Enhanced IBM MQ Insights Through Observability and the Power of OpenTelemetry

Toby Keegan
Software Engineer, IBM MQ for z/OS
toby.keegan@ibm.com

Agenda

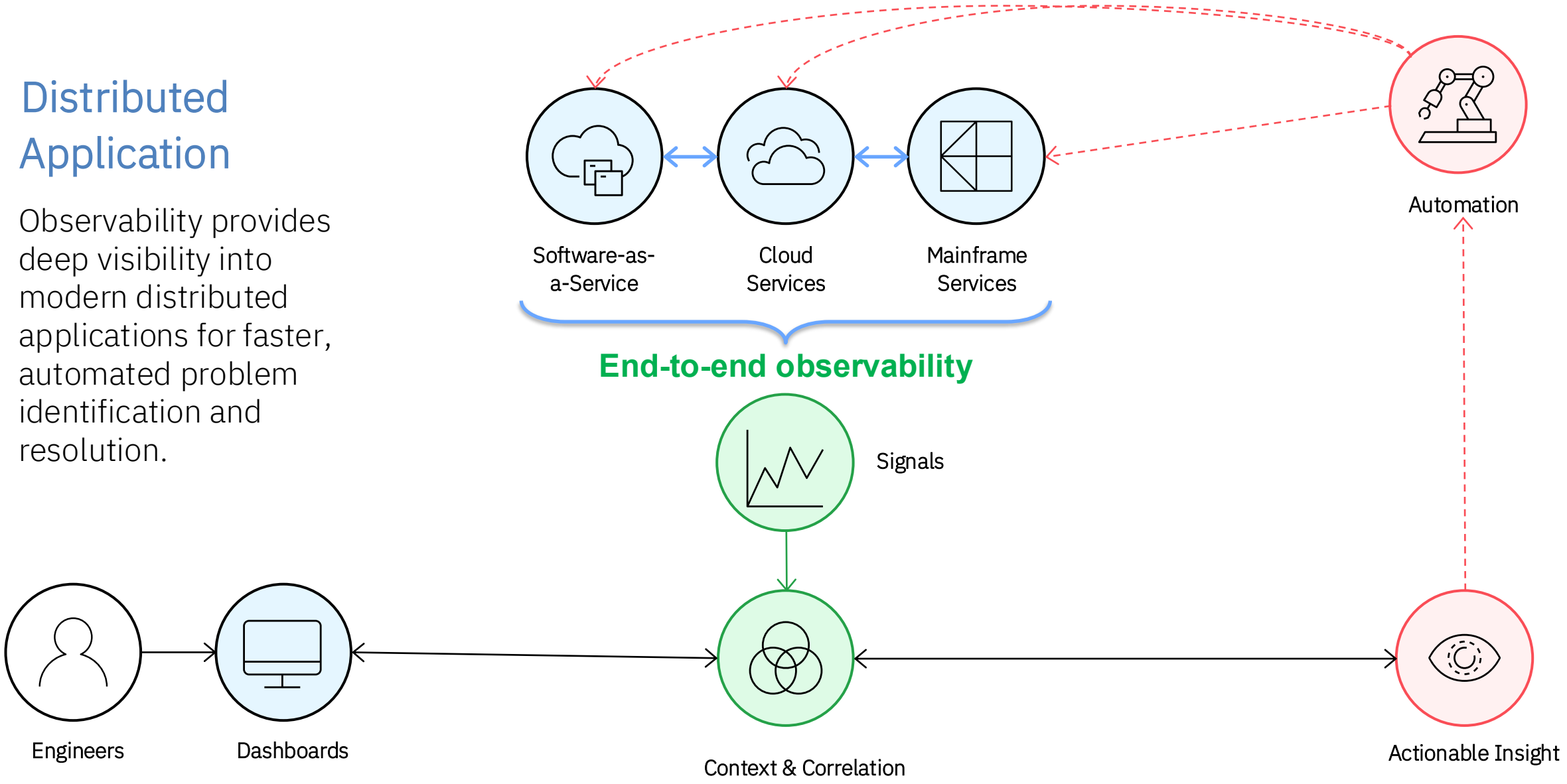
- Why observability matters!
- What is OpenTelemetry?
- OpenTelemetry with MQ on distributed
- OpenTelemetry with MQ on z/OS
- Live demo
- A quick flick at the MQ Console

Why observability matters!

Why end-to-end observability?

Distributed Application

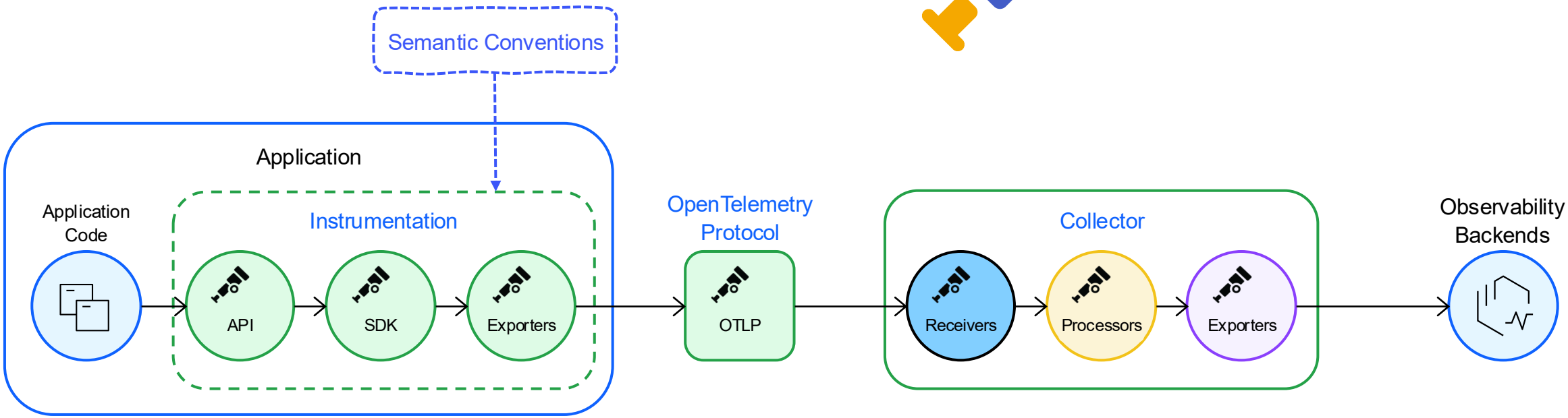
Observability provides deep visibility into modern distributed applications for faster, automated problem identification and resolution.



Source: IBM – What is observability?

What is OpenTelemetry?

What is OpenTelemetry?



OpenTelemetry is a vendor-agnostic observability framework that assists in generating, processing, and distributing telemetry data such as metrics, traces, and logs.

OpenTelemetry has lots of support already out there!

Apache License 2.0 • 107 • 626 • 0 • 0 • Updated on Dec 13, 2024

opentelemetry-dotnet-contrib Public

This repository contains set of components extending functionality of the OpenTelemetry .NET SDK. Instrumentation libraries, exporters, and other components can find their home here.

dotnet dotnet-core opentelemetry

C# • Apache License 2.0 • 351 • 598 • 219 • 7 • Updated 14 hours ago

opentelemetry-ruby Public

OpenTelemetry Ruby API & SDK, and related gems

metrics telemetry hacktoberfest distributed-tracing opentelemetry opentelemetry-ruby opentelemetry-api

Ruby • Apache License 2.0 • 270 • 544 • 57 • 16 • Updated 5 days ago

opentelemetry-helm-charts Public

OpenTelemetry Helm Charts

Smarty • Apache License 2.0 • 664 • 492 • 121 • 15 • Updated yesterday

opentelemetry-dotnet-instrumentation Public

OpenTelemetry .NET Automatic Instrumentation

C++ • Apache License 2.0 • 125 • 438 • 104 • 12 • Updated 1 hour ago

semantic-conventions Public

Defines standards for generating consistent, accessible telemetry across a variety of domains

