

Observability

2022 • GIORGIA BERTACCHINI

MLOps • 2022

Monitoring

GROUND

Monitor performance metrics

- Statistical measures. Accuracy, precision...

Monitor performance resources

- Track the system metrics to understand which models are consuming the most application resources (CPU, memory, and GPU usage).

Supervised learning methods

- alarms

<https://madewithml.com/courses/mlops/monitoring/>

System health

- to ensure that the actual system is up and running as it should. This can include metrics specific to service requests such as latency, throughput, error rates, etc. as well as infrastructure utilization such as CPU/GPU utilization, memory, etc.

Performance

- These could be quantitative evaluation metrics that we used during model evaluation (accuracy, precision, f1, etc.) but also key business metrics that the model influences (ROI, click rate, etc.).

Monitoring

<https://www.jeremyjordan.me/ml-monitoring/>

What should we be monitoring?

At a high level, there's three classes of metrics that we'll want to track and monitor.

Model metrics

- Prediction distributions
- Feature distributions
- Evaluation metrics (when ground truth is available)

System metrics

- Request throughput
- Error rate
- Request latencies
- Request body size
- Response body size

Resource metrics

- CPU utilization
- Memory utilization
- Network data transfer
- Disk I/O

Monitoring

ML DRIFT

Label drift: output data shifts (Target drift)

Feature drift: input data shifts (Features drift)

- Data drift: when the distribution of the production data is different from the training data.

Entity	Description	Drift
X	inputs (features)	data drift $\rightarrow P(X) \neq P_{ref}(X)$
y	outputs (ground-truth)	target drift $\rightarrow P(y) \neq P_{ref}(y)$
$P(y X)$	actual relationship between X and y	concept drift $\rightarrow P(y X) \neq P_{ref}(y X)$

Grafana + Prometheus

CLOUD

The good news are that in the Cloud Native world, there are well established tools to achieve that, namely Prometheus and Grafana.

In a nutshell, **Prometheus** is a platform for monitoring application metrics and generating alerts for these metrics, while **Grafana** provides a platform to visualizing in real-time.

PROMETHEUS

Prometheus is **Docker and Kubernetes compatible** and available on the Docker Hub.

The **Prometheus server** has its own self-contained unit that does not depend on network storage or external services. So it doesn't require a lot of work to deploy additional infrastructure or software.

Grafana + Prometheus

PROMETHEUS

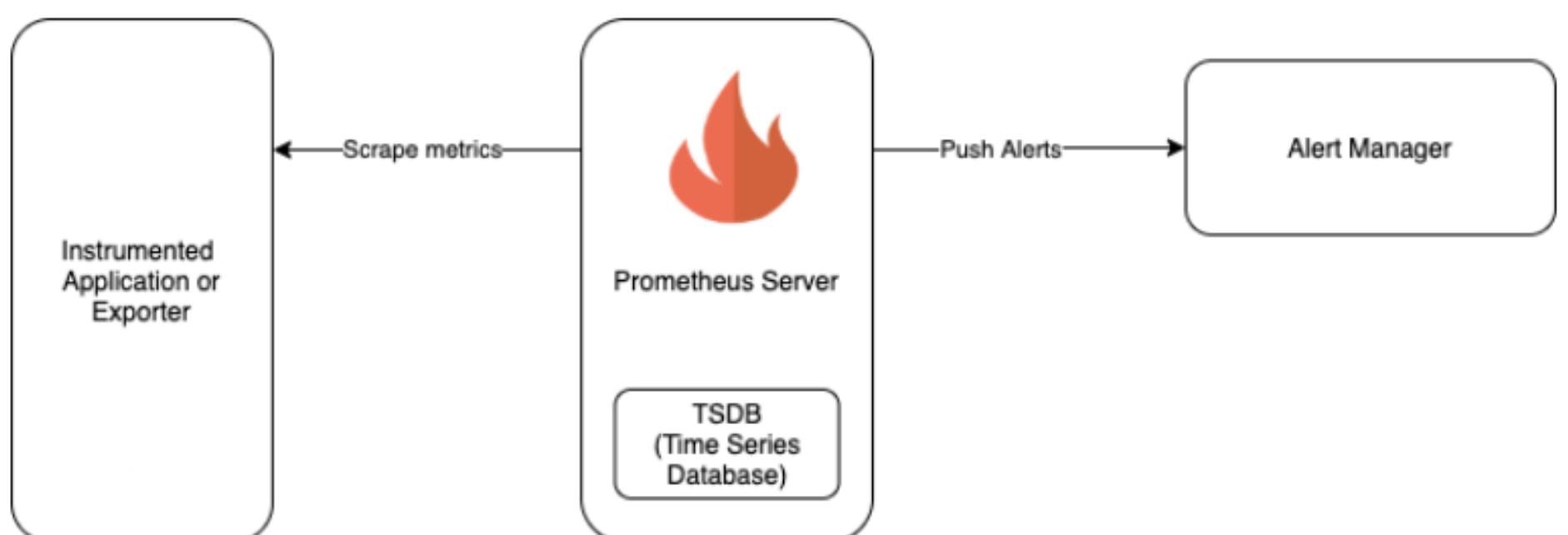
Each element that you want to monitor is called a **metric**.

The Prometheus server **reads targets** at an interval that you define to collect metrics and stores them in a **time series database**.

You query the Prometheus time series database for where metrics are stored using the **PromQL query language**.

The basic components of a Prometheus setup are:

- **Prometheus Server** (the server which scrapes and stores the metrics data).
- **Targets** to be scraped, for example an instrumented application that exposes its metrics, or an exporter that exposes metrics of another application.
- **Exporters** are optional external programs that ingest data from a variety of sources and convert it to metrics that Prometheus can scrape.
- **Alertmanager** to raise alerts based on preset rules.



