Observability in a Quarkus Application

{RIVIERADEV} 2023

Bruno Baptista

Haoyu Sun

@brunobat_

@vachefoule

Presenter

Bruno Baptista,
@brunobat_

Principal Engineer at Red Hat
https://www.redhat.com working on the
https://quarkus.io/ team

Eclipse Foundation Committer

Coimbra JUG
https://www.meetup.com/Coimbra-JUG/

JNation conference
https://jnation.pt/

Haoyu Sun,
@vachefoule

Senior Engineer at Red Hat
Openshift Monitoring Team

Red Hat

Agenda

- Observability Concepts
- Metrics
- Tracing
- Logging
- Setup Quarkus and observability on Openshift
- Conclusion

Observability

- Aims to provide an holistic view of a system's behavior.
- Directed to complex environments.
- Works best when there's collaboration between developers and devOps.

System relying on many services require automation at all levels...

... You cannot control what you cannot “see”.

Observability

Measure of how well internal states of a system can be inferred from knowledge of its external outputs. The observability and controllability of a linear system are mathematical duals.

Observability

...for a software system is its “capability to allow a human to ask and answer questions”. According to Bryan Cantrill.

Traditional Monitoring vs. Observability

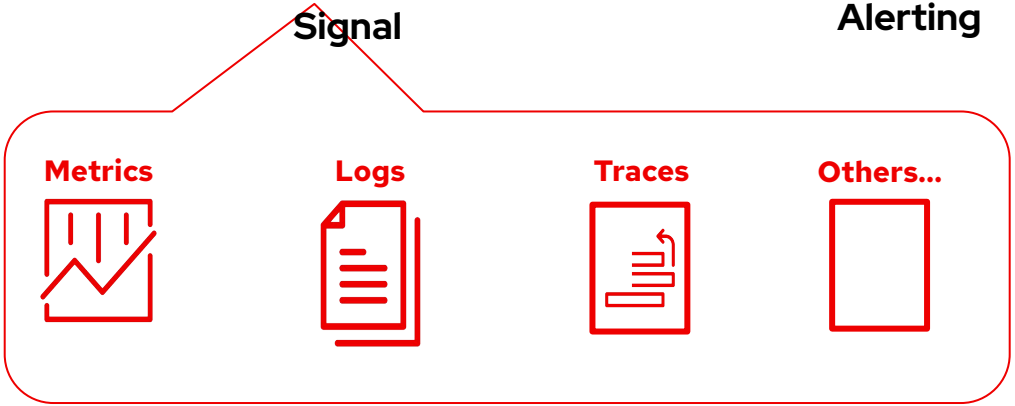
	Application Monitoring	**Observability**
Focus	Predefined metrics to maximize health and availability	Complete understanding of the system's internal state.
Analysis and Troubleshooting	Reactive.	Proactive.
Complexity Handling	Limited visibility into interconnected components	Handles complexity and distributed systems. Data can be correlated with traces

Observability Concepts

Signal

Alerting

Visualisation

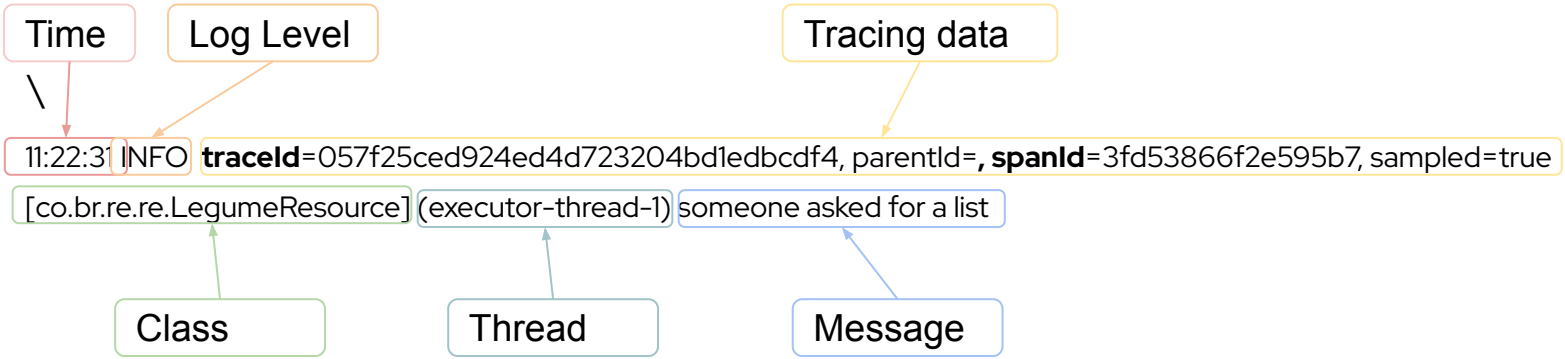


Logging

A log entry is an event.

It contains an occurrence that happened in the system.

Quarkus devmode plain text example:



Logging

- Centralized logging. ELK, OpenTelemetry
- Has context in every log event
- JSON format in prod. for smart search
- Proper log levels that you want to see:
  - ERROR: Execution is aborted;
  - WARN: Something wrong, but execution can continue;
  - INFO: Validate proper execution;
- Log system and job boundaries
- Too much log will harm you

Metrics

- Metrics are produced with constant cadence.
- Data grows linearly with time.
- Quantitative data
- Capture key indicators
- Are the foundation for understanding trends, identifying anomalies, and finding performance benchmarks.
- Base for Service-level agreements (SLAs).

