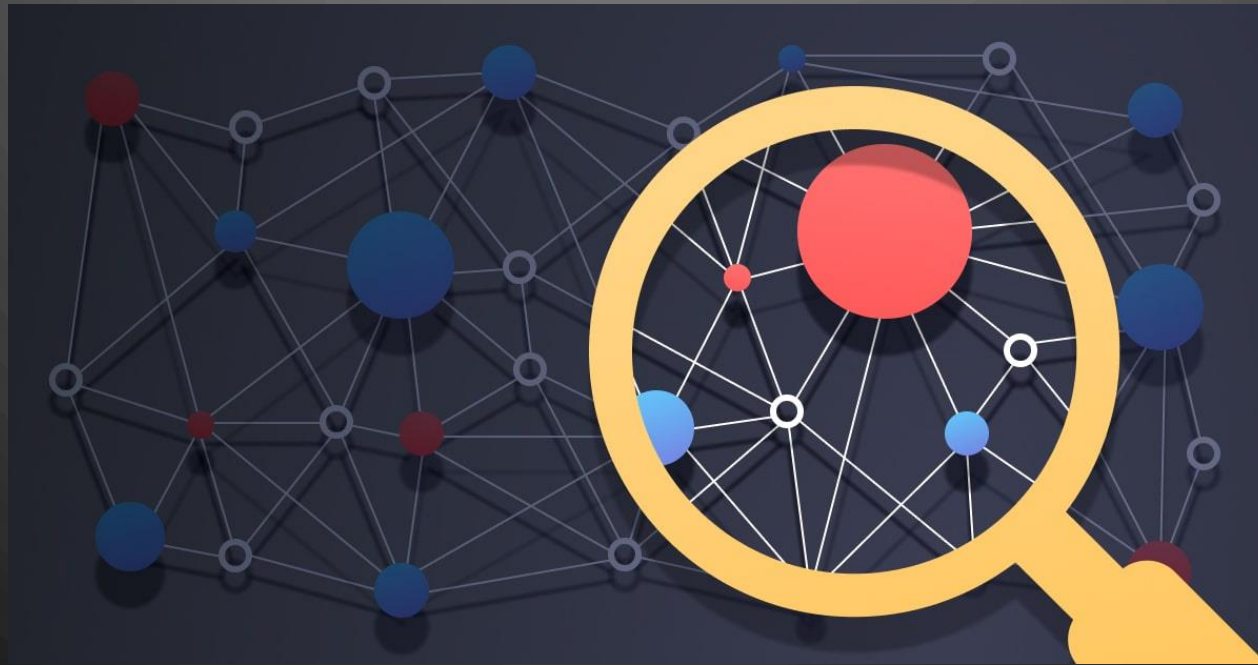


DISTRIBUTED TRACING



AGENDA

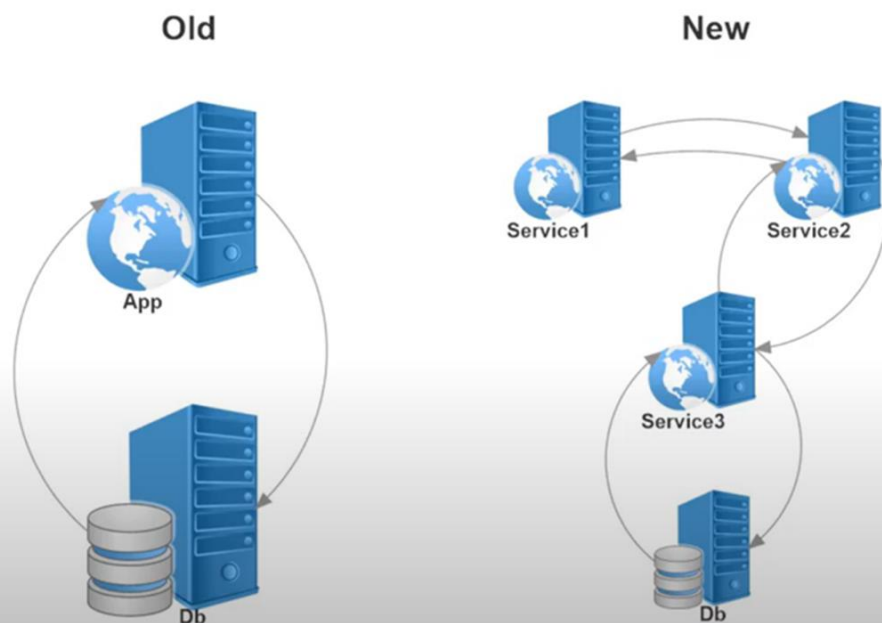
1. Czym jest i w jaki sposób działa Distributed Tracing.
2. Popularne narzędzia do Distributed Tracing.
3. Działanie Zipkin'a oraz Sleuth.
4. OpenTelemetry.
5. Signoz.
6. Live demo.

