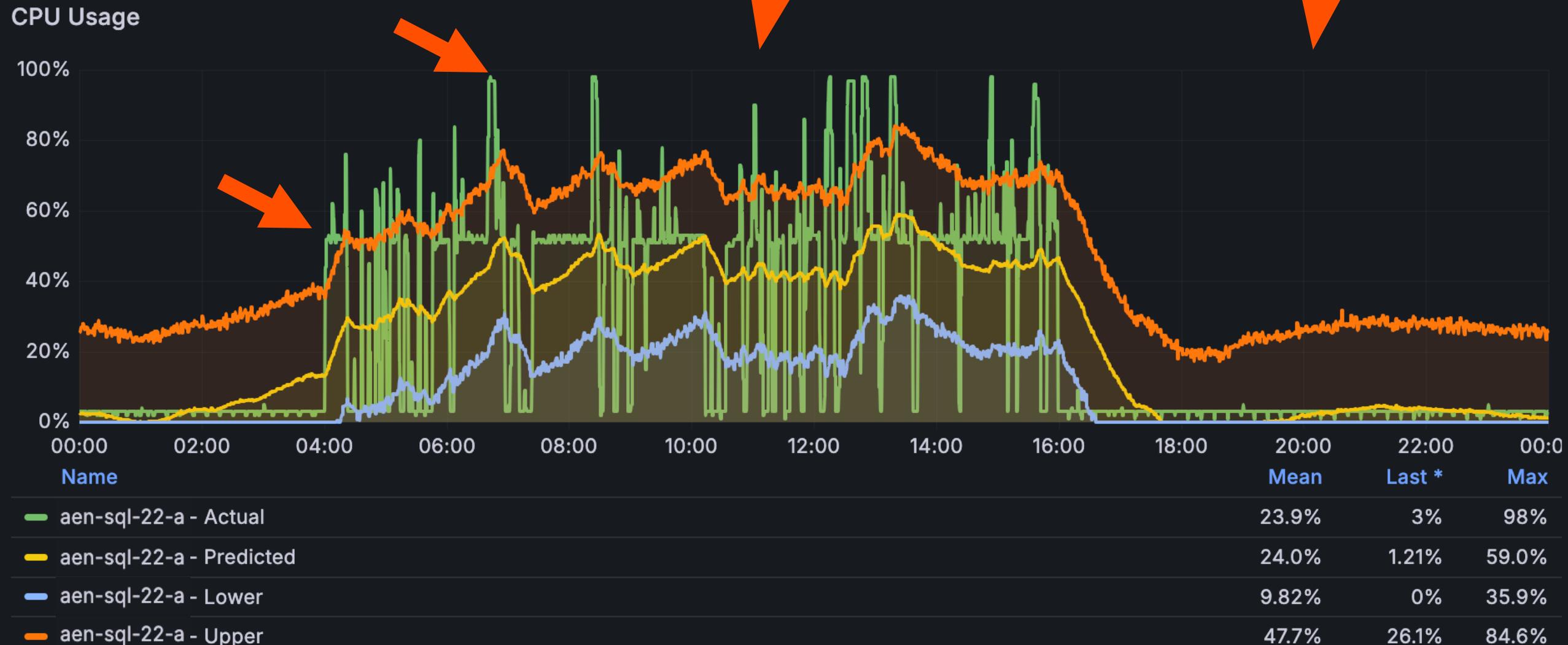


Name	Mean	Last *	Max
aen-sql-22-a - Actual	46.1%	98%	99%
aen-sql-22-a - Predicted	47.2%	58.8%	78.5%
aen-sql-22-a - Lower	27.9%	36.6%	56.2%
aen-sql-22-a - Upper	68.7%	81.2%	101%

Example Chart

A little more tuning is needed

Delayed alerting can help



Next Steps...

- More tuning in Prophet
- Lots of tuning
- Generalize for all metrics
- Parallelize the model training/prediction
- Configure alerting

Review

- Introduction to Telegraf, Prometheus and Grafana
- Building an Anomaly Detection Model
- Practical Use Cases for Anomaly Detection

THANK YOU

Please review this session



Session Review

April 8-11, 2024