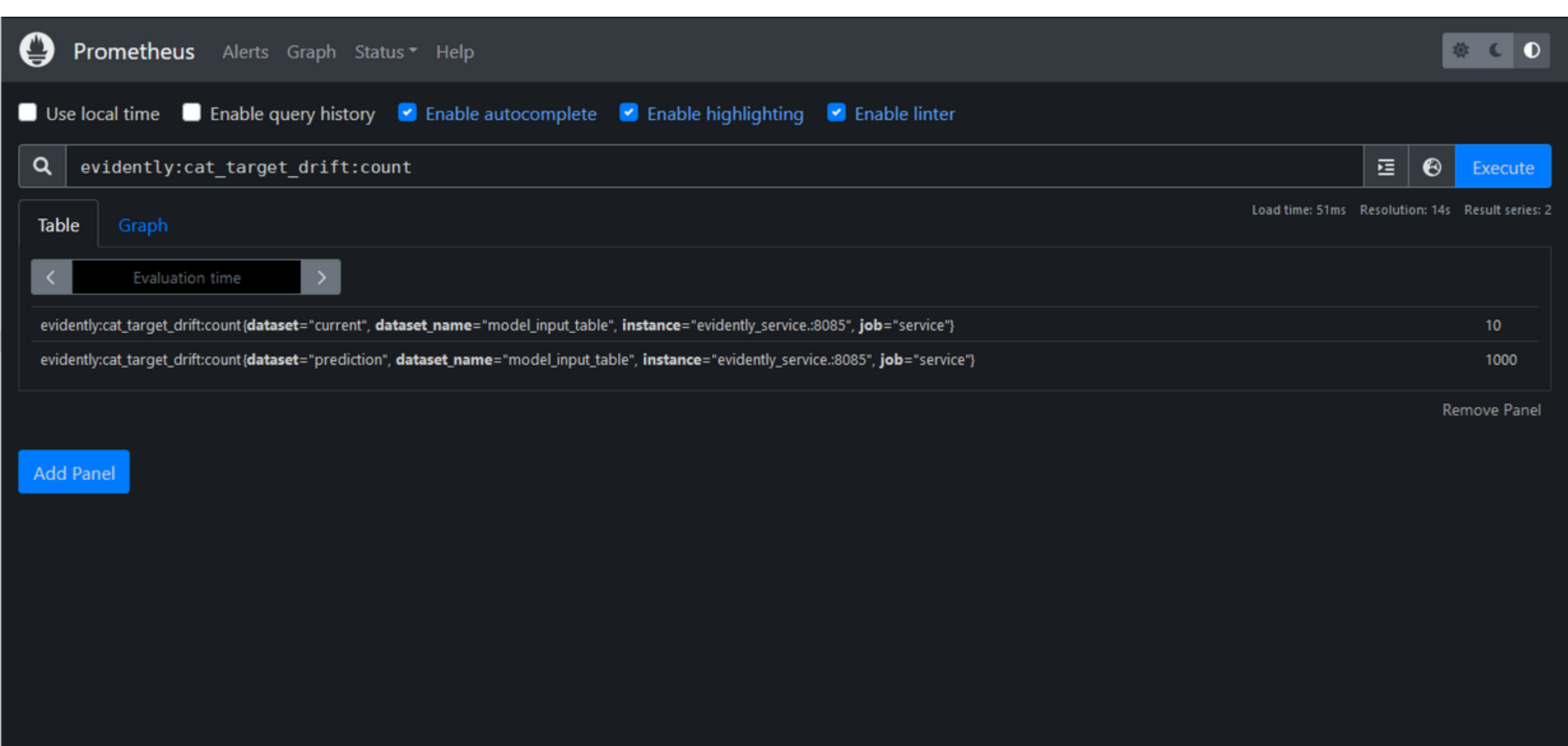
Grafana + Prometheus

PROMETHEUS QUERY

Prometheus provides a functional query language called **PromQL** (**Prometheus Query Language**) that lets the user select and aggregate time series data in real time. The result of an expression can either be shown as a graph, viewed as tabular data in Prometheus's expression browser.

Query examples:

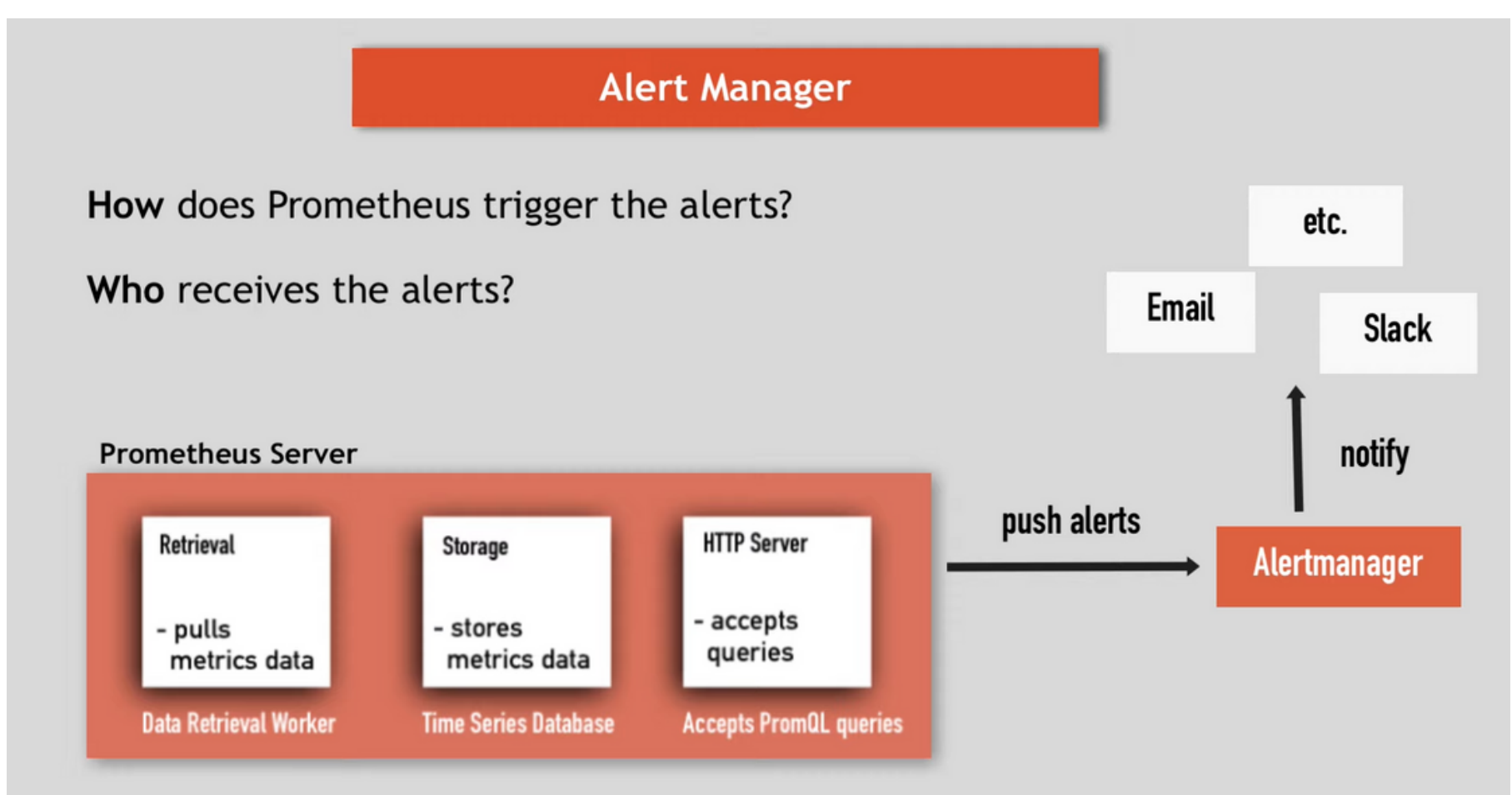
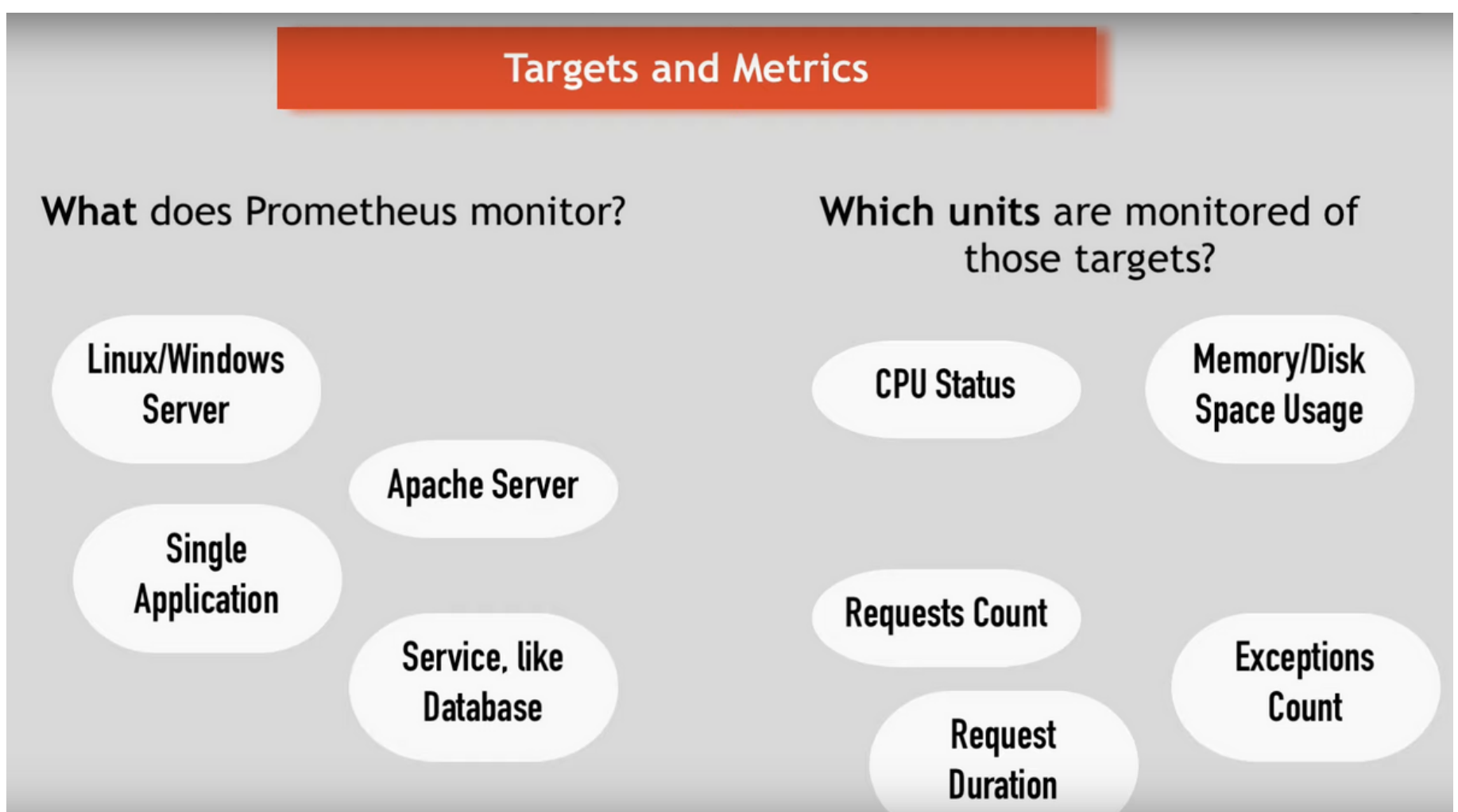
- `http_request_duration_seconds{instance="localhost:3005"} [1m]`
 - specific a label value ("localhost:3005")
 - specific that take the results of last 1 min
- You can use also Function as sum, rate.



