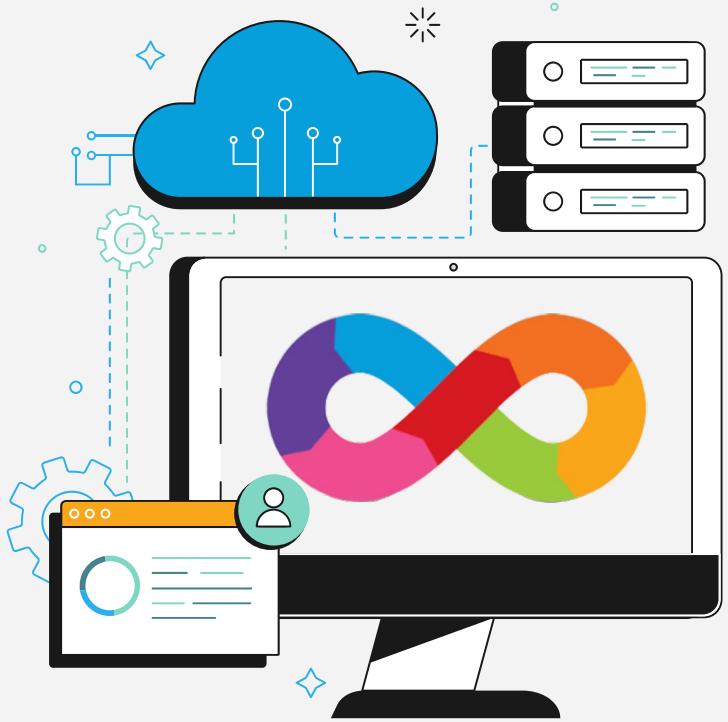
Introduction to DevOps

@ IBA - SMCS

Week 08

Observability & Monitoring



Obaid ur Rehman
Software Architect / Engineering Manager @ Folio3

Observability & Monitoring - Agenda

- Observability & Monitoring
- Difference b/w O11y & Monitoring
- Golden Signals of Monitoring
- The 3 Pillars of Observability
- What is APM?
- Tooling for Monitoring and Observability
- Prometheus & Grafana
- Architecture of Prometheus
- ELK Stack
- Hands-on: Prometheus & Grafana in K8s.

What is Observability & Monitoring?

Monitoring and observability are two ways to identify the underlying cause of problems.

Monitoring tells you when something is wrong, while observability can tell you what’s happening, why it’s happening and how to fix it.

Observability & Monitoring

Observability and monitoring are related concepts that are often used in the context of system administration, DevOps, and software engineering.

While they are related, Observability vs Monitoring are two distinct terms. In order to understand the importance of each and how they can be used in tandem, let’s take a look at what those terms mean.

Monitoring

Textbook Definition: Monitoring is the process of collecting **data** and generating reports on different metrics that define system health.

Example: You set up a Disk usage or CPU usage monitoring on DB that alerts you whenever it goes beyond 90%. It might indicate a problem.

🟢 Remember Monitoring is a critical core component of observability.

