Monitoring of services in distributed Cloud-ready environments

Filippo Berto – Università degli Studi di Milano

28/05/2024

Introduction

- Introduction
- Means of monitoring
- Monitoring flow
- Active vs Passive monitoring
- Push vs Pull approach
- Current standards
- Example deployment
- Issues with scalability
- Extensions and projects

Introduction

Why monitoring?

- System health and forward comparison
- Implementation errors
- User misbehavior
- Resource usage
- Alerts and prevention
- Incident analysis

Means of monitoring: Metrics

- Timeseries-based measurements of system aspects
  - **Numeric**
    e.g. CPU and RAM usage, storage available
  - **Categorical**
    e.g. system on/off, pipeline execution stages

- Metrics provide **objective** and **structured** measurements of a system behavior

Means of monitoring: Logs

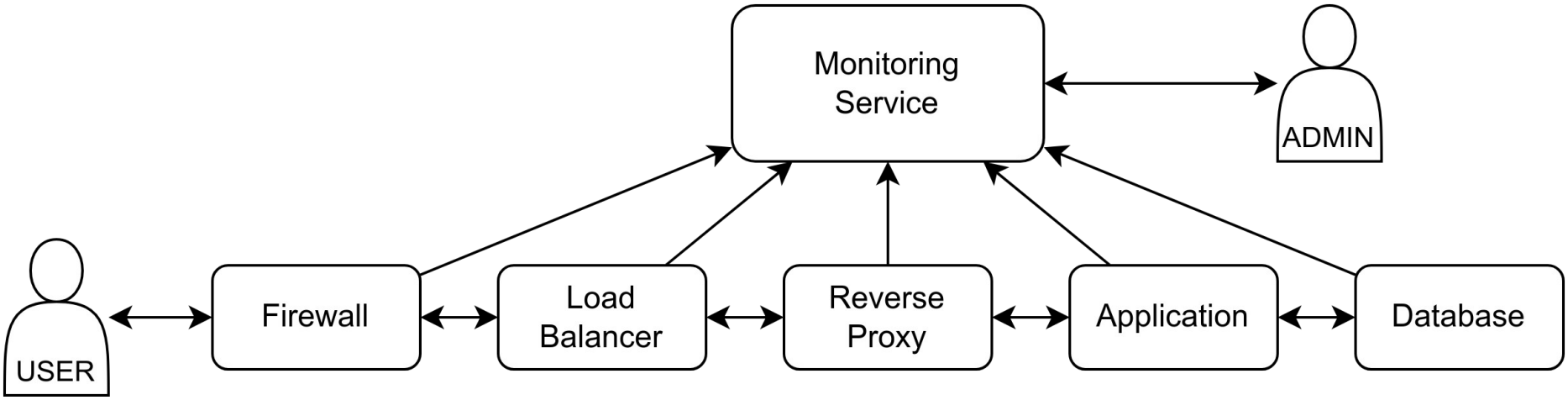
- Text records of application **interaction with the system**
  - e.g. warning and error messages, access logs
- Organized in **level of detail**:
  - e.g. error, warning, info, debug, trace
- Structured logs allow metadata and ease of integration
- Many (textual and binary) formats:
  JSON/YAML, GELF, CEF, Windows event log, NCSA, ELF, …
  - Need for extended support or adapters

Means of monitoring: Traces

- Traces record interactions **between system components**
  - Where the event is originated
  - Context of execution (methods, variables, time)

- Contexts **can be nested** and are bound to detail levels
- Especially useful when debugging large event-based systems:
  - you can **follow the event** while it flows in the infrastructure

Monitoring Flow



Agent based vs Agentless monitoring

- **Agent based monitoring**
  An agent installed in the host collects measurements on the application behavior
  - App-specific
  - Performance overhead
  - May need high privileges

Agent based vs Agentless monitoring

- **Agentless monitoring**
  The application itself implements the monitoring service and provides its own measurements
  - Code-driven: part of the application and versioned
  - We are trusting the application-provided data