Open Policy Agent • Apache License 2.0 • 272 • 431 • 649 • 130 • Updated 8 hours ago

opentelemetry-collector-releases Public

OpenTelemetry Collector Official Releases

opentelemetry open-telemetry

Go • Apache License 2.0 • 213 • 398 • 41 • 6 • Updated 4 hours ago

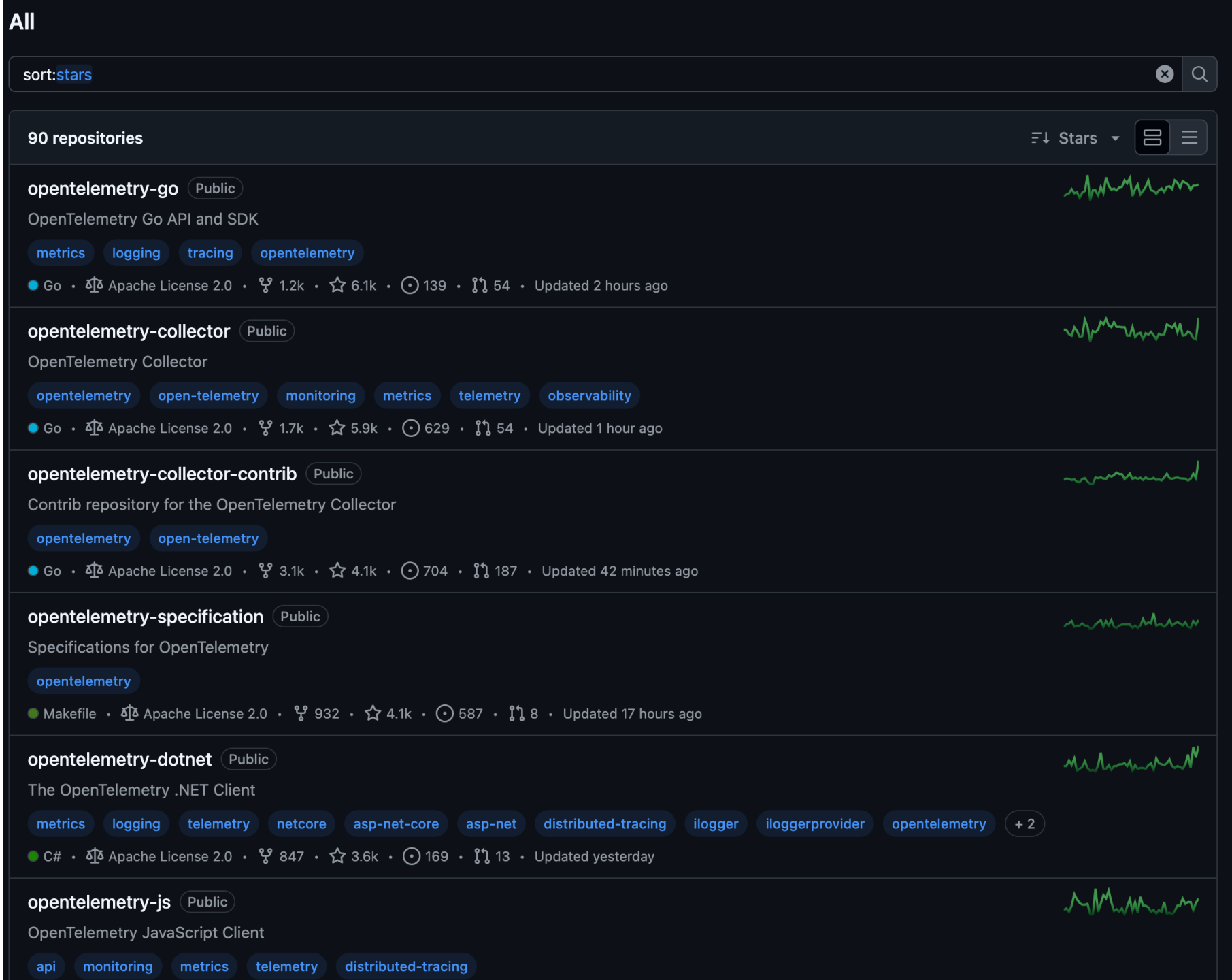
opentelemetry-network Public

eBPF Collector

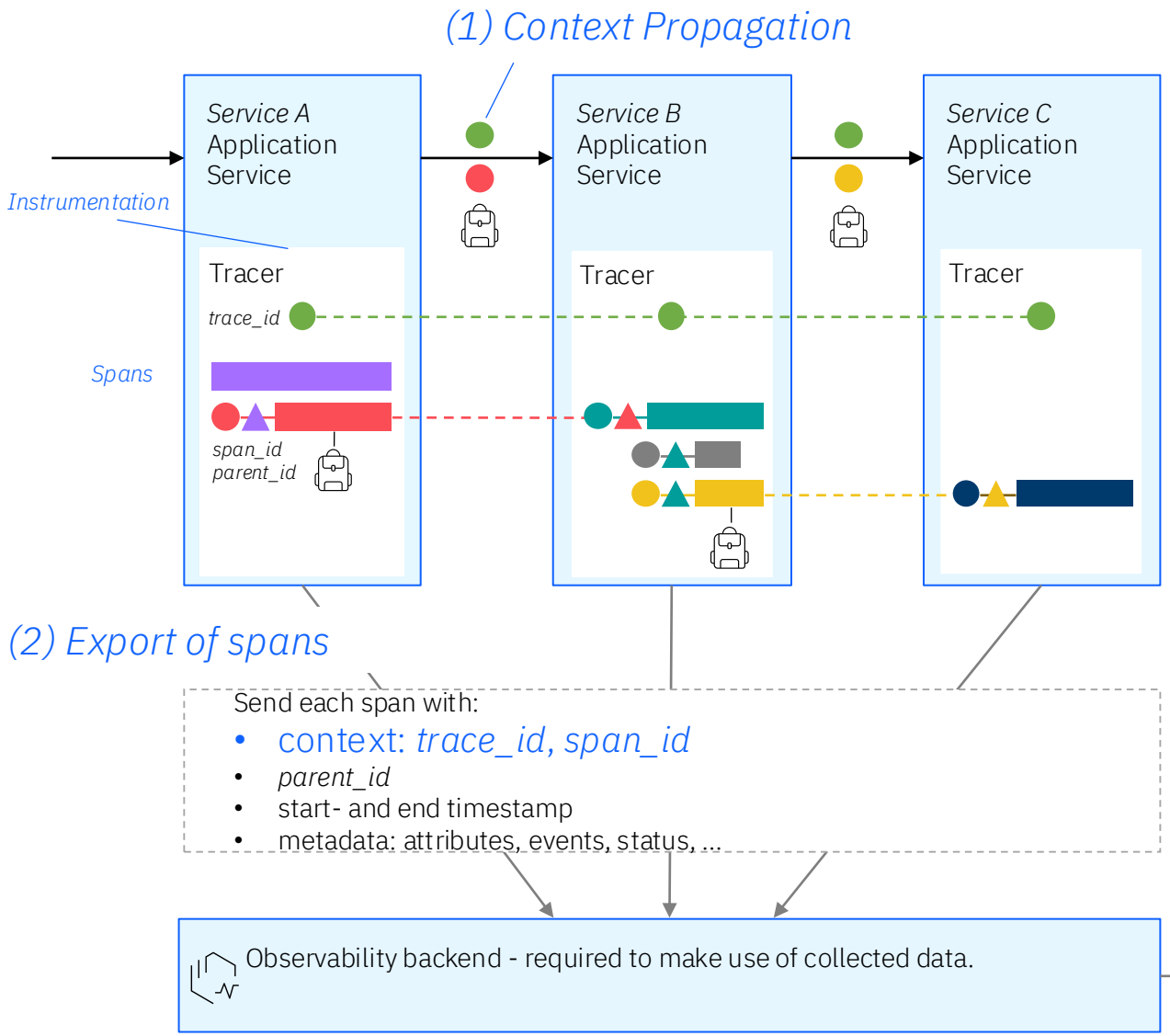
ebpf opentelemetry open-telemetry

C++ • Apache License 2.0 • 63 • 386 • 33 • 6 • Updated 7 hours ago

OpenTelemetry has lots of support already out there!



