

# L'Observabilité pour les devs

Florian Meuleman - Alexandre Moray

“



**La bête noire du dev**

”

“

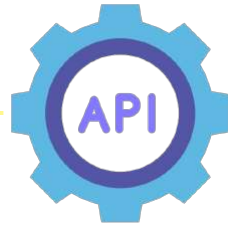


**Du Bug en prod**

”



**Front**



**Back**



**DB**

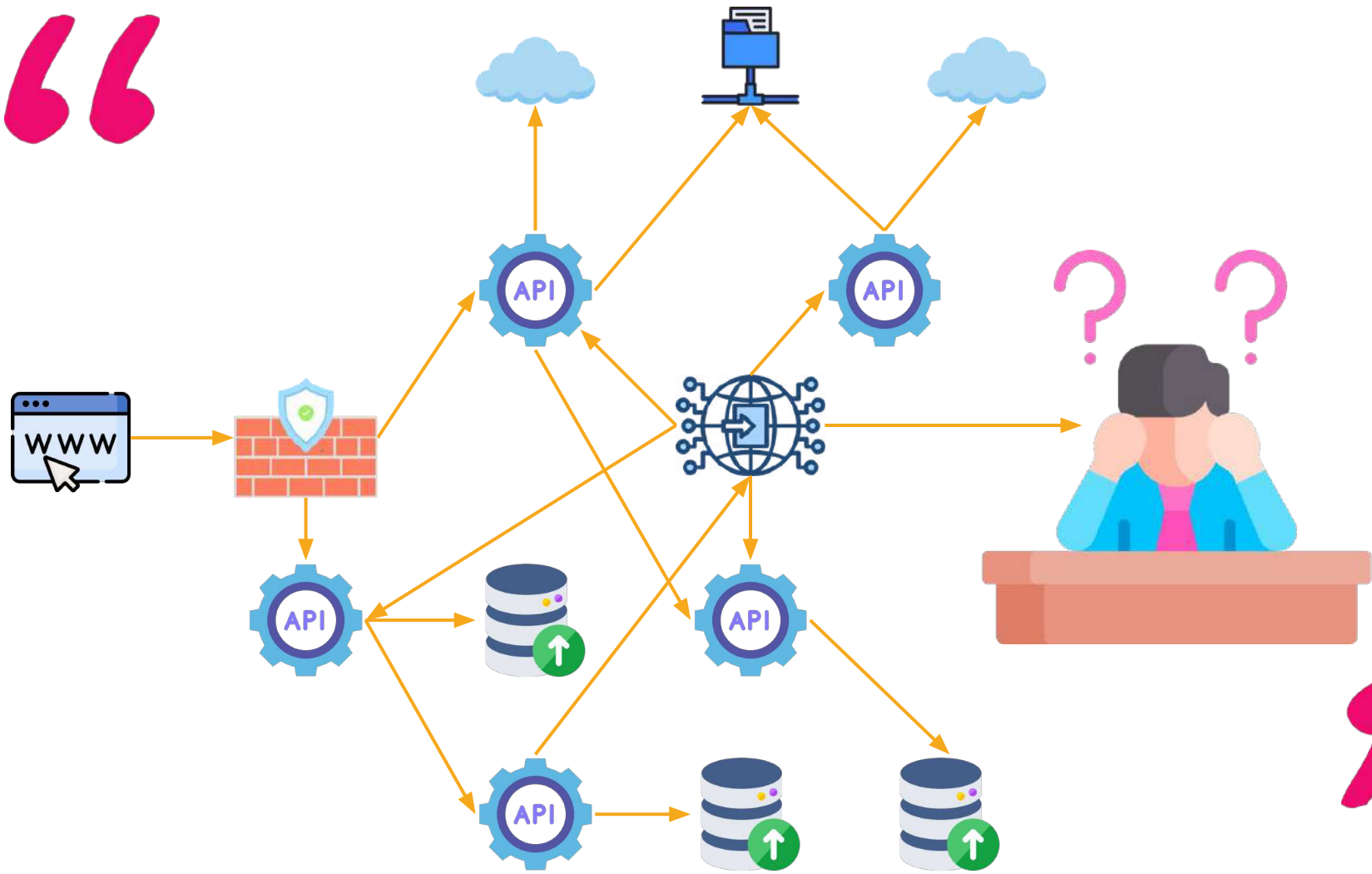
“



**La réalité**

”

