Monitoring and Tracing Applications in the Cloud with OpenCensus

Agenda

1. Introduction
    a. What is OpenCensus?
    b. What is the value of OpenCensus?
2. OpenCensus concepts
    a. Monitoring Metrics
    b. Tracing
    c. Exporters
    d. Agent
3. Monitoring and tracing ecosystem
    a. How does OpenCensus relate to Stackdriver?
    b. How does OpenCensus relate to other open source projects?
4. Workflow in using OpenCensus to debug issues
    a. Demo

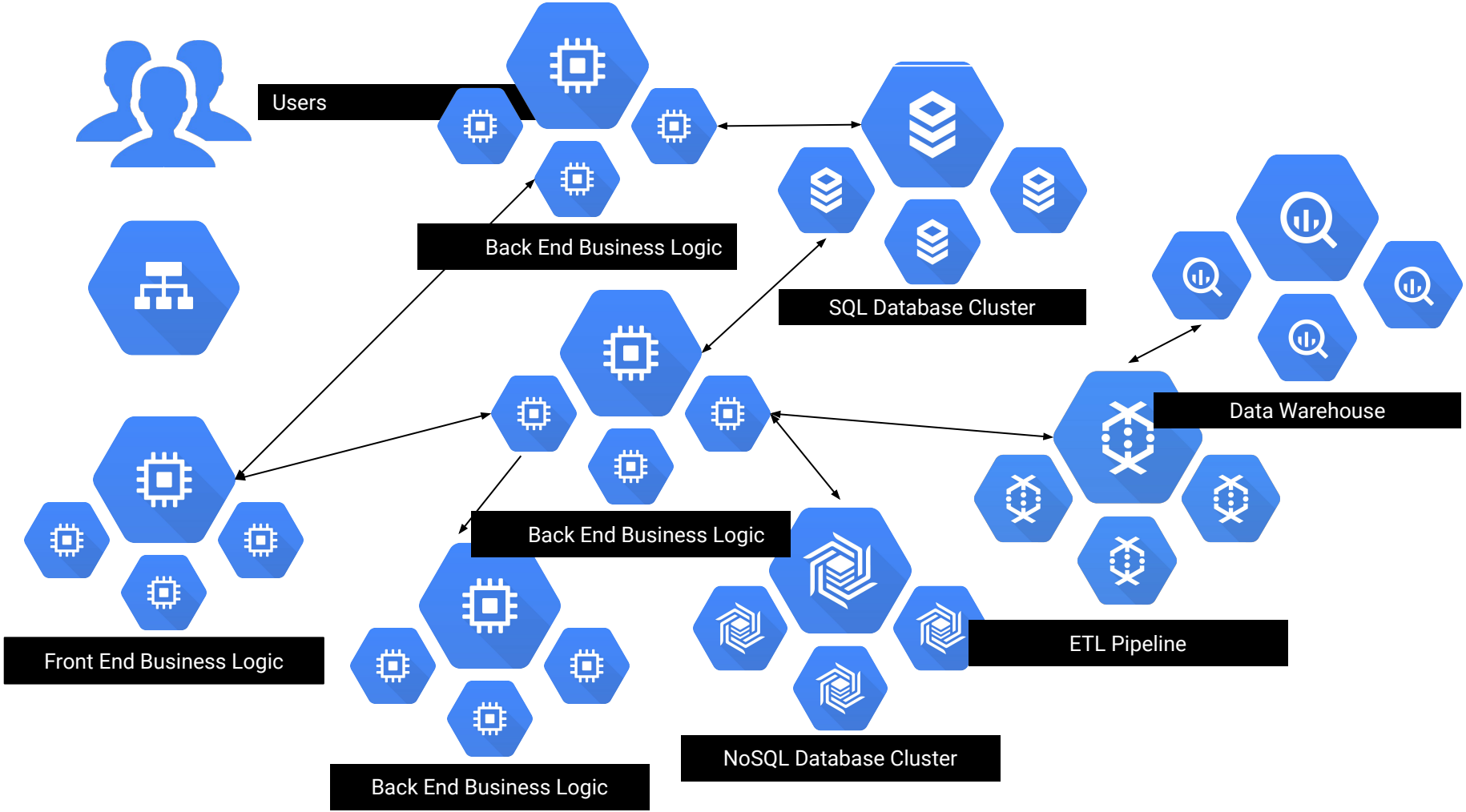
Introduction

What is OpenCensus?