Czym jest Distributed Tracing?

Distributed Tracing to sposób na obserwację działania aplikacji opartej na architekturze mikroservisów, za pomocą śledzenia zapytań przechodzących przez kolejne serwisy w aplikacji.

Usprawnia debugowanie oraz diagnostykę aplikacji. Pozwala lepiej zrozumieć zależności między serwisami. Umożliwia analizę czasu przetwarzania zapytania przez konkretne serwisy, dzięki czemu można znaleźć elementy, które spowalniają aplikację.


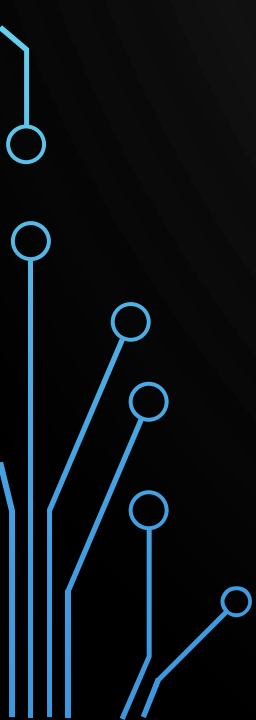
Monoliths and Microservices



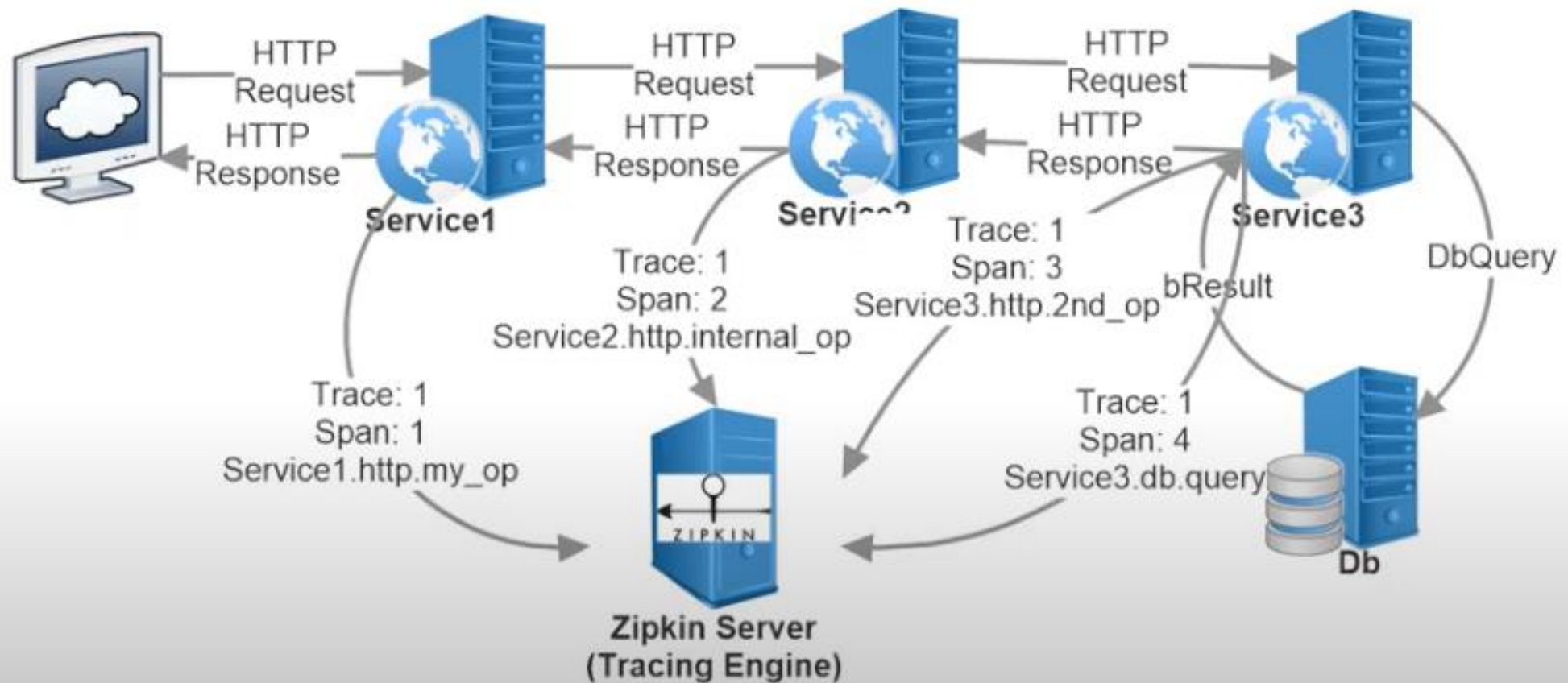


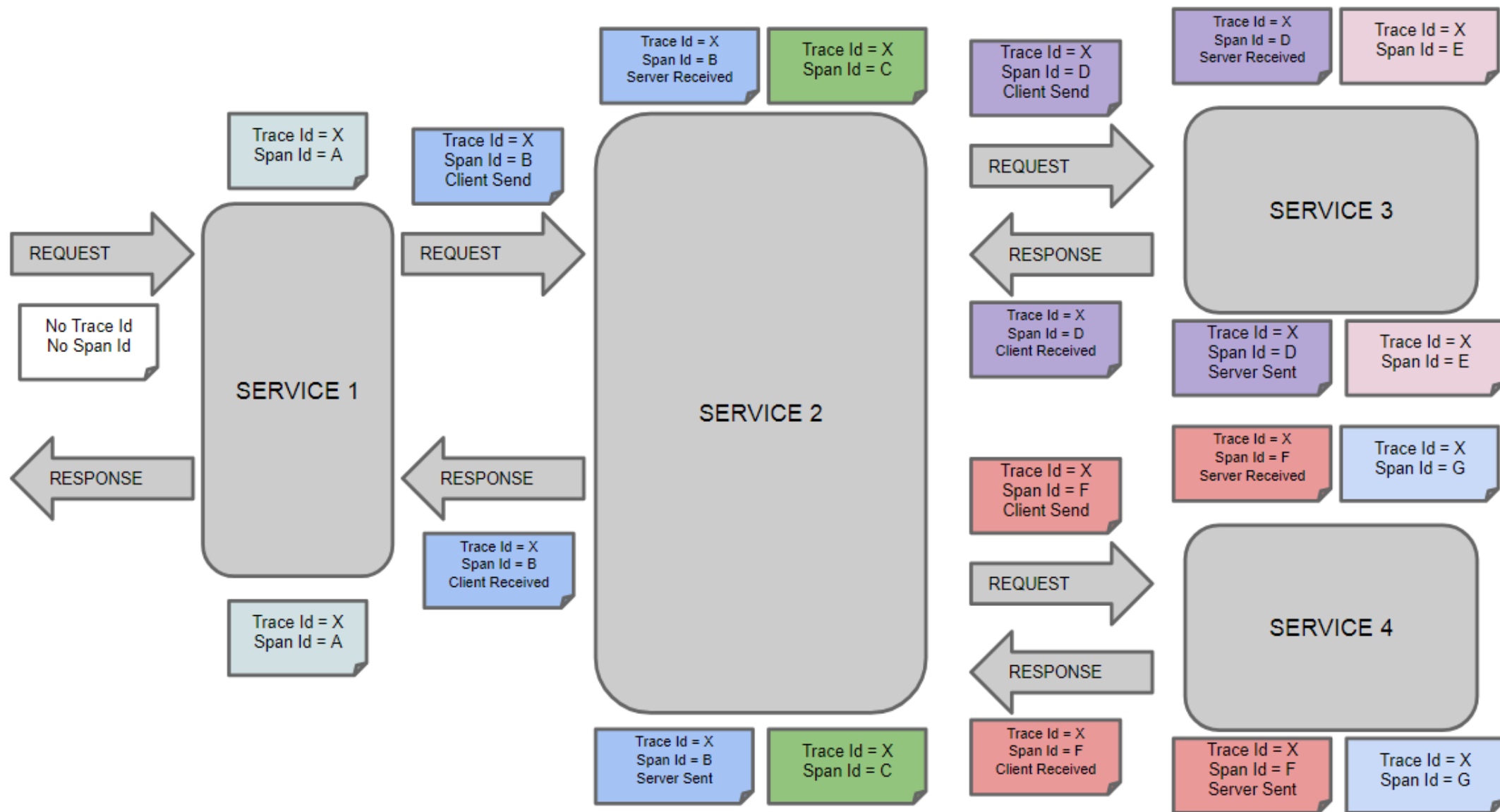
W jaki sposób działa Distributed Tracing?