“



”



**Florian Meuleman**

DevOps & Backend Engineer  
@ Takima

---



**Alexandre Moray**

Lead dev Fullstack  
@ Takima

---

“



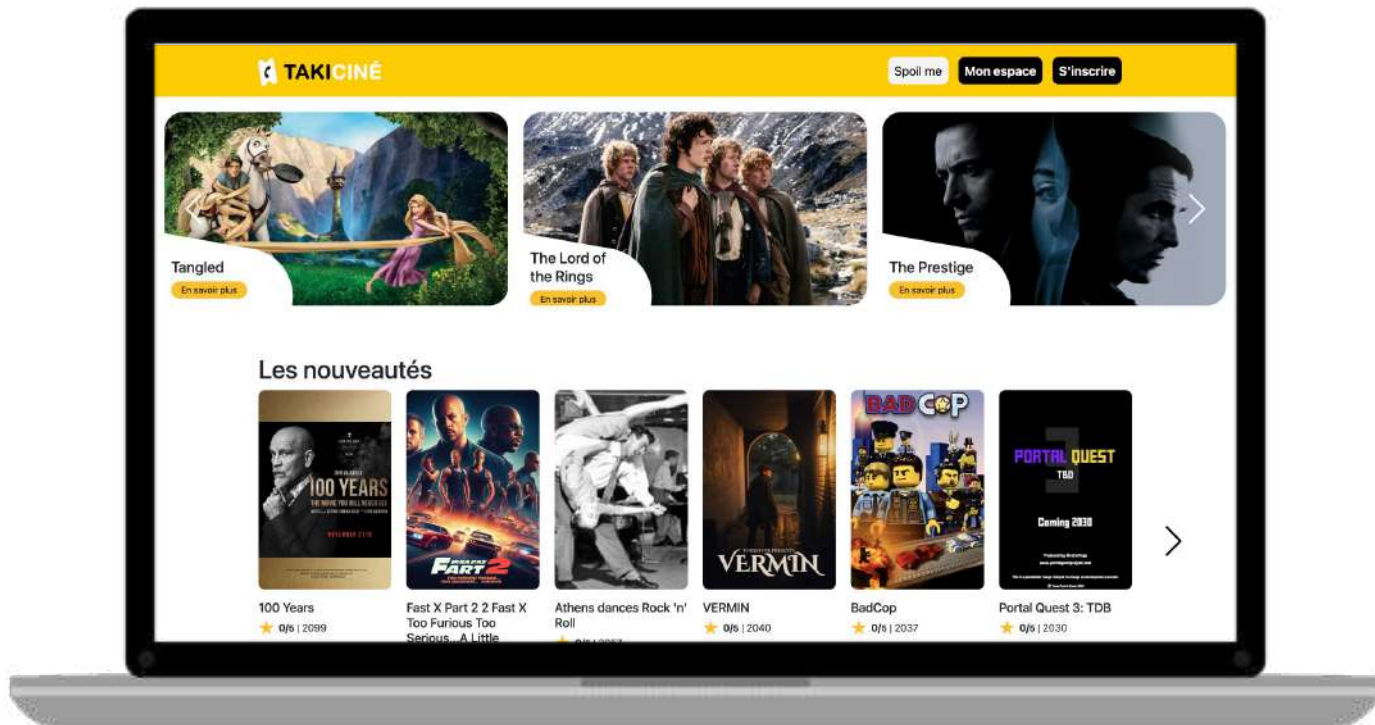
**On observe quoi ?**

”



Et on va observer quoi ?