OpenCensus is a set of libraries to collect application specific metrics and distributed trace data and export it to backend monitoring and trace systems.

OpenCensus supports a number of languages, including C#, C++, Go, Java, JavaScript, and Python (see Quickstarts)

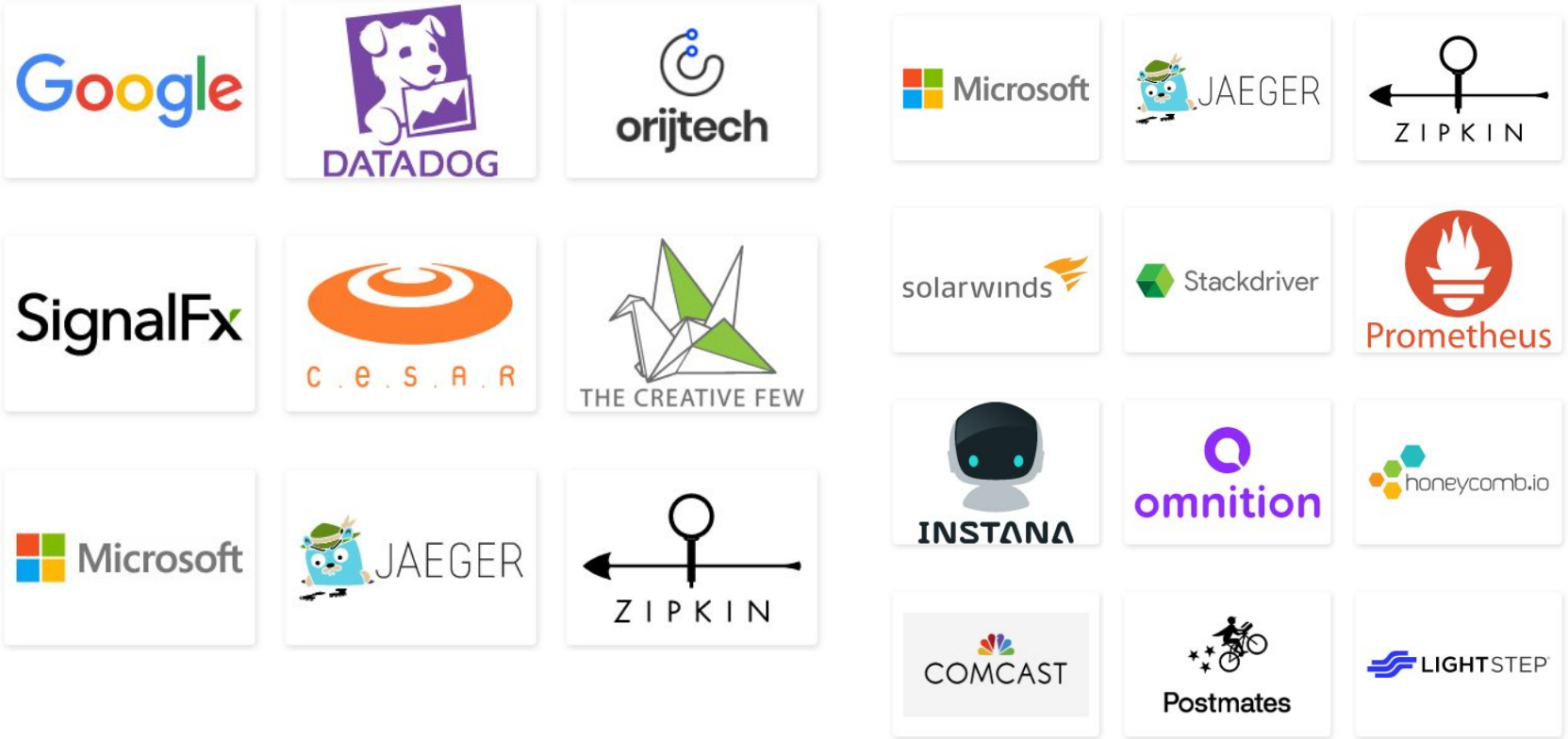
OpenCensus can export data to backends, including Stackdriver, Prometheus, Zipkin, AWS, and Azure