Distributed Tracing rozpoczyna się w momencie w którym użytkownik wchodzi w interakcję z aplikacją np. wysyła zapytanie HTTP. Zapytaniu temu zostaje przypisany unikatowy *Trace ID*. W momencie gdy zapytanie przechodzi przez system, każda operacja (*span*) przeprowadzona na nim jest oznaczona własnym ID (*span ID*) oraz *parent ID*, czyli ID operacji, która wywołała aktualną operację.

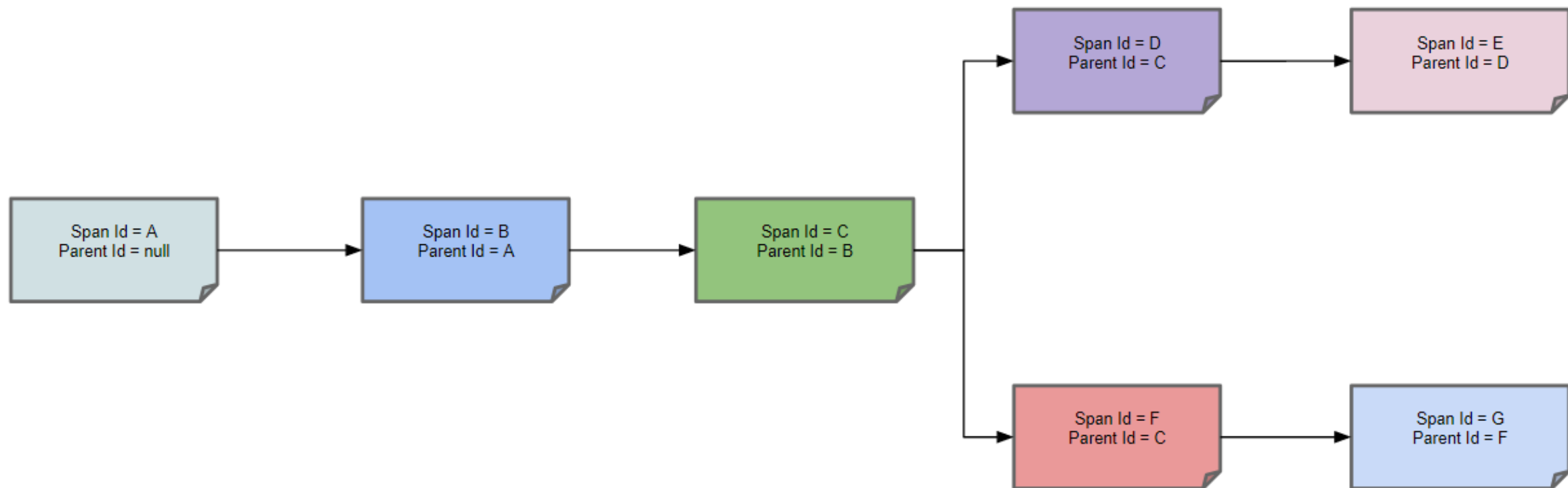


Collecting Distributed Traces





TRACE



POPULAR DISTRIBUTED TRACING TOOLS



Zaimplementowany w Javie.
Stworzony przez Twitter.
Obecnie open-source.



Zaimplementowany w Go.
Stworzony przez Uber.
Obecnie open-source.