Push vs Pull approach

- **Pull approach**
  The monitoring service collects the measurements from the metrics service
  - Update frequency defined by the monitoring service
    - Resource usage is easier to predict
  - Requires an open port to the target
    - The network topology can become quite large
    - Many devices with potentially confidential data that needs to be accessed securely

Push vs Pull approach

- **Push approach**
  The application sends updated measurements to the monitoring service
  - We only require one open endpoint on the monitoring service
    - Easier to manage
  - The application handles the data transmission
    - High spikes in traffic can saturate the monitoring service if scaling is slow
    - Good targets for DOS attacks

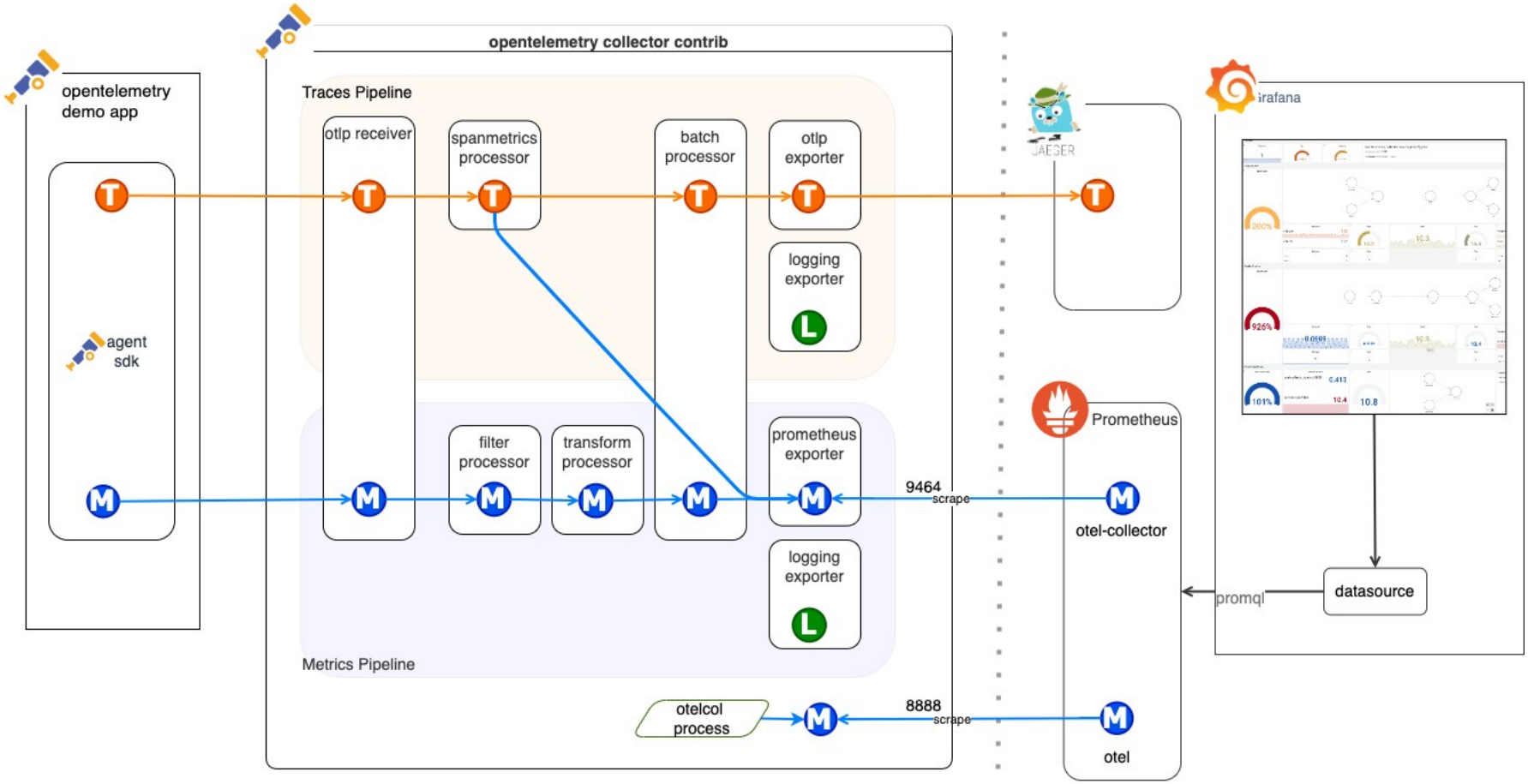
Current standards: Prometheus

- De facto standard in cloud applications metrics monitoring
- Several data types (gauges, counters, histograms, summaries)
- Pull approach:
  - Exporters expose metrics as a web service
  - Scraping configurations for each service
  - Data stored in timeseries
- Simple to implement (single web page)
- Many agents for specific applications compatability