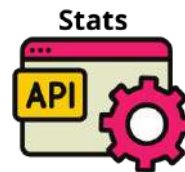
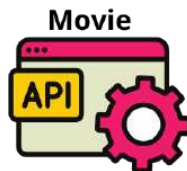
# Voyons ça sur notre appli



[www.takicine.fr](http://www.takicine.fr)

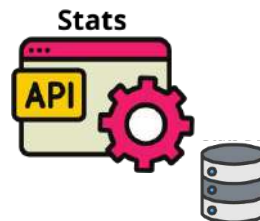
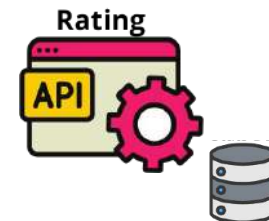
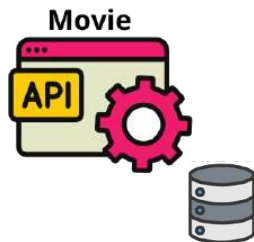
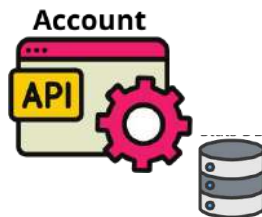
Et on va observer quoi ?

## Côté architecture



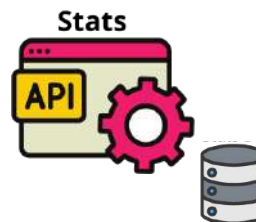
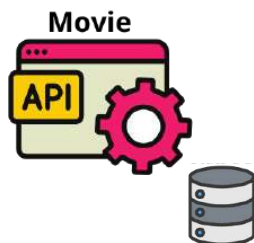
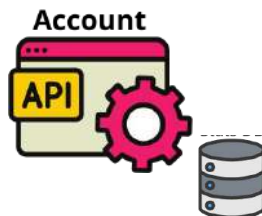
Et on va observer quoi ?

## Côté architecture



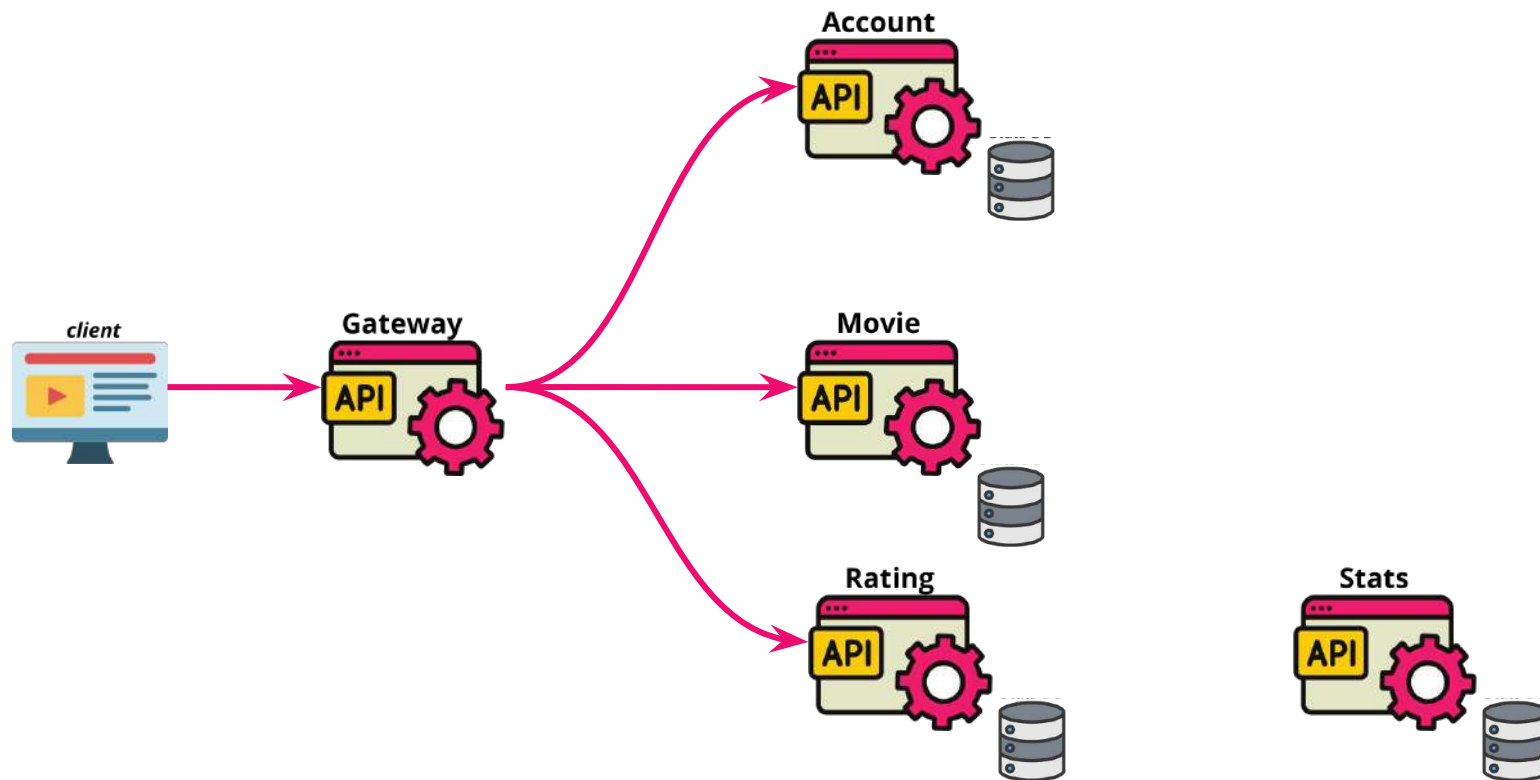
Et on va observer quoi ?