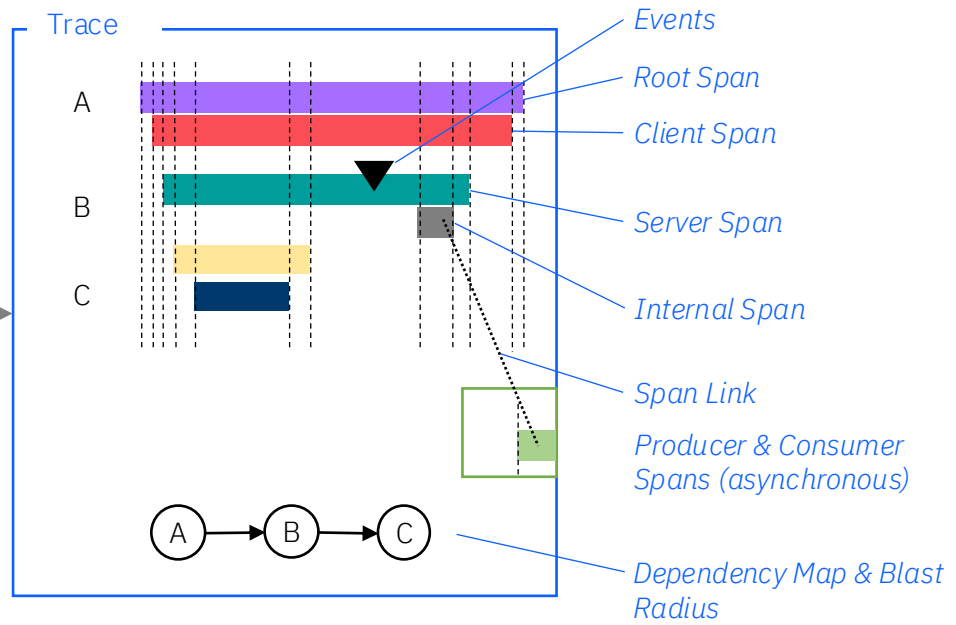
Distributed Tracing – a conceptual view



Trace: single request and its path through a system of services

Span (timespan): represents a unit of work or operation

Tracers are part of the runtime and collect data.



177 Traces

- GET /
  2024-02-27, 22:03:50 4ms
- GET /owners/find
  2024-02-27, 22:03:49 6ms
- GET /vets.html
  2024-02-27, 22:03:46 15ms
- GET /vets.html
  2024-02-27, 22:03:44 32ms
- GET /owners/{ownerId}
  2024-02-27, 22:03:40 21ms
- POST /owners/{ownerId}/pets/{petId...
  2024-02-27, 22:03:40 26ms
- GET /owners/{ownerId}/pets/{petId}/...
  2024-02-27, 22:03:34 43ms
- GET /owners/{ownerId}
  2024-02-27, 22:03:27 16ms
- POST /owners/{ownerId}/edit
  2024-02-27, 22:03:27 167ms
- GET /owners/{ownerId}/edit
  2024-02-27, 22:03:15 52ms
- GET /owners/{ownerId}
  2024-02-27, 22:03:13 51ms
- GET /owners
  2024-02-27, 22:03:11 21ms
- GET /owners
  2024-02-27, 22:03:09 199ms
- GET /owners/find
  2024-02-27, 22:03:07 34ms
- GET /webjars/**
  2024-02-27, 22:03:05 4ms
- GET /**
  2024-02-27, 22:03:05 3ms
- GET /**
  2024-02-27, 22:03:05 7ms
- GET /**
  2024-02-27, 22:03:05 7ms
- GET /**
  2024-02-27, 22:03:05 3ms

POST /owners/{ownerId}/pets/{petId}/visits/new

Trace ID: f105793e1f39dbd6c84dac361992a4cf

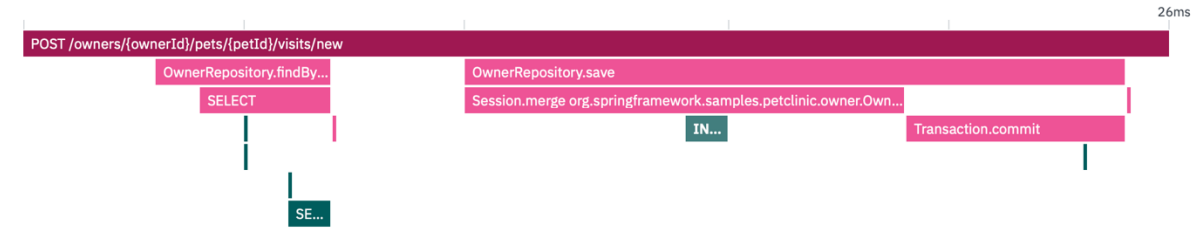
Download | Analyze Calls of this Trace

Sub Calls	Erroneous Calls	Error Logs	Warn Logs	Duration
14	0	0	0	26ms

Timeline

Colorize by Endpoint **Technology**

Started: 2024-02-27, 22:03:40



Calls

Colorize by Endpoint **Technology**

Started: 2024-02-27, 22:03:40 — Root call latency 26ms

26ms

- POST /owners/{ownerId}/pets/{petId}/visits/n... HTTP **26ms**
  To POST /owners/{ownerId}/pets/{petId}/visits/new of s390x-tutorial
  - OwnerRepository.findById INTERNAL 4ms
    In POST /owners/{ownerId}/pets/{petId}/visits/new of s390x-tutorial Inherited from POST /owners/{ownerId}/pets/{petId}/visits/new
  - OwnerRepository.save INTERNAL **15ms**
    In POST /owners/{ownerId}/pets/{petId}/visits/new of s390x-tutorial Inherited from POST /owners/{ownerId}/pets/{petId}/visits/new
    - Session.merge org.springframew... INTERNAL **10ms**
      In POST /owners/{ownerId}/pets/{petId}/visits/new of s390x-tutorial Inherited from POST /owners/{ownerId}/pets/{petId}/visits/new
      - **INSERT d5421aef-a345-4b3... DATABASE** **1ms**
        To INSERT of d5421aef-a345-4b3c-a9f1-7a543c0d7351
  - Transaction.commit INTERNAL 5ms

INSERT d5421aef-a345-4b3c-a9f1-7a543c0d... DATABASE

SOURCE s390x-tutorial

Details

Type	Call
Category	generic
Service	s390x-tutorial
Operation	INSERT d5421aef-a345-4b3c-a9f1-7a543c0d7351.visits
Tags	

Copy

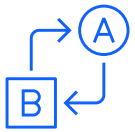

```
{
  "db.connection_string": "h2:mem:",
  "thread.name": "http-nio-8080-exec-8",
  "db.operation": "INSERT",
  "db.name": "d5421aef-a345-4b3c-a9f1-7a543c0d7351",
  "db.user": "sa",
  "db.sql.table": "visits",
  "db.system": "h2",
  "thread.id": "43",
  "db.statement": "insert into visits (visit_date,description,id) vi
}
```


Resource

Copy


```
{
  "process.runtime.name": "OpenJDK Runtime Environment",
  "telemetry.distro.name": "opentelemetry-java-instrumentation",
  "telemetry.sdk.language": "java",
  "telemetry.distro.version": "2.1.0",
  "process.runtime.version": "17.0.9+9-Ubuntu-122.04",
  "os.type": "linux",
  "service.name": "s390x-tutorial",
  "process.runtime.description": "Private Build OpenJDK 64-Bit Serv
  "process.executable.path": "/usr/lib/jvm/java-17-openjdk-s390x/bi
  "process.pid": "1119360",
  "telemetry.sdk.version": "1.35.0",
  "process.command_args": [
    "/usr/lib/jvm/java-17-openjdk-s390x/bin/java",
    "-javaagent:./otelagent.jar",
    "-Dotel.service.name=s390x-tutorial",
    "-jar",
    "./target/spring-petclinic-3.2.0-SNAPSHOT.jar"
  ],
  "host.name": "vsi-s390x-01",
  "os.description": "Linux 5.15.0-73-generic",
  "telemetry.sdk.name": "opentelemetry",
  "host.arch": "s390x",
```

The default propagators of OpenTelemetry are the W3C Trace Context and Baggage.



Traceparent Header

Enables correlation of requests across service boundaries.

Composed of version, trace ID, parent ID, and additional trace flags.

Sampled: if 1, trace is sampled and should be collected.

Sampled is set by vendors and tracing systems as part of their sampling strategy.

If a traceparent is marked as sampled, the spans of this trace are to be recorded.

Tracestate Header

Allows for vendor-specific or system-specific trace context to be propagated across services.

Tracestate is used to propagate sampling-related metadata to down-stream services and across multiple tracing systems to ensure a consistency in the recording of traces.

Tracestate key-value pairs are recorded in all the spans emitted as part of the trace.

Tracestate is always truncated at 512 bytes.

Baggage

Allows propagation of arbitrary key-value pairs across services.

Baggage attributes are not automatically added to span attributes.