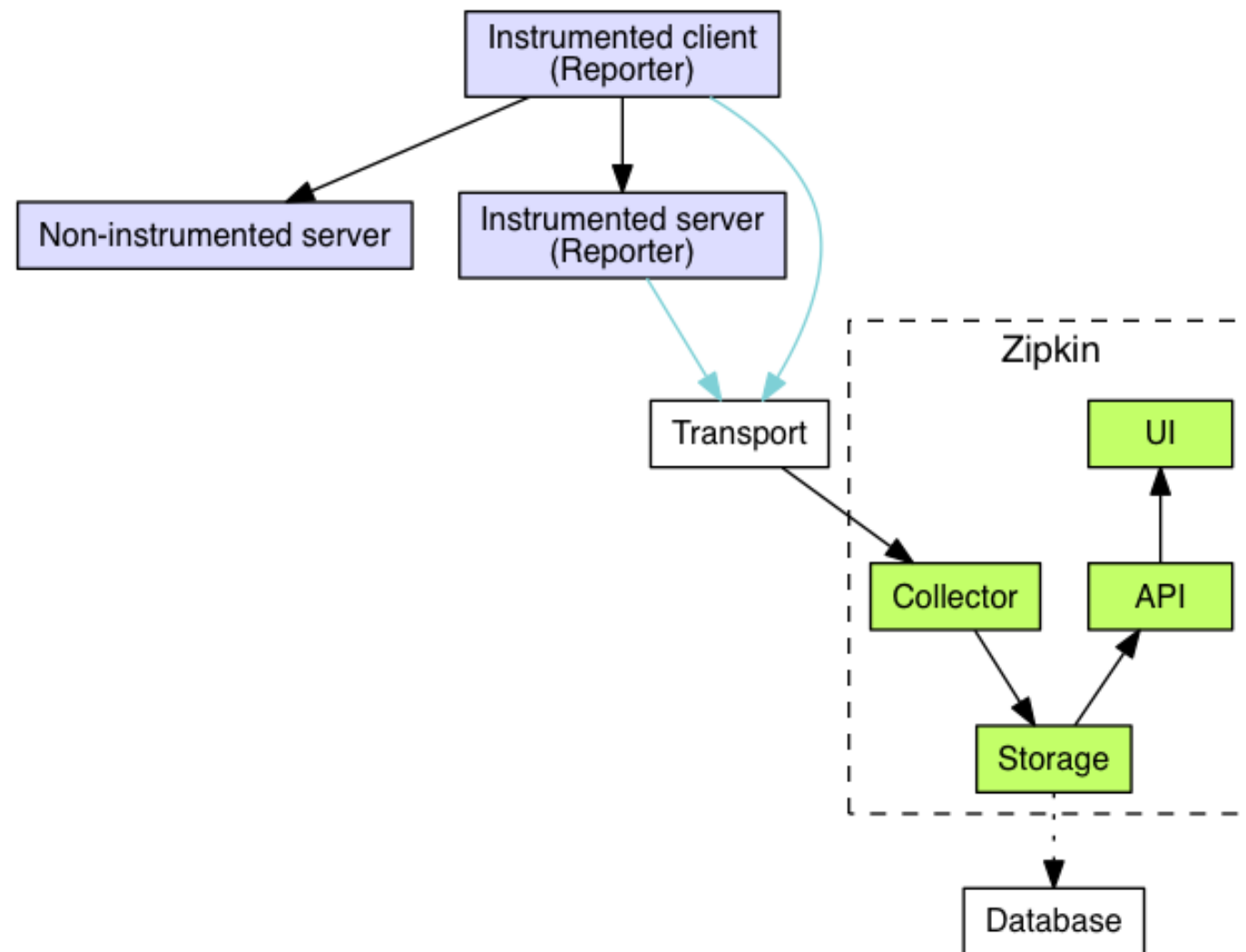


JAK DZIAŁA ZIPKIN?