Tracing

- End-to-end view of the flow of requests and transactions within a distributed system.
- Highlights latency and dependencies.
- Identifies performance bottlenecks and root causes.
- It's the glue

Combined analysis example

By correlating logs, metrics, and traces, we can form a comprehensive understanding of faults and root causes.

Case by case, not just from past experiences

QUARKUS

Observability in Quarkus

- Multi framework approach.
- Preferred output with OpenTelemetry (OTLP)

Observability in Quarkus

- Signal output convergence with OTel's OTLP protocol.
- Logs
  - Quarkus uses JBoss Log Manager + JBoss Logging facade.
  - Adapters for commons-logging, Log4j, Log4j2, Slf4j, etc...
  - In the future you'll be able to send to OTel Logging.
- Metrics
  - Micrometer. Default output in prometheus format, OTLP available.
- Tracing
  - OpenTelemetry Tracing

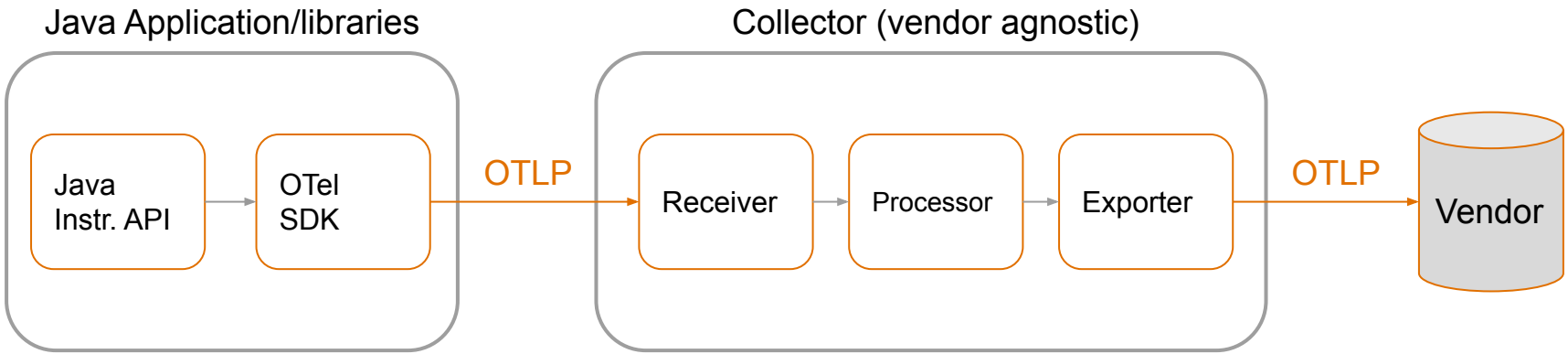
OpenTelemetry

- Software and tools
- Set of Specifications, protocols, formats, APIs and libraries
- Client side only
- It's not:
  - Data Ingestion
  - An Observability backend like Jaeger
  - Storage

OpenTelemetry Components

- **opentelemetry-specification**
  - API (baggage, traces, metrics, logs, etc...)
  - SDK
  - Semantic conventions
  - Data: The OpenTelemetry protocol (OTLP)
- Java Implementation projects
  - **opentelemetry-java**: API, SDK, extensions and the OpenTracing shim.
  - **opentelemetry-java-instrumentation**: The instrumentation API, the Java Agent and the instrumentation libraries.
  - Others like Contrib and Docs
- **opentelemetry-collector** (reference implementation)

OpenTelemetry Components



JSon logs in Quarkus

pom.xml

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-logging-json</artifactId>
</dependency>
```

application.properties

```
quarkus.log.console.format=%d{HH:mm:ss} %-5p traceId=%X{traceId},
parentId=%X{parentId}, spanId=%X{spanId}, sampled=%X{sampled} [%c{2.}]
(%t) %s%e%n

%dev.quarkus.log.console.json=false

%test.quarkus.log.console.json=false
```

Micrometer through OTLP in Quarkus

pom.xml

```
<dependency>
    <groupId>io.quarkiverse.micrometer.registry</groupId>
    <artifactId>quarkus-micrometer-registry-otlp</artifactId>
    <version>${quarkus-micrometer-registry-otlp.version}</version>
</dependency>
```

application.properties

```
quarkus.micrometer.export.otlp.url=http://localhost:4318/v1/metrics
```

OpenTelemetry Tracing in Quarkus

pom.xml

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-opentelemetry</artifactId>
</dependency>
```

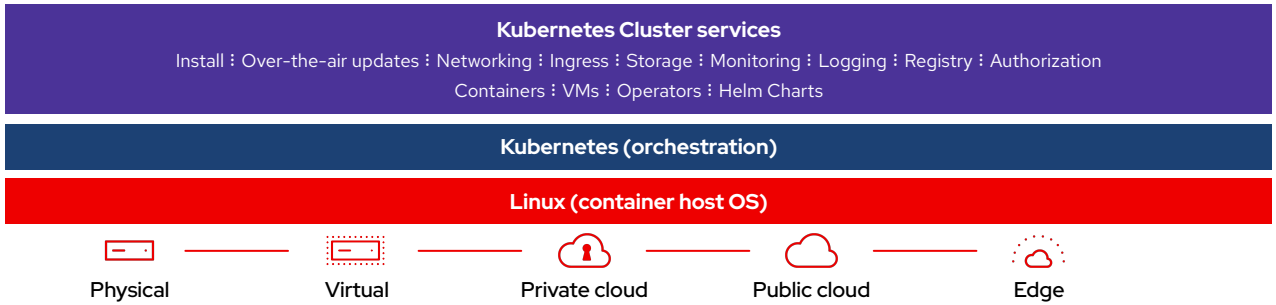
application.properties

```
quarkus.otel.tracer.exporter.otlp.endpoint=http://localhost:4317
```