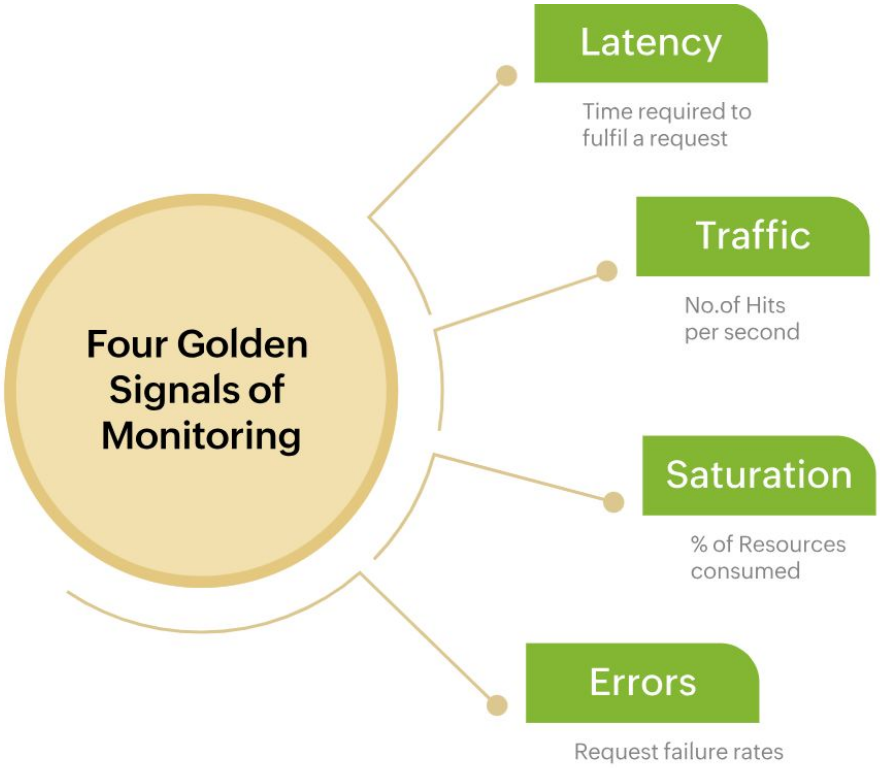
Golden Signals

Obviously you cannot monitor every thing...

Golden Signals were first introduced by Google in the context of Site Reliability Engineering (SRE) practices. The concept was originally presented in a talk by Google software engineers, Dave Rensin and Kevin Smathers, at the 2016 O'Reilly Velocity Conference.

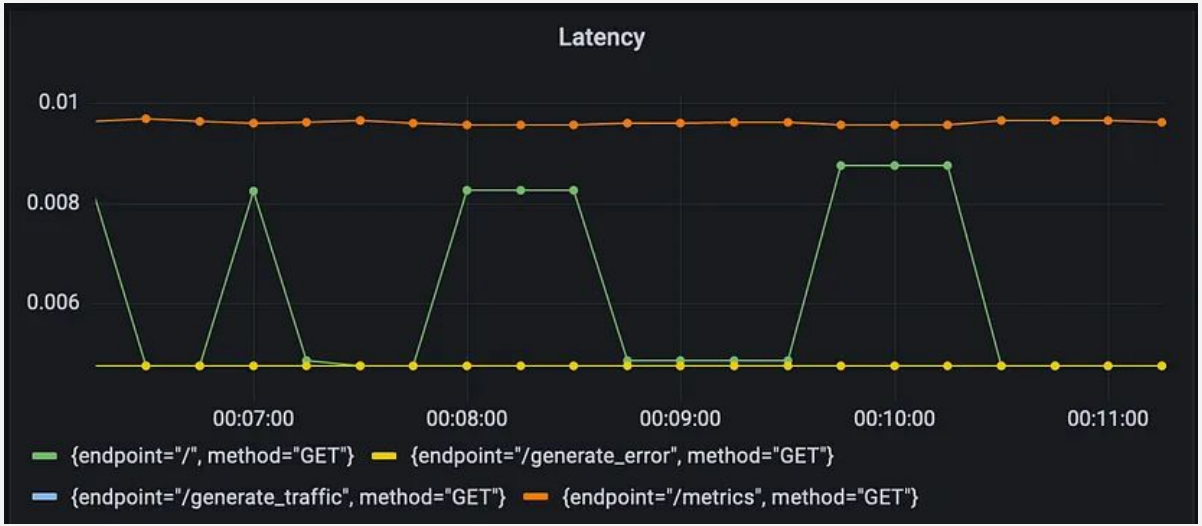
The idea behind Golden Signals was to provide a set of key performance indicators (KPIs) that could be used to measure and monitor the health of complex, distributed systems.

https://sre.google/sre-book/monitoring-distributed-systems/


Four Golden Signals of Monitoring
Latency
Time required to fulfil a request
Traffic
No.of Hits per second
Saturation
% of Resources consumed
Errors
Request failure rates