Current standards: OpenTelemetry

- Focus on maximum extensibility, usability and performance
- Supports metrics, logs and traces
- Mixed approach:
  - Receivers collect measurements from applications and handle their transmission to the monitoring service
  - Intermediate processors can transform the data
    e.g. filter, anonymize, convert
  - The data is exported in several formats
    e.g. Prometheus, Jaeger
- Can convert from one format to another
- Libraries available for many programming languages

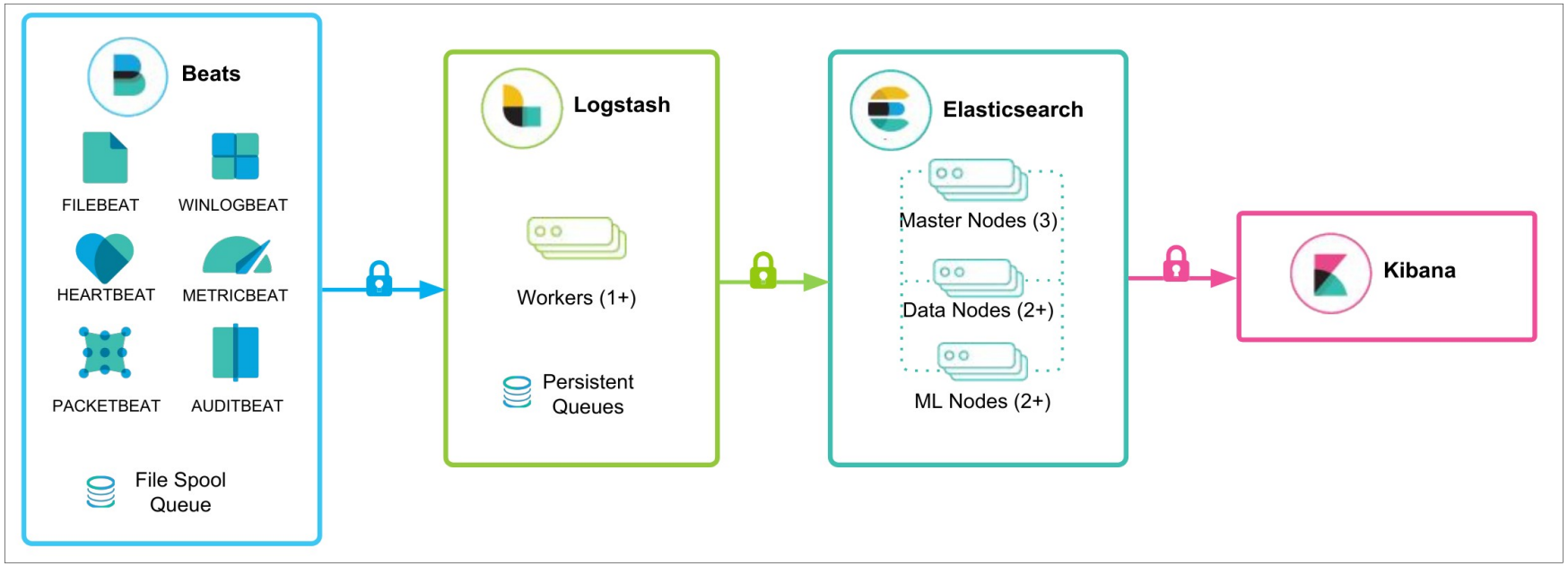
Current standards: OpenTelemetry



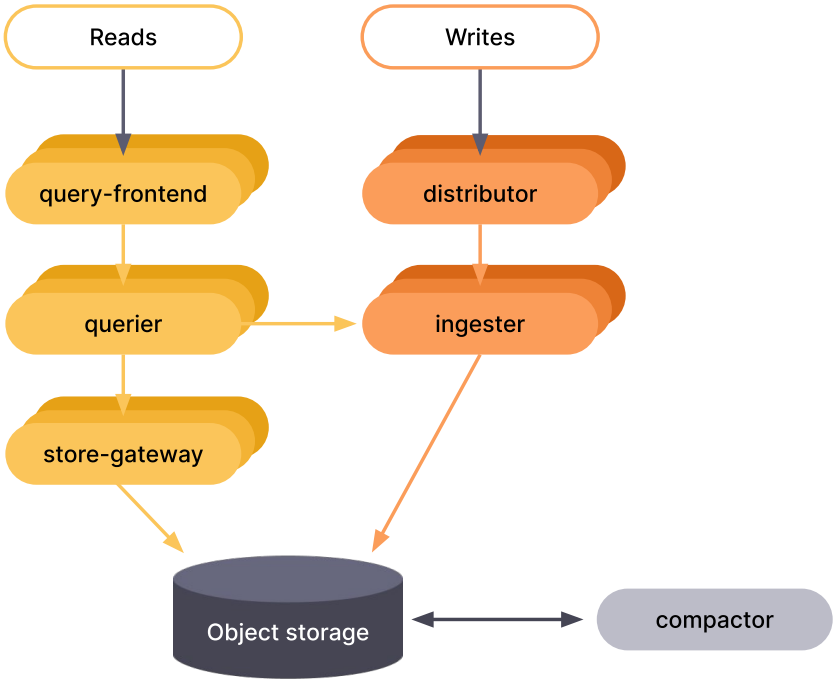
Current standards: Elastic

- Commercial solution with advanced features
- Supports metrics, logs and traces
