



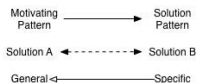
# INFRASTRUCTURE DE PRODUCTION

---

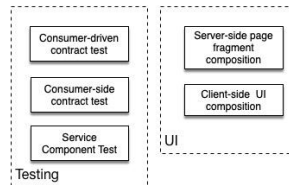
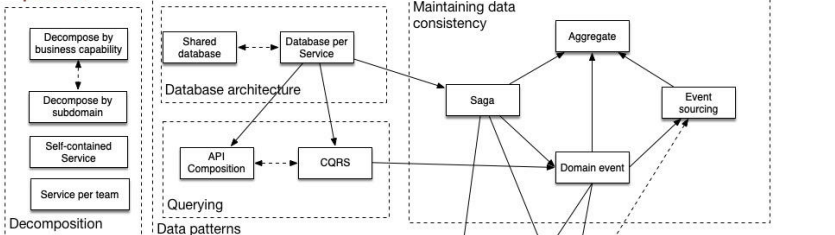
## TRACING

00.00 2017

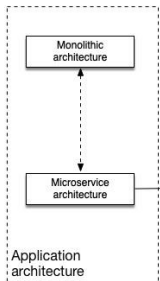
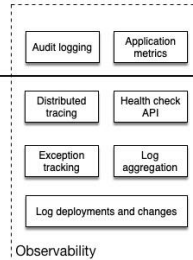
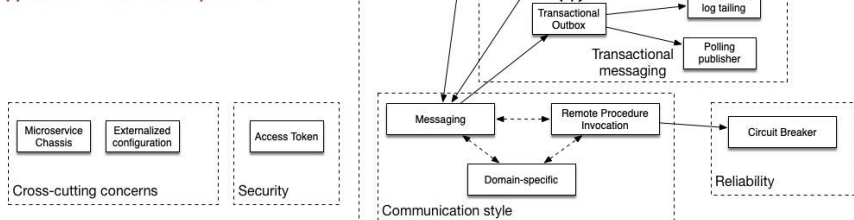
Photo by Diogène Morin on [Unsplash](#)



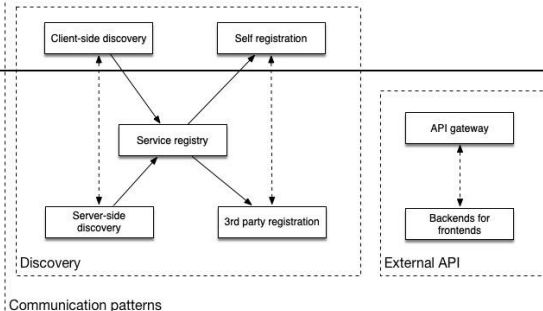
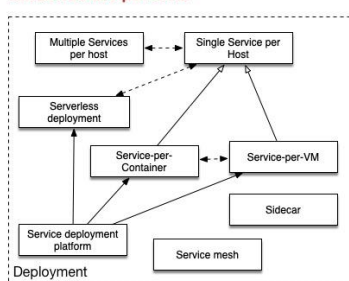
## Application patterns














## Application Infrastructure patterns



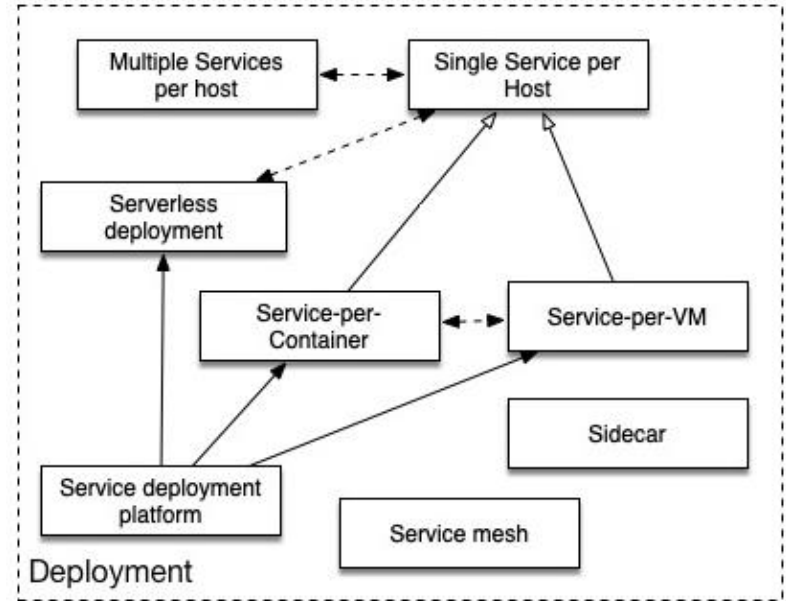
## Infrastructure patterns



Microservice patterns

	METRICS	TRACES	LOGS
APP	 Micrometer	 Opentelemetry	 Log4j2
COLLECTEUR	 Prometheus (poll)	 APM	 Logstash
STORAGE	 Prometheus	 Elastic Search	 Elastic Search
VISU	 Grafana	 Kibana	 Kibana

- Single Service per Host
- Multiple Services per Host
- Serverless deployment
  - Oracle Fx
  - AWS Lambda
- Service-per-Container
- Service-per-VM
- Service Deployment Platform
  - Docker orchestration (ex: k8s, Docker Swarm mode)