OPENSHIFT

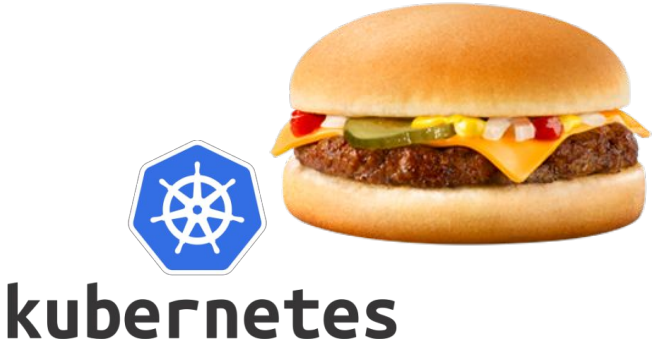
OpenShift enables enterprise ready containers

- Fully automated OpenShift installer provisioned infrastructure
- Immutable over the air or air gapped updates with telemetry
- Integrated OpenShift SDN (OVS/OVN), ingress & CNI plugins
- Kubernetes native storage (Ceph/Rook) & certified CSI plugins
- Deploy and manage both Containers & VMs (Kubevirt)
- Monitoring and alerts to manage your clusters
- Log collection & forwarding to external log management systems
- Built in OpenShift Registry to securely manage container images
- Integrated authentication, authorization & deployment policies
- Leverage Kubernetes Operators & Helm Charts to manage deployments

