

The background features a vibrant, abstract design with a color gradient from dark blue on the left to bright yellow and white on the right. The design consists of overlapping, wavy horizontal bands and a radial pattern of lines emanating from a bright white point on the right side, creating a sense of motion and energy.

CISCO *Live!*

Let's go



The bridge to possible

# ThousandEyes and OpenTelemetry

From Theory to Practice!

Hatam Shukur, Solutions Architect @ ThousandEyes (part of Cisco)

<https://www.linkedin.com/in/hatamshukur/>

# Raise your hand if you are ...

- using ThousandEyes in your environment
- running ThousandEyes and OpenTelemetry integration



# About me

- Based in Kraków, Poland
- CCIEx2 58771 (RS/SP)
- 11 years of experience in IT
- Been with Cisco for 7 years
- Solutions Architect @ ThousandEyes (part of Cisco)



# Agenda

- Introduction to OpenTelemetry
- OpenTelemetry and ThousandEyes
- Configuration steps
- Demo

# Introduction to OpenTelemetry



# What is OpenTelemetry ?

OpenTelemetry, also known as OTel for short

Set of standardized vendor-agnostic tools for ingesting, transforming, and sending data to an Observability back-end

Having a common format for how observability data is collected and sent is where OpenTelemetry comes into play

*Standards are like toothbrushes.  
Everybody wants one but nobody  
wants to use anybody else's.*

Connie Morella



# What was before OpenTelemetry?

## OpenTracing

- CNCF project
- Provided a vendor-neutral API for sending telemetry data over to an Observability back-end
- Relied on developers to implement their own libraries to meet the specification.



## OpenCensus

- Google Open Source community project
- Provided a set of language-specific libraries that developers could use to instrument their code



# OpenCensus + OpenTracing = OpenTelemetry



In May 2019 OpenCensus and OpenTracing were merged to form OpenTelemetry and became a CNCF project



The OpenTelemetry logo consists of four colored squares (yellow, blue, blue, yellow) arranged in a 2x2 grid, with the word "OpenTelemetry" in a sans-serif font to its right.

# OpenTelemetry main components

## OpenTelemetry Protocol (OTLP)

- specification describes the encoding, transport, and delivery mechanism of telemetry data between telemetry sources, intermediate nodes such as collectors, and telemetry backends.

## OpenTelemetry Collector

- offers a vendor-agnostic implementation on receiving, processing, and exporting telemetry data

## APIs and SDKs

- allows developers to easily integrate their applications

# OpenTelemetry and ThousandEyes



# ThousandEyes and OpenTelemetry

How does it work?