## ■ Discovery

### ■ Service Registry

### ■ Server-side discovery

### ■ Client-side discovery

### ■ Self Registration

### ■ 3rdParty Registration

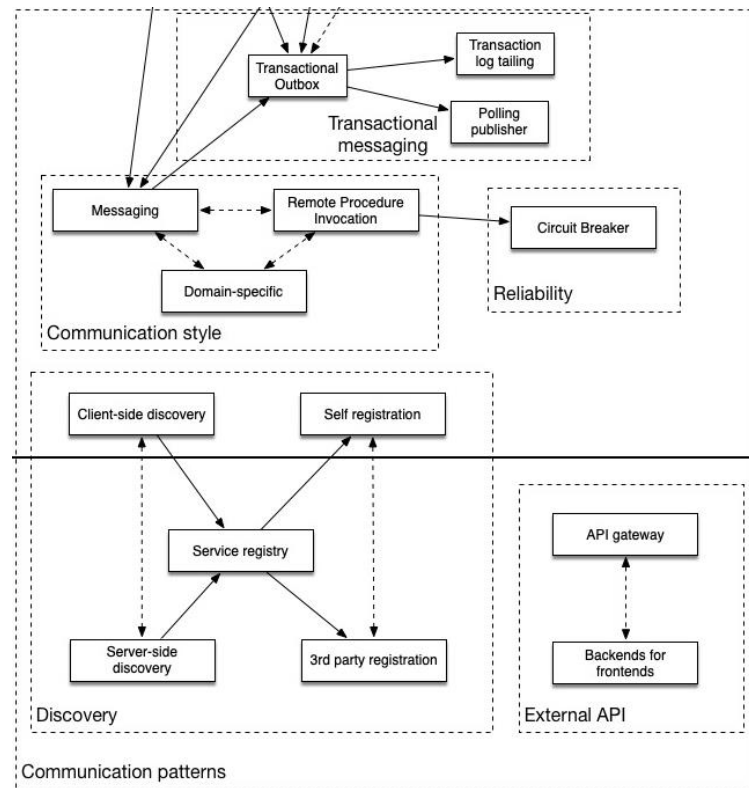
## ■ External API

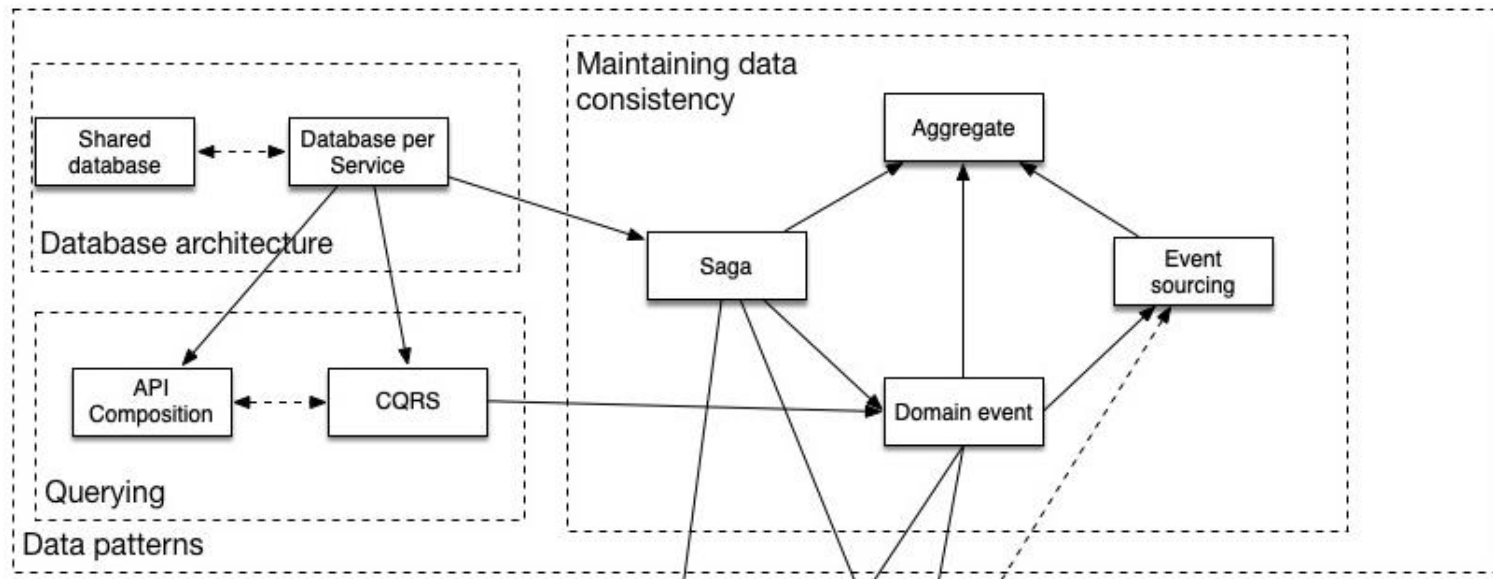
### ■ API Gateway

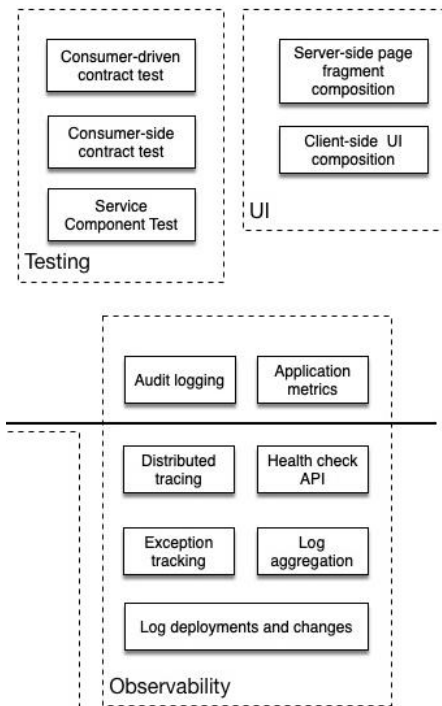
### ■ Backend for front end

## ■ Communication Style

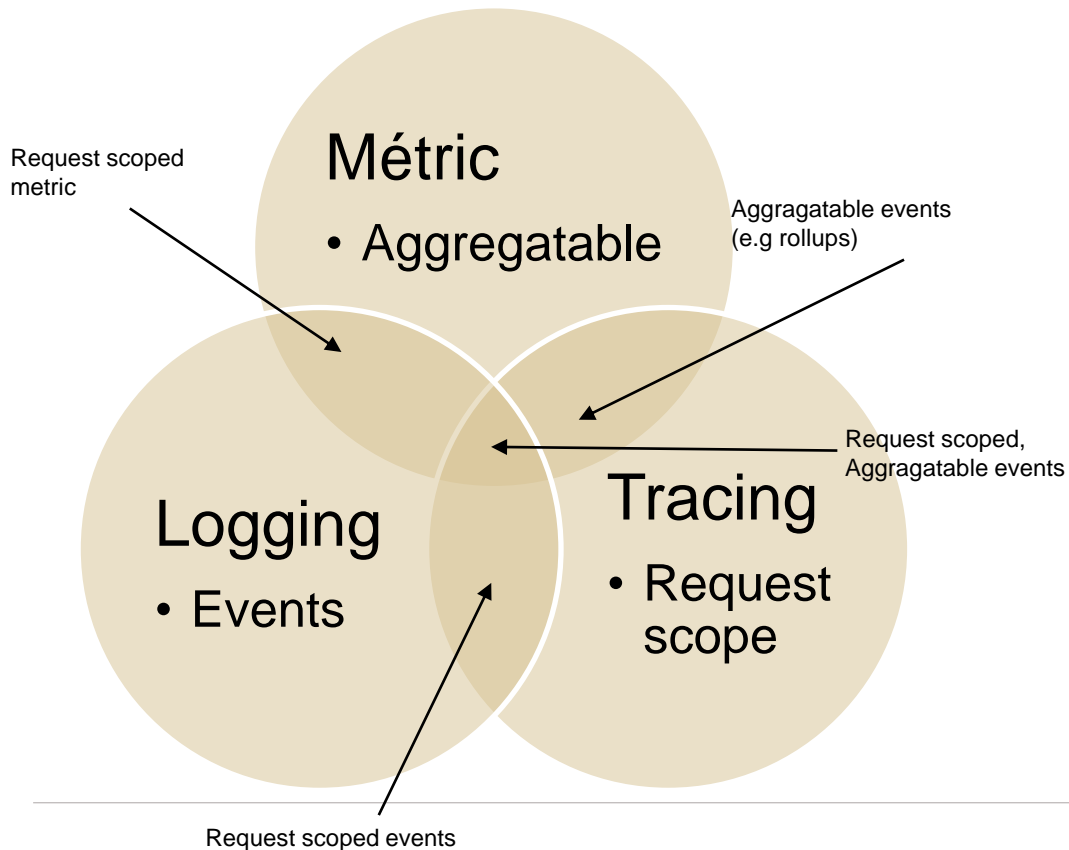
## ■ Reliability







# OBSERVABILITÉ



- Les logs sont fait pour enregistrer des évènements discrets. Comme les informations de debug ou d'erreur d'une application pour permettre un diagnostic simple.
- Les métriques sont des enregistrements utilisables pour faire des aggregation. Comme le nombre d'éléments dans une queue, le nombre de session, le nombre de requetes http, ....
- Les traces sont des enregistrements d'informations associées à une requête (request scope). Comme le temps passé pour faire l'appel d'une méthode distante. C'est l'outil a utiliser pour investiguer les problems de performance system, de Logging, metrics, ...



# MONITORING == OBSERVING EVENTS

---

Low volume

Metrics – aggregation des enregistrements des évènements (ex. compteurs)

Tracing – Enregistrement des évènements lié a la transaction

High volume

Logging - Record unique events

# OBSERVABILITY

---

Le terme «observabilité» dans la théorie du contrôle indique que le système est observable si les états internes du système et, par conséquent, son comportement, peuvent être déterminés en ne regardant que ses entrées et ses sorties