## Côté architecture



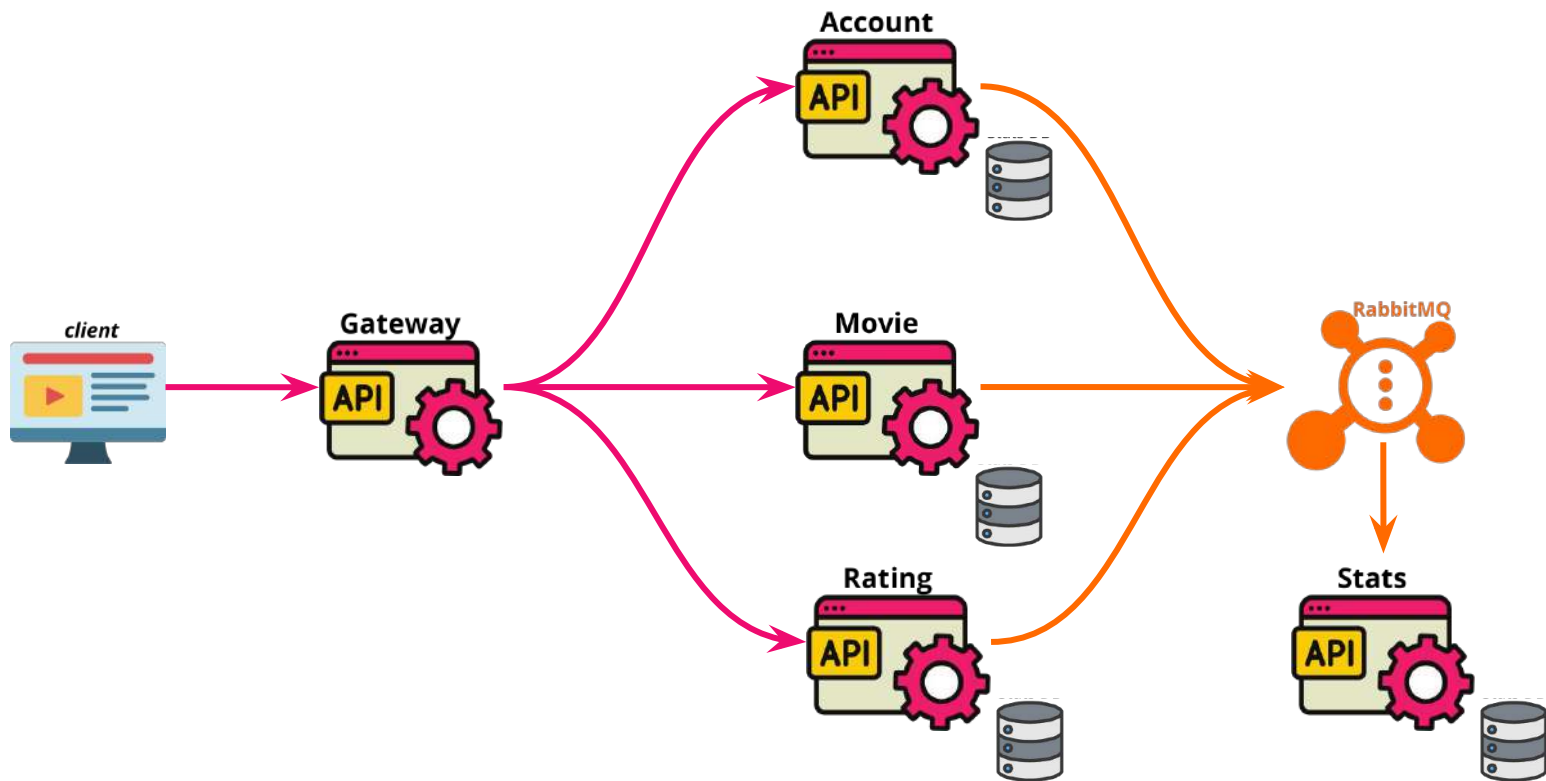
Et on va observer quoi ?