Grafana + Prometheus

PROMETHEUS

- <https://www.youtube.com/watch?v=h4Sl21AKiDg&t=1013s>

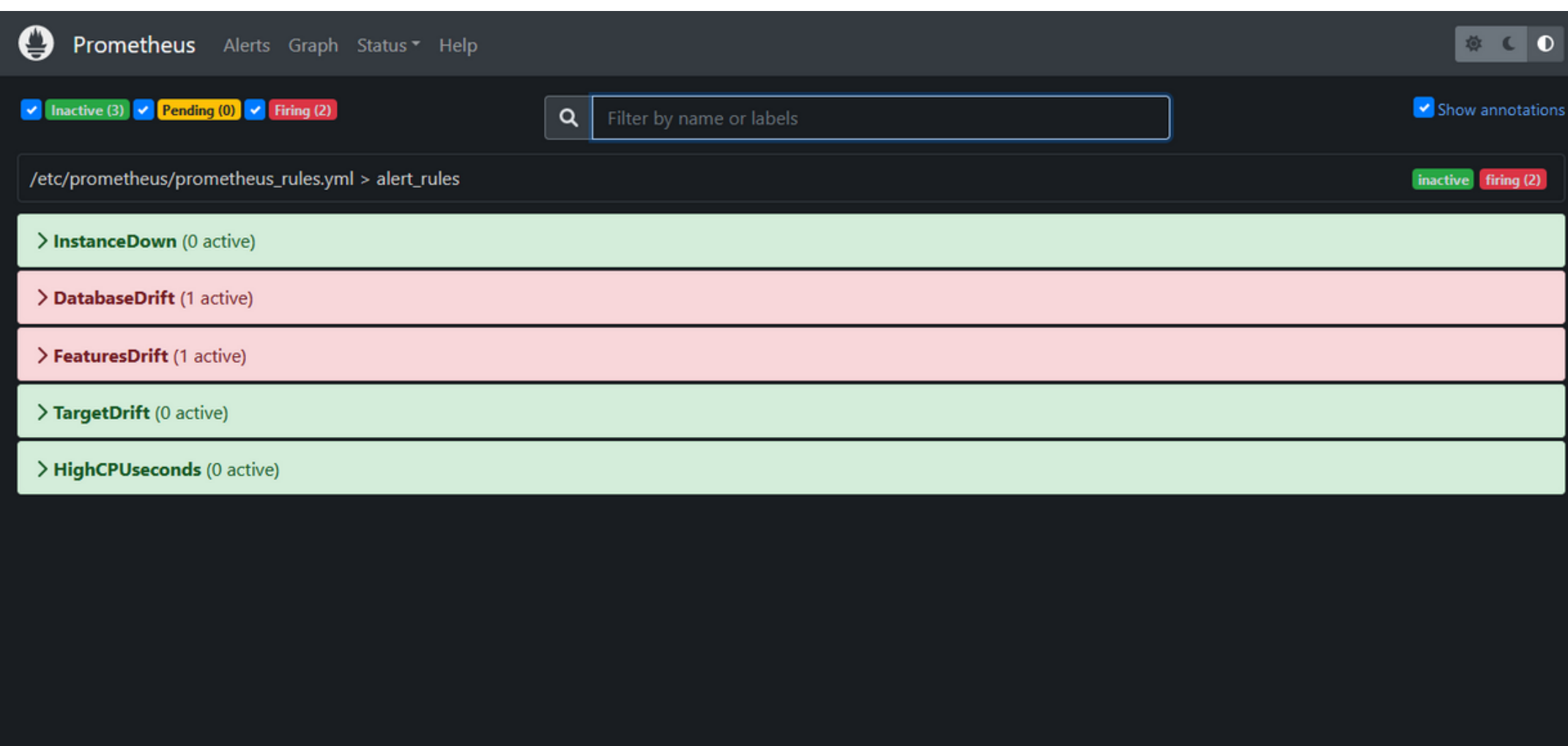


Grafana + Prometheus

ALERTING

Alerting with Prometheus is separated into two parts.