Zipkin składa się z czterech komponentów:

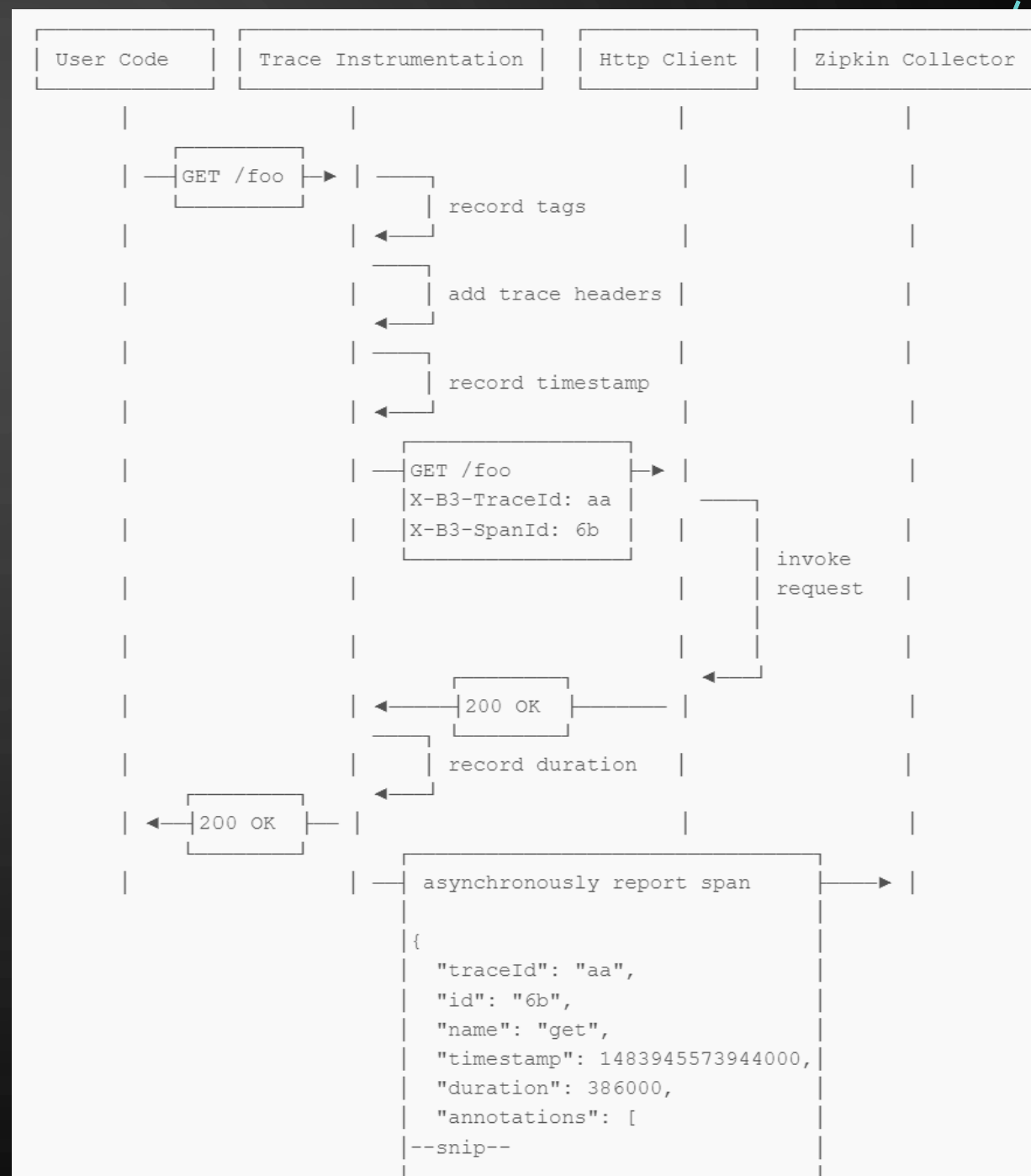
- Collector,
- Storage,
- Search,
- Web UI.

Collector dokonuje walidacji przychodzących danych, następnie przekazuje je do storage component. Użytkownik w dowolnej chwili może uzyskać dostęp do danych poprzez search component oraz web UI.



JAK DZIAŁA ZIPKIN? CD.2 (SLEUTH)

Jak wcześniej wspomnieliśmy Zipkin Collector otrzymuje dane, więc musi te dane skądś otrzymać i otrzymać je w odpowiedniej formie tj. wraz z Trace ID, Span ID,... Tę funkcjonalność zapewnia biblioteka **Sleuth**. Dodaje ona m.in. niezbędne nagłówki (Trace ID itp.) oraz czas trwania operacji a także przesyła te dane do Zipkin Collector.