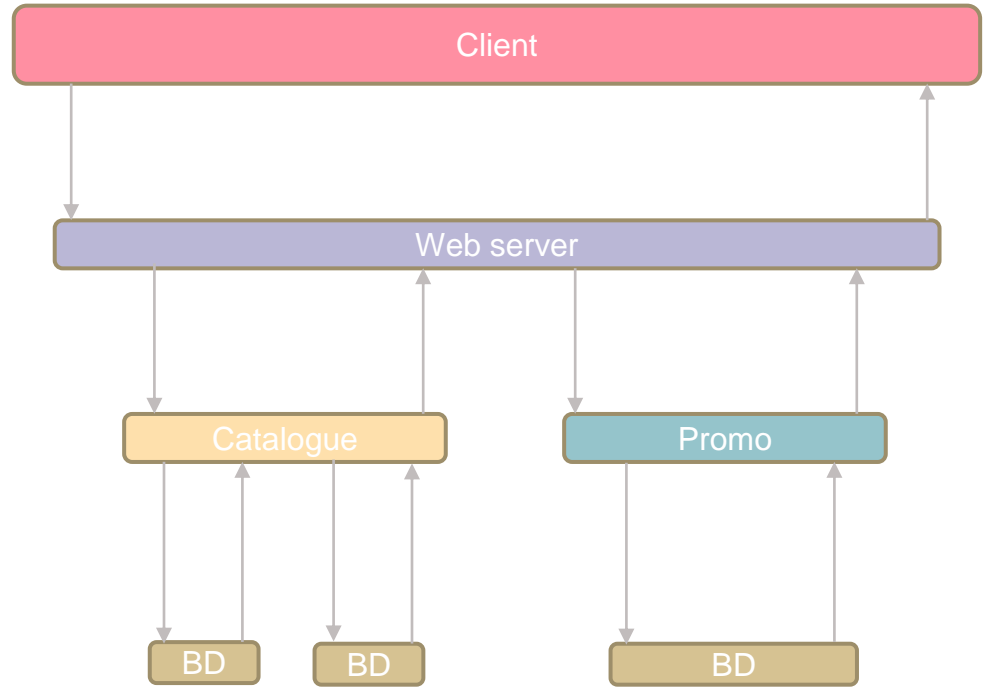
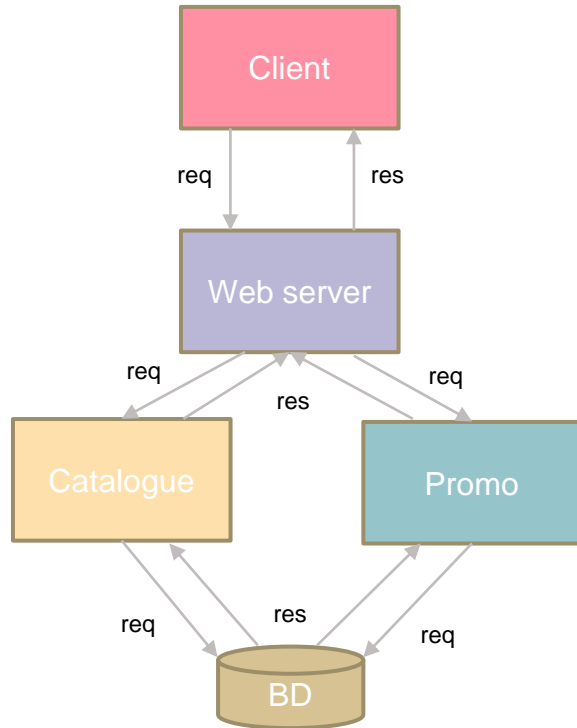
## The pillars of observability



- Much has been made of the so-called “pillars of observability”: monitoring, logging and instrumentation
- Each of these is important, for each has within it the capacity to answer questions about the system
- But each also has limitations!
- Their shared limitation: each can only be as effective as the observer — they cannot answer questions not asked!
- Observability seeks to answer questions asked and prompt new ones: **the human is the foundation of observability!**

