

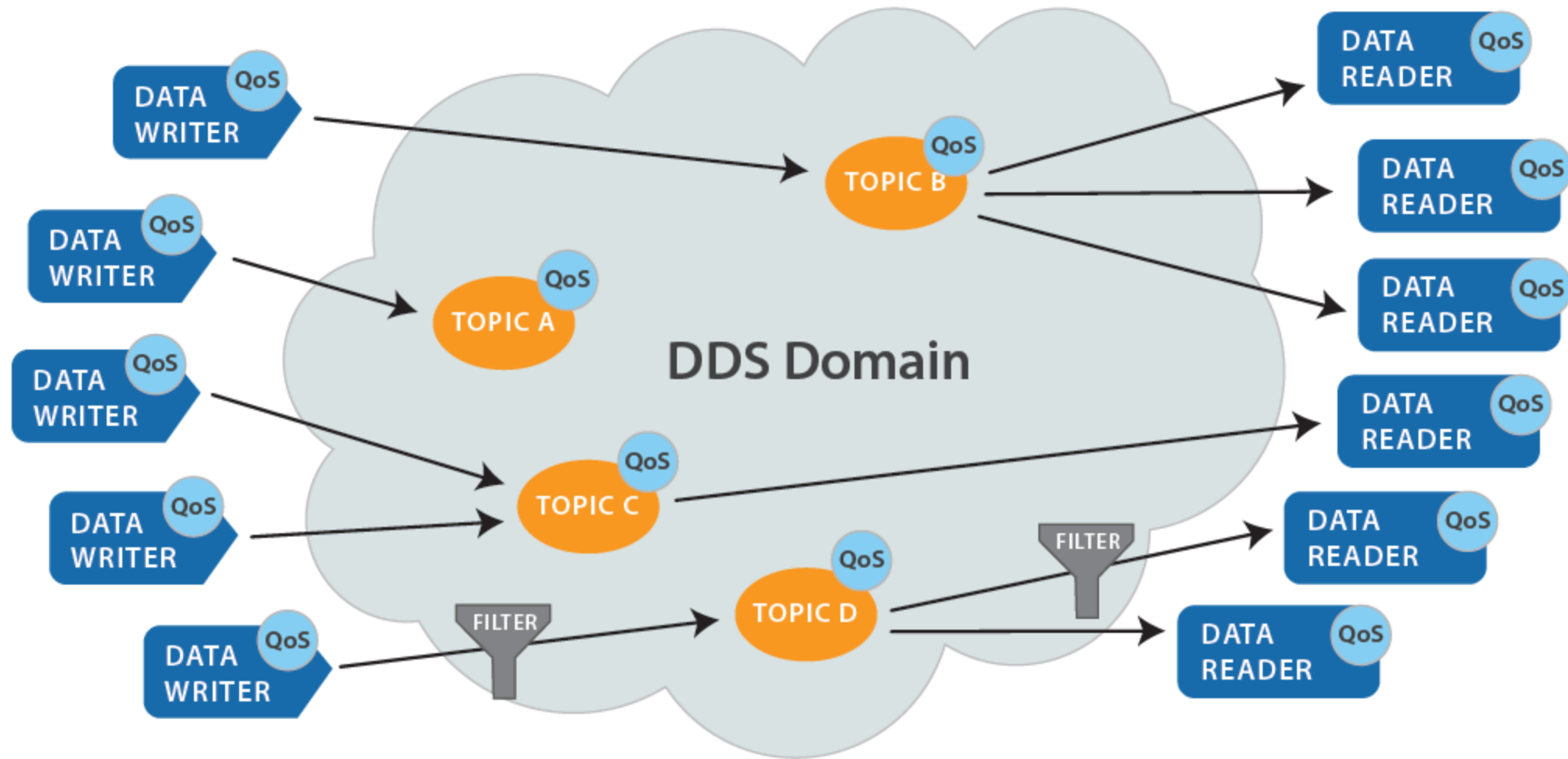
# Monitoring of RTI Connex DDS with Prometheus