Allows for much larger quantities of data to be attached to trace, if required.

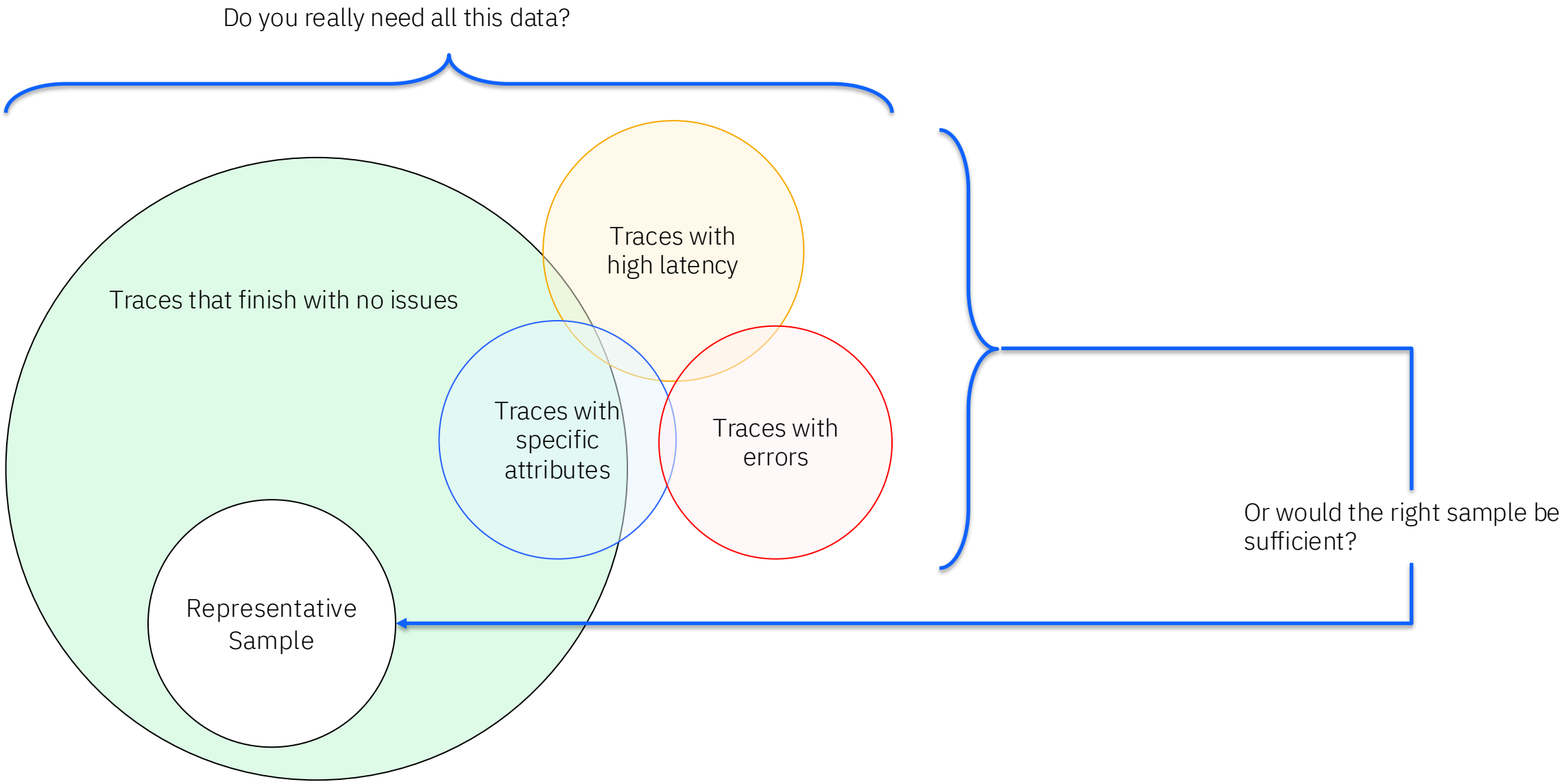
If you need baggage information in spans, this is usually done explicitly in your application code by reading attributes from the baggage and adding them to the span.

OpenTelemetry Trace Format

Resource Attributes	Value
host.name	vsi-s390x-01
host.arch	s390x
os.type	linux
os.description	Linux 5.15.0-113-generic
process.command_args	/usr/lib/jvm/<...>/bin/java -javaagent:./otelagent.jar Client
process.executable.path	/usr/lib/jvm/java-17-openjdk-s390x/bin/java
process.pid	27349
process.runtime. description	Ubuntu OpenJDK 64-Bit Server VM 17.0.12+7-Ubuntu-1ubuntu222.04
process.runtime.name	OpenJDK Runtime Environment
process.runtime.version	17.0.12+7-Ubuntu-1ubuntu222.04
service.instance.id	b50f550b-347a-4d1f-ae20-753efcd3a28
service.name	http_demo:java

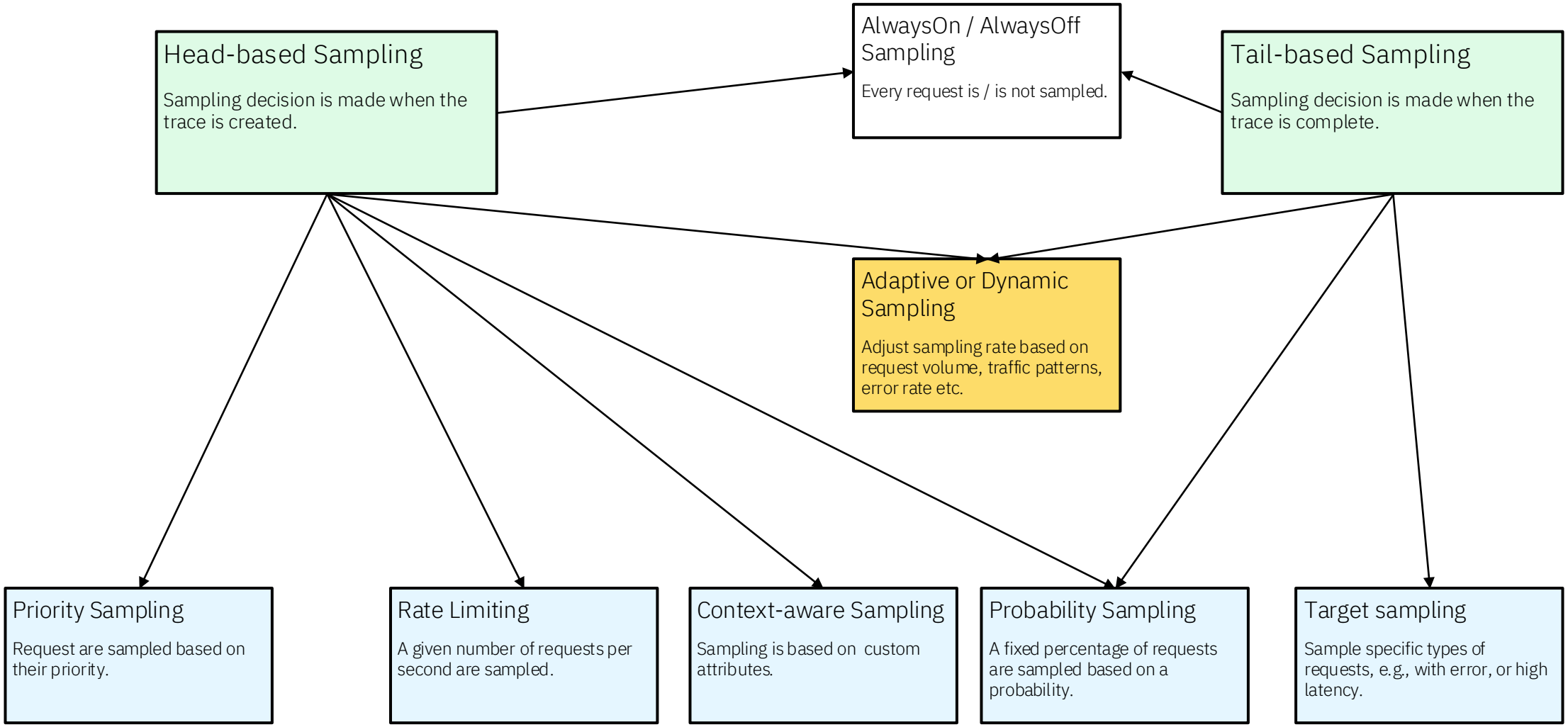
Span Attributes	Value
server.port	8000
server.address	localhost
url.full	http://localhost:8000/span-demo/
http.response.status_ code	200
http.request.method	GET
network.protocol. version	1.1
thread.name	Main
thread.id	1