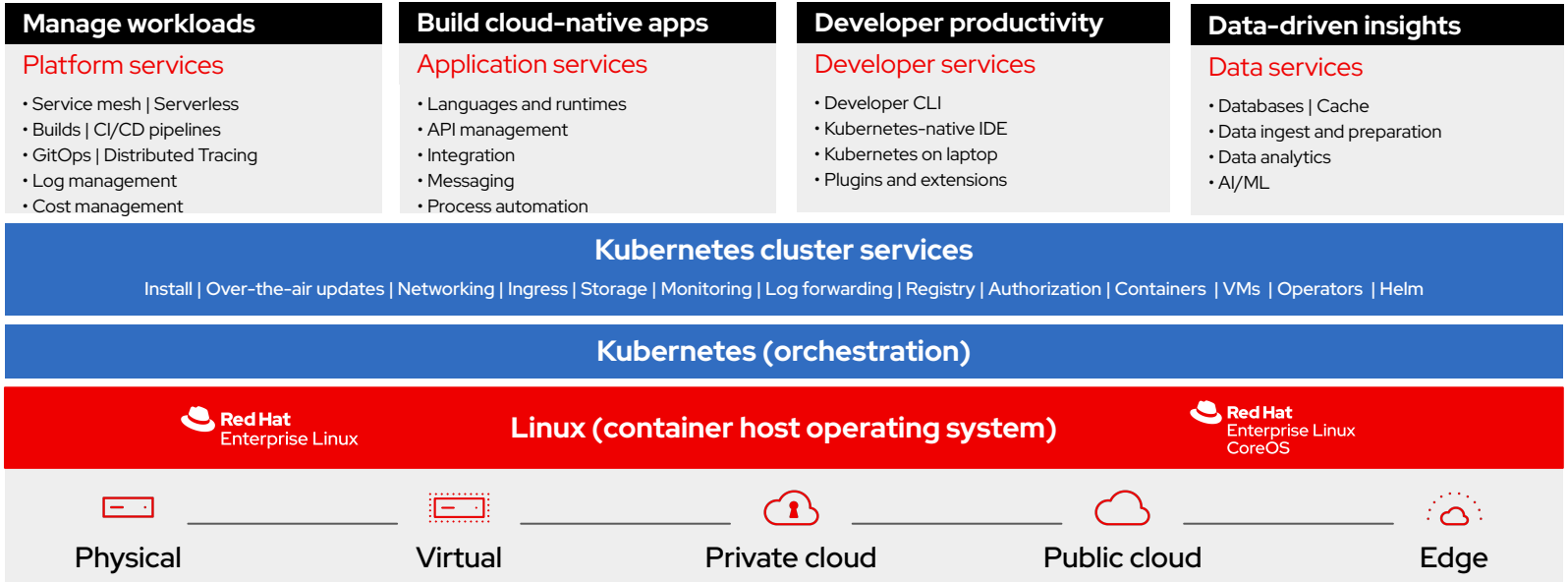


Red Hat

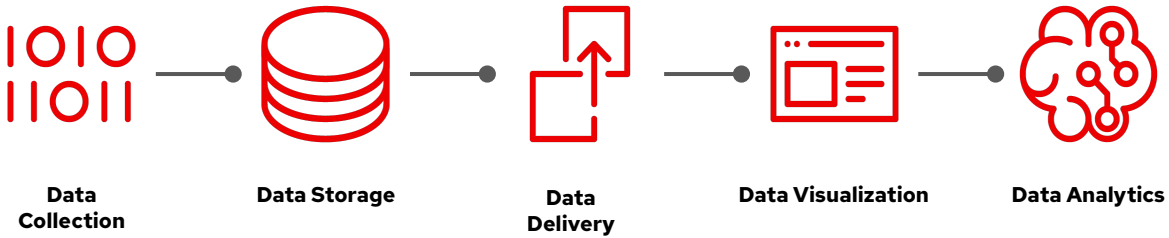
What is OpenShift



What is OpenShift



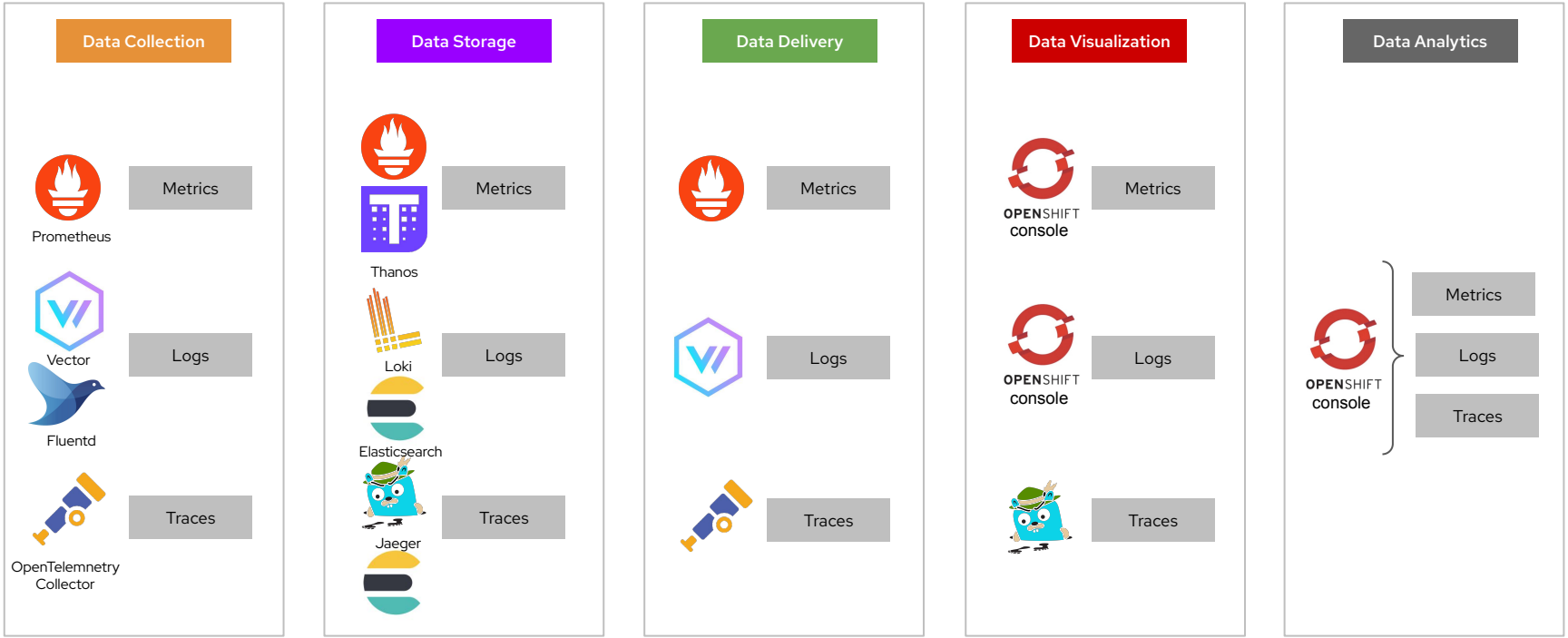
Red Hat


Data Collection
Data Storage
Data Delivery
Data Visualization
Data Analytics

OpenShift Observability