Users
Back End Business Logic
SQL Database Cluster
Data Warehouse
Front End Business Logic
Back End Business Logic
ETL Pipeline
Back End Business Logic
NoSQL Database Cluster

What is the value of OpenCensus?

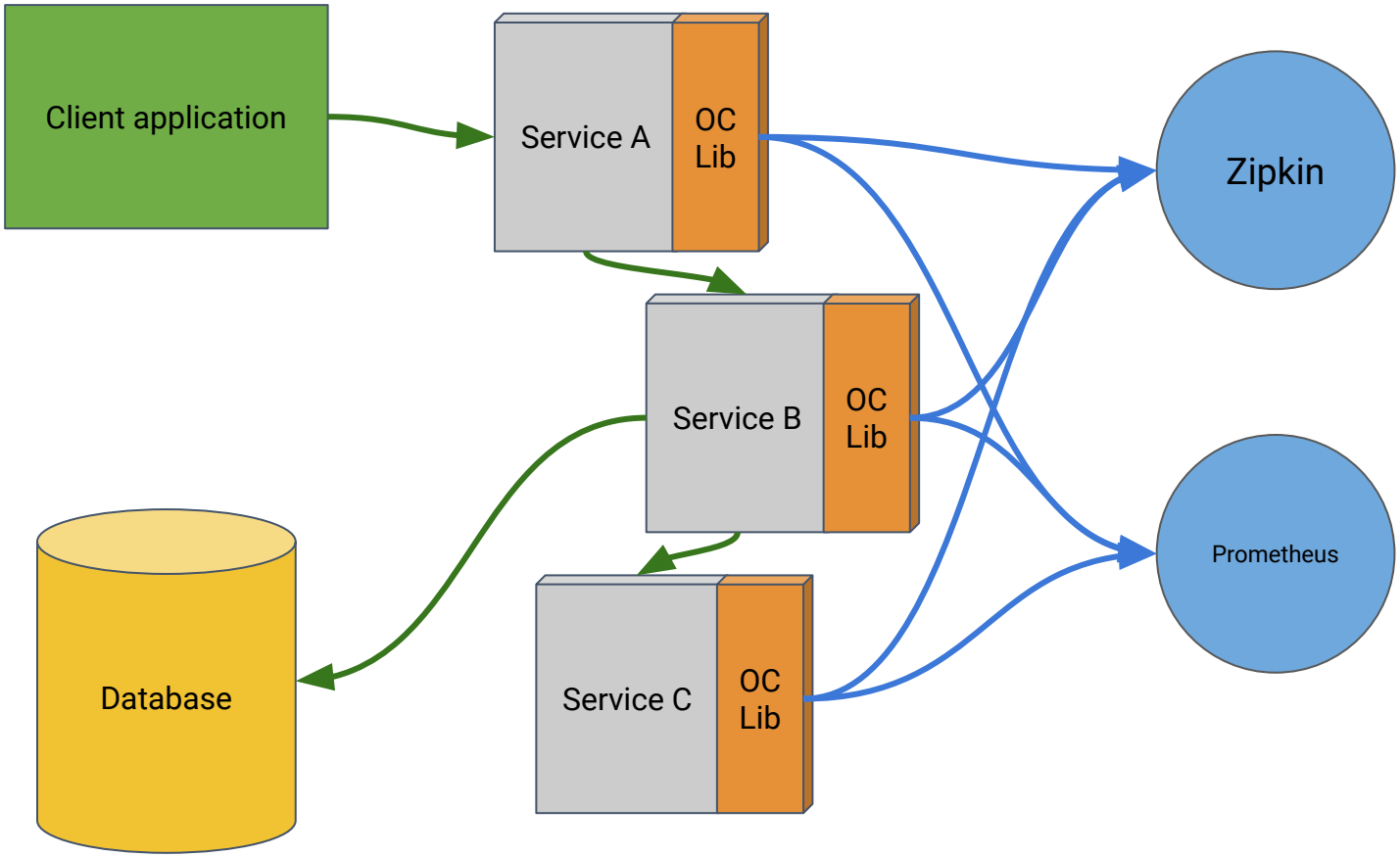
- Enables customers to monitor the health of distributed applications
- Enables customers to identify sources of latency
- It is an open source project that enables customers to be independent of particular cloud providers
- Integrates well with GCP, including Stackdriver, client libraries, and GCP services
- OpenCensus includes integrations with many frameworks, such as HTTP, gRPC, databases, and GCP APIs, so customers do not have to write code for that.
- Has an open source ecosystem beyond Google
- Like a public road linking a network of proprietary monitoring and tracing fiefdoms