- **Alerting rules** in Prometheus servers send alerts to an Alertmanager.
- The **Alertmanager** then manages those alerts and sending out notifications via methods such as
 - email
 - slack
 - webhook
 - telegram



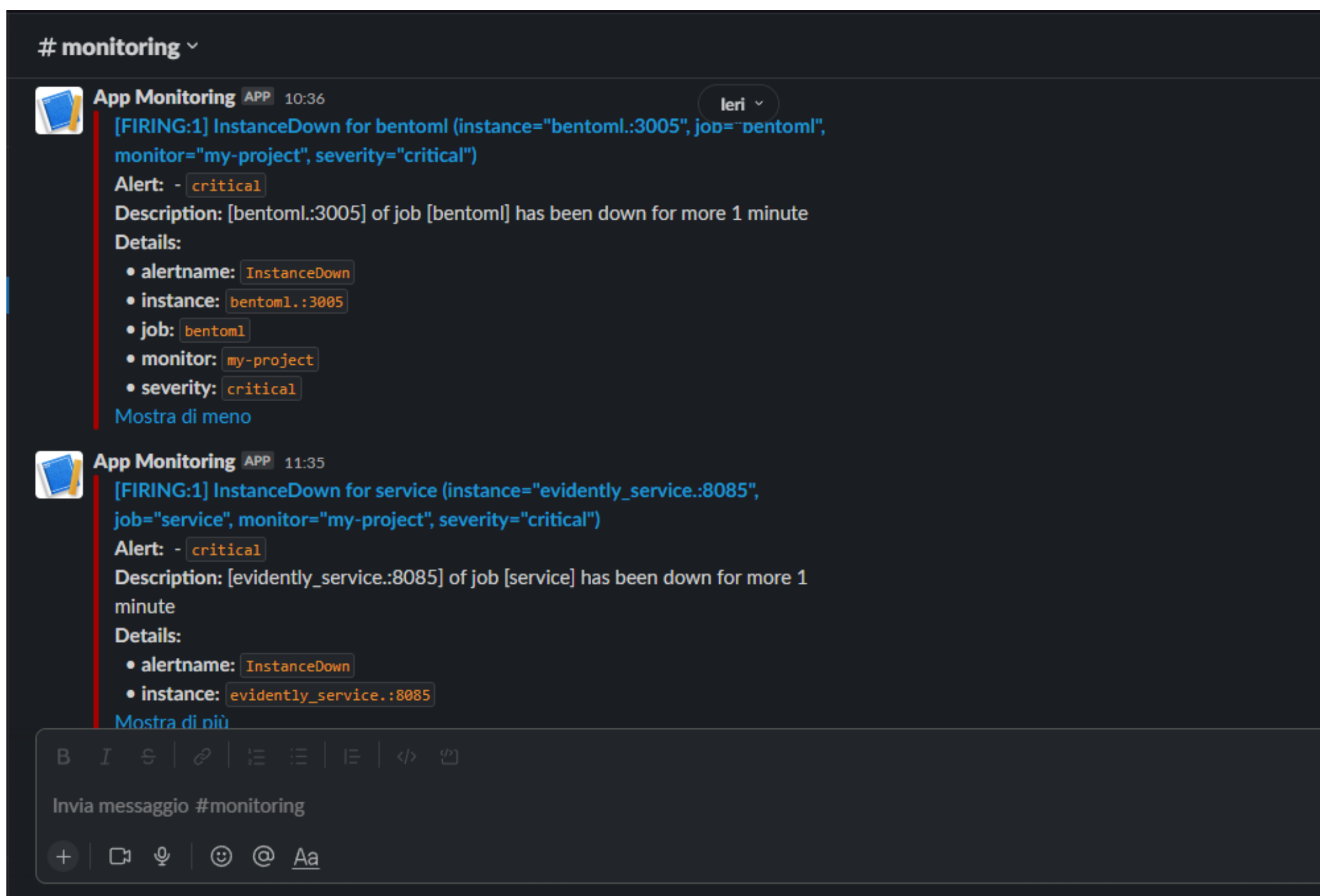
The screenshot shows the Prometheus Alerting interface. At the top, there's a navigation bar with 'Prometheus', 'Alerts', 'Graph', 'Status', and 'Help'. Below this, there's a search bar with the text 'Filter by name or labels'. To the right of the search bar, there's a checkbox for 'Show annotations'. Below the search bar, there's a breadcrumb trail: '/etc/prometheus/prometheus_rules.yml > alert_rules'. To the right of the breadcrumb trail, there are two status indicators: 'inactive' and 'firing (2)'. Below the breadcrumb trail, there's a list of alerting rules. Each rule is represented by a colored bar with a chevron icon and text indicating the rule name and the number of active alerts. The rules are: 'InstanceDown (0 active)' (green), 'DatabaseDrift (1 active)' (pink), 'FeaturesDrift (1 active)' (pink), 'TargetDrift (0 active)' (green), and 'HighCPUseconds (0 active)' (green).

Alerting Rule	Status	Active Alerts
InstanceDown	Inactive	0
DatabaseDrift	Firing	1
FeaturesDrift	Firing	1
TargetDrift	Inactive	0
HighCPUseconds	Inactive	0

Grafana + Prometheus

ALERTING ON SLACK

Show a example of chat where Alertmanager send alert, with details.



Grafana + Prometheus



BENTOML

BentoML offer a integration with Prometheus.

So Prometheus monitor and take metrics about the BentoML service from `/metrics` endpoint, which includes the essential metrics for model serving and the ability to create and customize new metrics base on needs.

