Dimitar Zahariev

Technical Trainer

Monitoring Kubernetes Clusters

Agenda

- Monitoring Kubernetes Clusters
  - Objectives and Challenges
  - Possible Approaches and Solutions
  - Prometheus, Grafana, and Something More
- Demo
- Q&A Session

Monitoring Kubernetes Clusters

Let’s explore the possibilities and challenges

Monitoring vs Observability

- **Monitoring** is the process of collecting, analyzing, and using data to track the performance and health of systems
- Focus Areas
  - **Metrics** - Numerical data representing system performance
  - **Alerts** - Notifications triggered by predefined conditions
  - **Dashboards** - Visual representations of metrics for quick insights

- **Observability** is the ability to understand the internal state of a system by examining its outputs
- Focus Areas
  - **Logs** - Detailed records of events within the system
  - **Metrics** - Quantitative data points for performance analysis
  - **Traces** - Records of the execution path and timings across services
  - **Correlation** - Linking logs, metrics, and traces to form a comprehensive view

Objectives

- **Comprehensive Monitoring**

  Achieve end-to-end monitoring of multiple Kubernetes clusters
- **Centralized Metrics**

  Aggregate and visualize metrics from all clusters in a single, centralized location
- **Scalability**

  Ensure the monitoring solution can scale with the number of clusters and workloads
- **High Availability**

  Maintain uptime and reliability of the monitoring system
- **Ease of Use**

  Provide intuitive dashboards and alerts for developers and operators

Challenges

- **Scalability Issues**

  Single Prometheus instance limitations in a multi-cluster environment

- **Data Storage**

  Efficiently storing large volumes of metrics data over time

- **High Availability**

  Ensuring the monitoring solution remains available despite failures

- **Multi-tenancy**

  Supporting multiple teams or projects with isolated metrics and dashboards

- **Data Aggregation**

  Combining metrics from multiple clusters without loss or duplication

Possible Approaches and Solutions

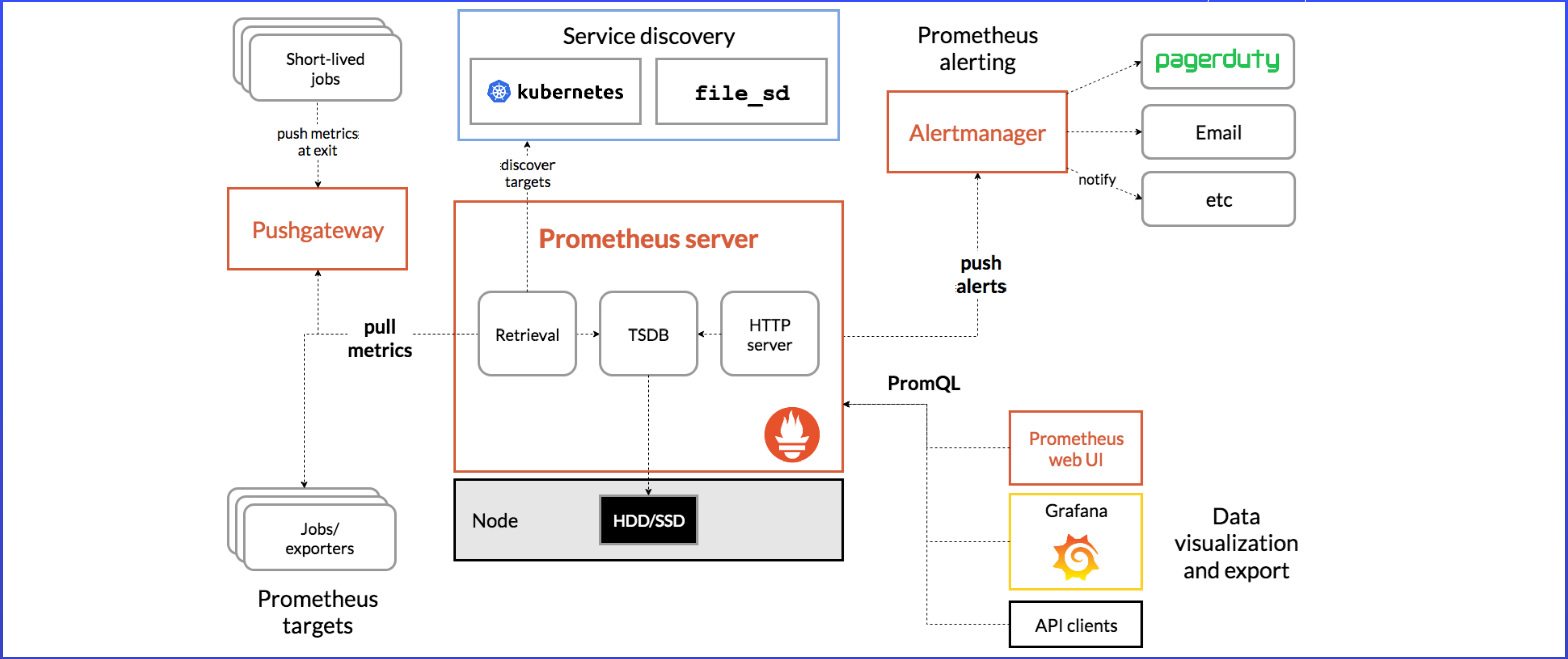
- **Single Prometheus per Cluster**
  - Description - Deploy a Prometheus instance in each Kubernetes cluster
  - Pros - Simplicity, isolated failure domains
  - Cons - Difficult to centralize data, scalability issues, higher maintenance overhead