## Côté architecture



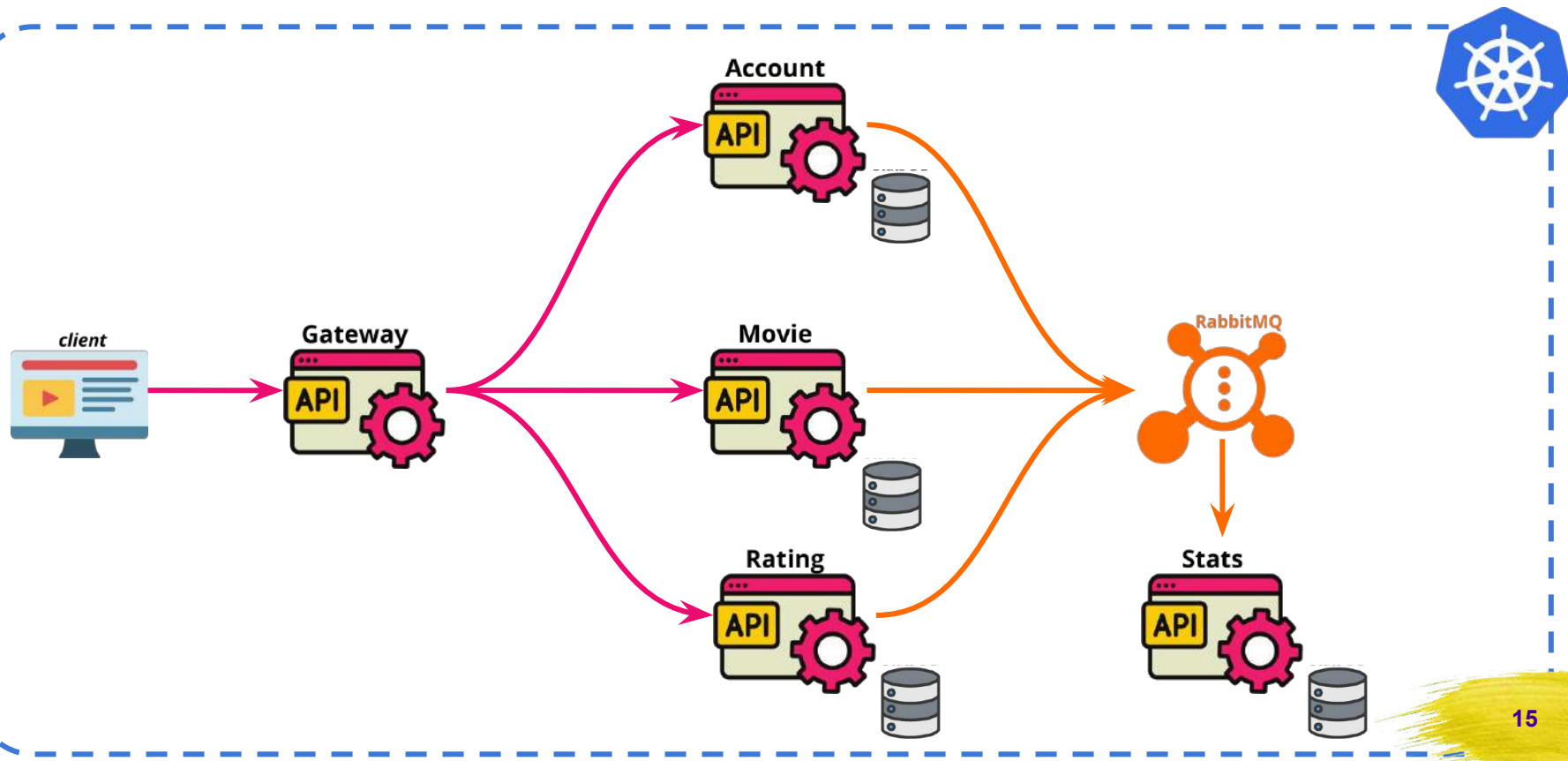
## Et on va observer quoi ?