~~OPEN TRACING~~ -----> OPEN TELEMETRY

OpenTelemetry is a collection of tools, APIs, and SDKs. Use it to instrument, generate, collect, and export telemetry data (metrics, logs, and traces) to help you analyze your software's performance and behavior.

OpenTelemetry powstało ze złączenia OpenCensus oraz OpenTracing.

OpenCensus było zbiorem bibliotek umożliwiającym zbieranie metryk oraz distributed traces aplikacji.

OpenTracing - Vendor-neutral APIs and instrumentation for distributed tracing.



SIGNOZ

Signoz jest narzędziem open-source umożliwiającym wizualizację danych zebranych przez OpenTelemetry.



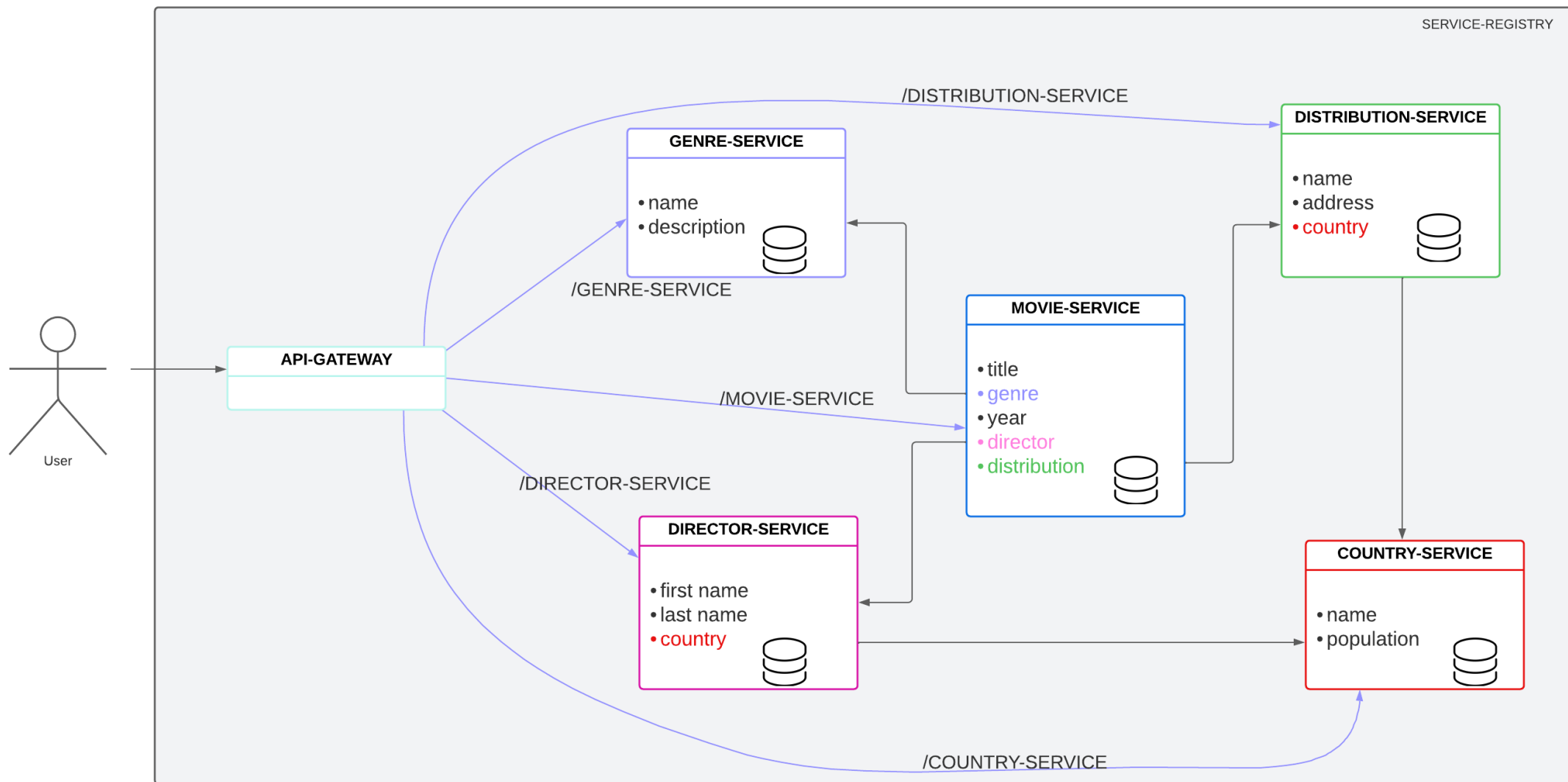
DISTRIBUTED TRACING NA BAZIE PRZYKŁADOWEJ APLIKACJI

The diagram illustrates a distributed application architecture for tracing, featuring a central API-GATEWAY and four microservices: GENRE-SERVICE, MOVIE-SERVICE, DIRECTOR-SERVICE, and DISTRIBUTION-SERVICE. A SERVICE-REGISTRY is also present. The architecture is as follows:

- API-GATEWAY**: Acts as the entry point for the User. It routes requests to the four microservices.
- GENRE-SERVICE**: Contains attributes `name` and `description`. It is reached via the `/GENRE-SERVICE` endpoint.
- MOVIE-SERVICE**: Contains attributes `title`, `genre`, `year`, `director`, and `distribution`. It is reached via the `/MOVIE-SERVICE` endpoint.
- DIRECTOR-SERVICE**: Contains attributes `first name`, `last name`, and `country`. It is reached via the `/DIRECTOR-SERVICE` endpoint.
- DISTRIBUTION-SERVICE**: Contains attributes `name`, `address`, and `country`. It is reached via the `/DISTRIBUTION-SERVICE` endpoint.
- COUNTRY-SERVICE**: Contains attributes `name` and `population`. It is reached via the `/COUNTRY-SERVICE` endpoint.
- SERVICE-REGISTRY**: A central registry that stores the location of the services. It is used by the API-GATEWAY to route requests and by the services to find each other.