OpenTelemetry Sampling



Source: OpenTelemetry - Sampling

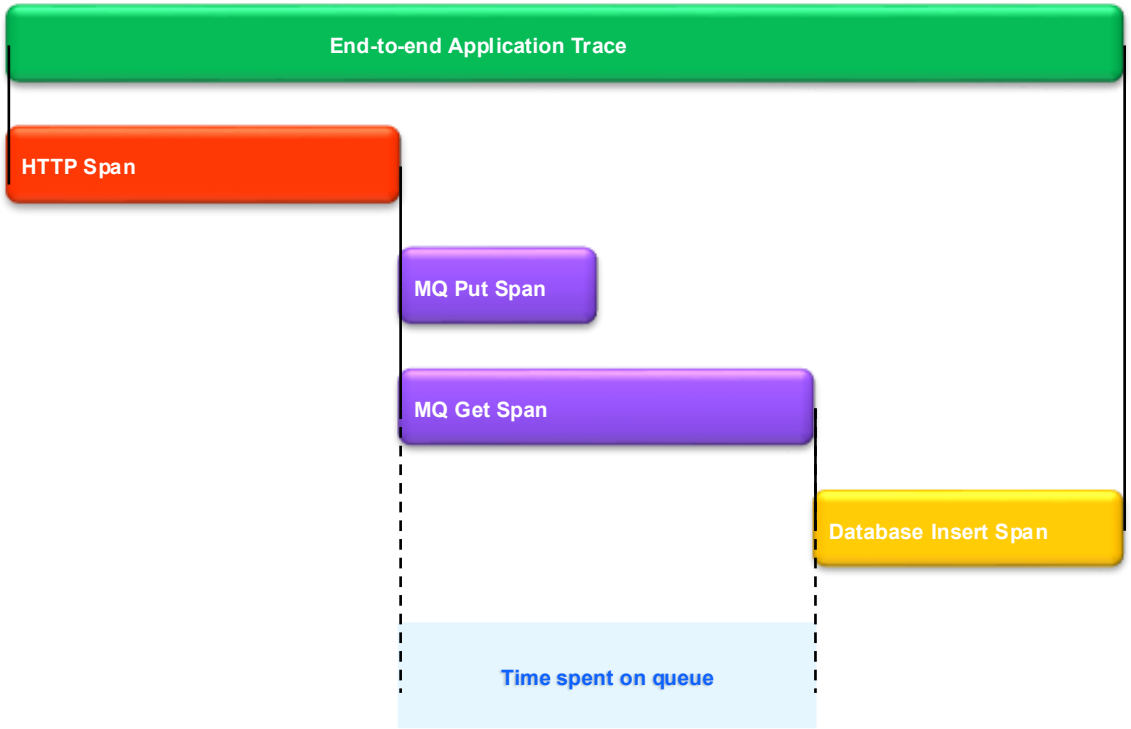
Trace sampling techniques



OpenTelemetry on Distributed MQ

Trace Support

- MQ 9.3.5 introduced support for Otel traces on Linux, AIX, and Windows
- This is provided in the form of an API exit which was developed by the Instana team
- Once installed, the exit will emit spans for MQPUT/MQGET calls
- Spans can be exported to Instana, or any other observability solution that understands the Otel spec (Jaeger, Prometheus, Grafana...)
- MQ Appliance introduced support in MQ 9.4 CD



Setting up the exit

- Download the exit from https://ibm.biz/mqinstanaexit
- Extract on queue manager machine
- Add to qm.ini or mqs.ini
- Configure the exit
- Start your queue manager
- Do some messaging!

```
ApiExitLocal:
   Sequence=100
   Function=EntryPoint
   Module=/var/mqm/exits64/mqtracingexit
   Name=TracingApiExit
```

```
LOG_LEVEL="debug"
SPAN_FORMAT="otel"
MONITOR_LEVEL="debug"
#IBMMQ_DEST_MONITOR_LEVEL_OFF = ""
#IBMMQ_DEST_MONITOR_LEVEL_QUIET = ""
#IBMMQ_DEST_MONITOR_LEVEL_NORMAL = ""
IBMMQ_DEST_MONITOR_LEVEL_DEBUG = "*"
IBMMQ_PUBSUB_SUPPORT = "on"

OTLP_EXPORTER_GRPC_ENDPOINT = "localhost:4317"
```

Implications of OTel

- OTel trace relies on the traceparent and tracestate headers
- Instana sets both, but only traceparent is mandatory
- Visible to applications, hence potentially problematic if your application can’t deal with RFH2 headers or doesn’t use the message property APIS
- PROPCTL(NONE) is a solution to this, but results in broken traces, and therefore reduced observability
- The exit also provides a number of options as to when to add OTel headers to a message: off (never), quiet (never, but still traced), normal (if already has headers), debug (always)

```
JMSMessage class: jms_text
JMSType:          null
JMSDeliveryMode:  1
JMSDeliveryDelay: 0
JMSDeliveryTime:  0
JMSExpiration:    0
JMSPriority:      6
JMSMessageID:     ID:414d5120465952452e4d51312020202031e81968012c0040
JMSTimestamp:     1746535936585
JMSCorrelationID: null
JMSDestination:   queue:///OTELQ
JMSReplyTo:       null
JMSRedelivered:   false
  JMSXAppID: gse.SimplePut
  JMSXDeliveryCount: 1
  JMSXUserID: root
  JMS_IBM_Character_Set: UTF-8
  JMS_IBM_Encoding: 273
  JMS_IBM_Format: MQSTR
  JMS_IBM_MsgType: 8
  JMS_IBM_PutApplType: 28
  JMS_IBM_PutDate: 20250506
  JMS_IBM_PutTime: 12520500
  traceparent: 00-00000000000000092203a247e2c01f8-977e9d5a4271f475-01
  tracestate: in=00000000000000092203a247e2c01f8;977e9d5a4271f475
```