- Mostly agent-based, but supports scraping
- Major components:
  - Beats: monitoring agents and exporters
  - Logstash: collection, parsing and processing of logs
  - Elasticsearch: logs database and query engine
  - Kibana: data visualization
- ML-based prediction solutions, strong integration with many enterprise applications and services on large scales

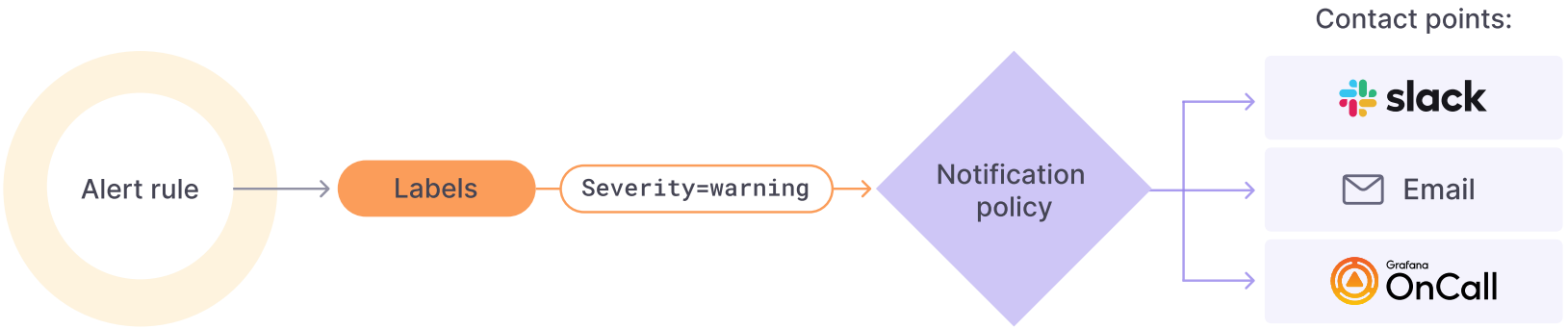
Current standards: Elasticsearch



Issues with scalability: Mimir's solution



Etensions: Alerts management



Etensions: ML-based predictions

Grafana Machine Learning

Grafana Machine Learning offers an expanding range of data analysis and generative AI capabilities, including creating alerts, forecasting capacity requirements, and identifying anomalous activities.