## Overview and demonstration

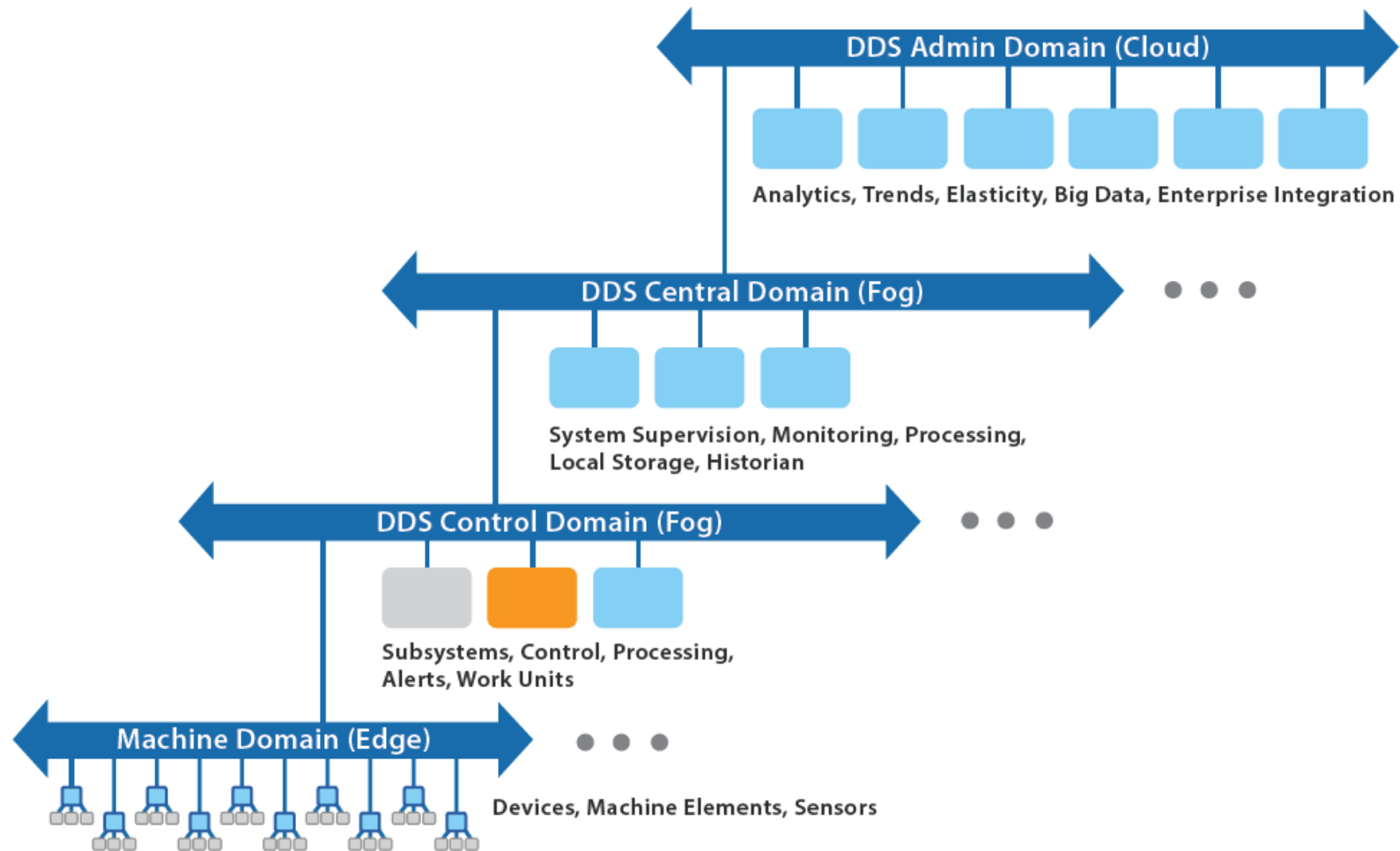
# Data Distribution Service

The Data Distribution Service (DDS) for real-time systems is an Object Management Group (OMG) machine-to-machine (sometimes called middleware or connectivity framework) standard that aims to enable dependable, high-performance, interoperable, real-time, scalable data exchanges using a publish–subscribe pattern.

# Data Distribution Service - Overview



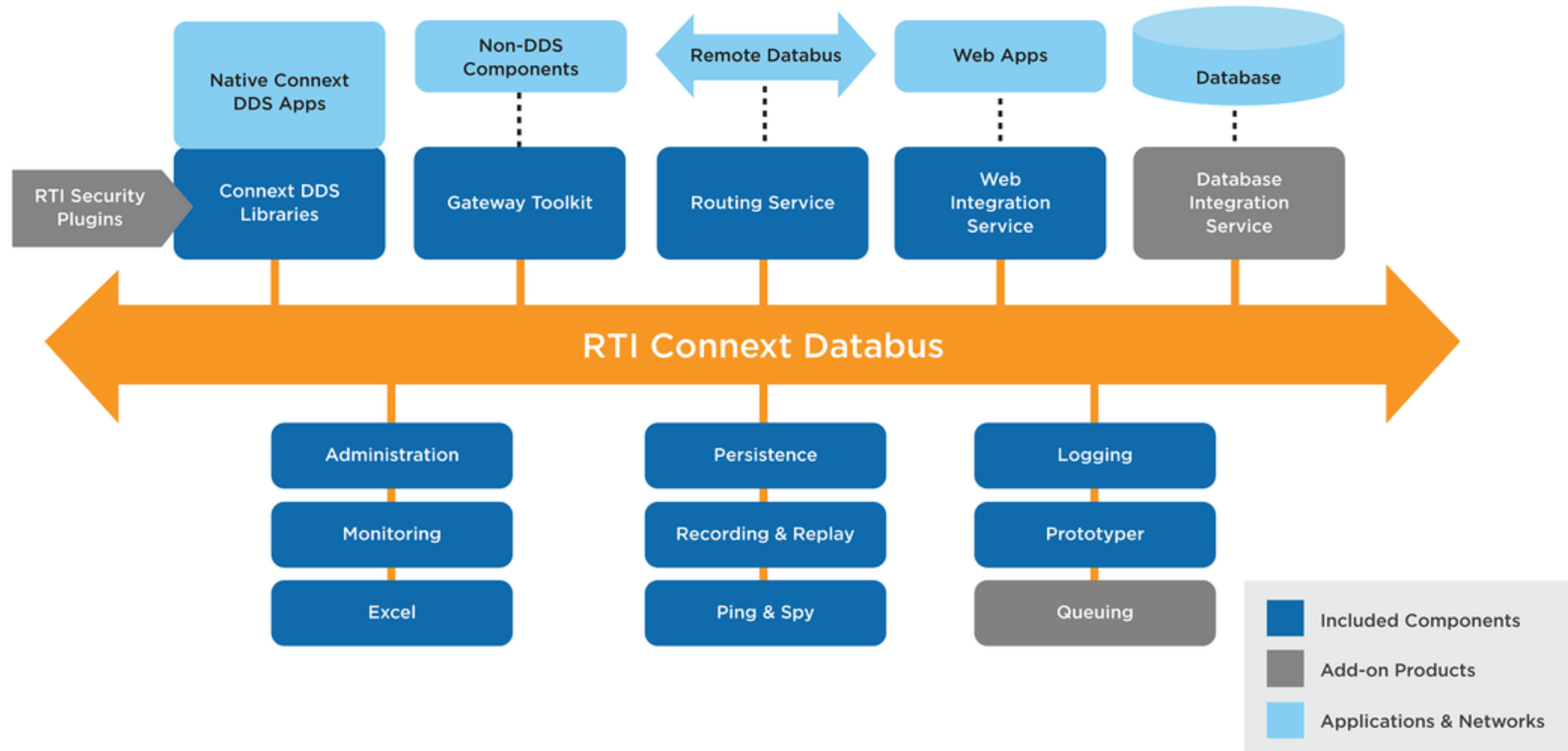
# Data Distribution Service - Layered Data Bus



# RTI Connex DDS

RTI Connex DDS implements the Data-Centric Publish-Subscribe (DCPS) API within the OMG's Data Distribution Service (DDS) for Real-Time Systems. DDS is the first standard developed for the needs of real-time systems. DCPS provides an efficient way to transfer data in a distributed system.