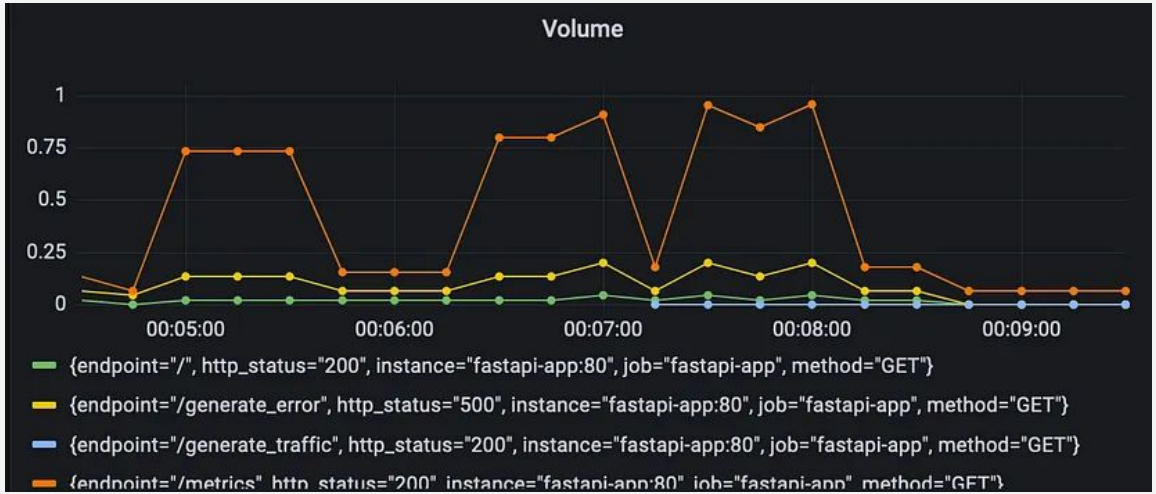
Latency

Latency measures the time it takes for a system to respond to a request. High latency can indicate that the system is overloaded or experiencing other performance issues.



Traffic

Traffic measures the amount of data or requests that are flowing through the system. High traffic can indicate that the system is experiencing high demand or that there are issues with the system's capacity.



Errors

Errors measure the number of errors that occur within the system. High error rates can indicate that there are bugs or other issues within the system.

Saturation

Saturation measures the resource utilization of the system. High saturation can indicate that the system is running out of resources, such as CPU or memory.

Observability

- Observability is the ability to understand a system's internal state by analyzing the data it generates, such as logs, metrics, and traces.

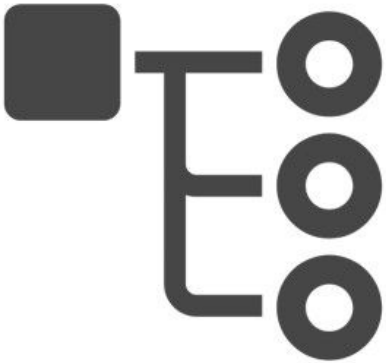
- Observability helps teams analyze what's happening in **context** across environments so you can detect and resolve the underlying causes of issues.

Observability & Monitoring

	Monitoring	**Observability**
Focus	Collect data to identify anomalous system events.	Investigate the root cause of anomalous system effects.
System error findings	The when and what.	The why and how.
Systems involved	Typically concerned with standalone systems.	

Three Pillars of Observability

Metrics



Traces

Logs

Logs

Logs are text records of certain events that took place over a specific period of time.

Structured logging is a must-have for complex ecosystems with multiple components, like Kubernetes services as it enables developers to spot unpredict behavior in a system.

Metrics

Metrics is another key pillar of application observability that defines a numerical representation of data that you can compile over a timeframe. There can be various sources of metrics, including the infrastructure, hosts, and cloud platforms.

Example: metrics provide information on the number of requests processed by a service every second or how much memory a particular pod consumes.

Tracing

Tracing represents activities of a request or transaction going on in a software product. You can identify bugs in the system and determine their root causes by capturing the traces of requests and figuring out what is happening across the request chain.