This guide will introduce how to use Prometheus and Grafana to monitor your BentoService:

- <https://docs.bentoml.org/en/0.13-lts/guides/monitoring.html>

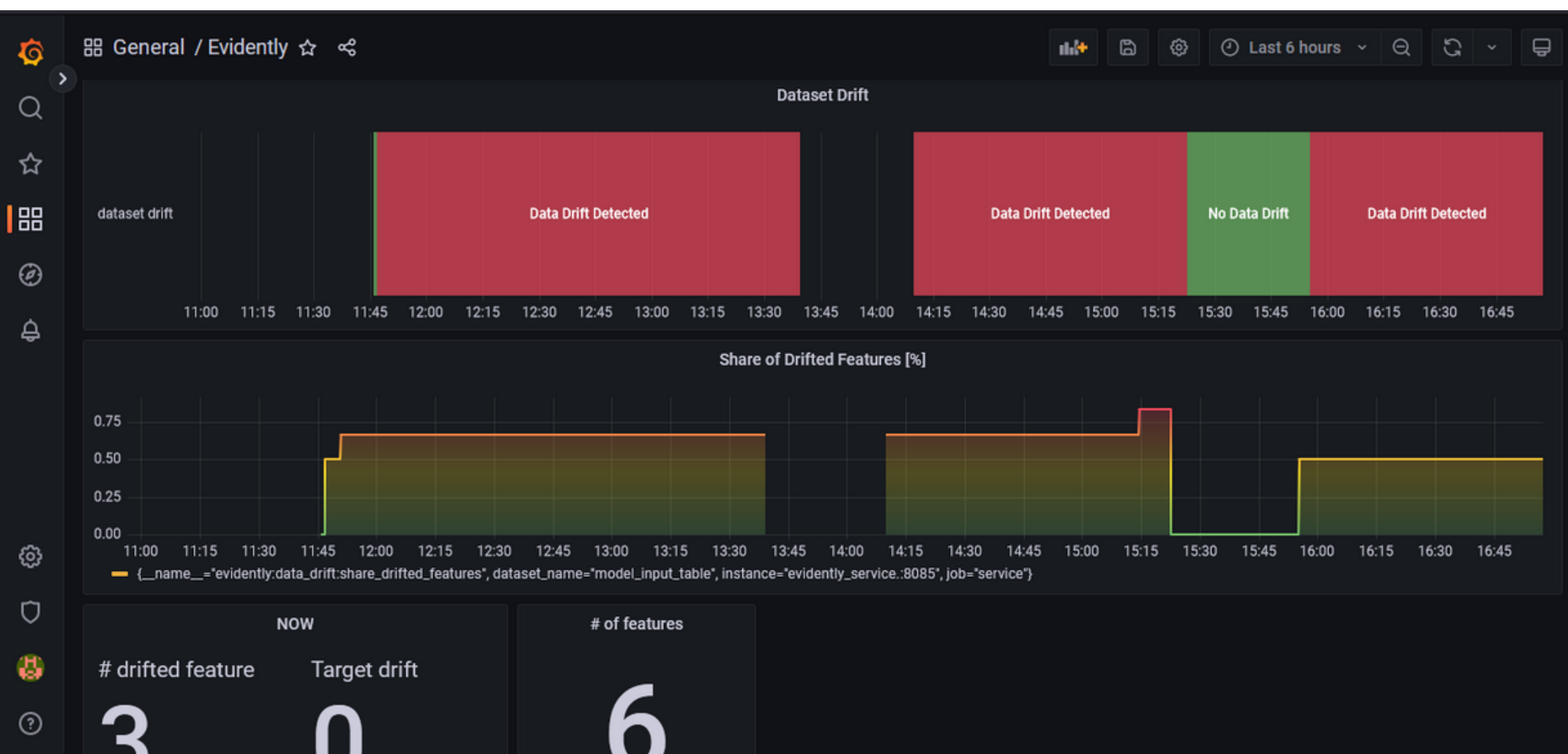
Grafana + Prometheus

GRAFANA

Grafana allows you to visualize monitoring metrics. It can visualize the results of monitoring work in the form of line graphs, heat maps, and histograms.

You use Grafana GUI boards to request metrics from the Prometheus server and render them in the Grafana dashboard.

Grafana dashboards are easy to setting and very flexible. You can also save the dashboards in json form to export.



Evidently

It helps **evaluate, test, and monitor the performance of ML models** from validation to production. The tool generates interactive reports from pandas **DataFrame**.

REPORTS

Currently, 6 reports are available:

1. Data Drift: detects changes in feature distribution
2. Numerical Target Drift: detects changes in the numerical target and feature behavior
3. Categorical Target Drift: detects changes in categorical target and feature behavior
4. Regression Model Performance: analyzes the performance of a regression model and model errors
5. Classification Model Performance: analyzes the performance and errors of a classification model. Works both for binary and multi-class models
6. Probabilistic Classification Model Performance: analyzes the performance of a probabilistic classification model, quality of model calibration, and model errors. Works both for binary and multi-class models

TARGET VARIABLE

The **target variable** is the feature of a dataset that you want to understand more clearly. It is the variable that the user would want to predict using the rest of the dataset.

Evidently

Evidently has three components: Reports, Tests, and Monitors (in development).

TEST

- Tests perform structured data and ML model quality checks. You typically compare two datasets: reference and current.
- Required input: one or two datasets as `pandas.DataFrames` or `csv`.
- How you get the output: as an HTML inside Jupyter notebook or Colab, as an exportable **HTML file**, as a **JSON**, or as a Python **dictionary**.

REPORTS

- Reports calculate various metrics and provide rich interactive visualizations.
- Required input: one or two datasets as `pandas.DataFrames` or `csv`.
- How you get the output: as an **HTML** inside Jupyter notebook or Colab, as an exportable **HTML file**, as **JSON**, or as a **Python dictionary**.

Evidently

MONITORS

- Evidently also has Monitors that collect data and model metrics from a deployed ML service.
- Note: this functionality is in early development and subject to an API change.
- You can use configuration to define the monitoring logic. Evidently calculates the metrics over the streaming data and emits them in **Prometheus format**. There are pre-built Grafana dashboards to visualize them.
- Required input: POST live data from the ML service.
- How you get the output: data and quality metrics in the Prometheus format.

Evidently

You will use Evidently test suites functionality.