2018 Observability  
Practitioners  
Summit, Bryan  
Cantrill, the CTO of  
Joyent

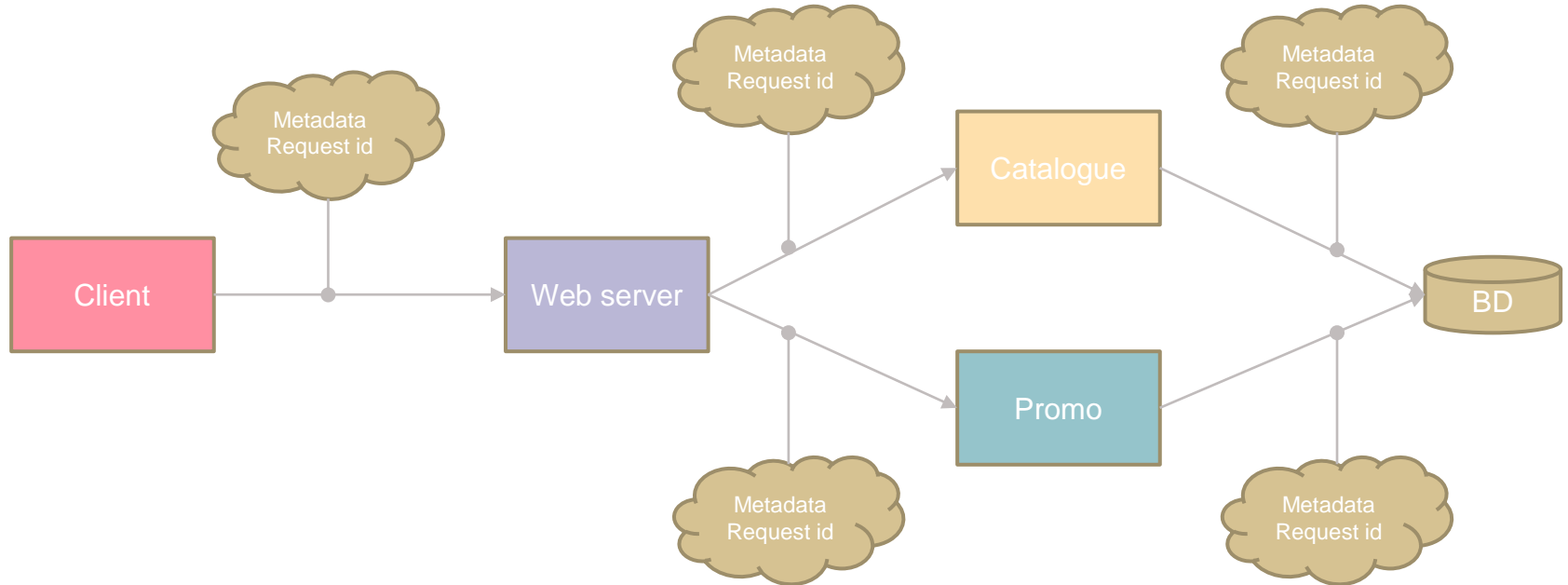
# IDÉE DE BASE SUR LE TRACING

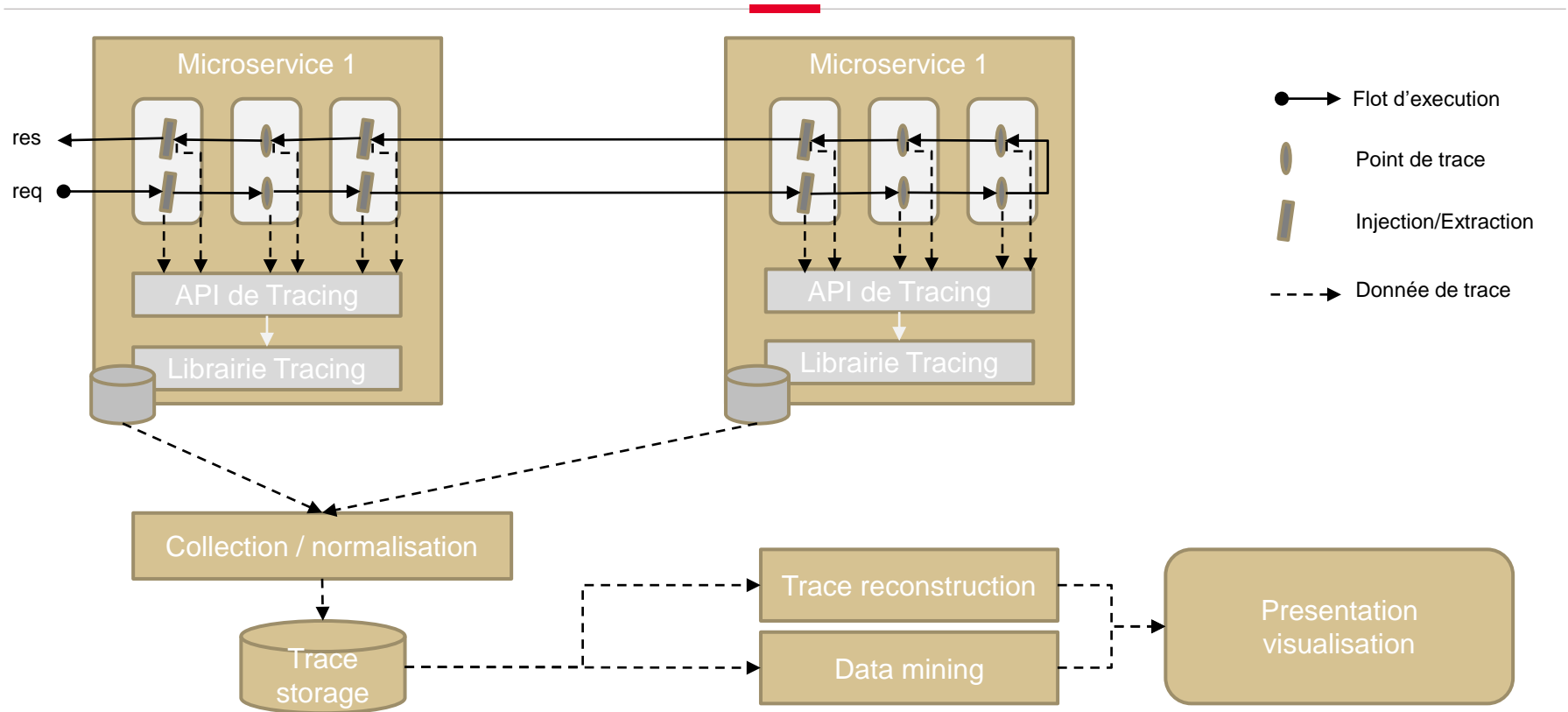


Time

# REQUEST CORRELATION

## Propagation des MetaData - Global ID(Execution ID or Request ID)





# FRAMEWORK DE TRACING

---

Dapper (Google): Foundation for all tracers  
Zipkin (Twitter)  
Jaeger (Uber)  
StackDriver Trace (Google)  
Appdash (golang)  
X-ray (AWS)



# TRACING SYSTEMS

---

Zipkin and OpenZipkin

Open Source Tracing system

2012 - Twitter

Tracer - Brave API - framework (ex: Spring, Spark, Kafka, gRPC)