APM

A subset of observability, Application performance monitoring (APM) includes the process, tools, and practices involved in monitoring and managing how applications perform and behave during execution.

Integrating APM into a system helps identify bottlenecks, pinpoint issues impacting user experience, optimize the system's performance, and make the system efficient.

Tooling

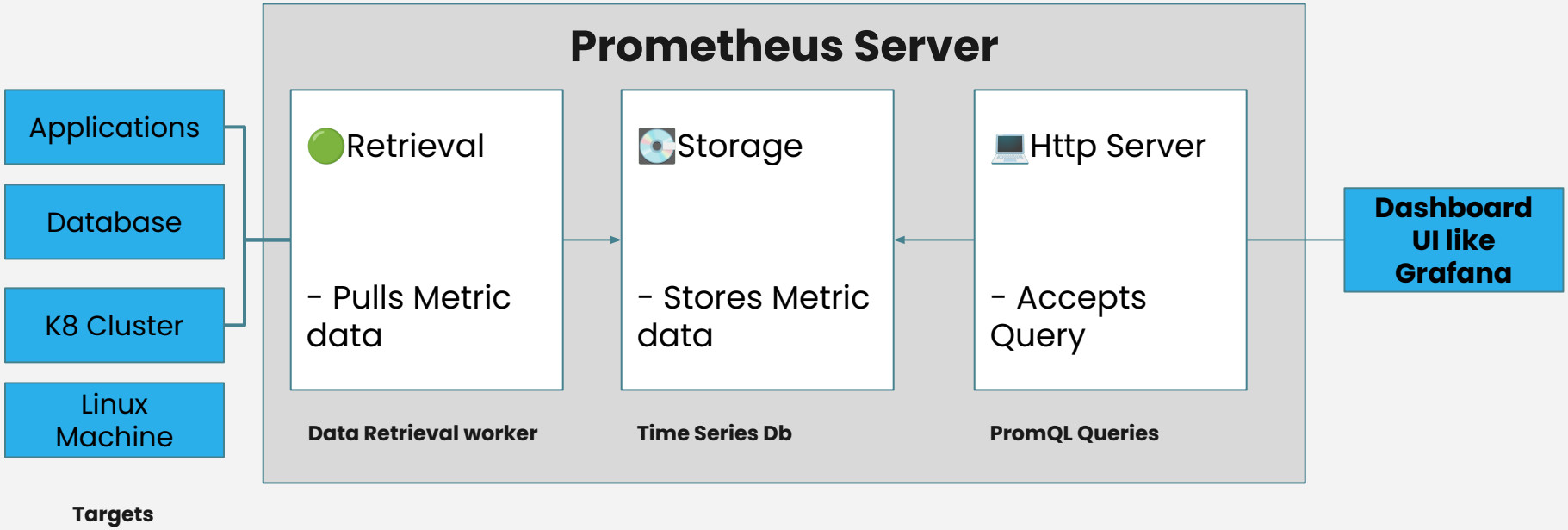
Prometheus is an open-source monitoring system that collects and stores time-series data from various sources such as application metrics, system metrics, and logs. It provides a powerful query language called PromQL to retrieve and analyze this data and has built-in alerting capabilities to notify users of issues in their systems.

Grafana is an open-source platform for data visualization and analysis that can be integrated with Prometheus to create interactive dashboards and alerts. It allows users to create customizable graphs, tables, and other visualizations from data collected by Prometheus.