Logical Architecture

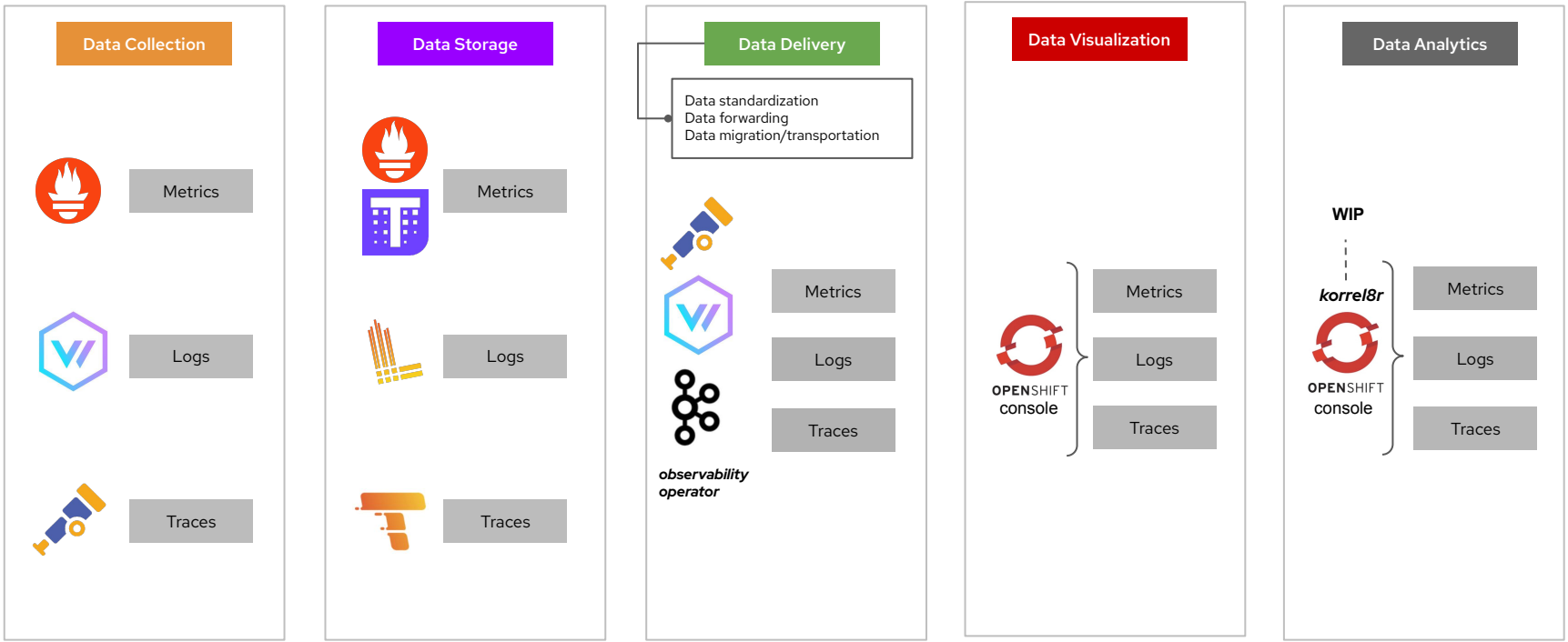
In-cluster Observability Experience



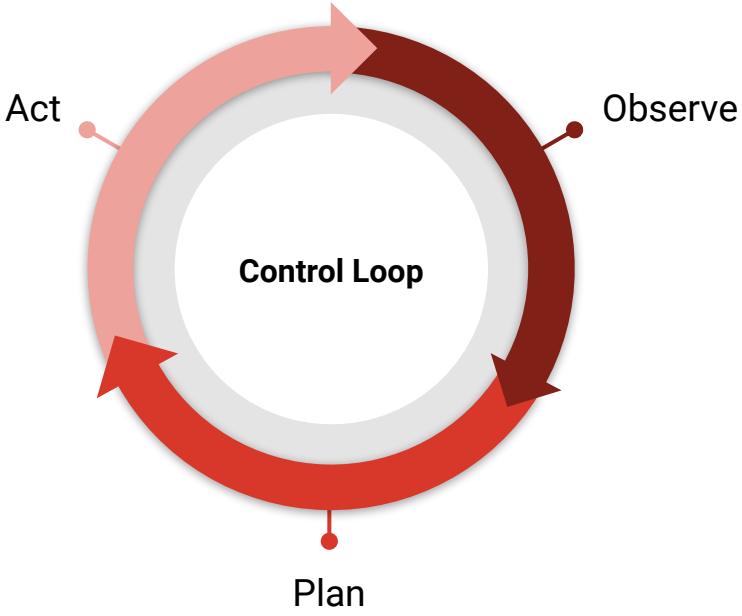
OpenShift Observability

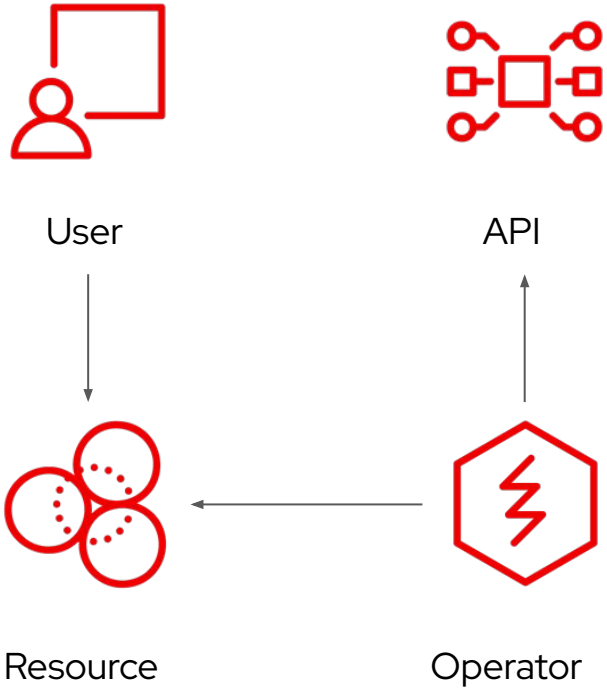
Logical Architecture

In-cluster Observability Experience

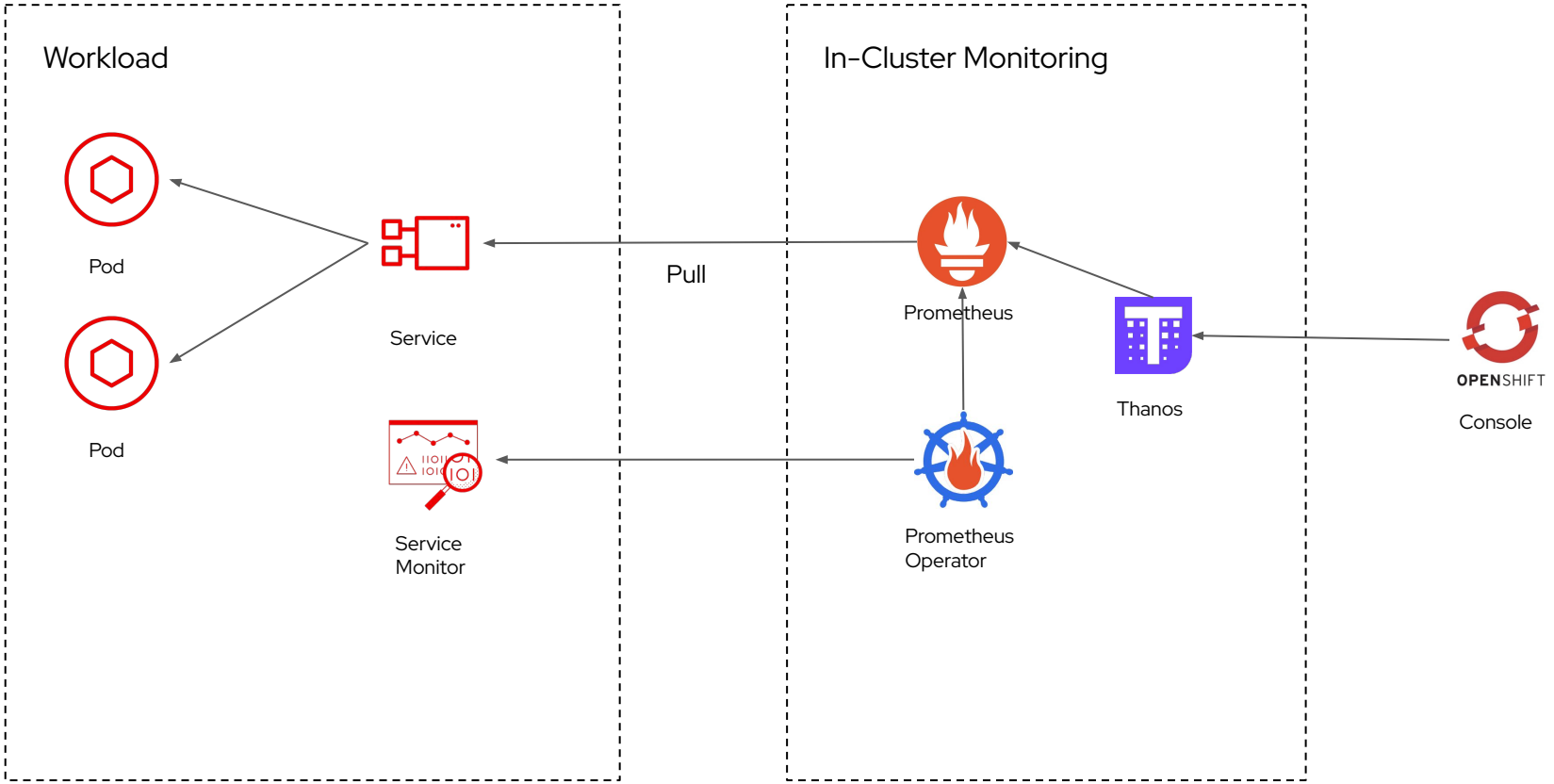