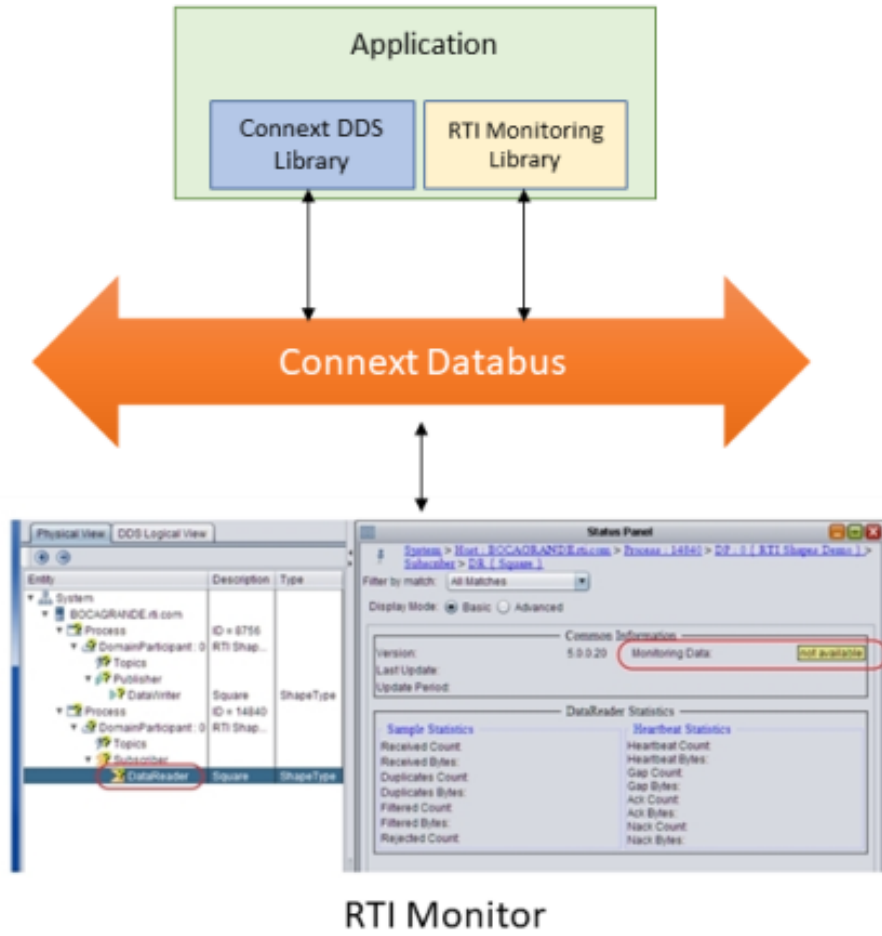
# RTI Connex DDS



# Monitoring Library

- Can be dynamically loaded and attached to RTI Connex DDS libraries
- Reads data from the core libraries and writes them to defined topics
- Several configuration parameters available, e.g. how often data is written
- RTI Monitor can read and show the monitoring data

# Monitoring Library





# Monitoring of Routing Service

- If enabled, Routing Service writes monitoring data to defined topics
- Several configuration parameters available, e.g. how often data is written
- RTI Admin Console can read and show the monitoring data

# Monitoring of Routing Service

RTI Administration Console

File View Visualization Help

dds-examples-routing-static : 38295

System → rhel-8.vm → Routing - dds-examples-routing-static

Routing Entities DDS Entities Log Routing Information Resource Charts

Metrics

dds-examples-routing-static

Default

DefaultShapes

Square Circle

Triangle

Square@Square

Input1 Processor Output1

1 2

19.59 S/s 783.53 B/s

Latency: 0.11 μs

Statistics Configuration

Selected entity dds-examples-routing-static

State Started

Name	State	Input Sam
[ RTI Routing Service ] dds-examples-routing-static	Started	
[ Domain Route ] Default	Started	19.59 S/s 783.53 B/s
[ Connection ] 1	N/A	
[ Connection ] 2	N/A	
[ Session ] DefaultShapes	Started	19.59 S/s 783.53 B/s
[ Auto Route ] Circle	Started	19.59 S/s 783.53 B/s
[ Auto Route ] Square	Started	19.59 S/s 783.53 B/s
[ Auto Route ] Triangle	Started	19.59 S/s 783.53 B/s
[ Route ] Square@Square	Running	19.59 S/s 783.53 B/s
[ Route Input ] Input1	Enabled	19.59 S/s 783.53 B/s
[ Processor ] Processor	N/A	
[ Route Output ] Output1	Enabled	

# Monitoring of Routing Service

RTI Administration Console

File View Visualization Help

dds-examples-routing-static : 38295

System → rhel-8.vm → Routing - dds-examples-routing-static

Routing Entities DDS Entities Log Routing Information Resource Charts

Metrics

Selected entity: dds-examples-routing-static

State: Started

Diagram of dds-examples-routing-static:

```
graph LR
    1((1)) --> Input1[Input1]
    Input1 -- "19.59 5%  
783.53 B/s" --> Processor[Processor  
Latency: 0.11 μs]
    Processor -- "19.59 5%  
783.53 B/s" --> Output1[Output1]
    Output1 --> 2((2))
```

Statistics

Name	State	Input Samples/s	Output Samples/s	Input Bytes/s	Output Bytes/s	Latency
[ RTI Routing Service ] dds-examples	Started					
[ Domain Route ] Default	Started	19.79	19.79	791.68	791.68	0.11
[ Connection ] 1	N/A					
[ Connection ] 2	N/A					
[ Session ] DefaultShapes	Started	19.79	19.79	791.68	791.68	0.11
[ Auto Route ] Circle	Started	0	0	0	0	0
[ Auto Route ] Square	Started	19.79	19.79	791.68	791.68	0.11
[ Auto Route ] Triangle	Started	0	0	0	0	0
[ Route ] Square@Square	Running	19.59	19.59	783.53	783.53	0.11