tracing System - data-format level

Jaeger - 2015 - Uber

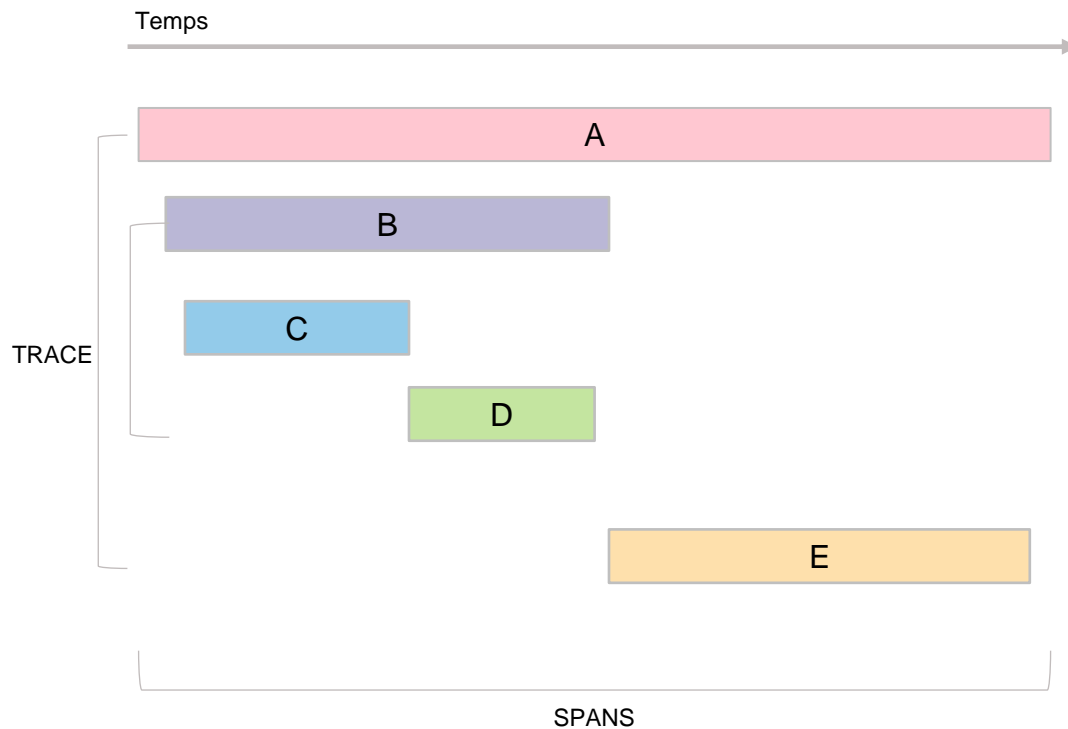
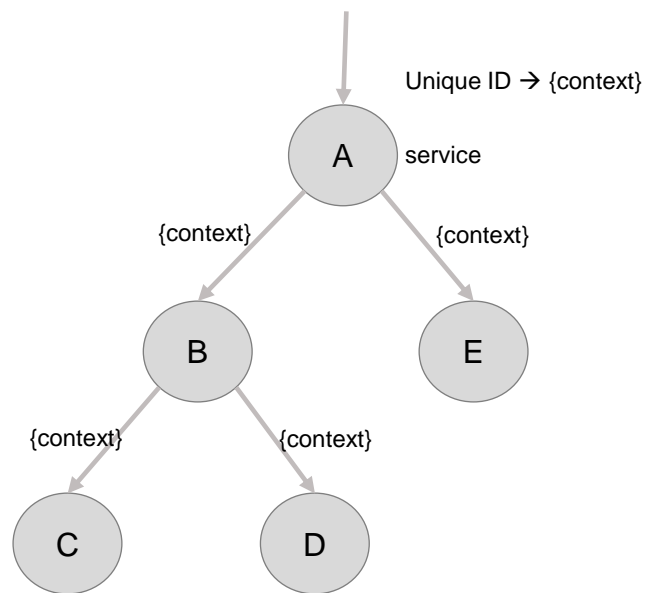
2017 - Open Source Project

CNCF - OpenTracing

Zipkin - (B3 metadata format)

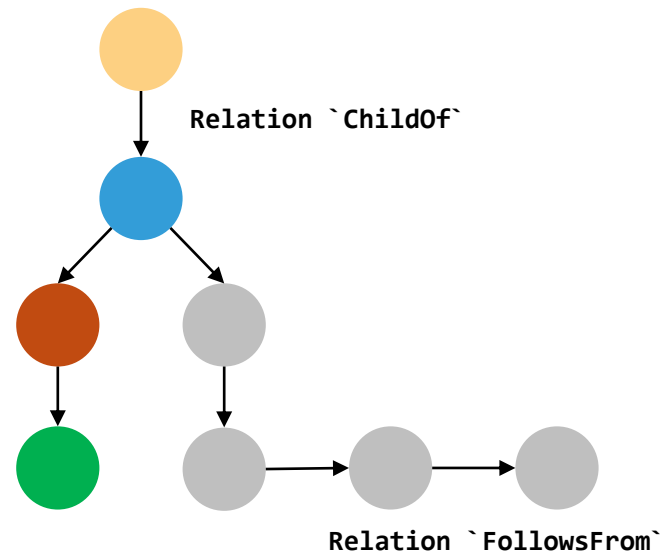
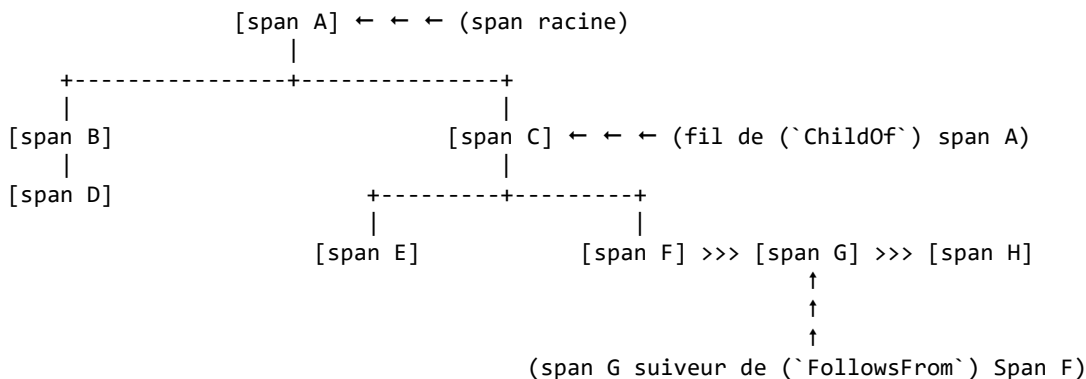
Instrumentation - OpenTracing-compatible tracer

# TRACE DISTRIBUÉE



## RELATION ENTRE LES SPANS

**Les Traces** dans l'OpenTracing sont définis implicitement par leurs **Spans**. En particulier, une **Trace** peut être considéré comme un graphe acyclique dirigé (DAG) de **Spans**, où les arêtes entre les travées sont appelées **References**. Par exemple, voici un exemple de Trace composé de 8 Spans : Relations causales entre Spans dans un seul Trace



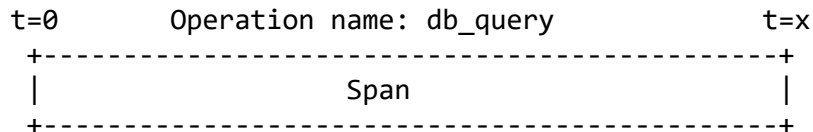
# QU'EST QU'UN SPAN

---

Le «**span**» est le principal élément constitutif d'une trace distribuée, représentant l'**unité de travail individuelle** fait dans un système distribué.