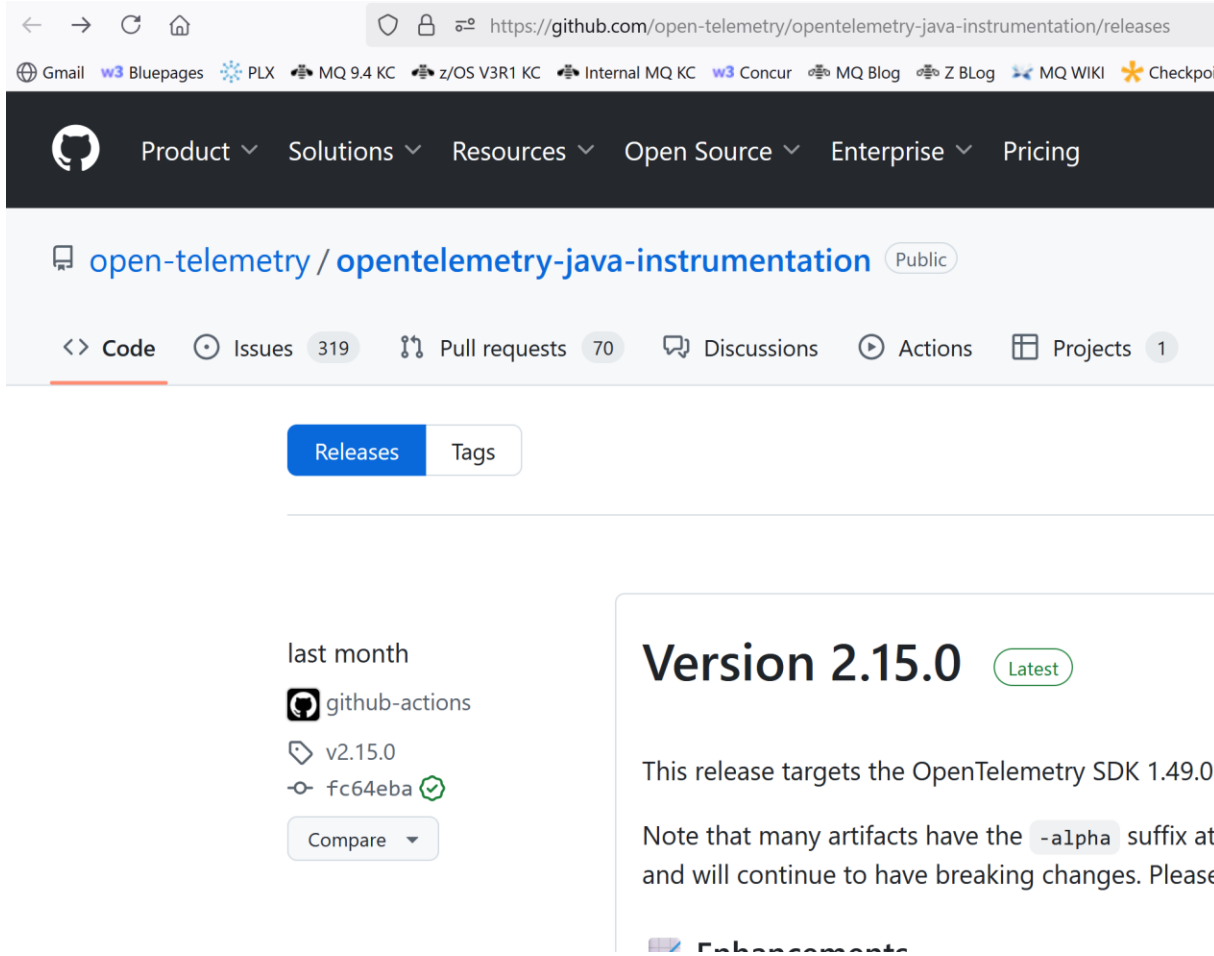
Instrumentation

For OTel to be useful you need to be able to propagate the trace through your application from end to end, and to have each piece of the application emit spans

The Instana exit is just one part of your application

You can manually instrument each component of your application, or there are lots of open-source tooling that will do this for you for a wide variety of languages and frameworks. A good example of this is for HTTP requests which are easy to automatically instrument

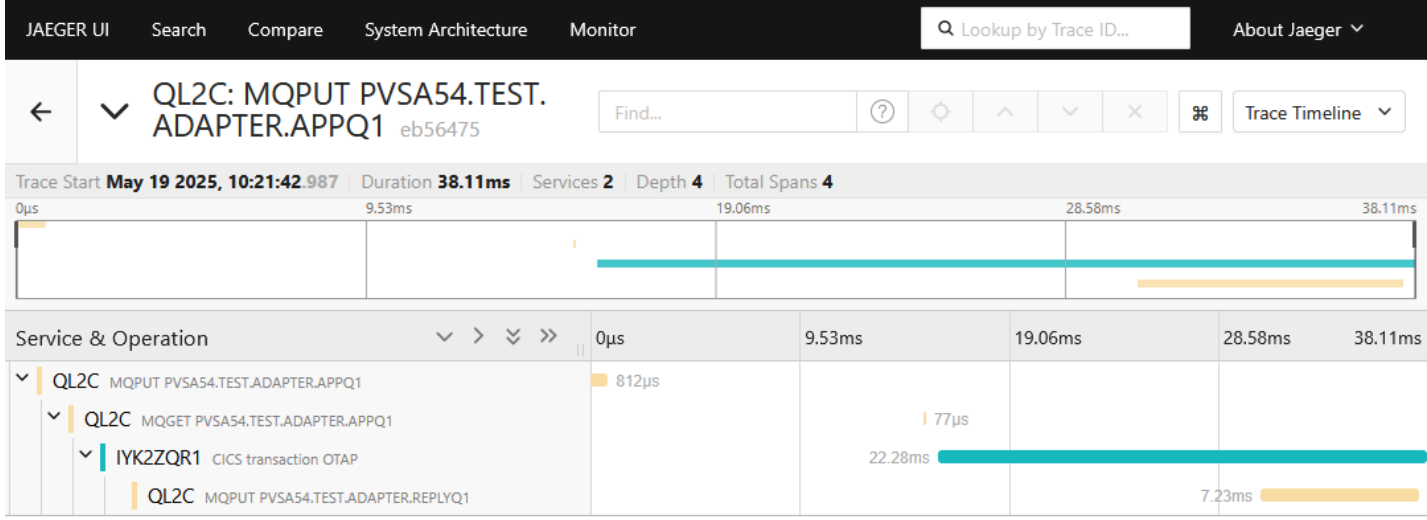
But this is also possible for messaging. For example there is instrumentation support for JMS



Coming back to z/OS...

IBM MQ for z/OS

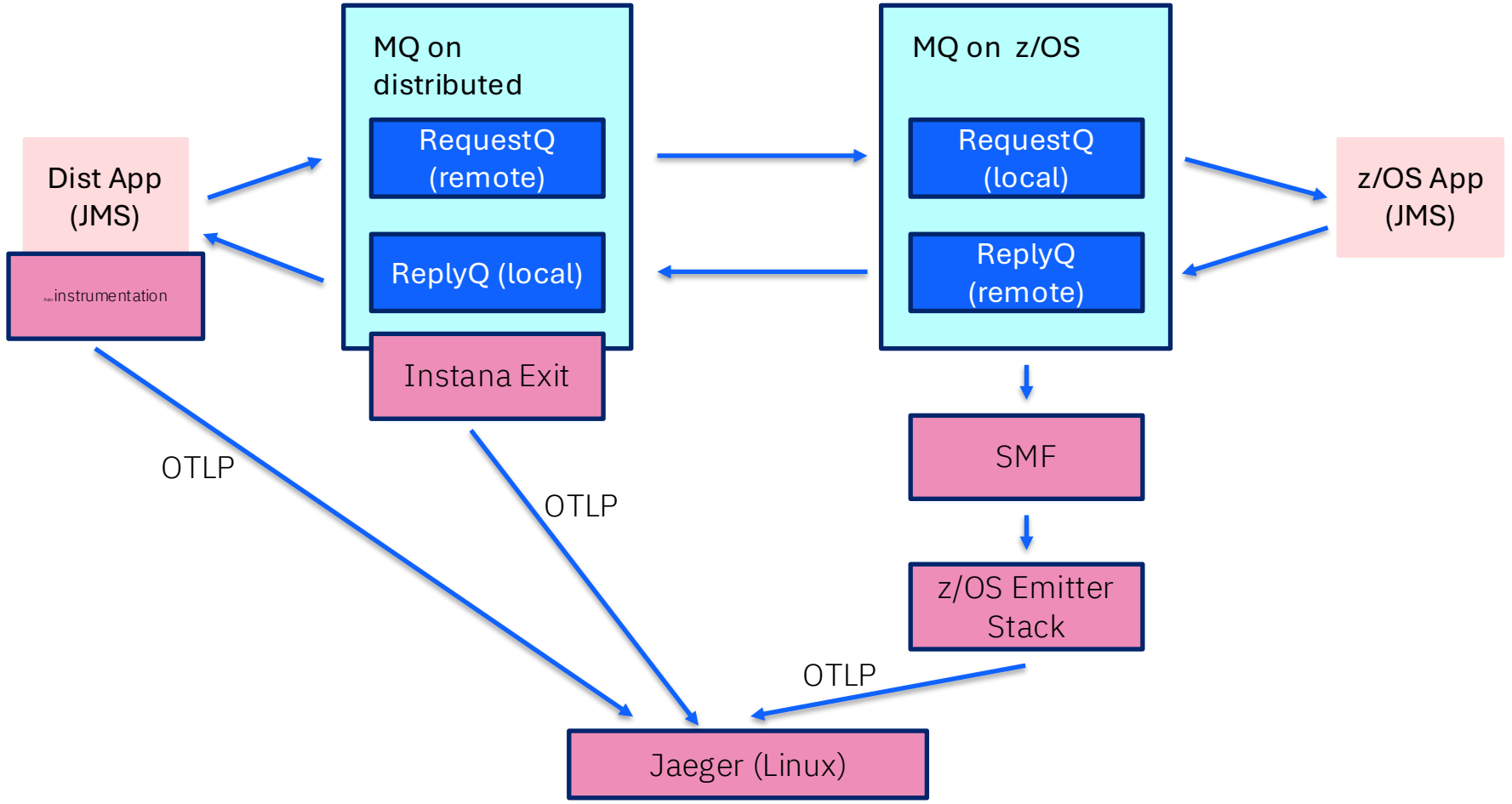
- From 9.4.3:
  - z/OS queue managers can take part in traces and emit spans for point-to-point messaging scenarios.
  - Traces can be propagated to and from distributed queue managers.
  - Applications can manually instrument messages by setting message properties or using open-source instrumentation (i.e. JMS).
  - Automatic context propagation CICS TS 6.3 without applications needing to be able to tolerate message properties.
- From 9.4.5:
  - Queue managers can participate in traces for pub/sub scenarios
  - Automatic context propagation with IMS via the IMS Bridge without applications needing to be able to tolerate message properties




```
!QL2C ALT QMGR OTELTRAC(ON) OTELPCTL(AUTO)

!QL2C ALT QL(MY.Q) OTELTRAC(ON) OTELPCTL(AUTO)
```