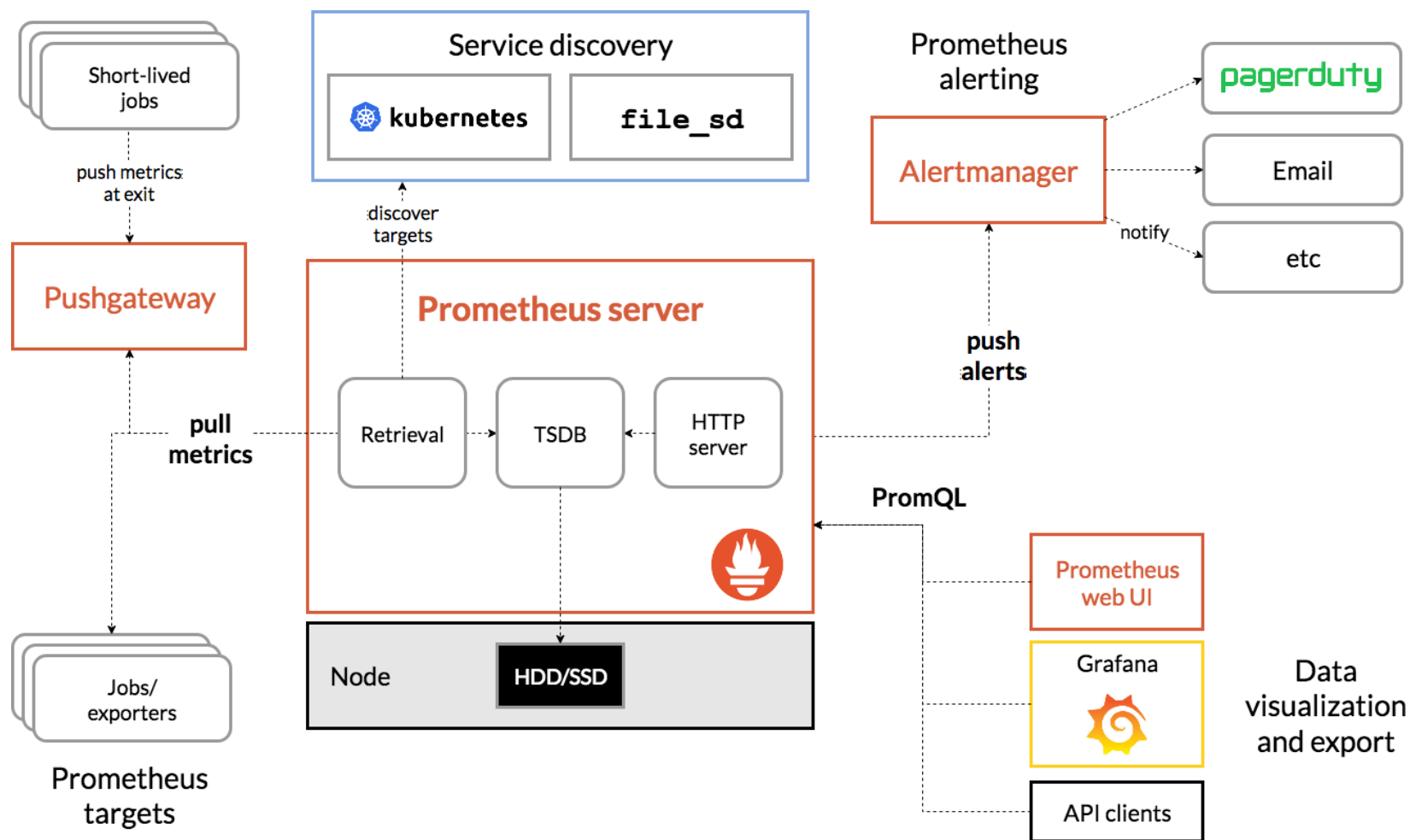
## Prometheus

Prometheus, a Cloud Native Computing Foundation project, is a systems and service monitoring system. It collects metrics from configured targets at given intervals, evaluates rule expressions, displays the results, and can trigger alerts if some condition is observed to be true.

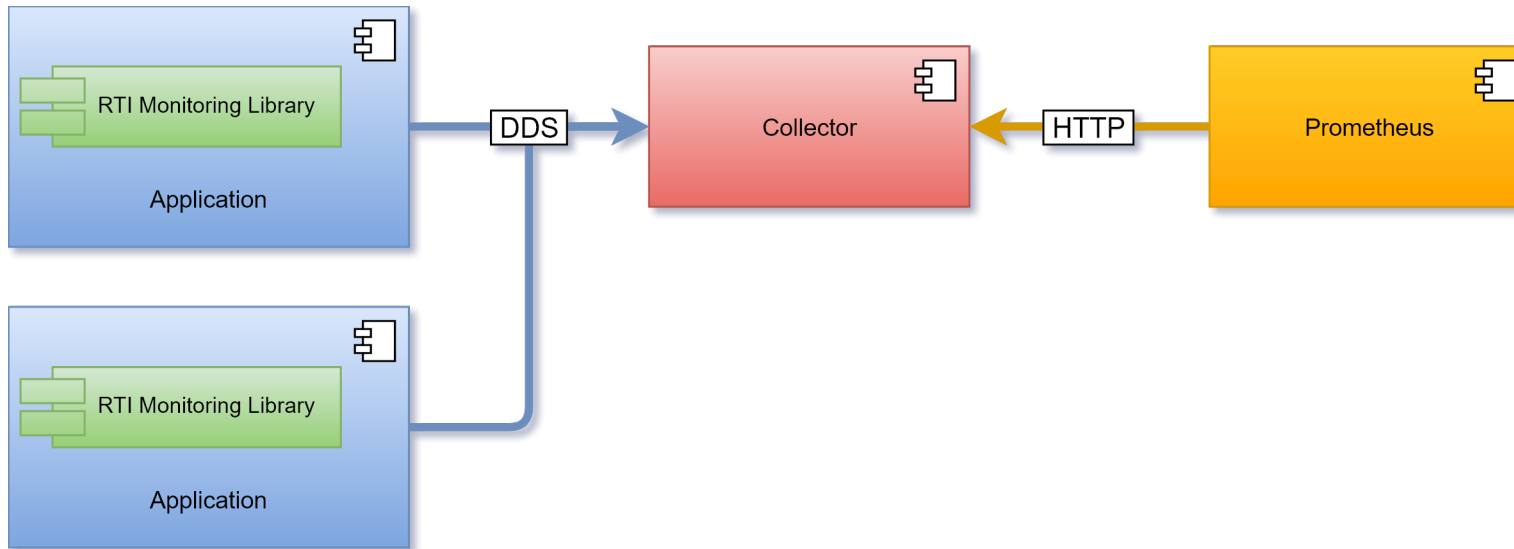
## Grafana

Grafana is an open source, feature rich metrics dashboard and graph editor for Graphite, Elasticsearch, OpenTSDB, Prometheus and InfluxDB.