# Côté architecture



Et on va observer quoi ?

## Côté architecture



“



**Ça fait beaucoup de choses**

”



“



Ça fait beaucoup de choses  
Qui peuvent **mal** se passer

”

“



**On monte une observabilité  
en 45 mins**

”

“



- ✓ **Solution simple**
- ✓ **Limiter l'impact sur notre code**
- ✓ **Ne pas avoir 15 000 outils**
- ✓ **Open Source**

”

“



**L'observabilité, c'est  
quoi ?**

”

# Monitoring vs Observabilité



*Monitoring*



*Observabilité*

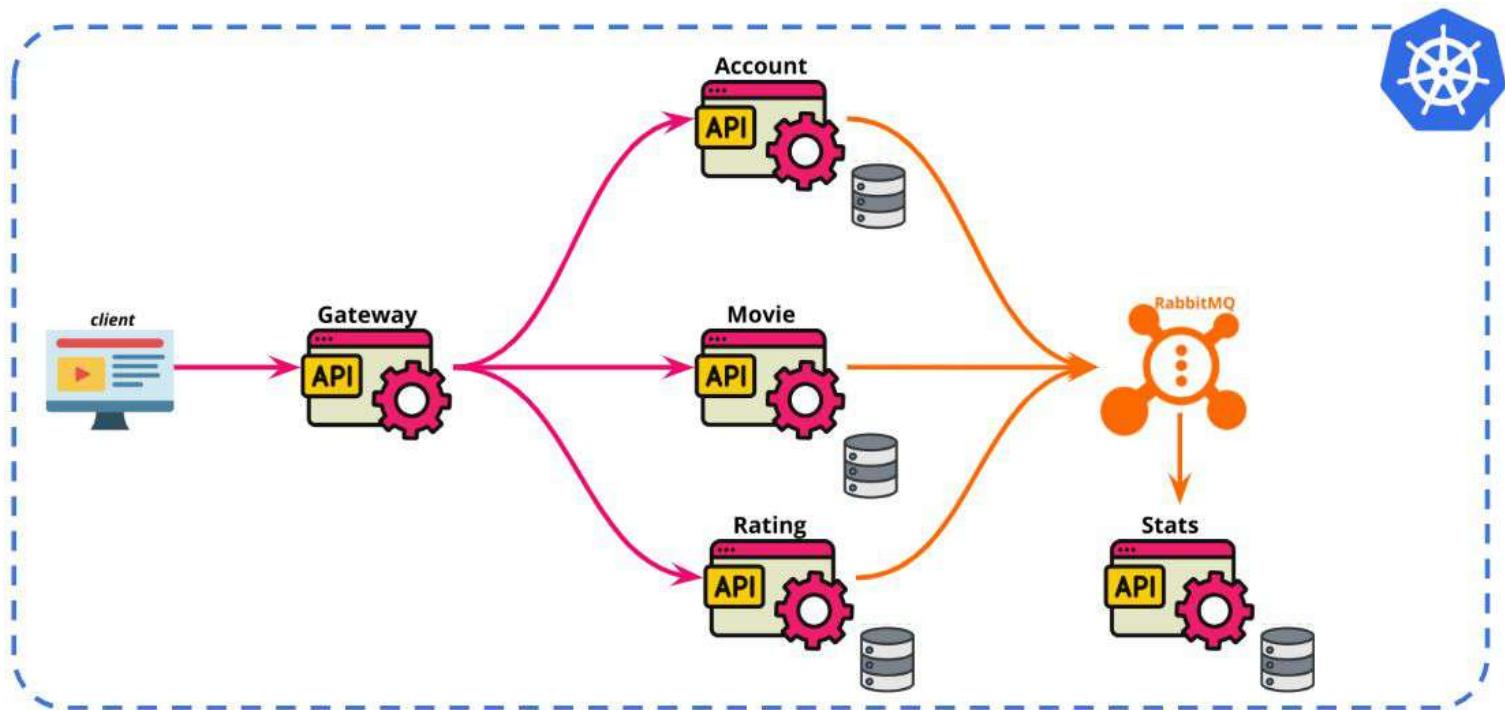
“



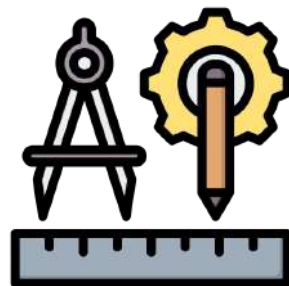
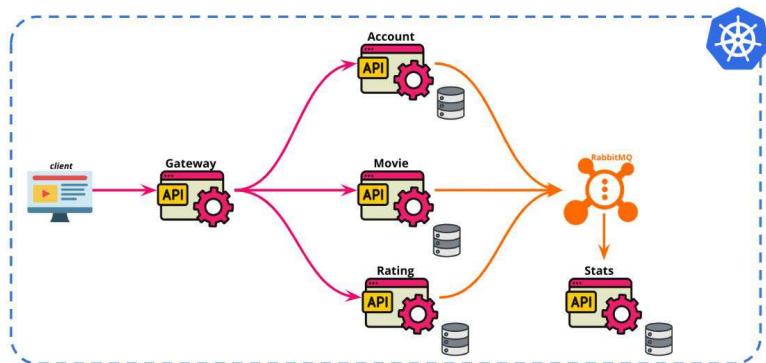
**Alors, comment peut-on  
détecter et comprendre ce qui  
ne va pas ?**

”

## Faire des mesures



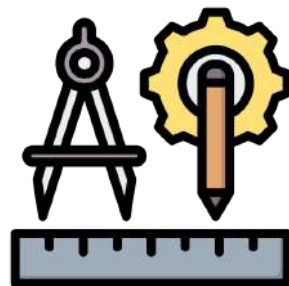
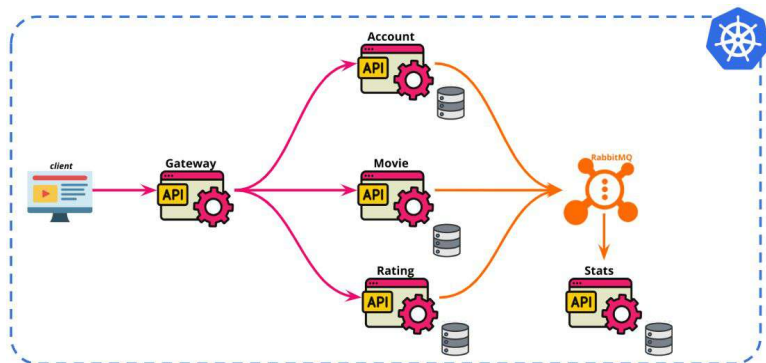
# Faire des mesures