Ecosystem


Google
DATADOG
orijtech
SignalFx
C.e.S.A.R
THE CREATIVE FEW
Microsoft
JAEGER
ZIPKIN
Microsoft
JAEGER
ZIPKIN
solarwinds
Stackdriver
Prometheus
INSTANA
omnition
honeycomb.io
COMCAST
Postmates
LIGHTSTEP

OpenCensus Concepts

Concepts

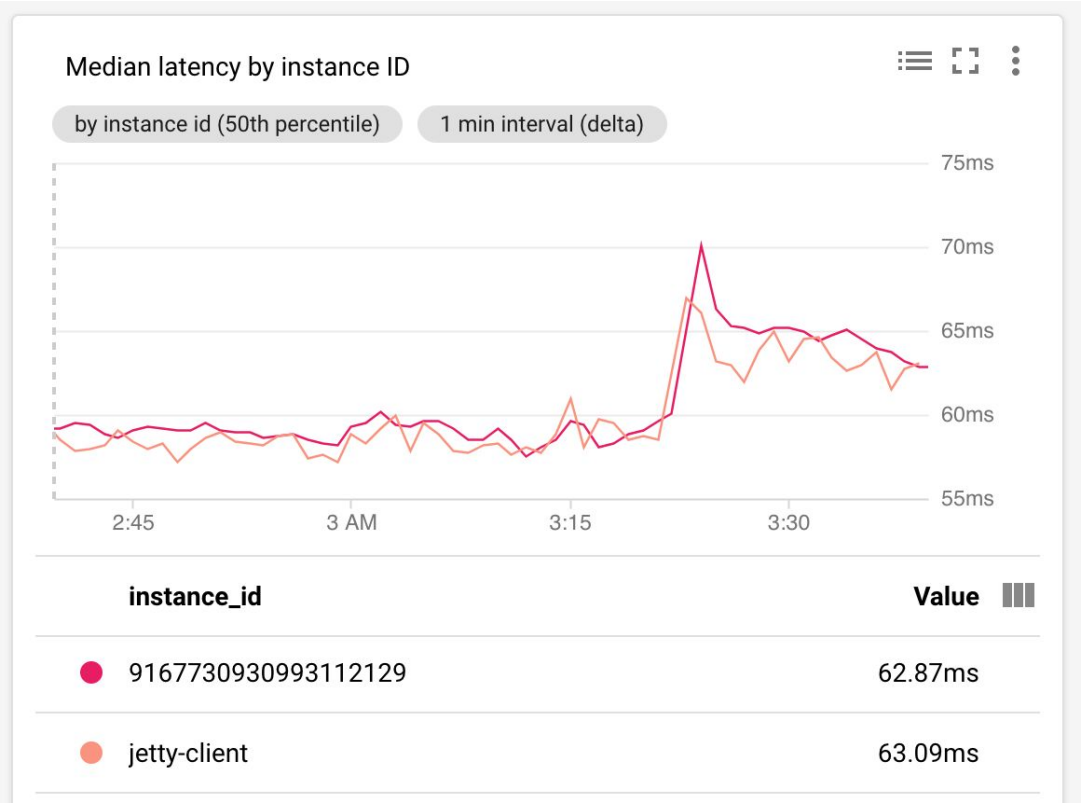

Client application
Service A
OC Lib
Service B
OC Lib
Service C
OC Lib
Database
Zipkin
Prometheus

Monitoring Metrics

OpenCensus enables customers to define application specific metrics that are meaningful to them.

- This is in contrast to generic metrics, like for the performance of cloud services.
- Example: latency as measured from mobile clients to the application backend vs the latency of a database operation
- The application can be instrumented for metrics that reflect user experience and user engagement

Metrics in Stackdriver Monitoring



Metric Types

Metrics are based on Measurements

Measurements are raw values that may be either integer or real valued

Measurements are produced by Measures that define the source of the data with a name, description, and unit. For example,

- Name: client_latency
- Description: latency measured from a mobile client
- Unit: ms

Metrics aggregate a measure over a time period using Views (name, description, measure, tags, aggregation)

Aggregations can be counts, sums, or distributions

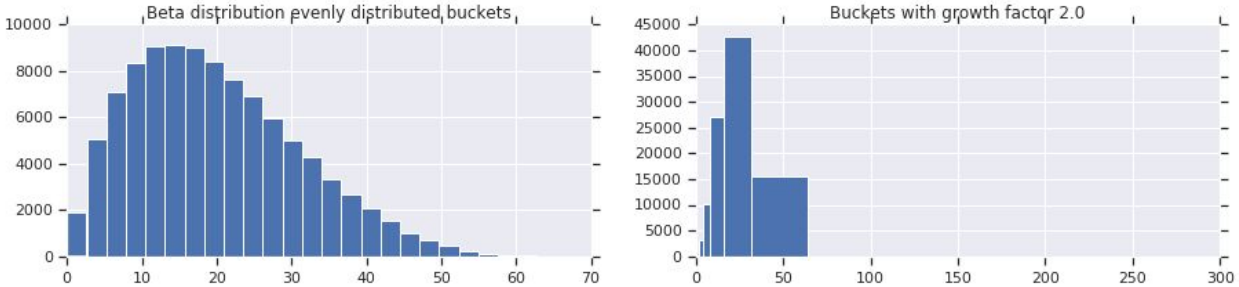
Distribution Metrics

Distributions aggregate measurements into histograms for each time period

The aggregation can be used to calculate median and tail values

Do not give the values for specific requests