# Architecture



# Collector - Data Flow



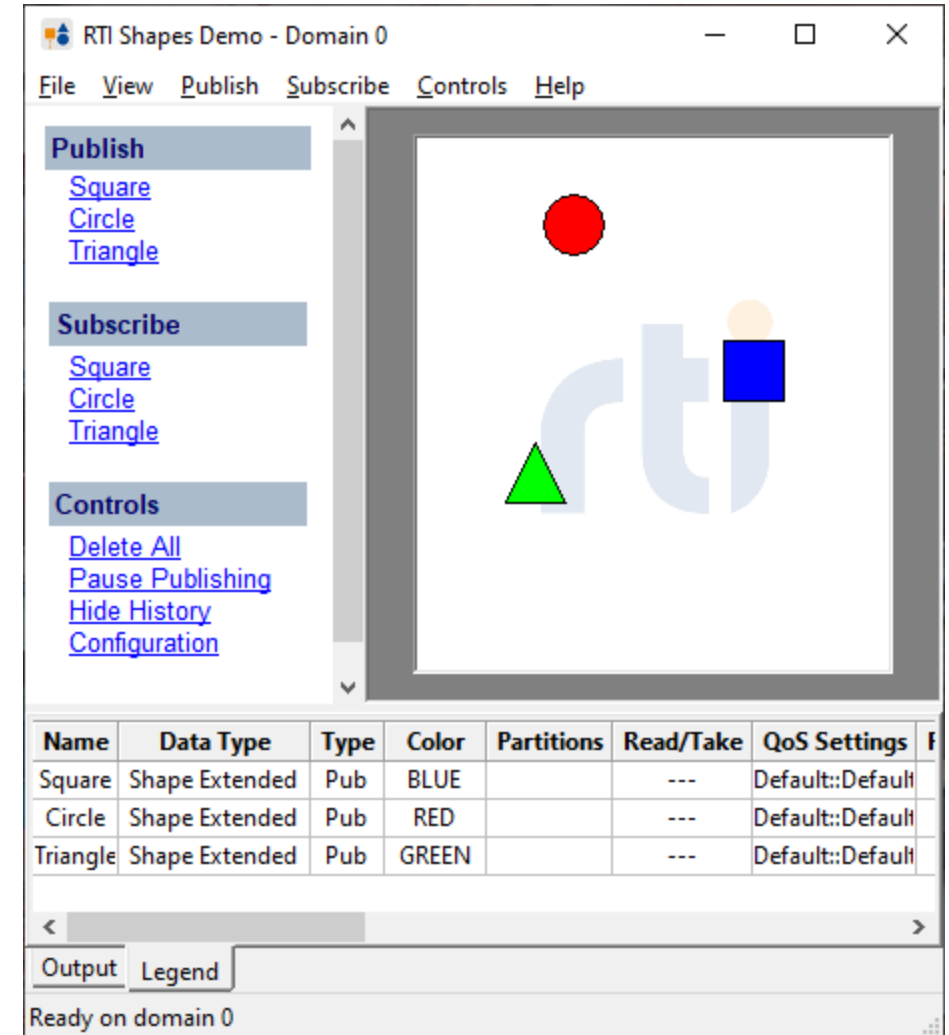
- Collector subscribes to DDS topics to receive the monitoring data
- Collector converts monitoring data into prometheus metrics (with labels)
- Prometheus is scraping the collector regularly to read metrics

# Collector - Metric example

```
# HELP dds_routing_service_process_uptime_seconds Time in seconds elapsed since the running service process started.  
# TYPE dds_routing_service_process_uptime_seconds gauge  
dds_routing_service_process_uptime_seconds{routing_service="default",} 301.0
```

# RTI Shapes Demo

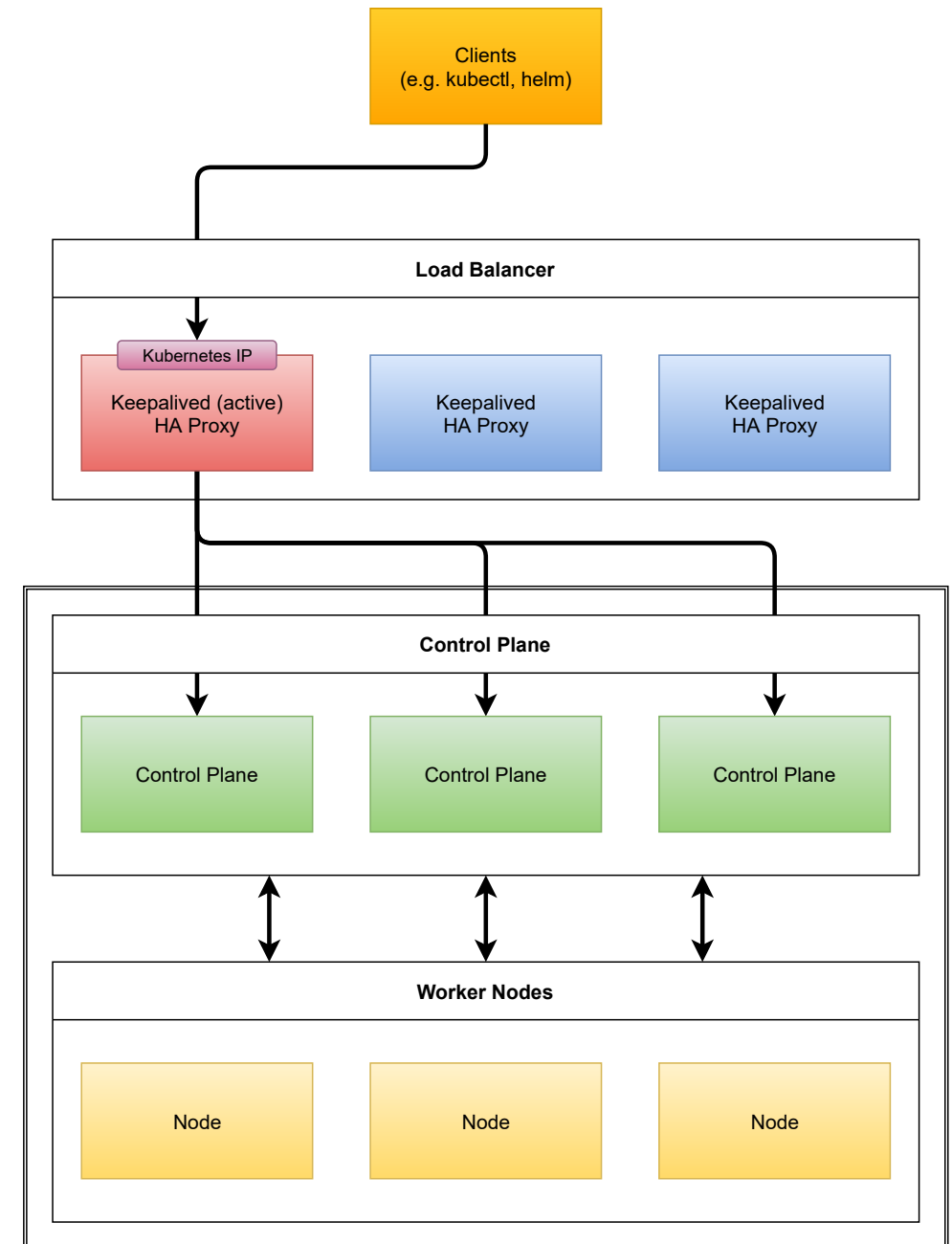
- Usage to improve understanding
- Usage for compatibility tests
- Repository dds-examples provides the following console applications:
  - ShapePublisher
  - ShapeSubscriber