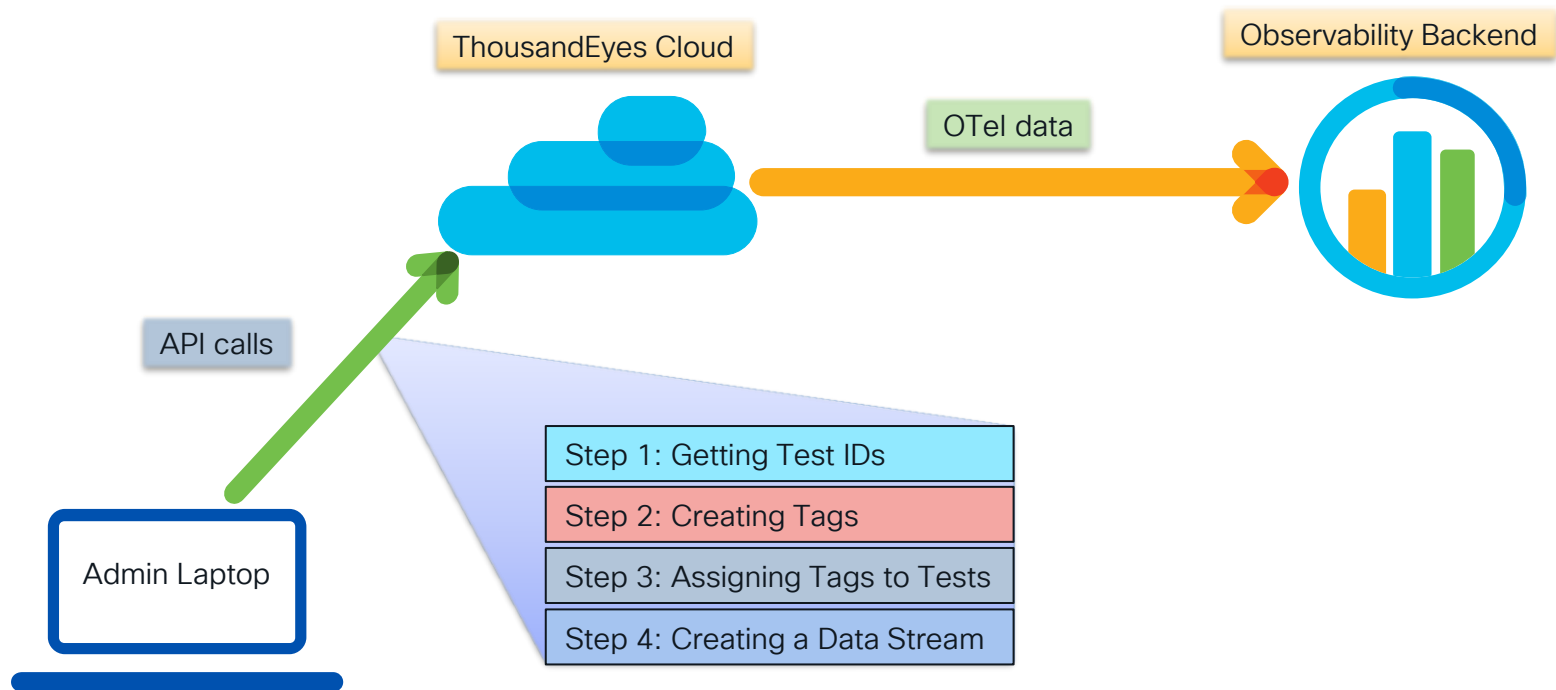
**Data streaming APIs** that you can use to configure and enable your ThousandEyes tests with OTel-compatible streams

**Collectors** actively fetch ThousandEyes test data, enrich the data with some additional detail, filter, and push the data to the customer-configured endpoints

**Third-party OTel collectors** that receive, transform, filter, and export different metrics to client applications such as AppD, or any other OTel-capable client configuration.

# ThousandEyes and OpenTelemetry

How does it work?





# Configuration steps

# Step 1: Getting Test IDs

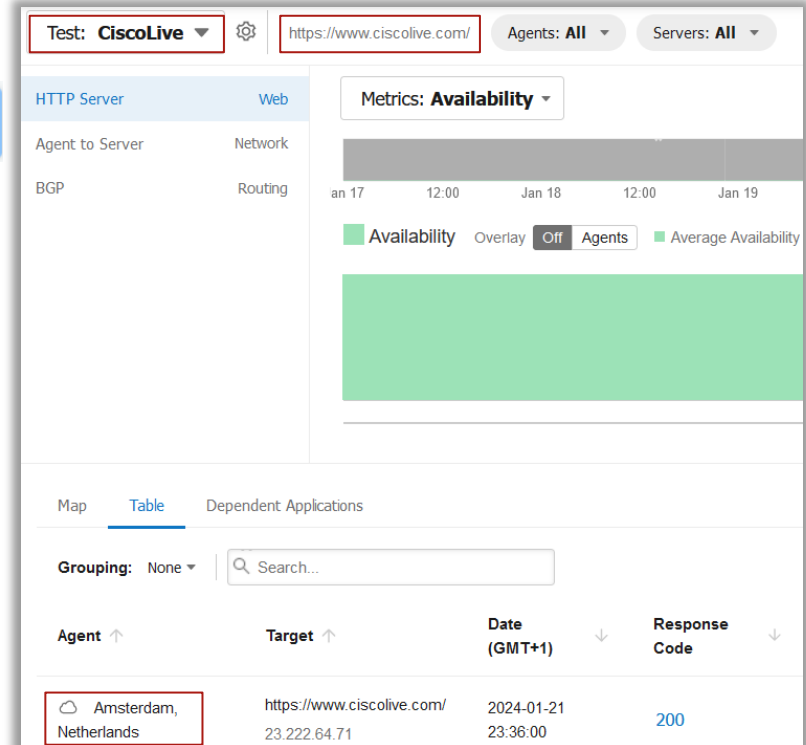


API call to get the test IDs:

**GET** `/v7/tests` List configured tests  

Response body:

```
"tests": [  
  {  
    "testId": "281474976744772",  
    "testName": "CiscoLive",  
    "type": "http-server",  
    "enabled": true,  
    "url": "https://www.ciscolive.com/"  
  }  
]
```









## Step 2: Creating Tags

API call to configure Tags:

**POST** `/v7/tags` Create tag  

Request body:

```
{  
  "key": "OTelKey1",  
  "value": "OTelValue1",  
  "objectType": "test",  
  "accessType": "all"  
}
```



Response body:

```
{  
  "id": "962e88bd-2426",  
  "key": "OTelKey",  
  "value": "OTelValue",  
}
```



## Step 3: Assigning Tags to Tests

API call to assign tags to tests:

**POST** `/v7/tags/assign` Assign multiple tags to multiple objects  

Request body:

```
"tags":
{
  "tagId": "962e88bd-2426",
  "assignments":
  {
    "id": "281474976744772",
    "type": "test"
  }
}
```

Response body:

```
"tags":
{
  "tagId": "962e88bd-2426",
  "assignments":
  {
    "id": "281474976744772",
    "type": "test"
  }
}
```

# Step 4: Creating a Data Stream

API call to create a data stream

POST

/v7/stream Create data stream



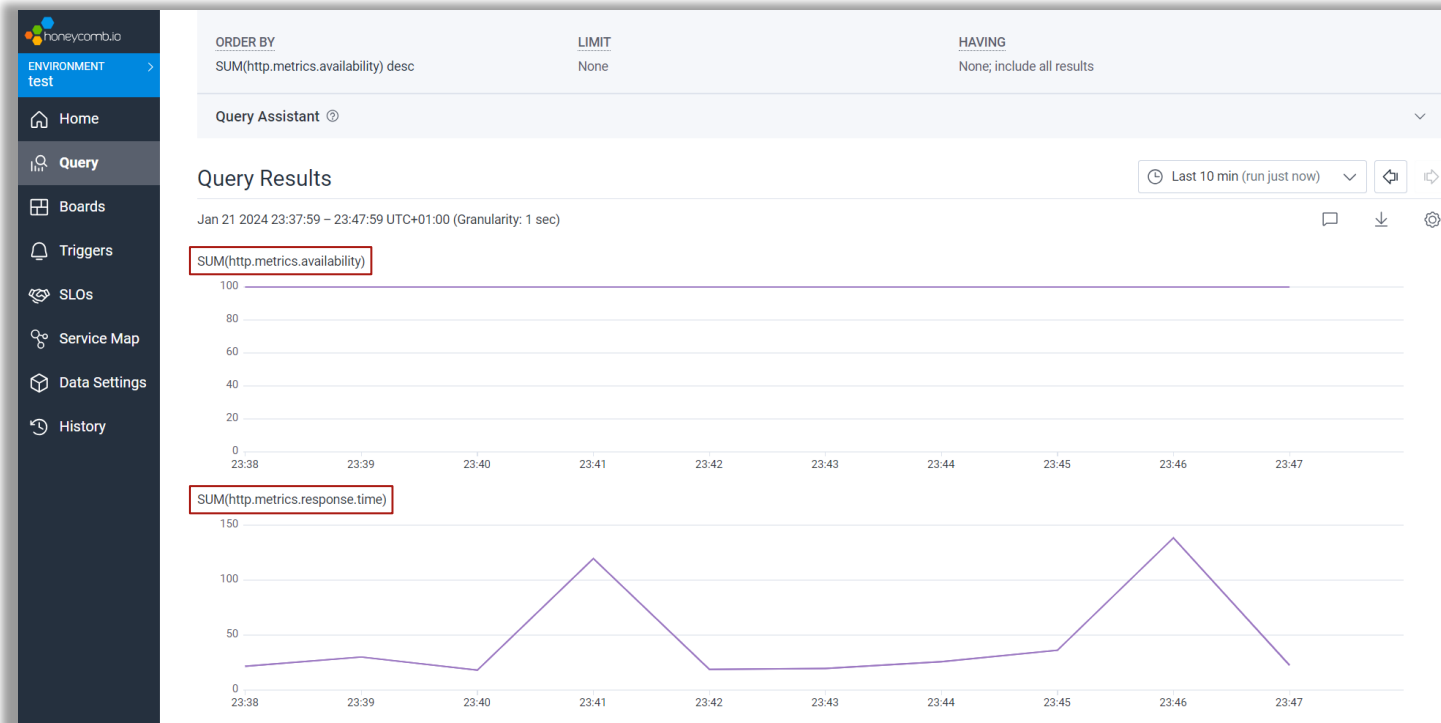
Request body:

```
{
  "type": "opentelemetry",
  "tagMatch": [{
    "key": "OTelKey1",
    "value": "OTelValue1",
    "objectType": "test"
  }],
  "endpointType": "grpc",
  "streamEndpointUrl": "https://api.honeycomb.io:443",
  "customHeaders" : {
    "x-honeycomb-team": "{{honeycomb_api_key}}",
    "x-honeycomb-dataset": "NEW-OTel-DataSet"
  }
}
```