Tail latency (eg, p99) may not be accurate - careful of bucket boundaries



Same data different bucket boundaries

Trace

Tracing tracks the progression of a single user request

Distributed tracing does this from clients to multiple servers

A span is one segment of a trace

You can add your own annotations to traces

Traces can be sampled for high traffic systems to reduce the amount of data to be managed

Stackdriver Trace

Exporters

Exporters send metrics and trace data to backend systems for storage, visualization, and alerting.

The exporters are libraries that you import into your application.

For integrations, all you need to do to instrument your application is to initialize the exporter libraries and register the views.

Exporter libraries may be more convenient than the agent, if the language x backend exporter exists

If there is no exporter for your backend then you can write your own custom exporter.

Service

The OpenCensus service enables sending of metrics and trace data to backends to an exporter in another process.

Advantages

- The exporters only need to be written once (in Go) for each backend, rather than for each backend x language combination (helps partners)
- More seamless integration in Kubernetes ecosystem using sidecars.

Disadvantage

- You need to manage a separate process

Monitoring and Tracing Ecosystem

How does OpenCensus relate to Stackdriver

OpenCensus collects data and sends it to Stackdriver or other backend

The OpenCensu APIs are suggested rather than direct us of Stackdriver collection APIs

Stackdriver also has APIs for dashboard creation and alerting.