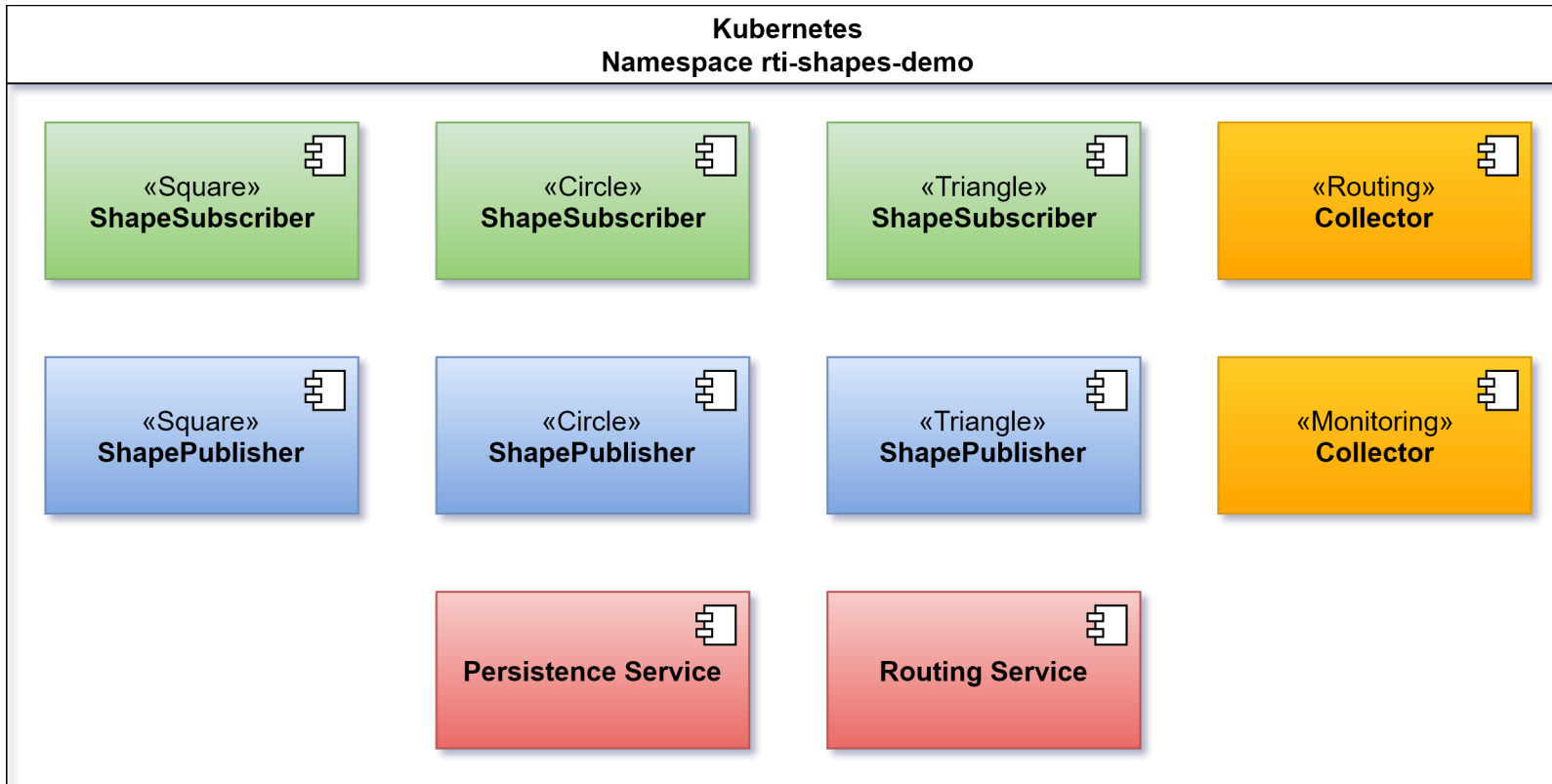


# Deployment - Kubernetes Structure

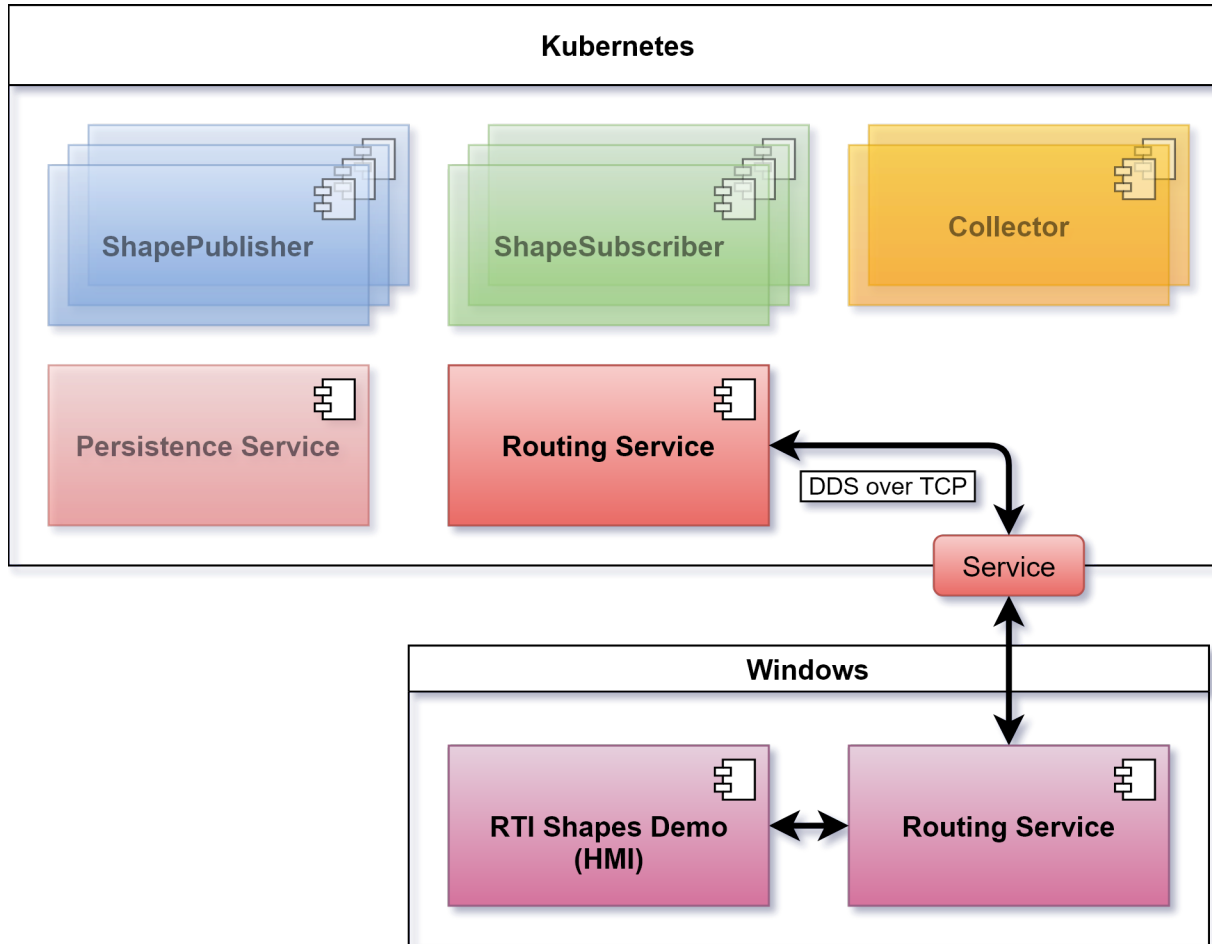
- Keepalived + HAProxy for load balancing of control plane