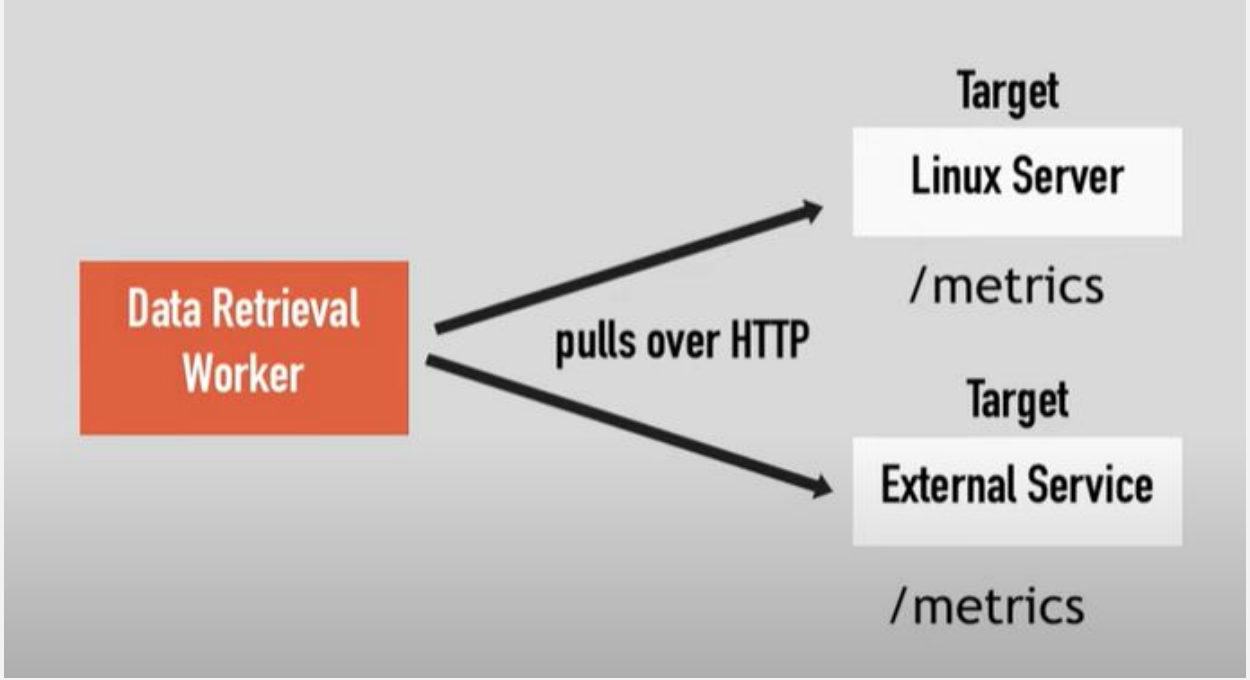
Prometheus Use Case

- A series of events that leads to a failure
- A pre-emptive alerting system to warn about potential issues and identify them before they occur.