A live demo

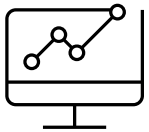


Wider support

Introducing **IBM Z Observability Connect**

Formerly IBM Z APM Connect

Demand for OpenTelemetry on IBM Z is growing, success is dependent simplicity of adoption

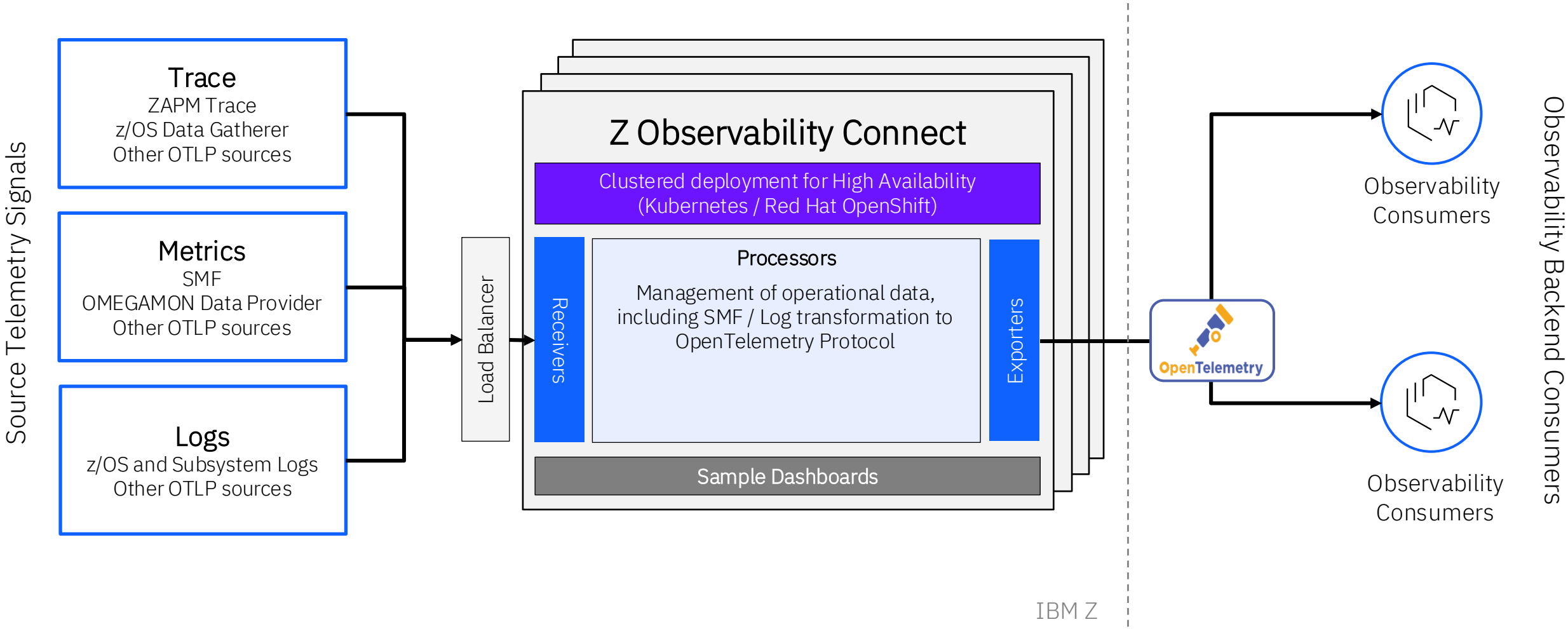


Observability platforms look to integrate trace, metrics and log signal types to provide end-to-end visibility

Clients are concerned about cost and management of enabling OpenTelemetry across critical z/OS resources

IBM Z Observability Connect is positioned to be the single data aggregator for mainframe operational data and deliver it to multiple consumers for observability of application and resources. It adds value by managing, filtering and enriching the source data for interchangeable consumers

Introducing Z Observability Connect

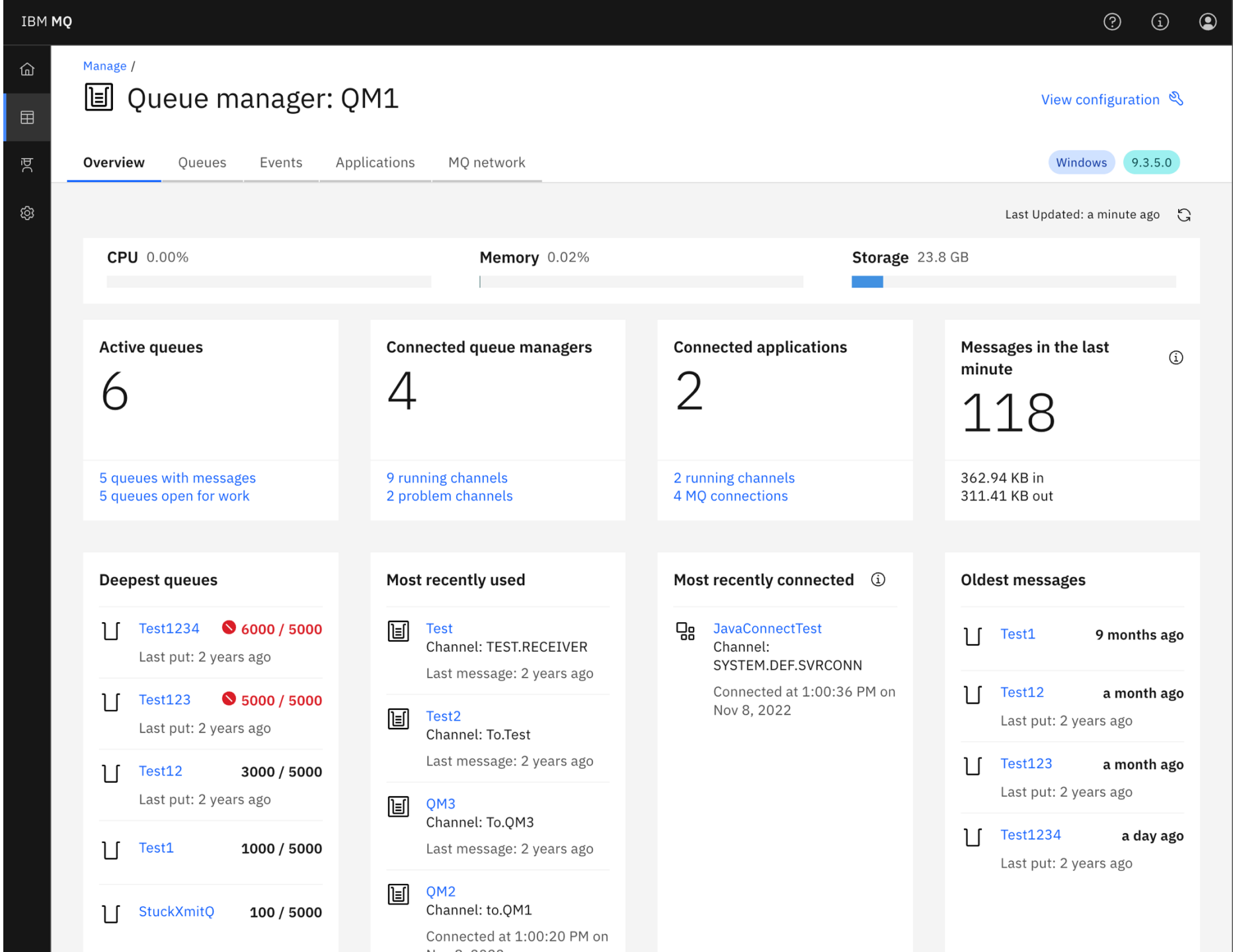


The MQ Console

MQ Console Dashboards

The IBM MQ Console allows common administrative tasks to be performed through a graphical user interface.