- **Prometheus + [Thanos | Cortex]**
  - Description - Use Thanos or Cortex to aggregate/centralize metrics from multiple Prometheus instances
  - Pros - Long-term storage, global querying, horizontal scalability, high availability, and multi-tenancy
  - Cons - Increased complexity, dependency on external components, requires careful planning and configuration

Prometheus

- An open-source systems monitoring and alerting toolkit
- Main Components
  - **Prometheus Server** - scrapes and stores metrics
  - **Alertmanager** - handles alerts
  - **Exporters** - collect metrics from various sources
- Key Features
  - **Scraping** - collects metrics from endpoints at specified intervals
  - **PromQL** - powerful query language for metric analysis
- Limitations
  - Limited scalability in a multi-cluster environment
  - Challenges in long-term storage and high availability

Prometheus Architecture



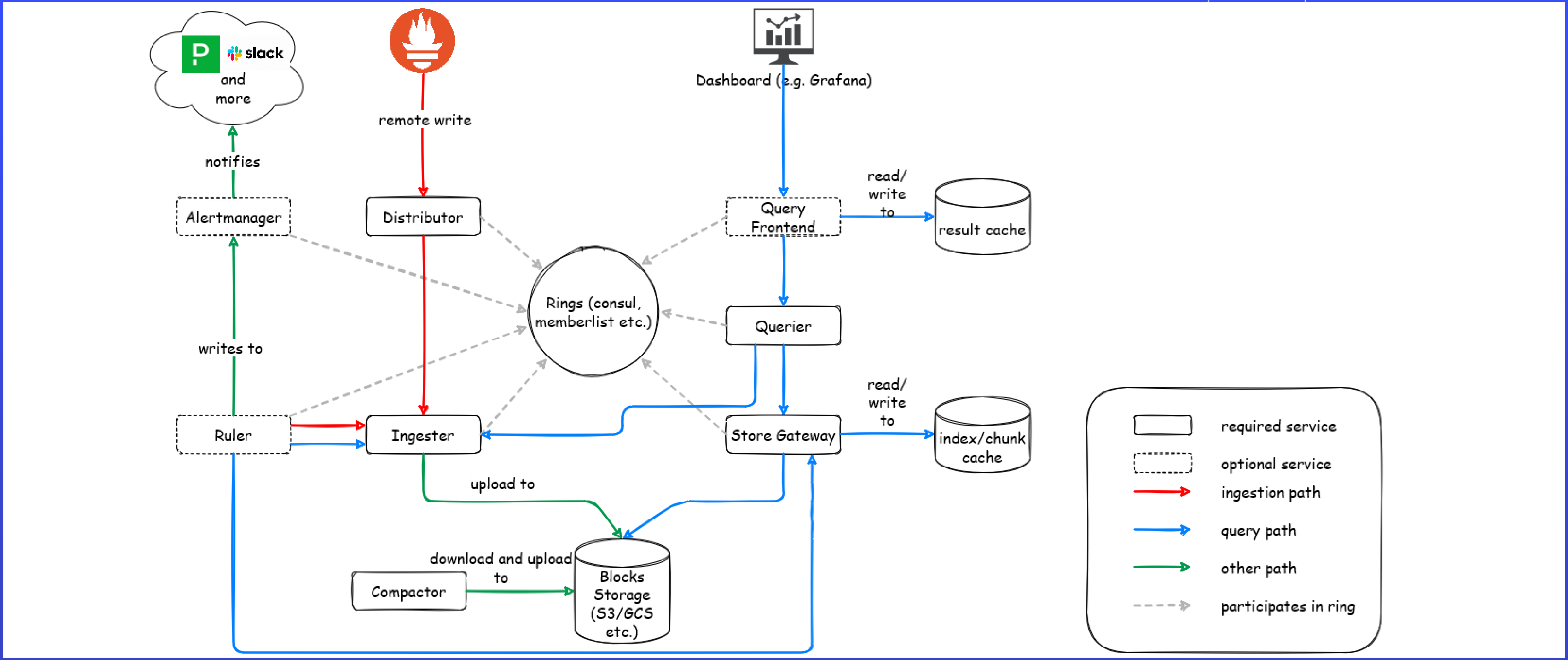
Grafana

- An open-source platform for visualization, monitoring, and observability
- Main Components
  - Dashboards - create and share visual representations of metrics
  - Data Sources - integrate with Prometheus, Cortex, and other data sources
  - Alerting - manage and visualize alerts
- Key Features
  - Flexible and customizable dashboards
  - Wide range of plugins and integrations
- Limitations
  - Dependent on the underlying data source for performance and availability