- **Test suites** help compare the two datasets in a structured way. A test suite contains several individual tests. Each test compares a specific metric against a defined condition and returns a pass/fail result.
- Let's start by running the **DataStability** preset. It will run several checks for data quality and integrity and help detect issues like feature values out of the expected range.
- The data stability test suite compares two batches of data you expect to be similar. It automatically derives descriptive statistics from the reference dataset and compares them with the current data.

Reports

- Evidently reports help to explore and debug data and model quality. Unlike tests, reports do not require defining explicit pass or fail conditions. Instead, they calculate various metrics and generate a dashboard with rich visuals.
- The **data drift report** compares the distributions of each feature in the two datasets.

Data quality

- It can detect issues like missing data, duplicates, or constant and almost constant features.
- <https://docs.evidentlyai.com/tests/data-quality>

more about Drift

- <https://www.evidentlyai.com/blog/tutorial-3-historical-data-drift>

Evidently

INTEGRATIONS

Evidently have two integrations:

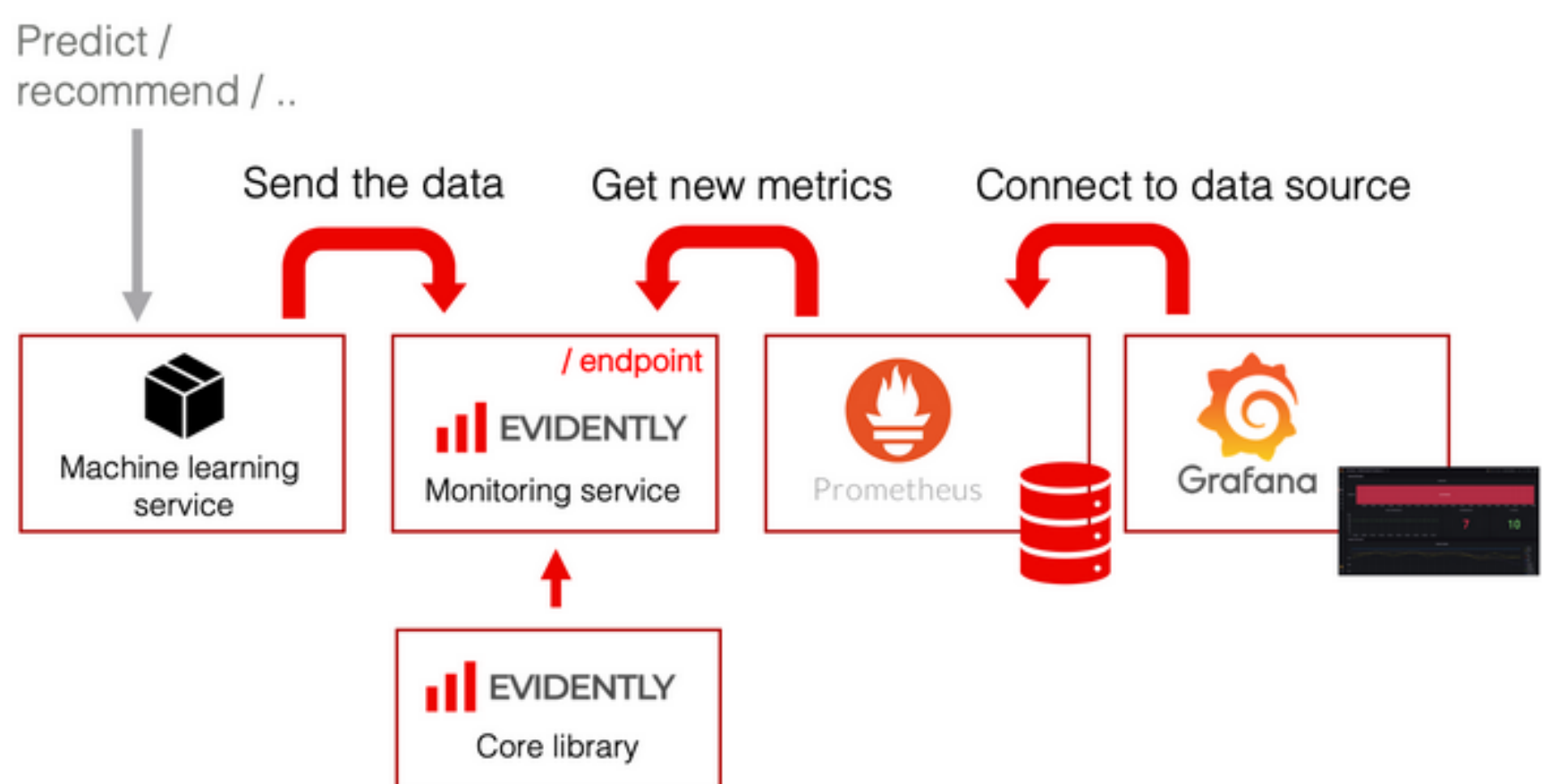
- Evidently with MLflow
- Evidently with Grafana and Prometheus

Evidently with MLflow

- You use Evidently to calculate the metrics and MLflow to log the results. You can then access the metrics in the MLflow interface.