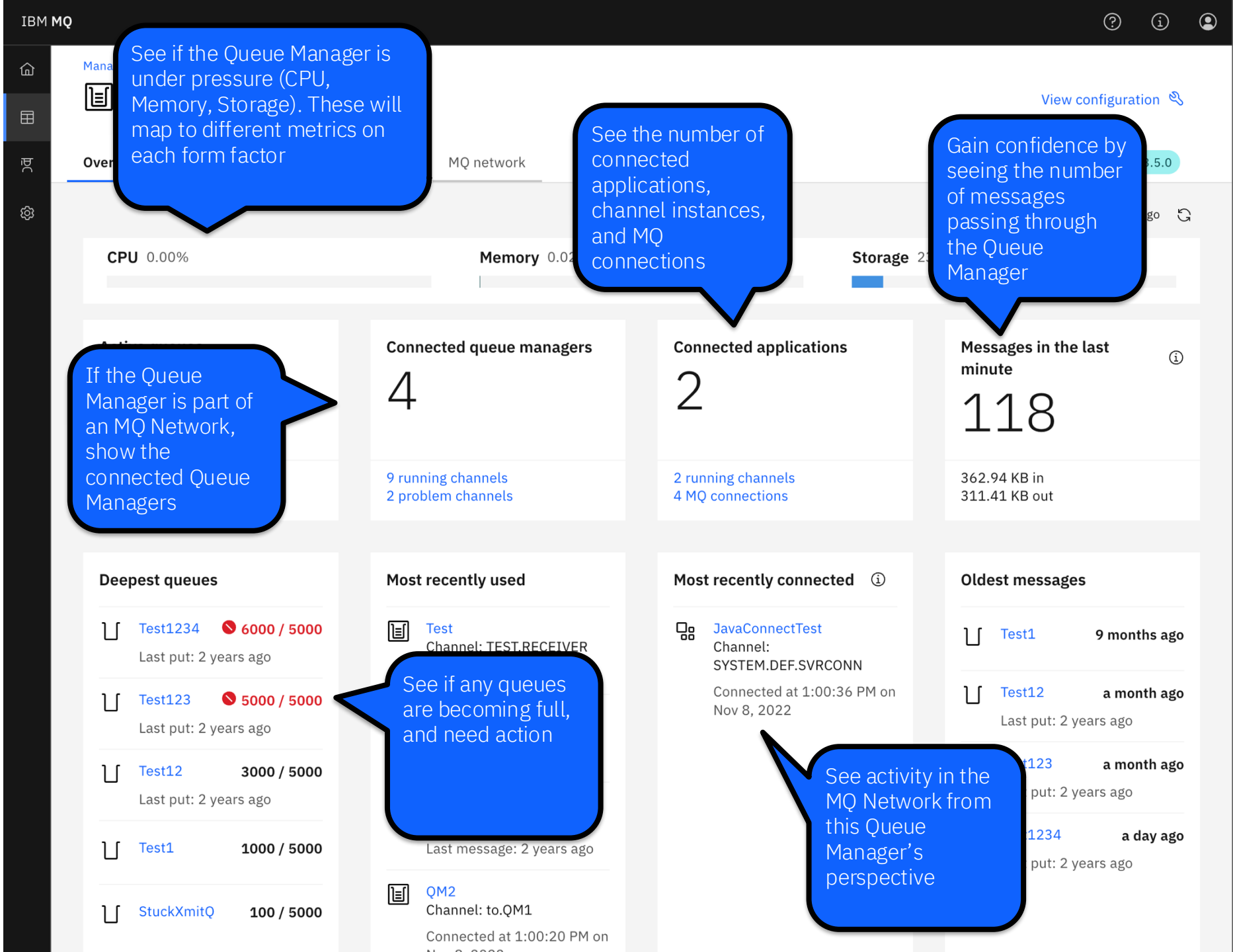
The Overview tab eliminates the manual look-up of key information so you can quickly understand the state of your queue manager, and act on emerging issues before they become problems.



MQ Console Dashboards

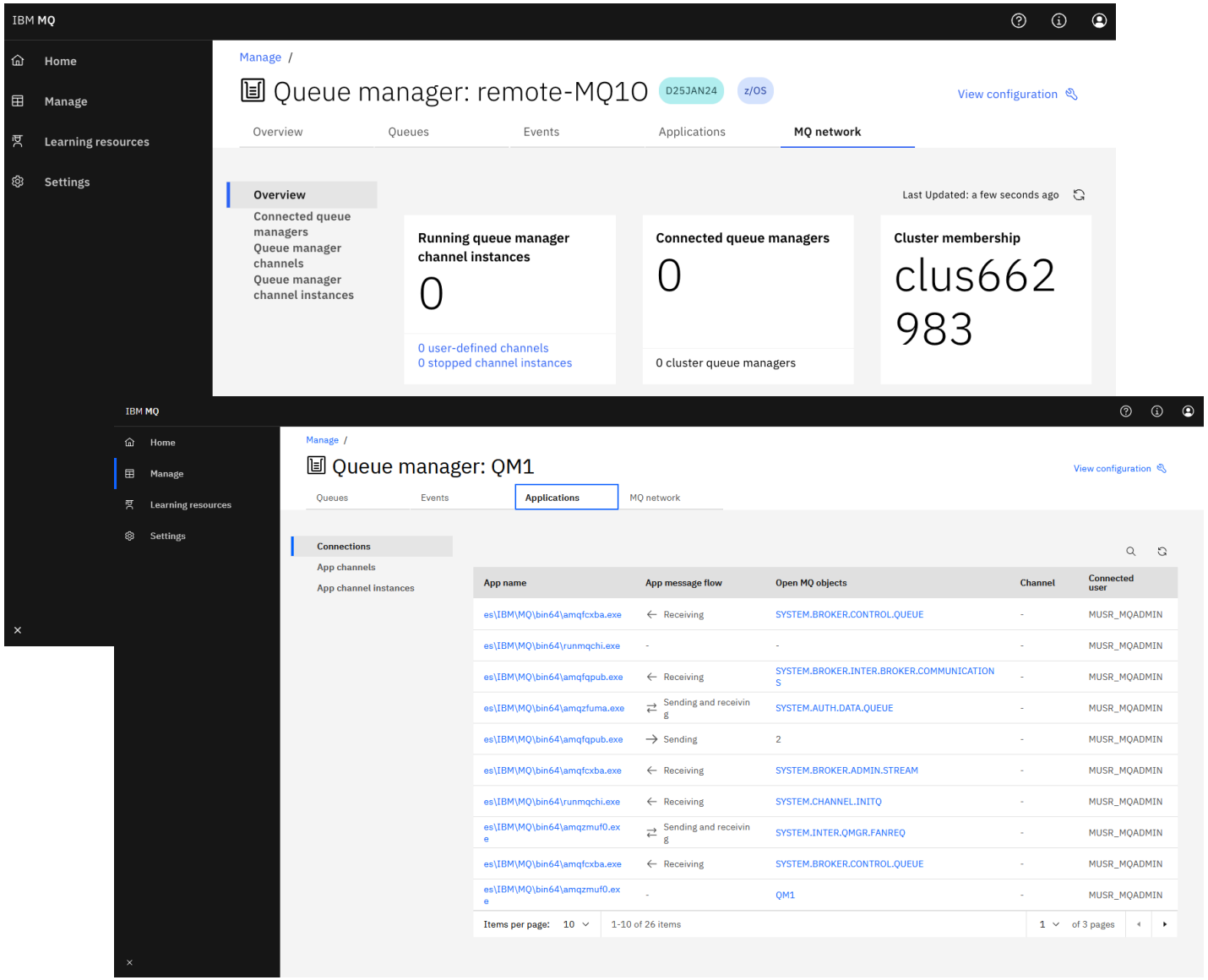
The IBM MQ Console allows common administrative tasks to be performed through a graphical user interface.

The Overview tab eliminates the manual look-up of key information so you can quickly understand the state of your queue manager, and act on emerging issues before they become problems.



MQ Console Dashboards

- The new Applications and MQ network tabs give you a quick view of the connected applications, and other connections in your MQ network.
- Easier for admins and developers to perform real-time analysis of:
  - Application connectivity
  - Queue manager to queue manager networking
- Reduce the time to resolution of issues
- Allows less skilled users to carry out analysis



Q&A