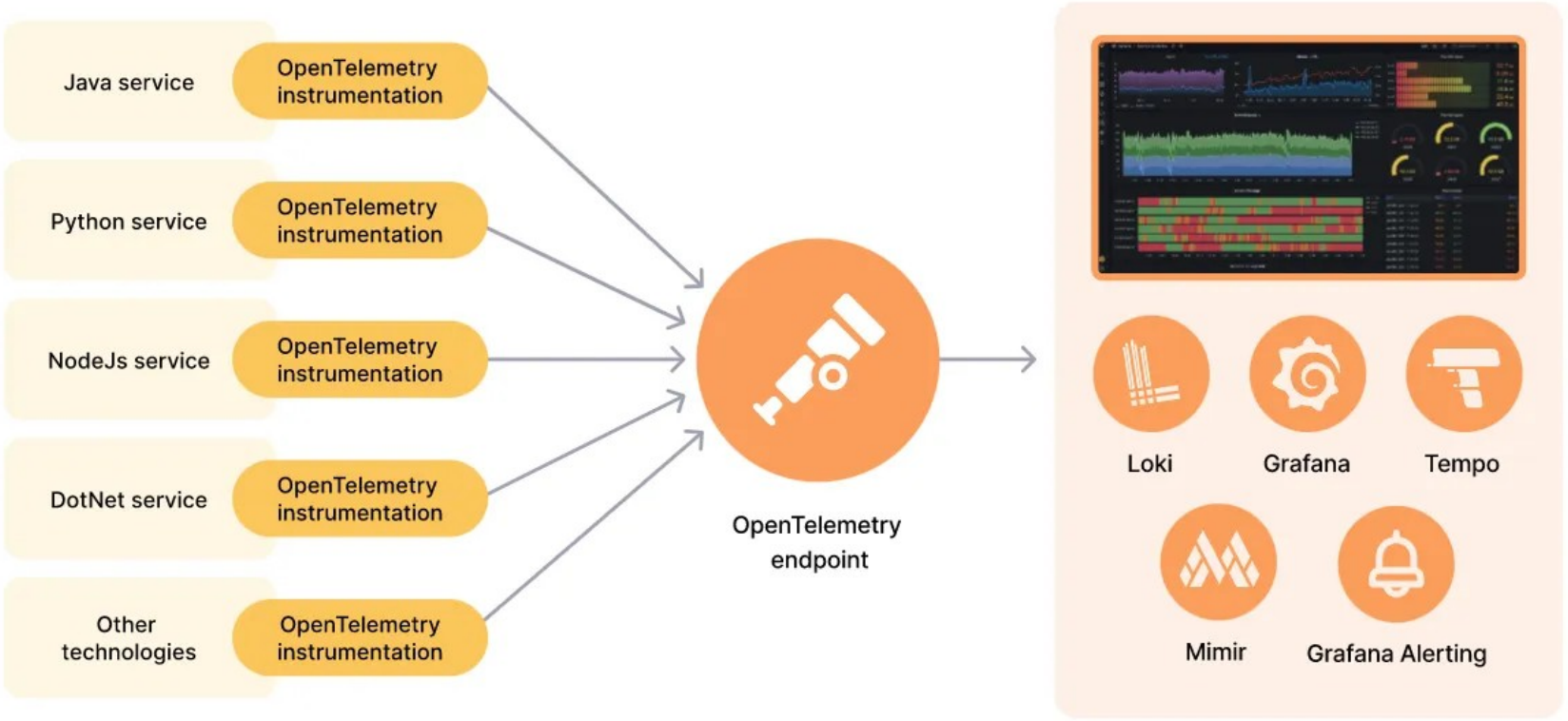
Overview

Grafana Machine Learning is a powerful tool with an expanding range of data analysis and generative AI capabilities aimed to facilitate and inform proactive decision-making and incident response.