Prometheus Architecture

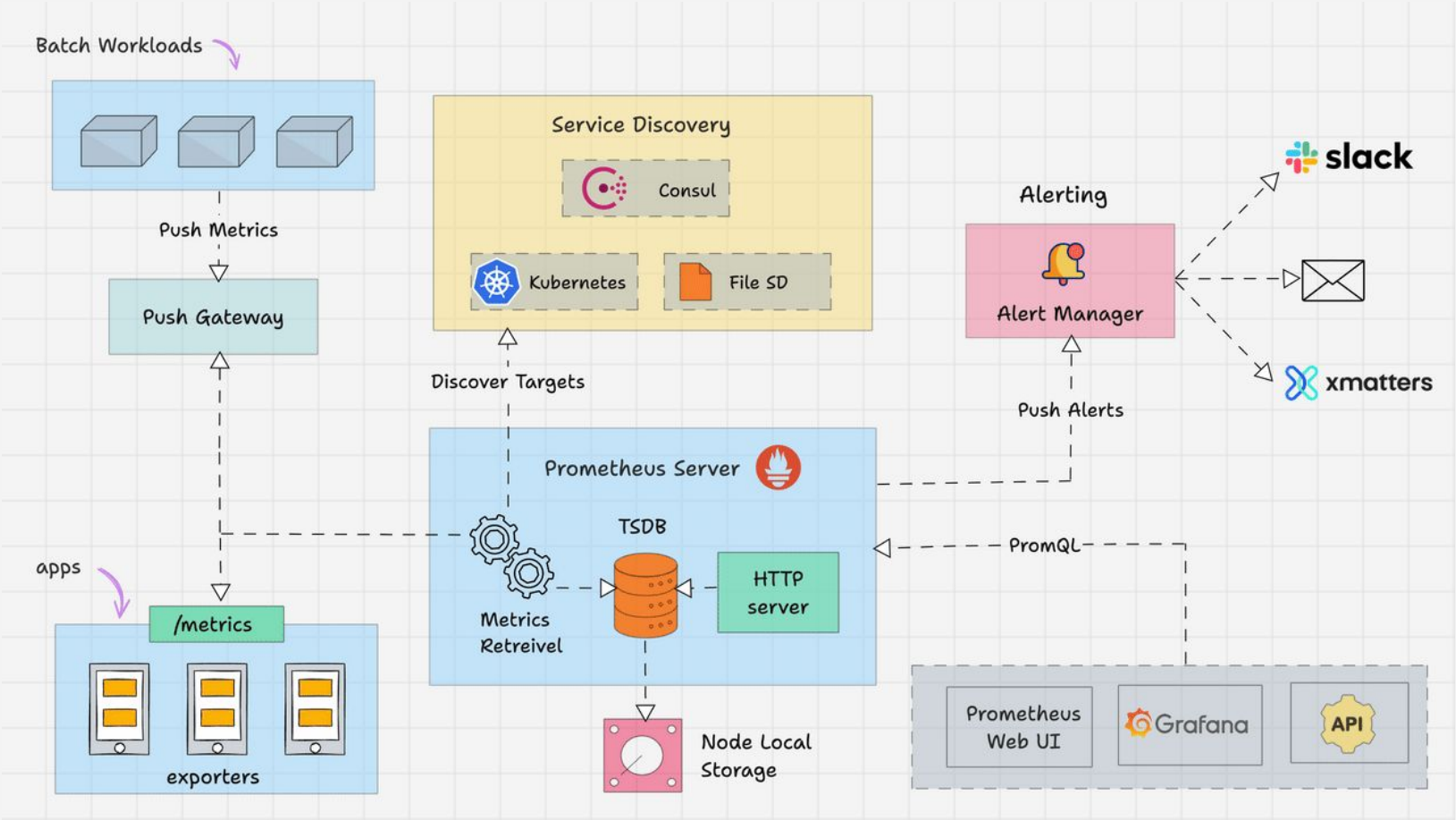


How does Prometheus Pulls data?



Prometheus client Libraries

https://prometheus.io/docs/instrumenting/clientlibs/



Source: https://devopscube.com/prometheus-architecture/

ELK Stack

The ELK stack is an acronym used to describe a stack that comprises three popular projects:

- Elasticsearch
- Logstash
- Kibana.

The ELK stack gives you the ability to aggregate logs from all your systems and applications, analyze these logs, and create visualizations for application and infrastructure monitoring, faster troubleshooting, security analytics, and more.

ELK Stack

E = Elasticsearch
Elasticsearch is a distributed search and analytics engine built on Apache Lucene. Support for various languages, high performance, and schema-free JSON documents makes Elasticsearch an ideal choice for various log analytics and search use cases.

L = Logstash
Logstash is an open-source data ingestion tool that allows you to collect data from various sources, transform it, and send it to your desired destination. With prebuilt filters and support for over 200 plugins, Logstash allows users to easily ingest data regardless of the data source or type.

ELK Stack

K = Kibana
Kibana is a data visualization and exploration tool used for log and time-series analytics, application monitoring, and operational intelligence use cases.

It offers powerful and easy-to-use features such as histograms, line graphs, pie charts, heat maps, and built-in geospatial support. Also, it provides tight integration with Elasticsearch, a popular analytics and search engine, which makes Kibana the default choice for visualizing data stored in Elasticsearch.

ELK Demo: https://www.elastic.co/demos

Hands on 👨‍💻

- Install Prometheus and Grafana in minikube and create dashboards.
- Part-2: Deploy application into k8s and send Logs to Prometheus and create alerts based on some metric using Loki.

References

- Obs V/s Monitoring: https://www.youtube.com/watch?v=vY61h6cSkVA

End

Q&A