Response body:

```
{
  "id": "90e91f2e-af16",
  "enabled": true,
  "type": "opentelemetry",
}
```

# Last step: Checking the Observability Backend





# Demo

CISCO *Live!*

# Continue to Learn, Code and Build with Cisco DevNet!

Get access to an exclusive learning module filled with digital learning opportunities on topics including **Full Stack Observability** and more.

Scan QR Code to get started.

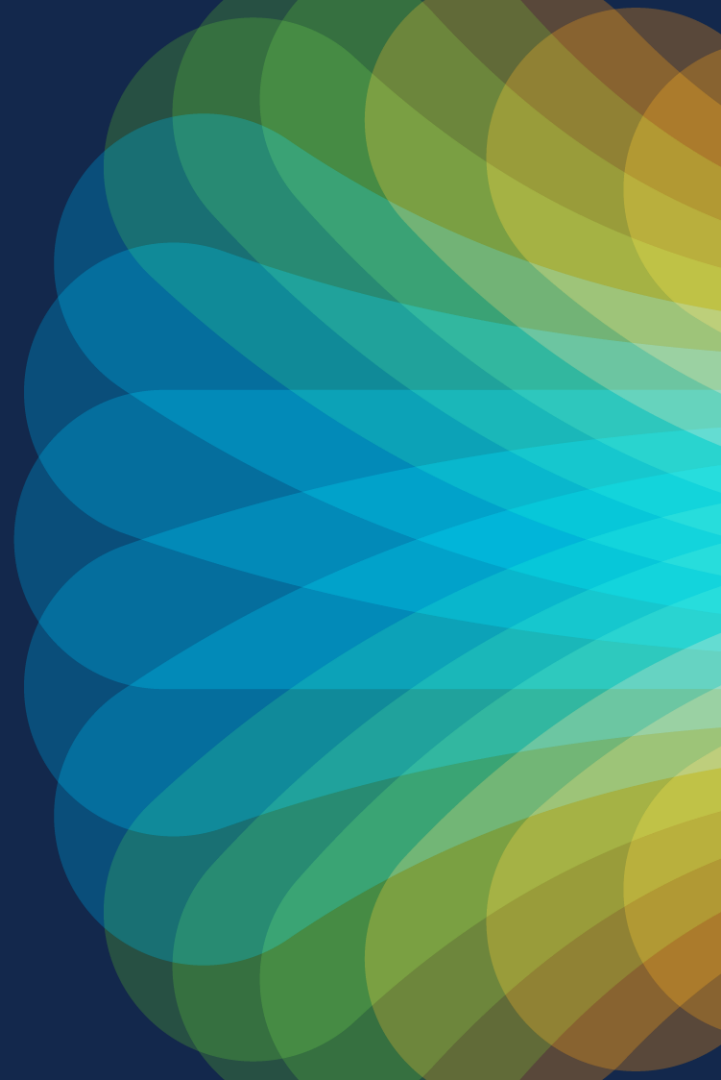




The bridge to possible

# Thank you

CISCO *Live!*



The background of the slide is a vibrant, abstract graphic. It features a large, stylized cloud on the left side, composed of overlapping, semi-transparent shapes in shades of red, orange, and yellow. To the right of the cloud, a bright, multi-colored sunburst or starburst pattern radiates from a central point, with rays extending towards the right edge of the frame. The colors in the sunburst transition through a rainbow spectrum, including yellow, green, blue, and purple. The overall effect is energetic and celebratory.

cisco *Live!*

Let's go