Chaque composant du système distribué contribue à une span - une opération nommée et chronométrée représentant une partie du flux de travail.

Les spans peuvent (et contiennent généralement) des «références» à d'autres spans, ce qui permet à plusieurs spans d'être assemblé en une seule trace complète : une visualisation de la vie d'une request a travers toute ses étapes dans un système distribué.



Chaque span encapsule l'état suivant selon la spécification OpenTracing :

- Un **nom** d'opération
- Un **timestamp** de début et de fin
- Un ensemble de span **Tags** de type key:value
- Un ensemble de span **Logs** de type key:value
- Un **SpanContext**

# LES TAGS ET LOGS

---

Les **Tags** sont des paires clé - valeur qui permettent l'annotation personnalisée des spans afin d'être requêtée, filtrée et de comprendre les données de trace.

Les tags de span doivent s'appliquer à toute la durée.

## Exemple

- `http.url = "http://google.com"`
- `http.status_code = 200`
- `peer.service = "mysql"`
- `db.statement = "select * from users"`

Les **logs** sont des paires clé - valeur qui sont utiles pour capturer les messages de journalisation spécifiques du span et autres débogage ou sortie d'information de l'application elle-même.

## Exemple

- `span.log_kv( {'event': 'open_conn', 'port': 433} )`



# DEMO



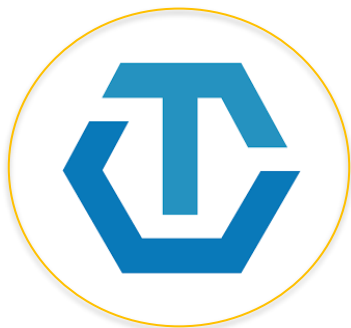
# JAEGER ARCHITECTURE

---



# JAEGER

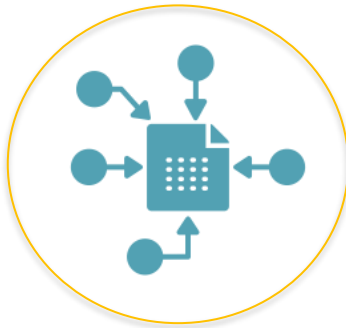
**Jaeger client**



**Agent**



**Collector**



**Data store**

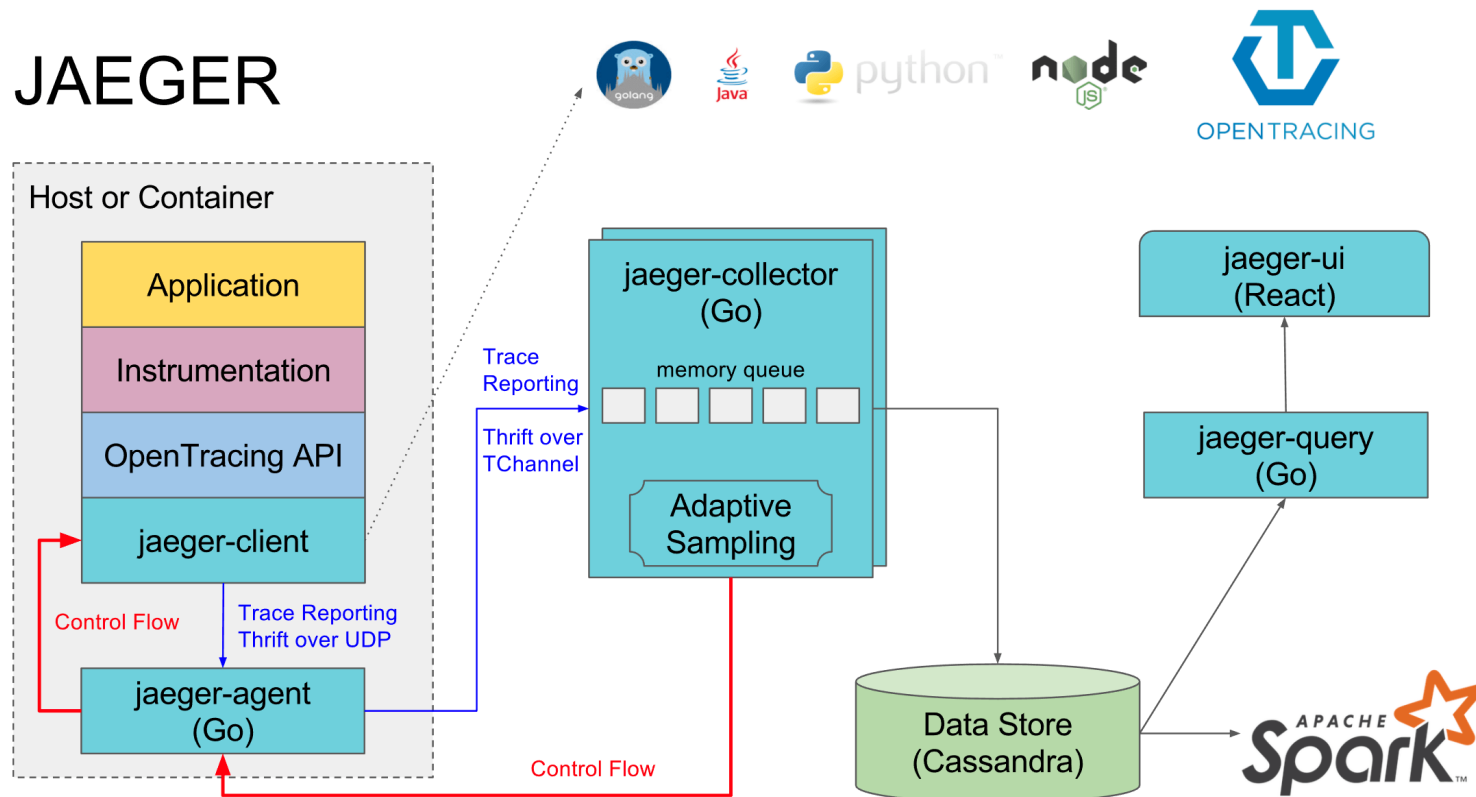


**Query**



# JAEGER

## JAEGER



# JAEGER

Jaeger UI

Lookup by Trace ID...

Dependencies

Search

## Find Traces

Service

Tags

Lookback

Min Duration

Max Duration

Limit Results

Find Traces



# JAEGER

Jaeger UI

Lookup by Trace ID...

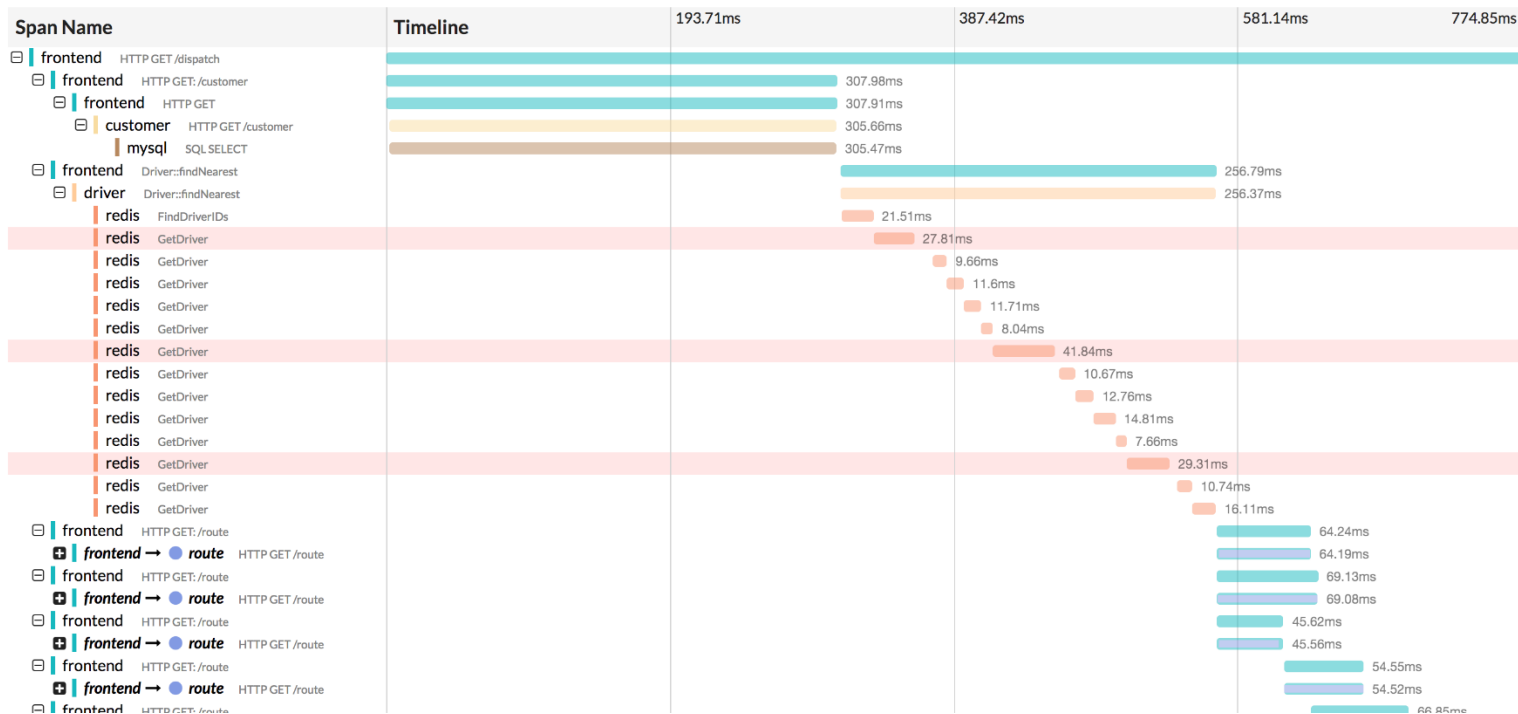
Dependencies

Search

» frontend: HTTP GET /dispatch

View Options ▾

Search...



A person with dark hair and a beard, wearing a dark blue t-shirt and a black neck strap, is seen from behind, sitting at a wooden table in a cafe. They are using a silver laptop. To their left, another person's hands are visible typing on a laptop. To their right, another silver laptop is open, displaying a document. A small potted plant in a white lace pot sits on the table. In the background, other people are seated at tables, and a water bottle is visible. The scene is lit with warm, natural light.

TP