If a customer has instrumented their application with OpenCensus then they can easily migrate to Stackdriver.

How does OpenCensus relate to other open source projects?

OpenCensus will merge with OpenTrace and be renamed to OpenTelemetry

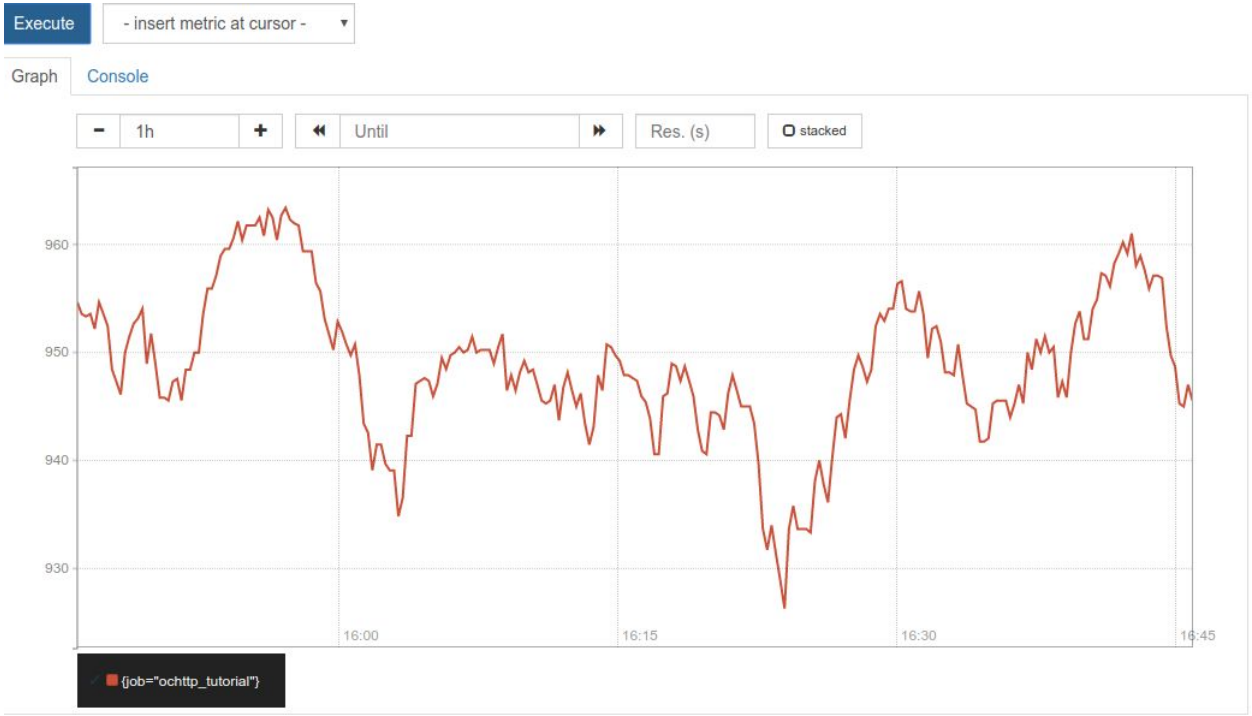
OpenCensus supports export to a number of metrics and tracing backends

- Metrics backends: Prometheus
- Tracing backends: Zipkin, Jaeger
- Also many partners and other clouds