- metallb for external access to services (in L2 mode)
- rook-ceph for storage
- local docker-registry for images



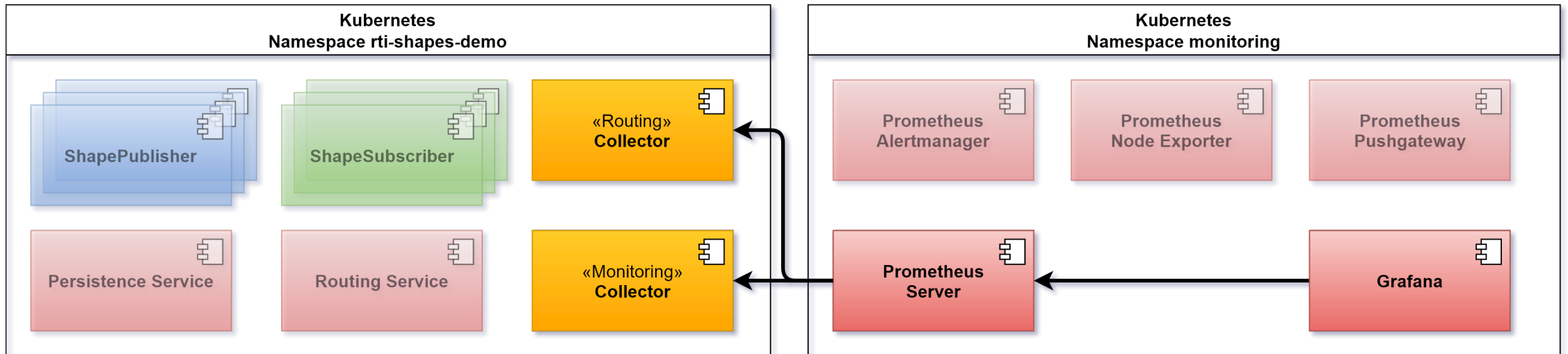
# Deployment - Namespace for Shapes Demo