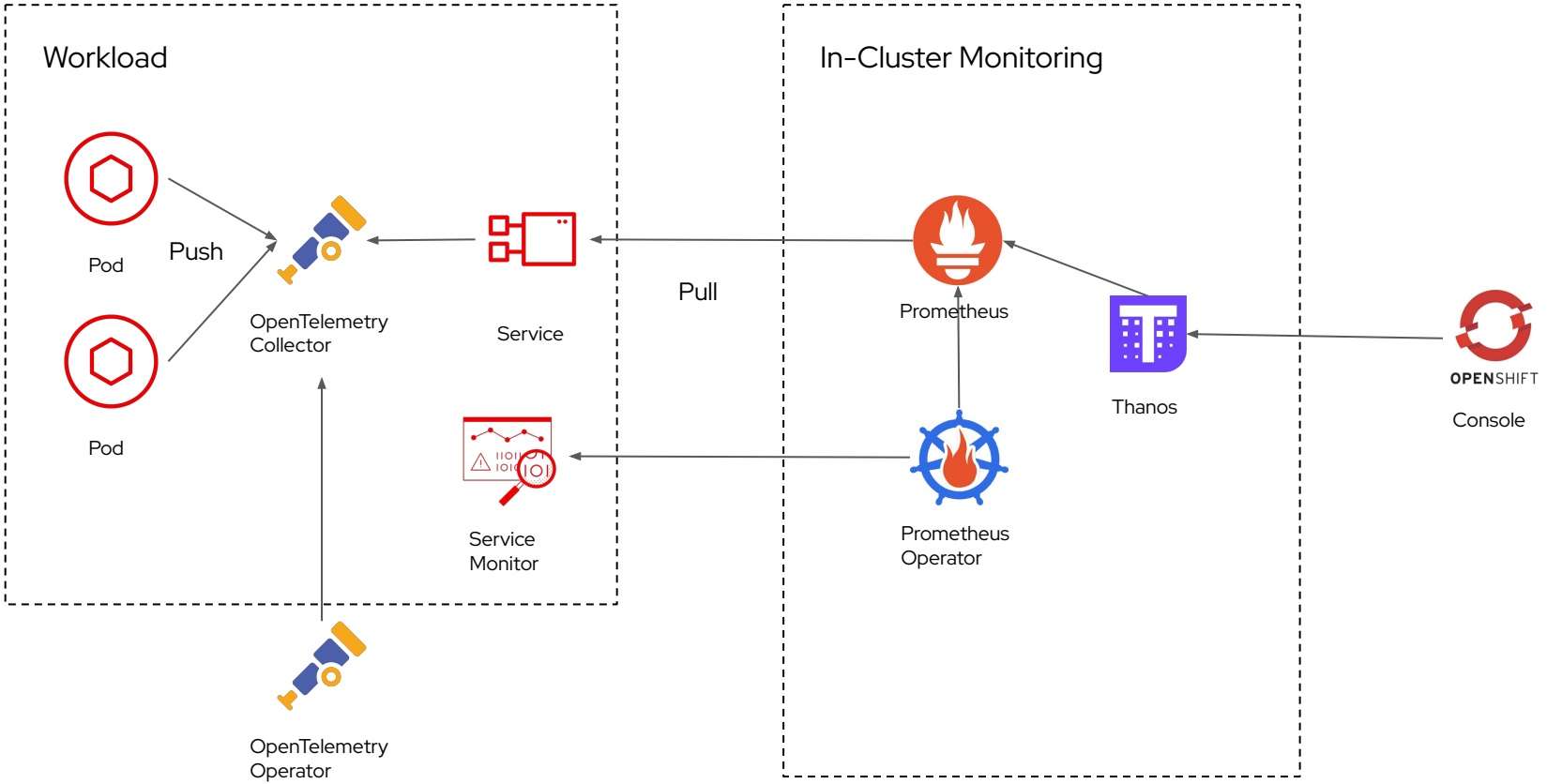
Operator

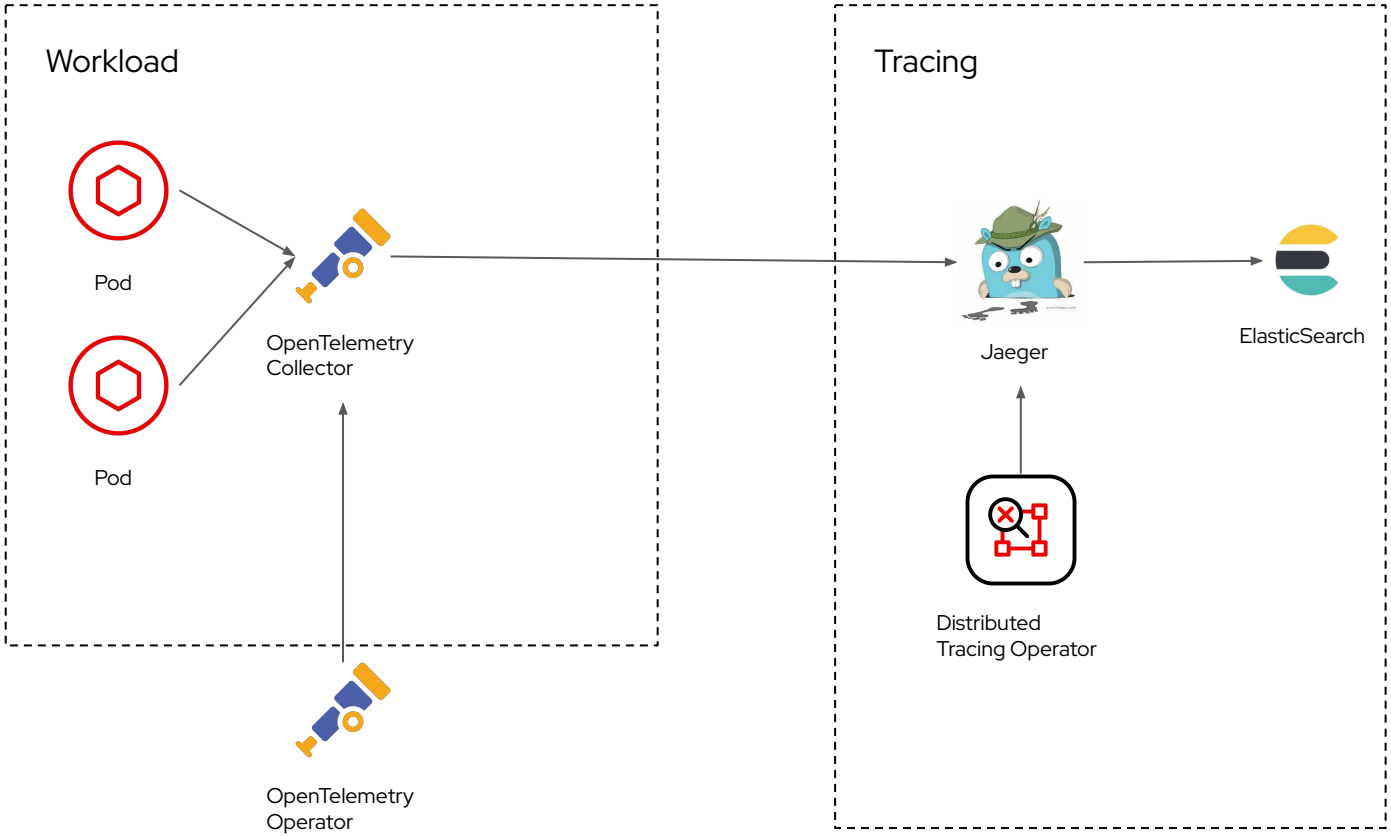




TLDR: *"Operators are Software SRE's"*


Workload
In-Cluster Monitoring
Pod
Pod
Service
Pull
Prometheus
Thanos
OPENSHIFT
Console
Service
Monitor
Prometheus
Operator


Workload
In-Cluster Monitoring
Pod
Push
Pod
OpenTelemetry Collector
Service
Pull
Prometheus
Thanos
OPENSHIFT
Console
Service Monitor
Prometheus Operator
OpenTelemetry Operator


Workload
Pod
Pod
OpenTelemetry
Collector
OpenTelemetry
Operator
Tracing
Jaeger
ElasticSearch
Distributed
Tracing Operator