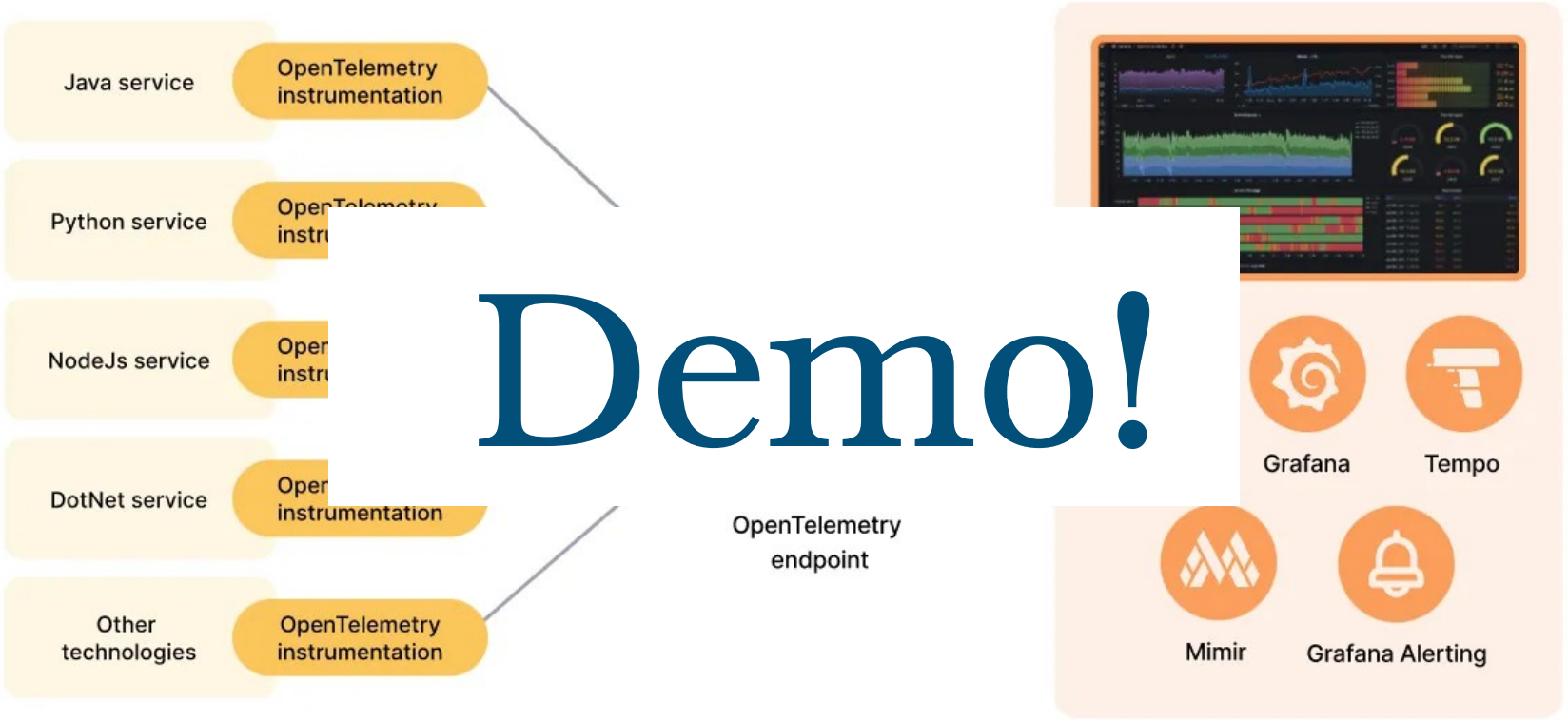
You can leverage ML features in Grafana Cloud to learn patterns in your data and get predictive insights for your time series. These forecasts can inform the creation of alerts, forecast capacity requirements, and identify anomalous activities.

Explore how Grafana ML can help you learn patterns in your data, investigate your infrastructure telemetry, and gain predictive insights. To get started, refer to Set up Grafana Machine Learning.

Example deployment



Example deployment



Demo!

Measurements as behavioral evidence

The collected metrics can be used as **evidence** in the **verification** of **non-functional properties** (e.g. QoS) and the **certification** of a system.

In **SESAR Lab** we are focusing on transparent monitoring solutions for **cloud-edge** environments for continuous QoS verification.

Any questions?