# Deployment - Connection of HMI

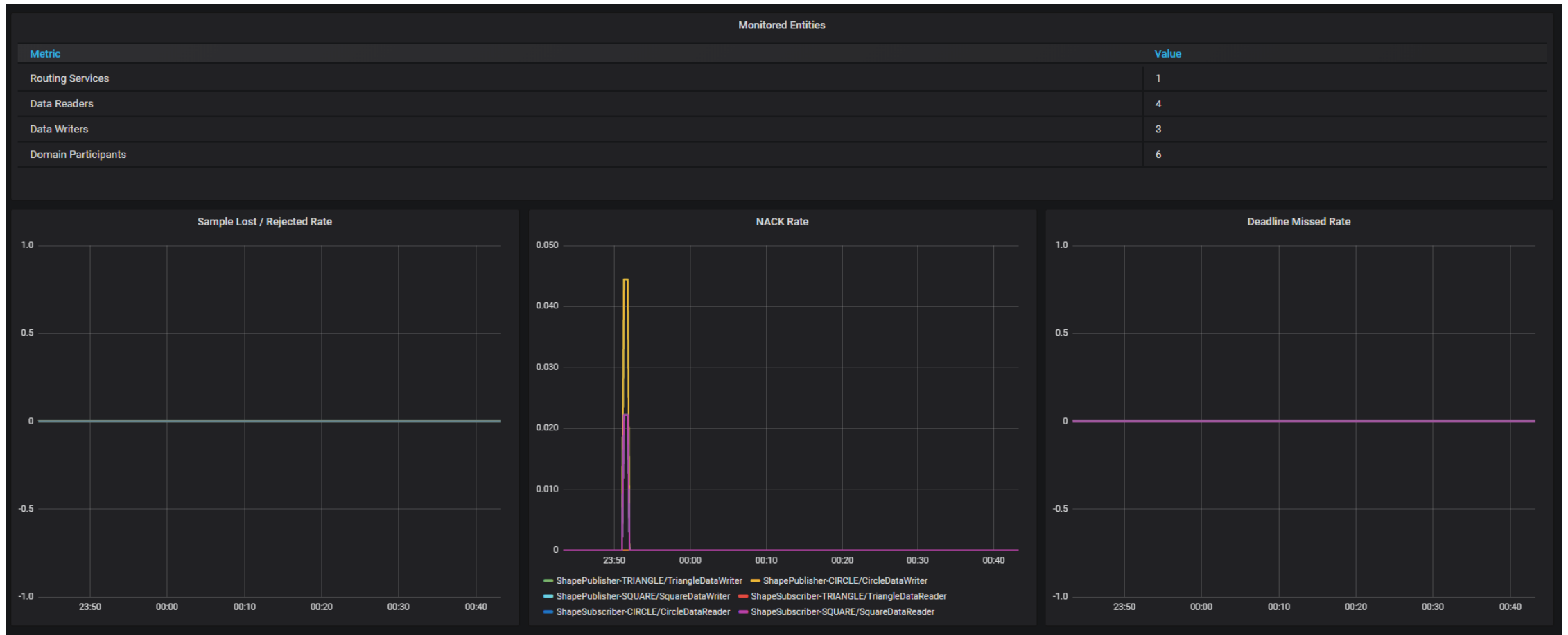


# Deployment - Scraping



# Demo

# Demo - Screenshot Overview



# Demo - Screenshot Application








# Demo - Screenshot Routing Service







# Conclusions - Advantages

-  Separation of monitoring, alerting from business logic
-  Prometheus / Grafana is a powerful tool for queries and analysis
-  can be used for long term monitoring and storage
-  data aggregation/compaction is possible
-  automatically taking advantage of new features available on the market

## Conclusions - Disadvantages

-  Additional DDS traffic is generated for monitoring
-  Prometheus does not support text, so information like QoS cannot be tracked

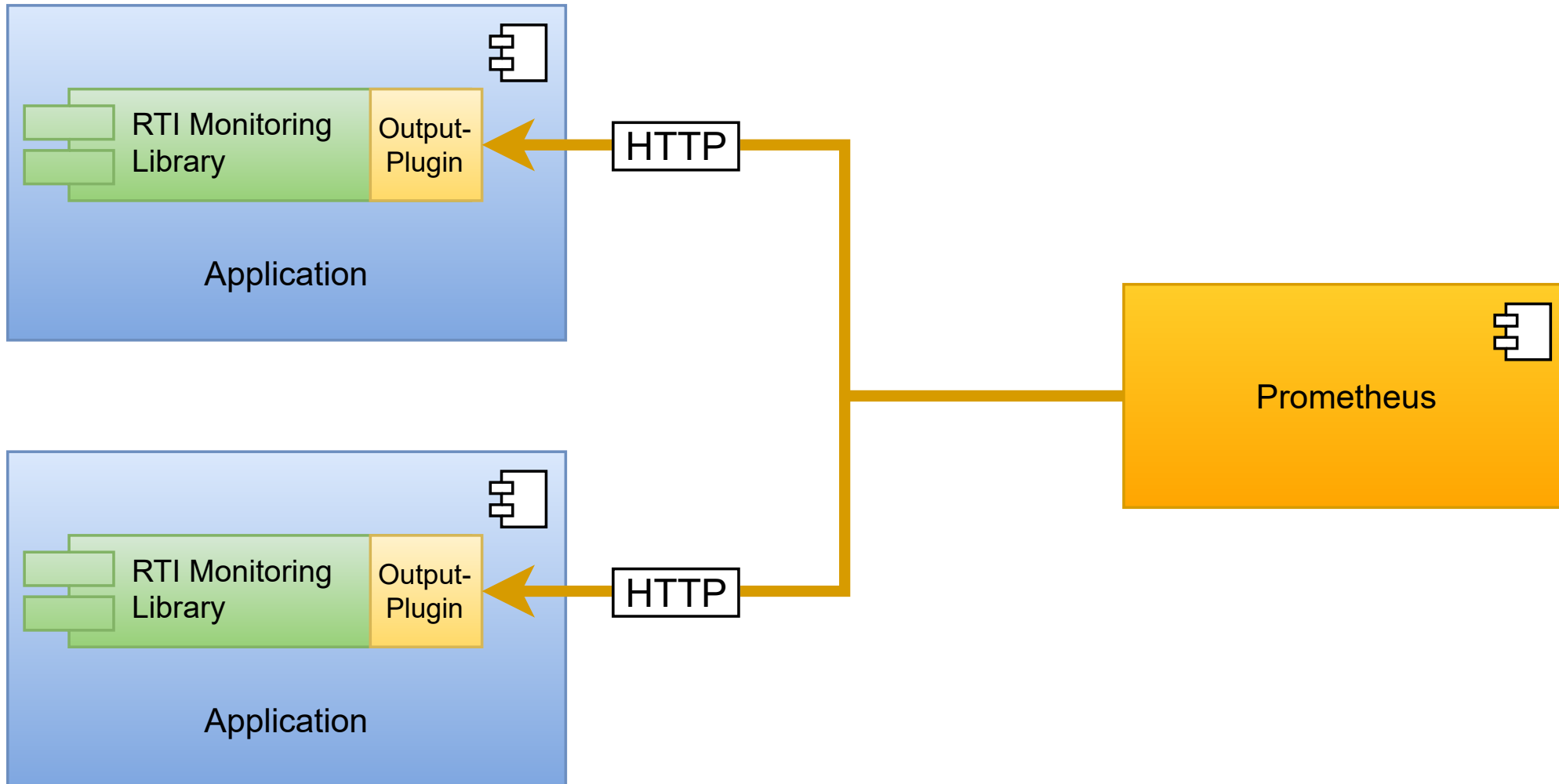
# Conclusions - Things to think about

- 🤔 Revisit of labels to be used (new set of labels create a new series)
- 🤔 Usage of collector as side-car to have a more clear mapping to pods





# Proposal

- split RTI Monitoring Library into collection and output part
- use plugin mechanism for output part to support multiple formats
- default plugin could be the todays interface
- foundation on OMG "Data Distribution Service (DDS) Status Monitoring"

# Proposal - Architecture



# Proposal - Advantages

-  mapping of monitoring data to pods will work correctly
-  supports both push and pull based collection
-  saves resources by using no additional...
  - ...DDS resources
  - ...processes like collector
  - ...network traffic
-  will allow easy integration of future technologies or custom solutions

 **Thank you for your attention!**

# Links

## Presentation

<https://www.github.com/aguther/presentations>

## Data Distribution Service

<https://www.dds-foundation.org>

<https://community.rti.com/documentation>

## Code

<https://www.github.com/aguther/dds-examples>

<https://www.github.com/aguther/deployment-kubernetes>

<https://www.github.com/aguther/deployment-containers-rti>