Infra

Setup observability infrastructure and metric collection

Build

Build Quarkus application with traces and metrics generation

Deploy

Deploy Quarkus application to Openshift Cluster

Infrastructure

Observability infrastructure

Metric collection

Build

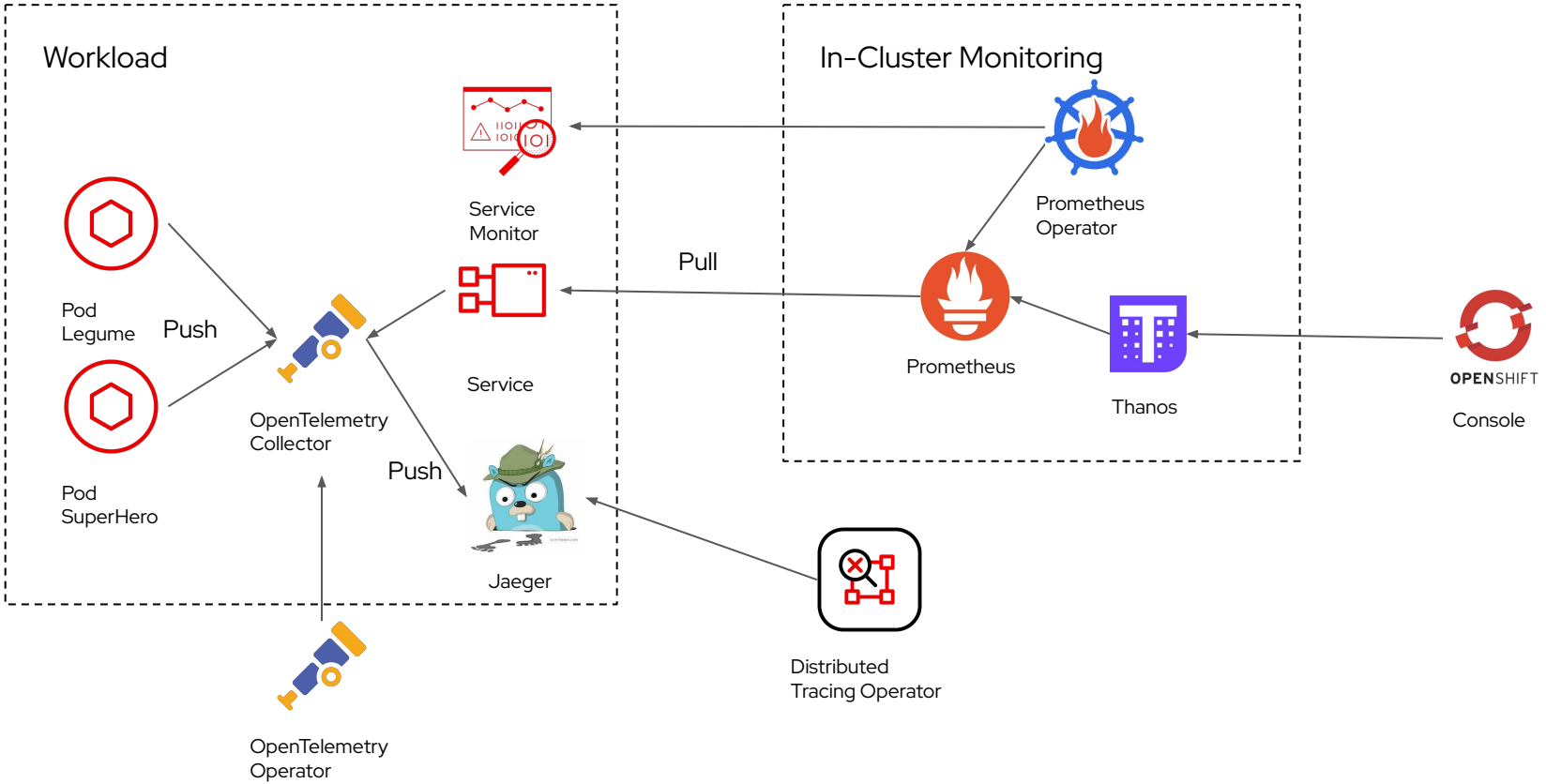
Quarkus Application

Exporting Trace & Metrics

Deployment

Quarkus application on Openshift Cluster

Demo - Architecture



Red Hat

Scan Me

Demo Code:
https://github.com/brunobat/quarkus-observability-demo/tree/main/quarkus-observability-demo-full

Red Hat

Observability

Infer internal state from external signals
Signals: Trace, Metric and Log

Quarkus

Instrument Quarkus application
OpenTelemtry Collector

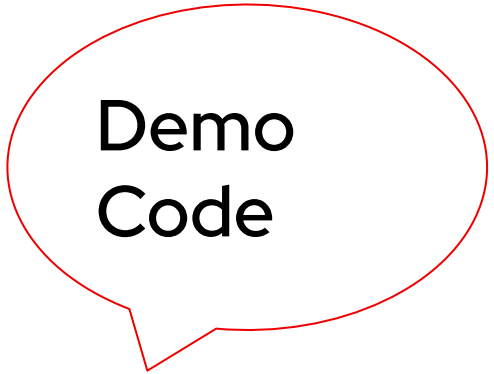
Openshift

Observability infrastructure & application integration

@brunobat_
@vachefoule

Red Hat

https://openfeedback.io/VWEMZHoBj0mPrdZ9Isso



Demo Code:
https://github.com/brunobat/quarkus-observability-demo/tree/main/quarkus-observability-demo-full