The diagram shows the following connections and data flow:

- User** → **API-GATEWAY**
- API-GATEWAY** → **GENRE-SERVICE** (`/GENRE-SERVICE`)
- API-GATEWAY** → **MOVIE-SERVICE** (`/MOVIE-SERVICE`)
- API-GATEWAY** → **DIRECTOR-SERVICE** (`/DIRECTOR-SERVICE`)
- API-GATEWAY** → **DISTRIBUTION-SERVICE** (`/DISTRIBUTION-SERVICE`)
- API-GATEWAY** → **COUNTRY-SERVICE** (`/COUNTRY-SERVICE`)
- GENRE-SERVICE** ↔ **MOVIE-SERVICE**
- MOVIE-SERVICE** ↔ **DIRECTOR-SERVICE**
- DIRECTOR-SERVICE** ↔ **COUNTRY-SERVICE**
- DISTRIBUTION-SERVICE** ↔ **COUNTRY-SERVICE**



PRZYDATNE INFORMACJE

- https://www.splunk.com/en_us/data-insider/what-is-distributed-tracing.html
- <https://www.dynatrace.com/news/blog/what-is-distributed-tracing/>
- <https://www.techtarget.com/searchapparchitecture/tip/3-distributed-tracing-tools-perfect-for-microservices>
- <https://zipkin.io/pages/architecture.html>
- <https://newrelic.com/resources/ebooks/quick-introduction-distributed-tracing>
- <https://howtodoinjava.com/spring-cloud/spring-cloud-zipkin-sleuth-tutorial/>
- <https://opentracing.io/docs/overview>
- <https://eng.uber.com/distributed-tracing>
- <https://newrelic.com/resources/ebooks/quick-introduction-distributed-tracing>
- <https://docs.spring.io/spring-cloud-sleuth/docs/current/reference/html/getting-started.html#getting-started>
- <https://cloud.spring.io/spring-cloud-sleuth/reference/html>
- <https://www.baeldung.com/spring-cloud-sleuth-get-trace-id>
- <https://ryanharrison.co.uk/2021/08/06/distributed-tracing-spring-boot-jaeger.html>

The background is a dark gray gradient with a series of concentric circles centered in the middle. In the four corners, there are stylized, light blue circuit-like lines with small circles at the ends, resembling a technical or digital theme.

DZIĘKUJĘ