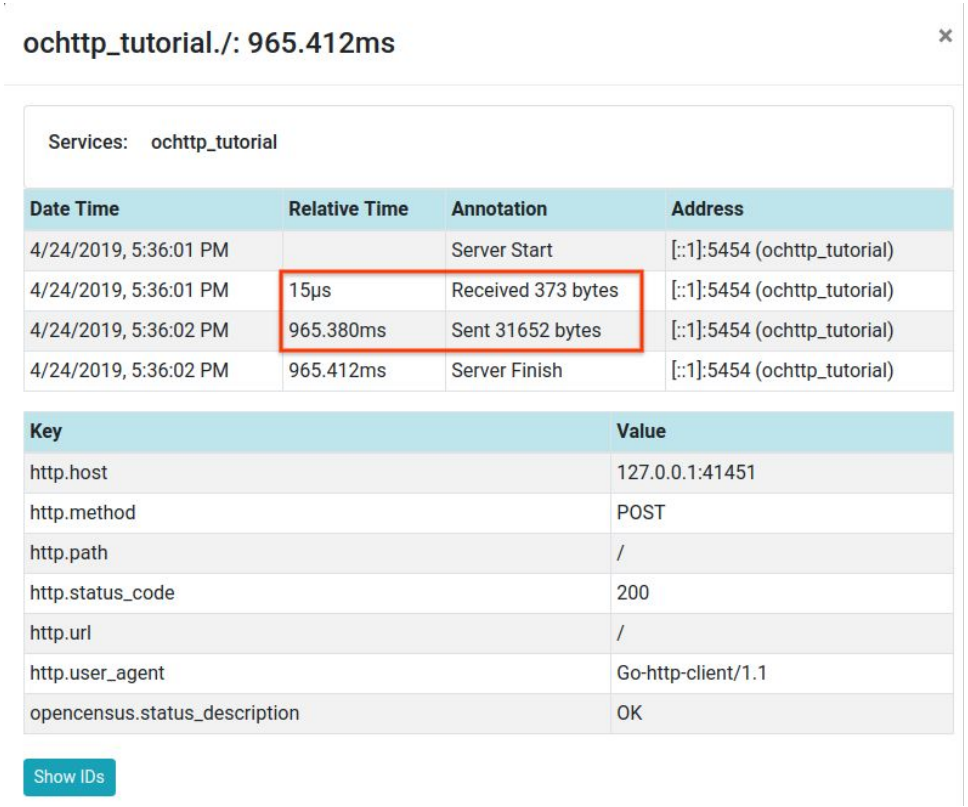
OpenCensus integrates with a number of open source projects

- Jetty, Go http, gRPC, Redis, Memcached, JDBC, MongoDB

Prometheus


Execute
- insert metric at cursor -
Graph
Console
1h
Until
Res. (s)
stacked
960
950
940
930
16:00
16:15
16:30
16:45
{job="ochttp_tutorial"}

Zipkin



ochttp_tutorial./: 965.412ms

Services: ochttp_tutorial

Date Time	Relative Time	Annotation	Address
4/24/2019, 5:36:01 PM		Server Start	[::1]:5454 (ochttp_tutorial)
4/24/2019, 5:36:01 PM	15µs	Received 373 bytes	[::1]:5454 (ochttp_tutorial)
4/24/2019, 5:36:02 PM	965.380ms	Sent 31652 bytes	[::1]:5454 (ochttp_tutorial)
4/24/2019, 5:36:02 PM	965.412ms	Server Finish	[::1]:5454 (ochttp_tutorial)

Key	Value
http.host	127.0.0.1:41451
http.method	POST
http.path	/
http.status_code	200
http.url	/
http.user_agent	Go-http-client/1.1
opencensus.status_description	OK

Show IDs

Workflow and Demo

Workflow

1. Instrument your application
    - Use the OpenCensus libraries
2. Set up metrics and tracing backends
    - Either Stackdriver, open source alternative, or 3rd party
3. Respond to alerts in an incident
    - Typically by integration with a paging system for condition that predicts an SLO violation that reflects impact to user experience
4. Assess the severity and scope of the incident
    - From an extensive set of dashboards showing the detailed health of your application
5. Identify the source of the problem
    - By drilling down into metrics, trace, and log information

Demo

Follow steps at https://opencensus.io/guides/debugging/incidents/

OpenTelemetry

OpenCensus and OpenTracing have merged into OpenTelemetry!

OpenTelemetry Demo: https://opentelemetry.io/community/demo/