Evidently with Grafana and Prometheus

- Evidently provides a metrics calculation layer, Prometheus is used to store the metrics, and Grafana is used to display the dashboards and manage alerts.
- <https://www.evidentlyai.com/blog/evidently-and-grafana-ml-monitoring-live-dashboards#how-can-i-use-it-for-my-production-service-2>



References and useful url

Best tool

- <https://neptune.ai/blog/ml-model-monitoring-best-tools>

PROMETHEUS AND GRAFANA

Theory

- <https://www.jeremyjordan.me/ml-monitoring/>
- <https://mlopshowto.com/a-journey-into-machine-learning-observability-with-prometheus-and-grafana-part-i-384da4b2d77f>

Tutorial

- https://prometheus.io/docs/tutorials/getting_started/

Complete video tutorial about Prometheus:

- <https://youtu.be/4lAtwCmG6cs>

With Docker Compose

- <https://www.theairtips.com/post/setting-up-alertmanager-with-docker-compose>

Alert prometheus

- <https://grafana.com/blog/2020/02/25/step-by-step-guide-to-setting-up-prometheus-alertmanager-with-slack-pagerduty-and-gmail/>

References and useful url

EVIDENTLY

Theory

- <https://docs.evidentlyai.com/>
- <https://medium.com/@lzhangbq/evidently-evaluation-and-monitoring-tool-for-machine-learning-601c1d23049e>

Tutorial

- <https://www.youtube.com/watch?v=RIc45j1c520>

more about Drift

- <https://www.evidentlyai.com/blog/tutorial-3-historical-data-drift>

Tutorial Evidently + Prometheus + Grafana + Fast API

- <https://github.com/vmallya-123/evidently-real-time>
- <https://medium.com/dkatalis/realtime-data-drift-detection-140ff0d586ea>

Streamlit

2022 • GIORGIA BERTACCHINI

MLOps • 2022

Streamlit



MODEL SERVING

Streamlit is an open-source **Python library** that facilitates building and deploying shareable web apps in minutes. It turns data scripts into customized, powerful, and shareable data apps.

Being a production-ready app framework, Streamlit offers the fastest way to build web apps for machine learning models.

<https://censius.ai/mlops-tools/streamlit>

WIDGET

Adding a widget is the same as declaring a variable. No need to write a backend, define routes, handle HTTP requests, connect a frontend, write HTML, CSS, JavaScript, ...

STREAMLIT CLOUD

Streamlit Cloud workspace helps teams to deploy, manage and collaborate on web apps.

Streamlit account is connected with a public or private GitHub repository to launch apps directly at scale.


API reference

CLOSE TO MACHINE LEARNING

When you're working with data, it is extremely valuable to visualize that data quickly, interactively, and from multiple different angles.

You can display data via charts, and you can display it in raw form.


- Data display elements: **DataFrame**, **Static tables**, **Metrics**, **Dicts** and **JSON**.



	1961	1962	1963	1964	1965	1966
Argentina	11.9504	12.5193	13.6947	12.9182	12.2474	13.4104
Brazil	16.5646	17.3449	17.4430	17.6581	20.3672	19.2388
China	58.3407	60.6909	63.9427	68.4626	74.7479	80.4459
France	23.4085	26.6540	25.2027	25.3211	26.7809	25.4474
Germany	22.0777	24.2415	25.3408	24.8436	23.9885	25.2363
India	50.1427	50.0158	51.3677	52.2677	49.4983	49.3096
Indonesia	7.3837	8.0549	7.6509	8.0770	7.9255	8.2192
Mexico	6.6326	7.0989	7.3942	8.0449	8.6041	9.1411
Spain	9.7984	9.8255	11.4204	10.6459	10.6434	11.6959
USSR	77.7688	77.9614	70.5310	81.7169	80.9420	90.7813

Dataframes
Display a dataframe as an interactive table.

```
st.dataframe(my_data_frame)
```

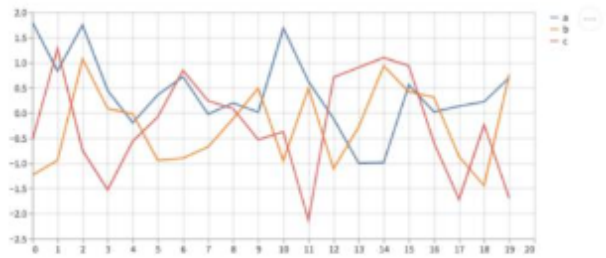


	A	B	C	D	E
0	1.000000	1.329212	nan	-0.316280	-0.990810
1	2.000000	-1.070816	-1.438713	0.564417	0.295722
2	3.000000	-1.626404	0.219565	0.678805	1.889273
3	4.000000	0.961538	0.104011	-0.481165	0.850229
4	5.000000	1.453425	1.857737	0.165562	0.515018

Static tables
Display a static table.

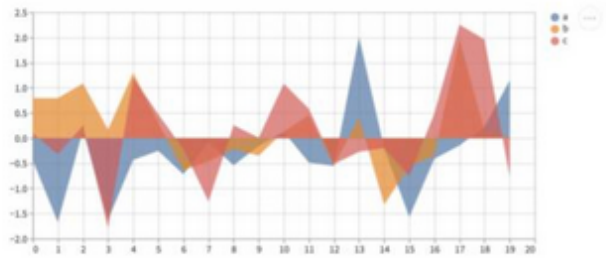
```
st.table(my_data_frame)
```

- Chart elements: Streamlit supports several different charting libraries as Matplotlib. And a few chart types that are "native" to Streamlit, like `st.line_chart` and `st.area_chart`



Simple line charts
Display a line chart.

```
st.line_chart(my_data_frame)
```



Simple area charts
Display an area chart.

```
st.area_chart(my_data_frame)
```

References and useful url

Doc

- <https://streamlit.io/>

Example: Streamlit + BentoML + DagsHub

- <https://modelserving.com/blog/the-easiest-way-to-deploy-your-machine-learning-models-in-2022-streamlit-bentoml-dagshub>

Download_button for download pickle file

- <https://discuss.streamlit.io/t/download-pickle-file-of-trained-model-using-st-download-button/27395/4>

Examples:

- <https://www.youtube.com/watch?v=snRsPcwTwSI>
- <https://www.youtube.com/watch?v=RVMlibDbzaE>