Cortex

- Horizontally scalable, highly available, multi-tenant, long term storage for Prometheus
- Main Components
  - **Distributor** - Receive, validate, and route samples to ingesters
  - **Ingester** - Temporarily store incoming series before writing them to a long-term backend storage
  - **Querier** - Fetch series samples both from the ingesters and long-term storage
  - **Store Gateway** - Responsible for querying series from blocks stored in the long-term storage
- Key Features
  - Horizontal Scalability - Easily scale out by adding more nodes
  - Multi-tenancy - Isolate data for different teams or projects
  - High Availability - Redundant components ensure reliability
  - Long-term Storage - Supports S3, GCS, Swift, and Microsoft Azure for storing metric data

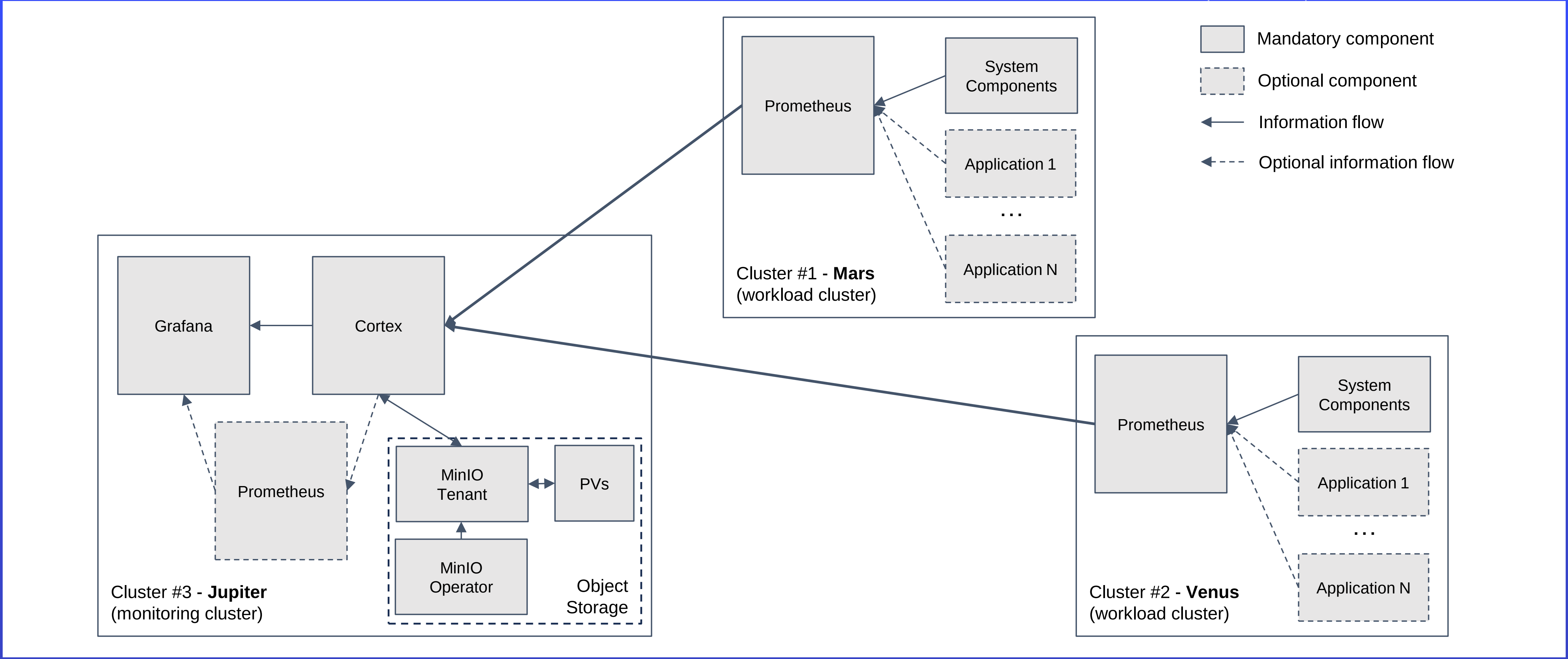
Cortex Architecture



Demo

Let’s see it in action

Our Scenario



Q&A Session

You ask, I answer (if I can)

Thank you!

Contacts:

https://www.linkedin.com/in/dzahariev/

https://github.com/shekeriev

СЛЕДВАЩО СЪБИТИЕ

Check regularly on **dev.bg** or subscribe to the monthly bulletin