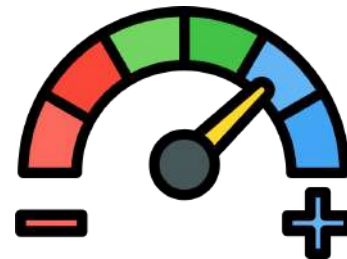
Mesures



## Faire des mesures



Mesures



Santé

## Quelques mesures courantes



Latences

## Quelques mesures courantes



**Latences**



**Nb d'erreurs**

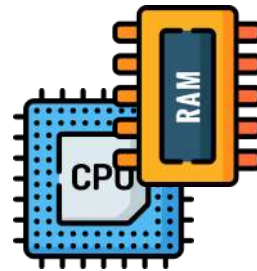
## Quelques mesures courantes



Latences



Nb d'erreurs



CPU / RAM

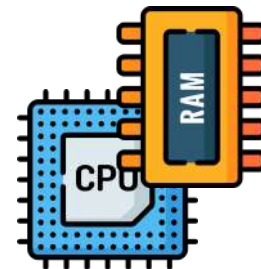
## Quelques mesures courantes



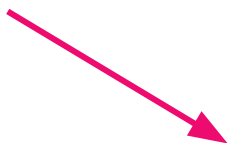
Latences



Nb d'erreurs



CPU / RAM



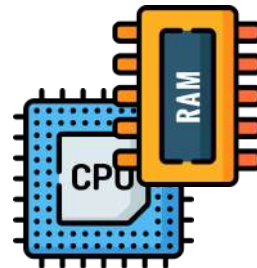
## Quelques mesures courantes



Latences



Nb d'erreurs



CPU / RAM

**Métriques**

“



**Les métriques te disent  
quand il y a un problème,  
mais pas pourquoi**

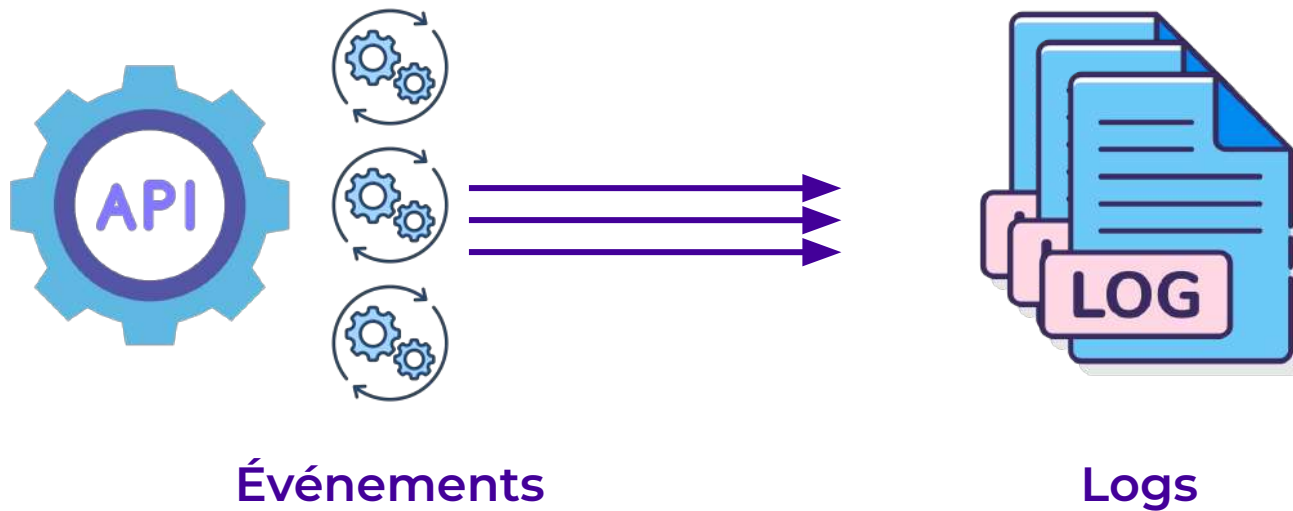
”

## Les logs

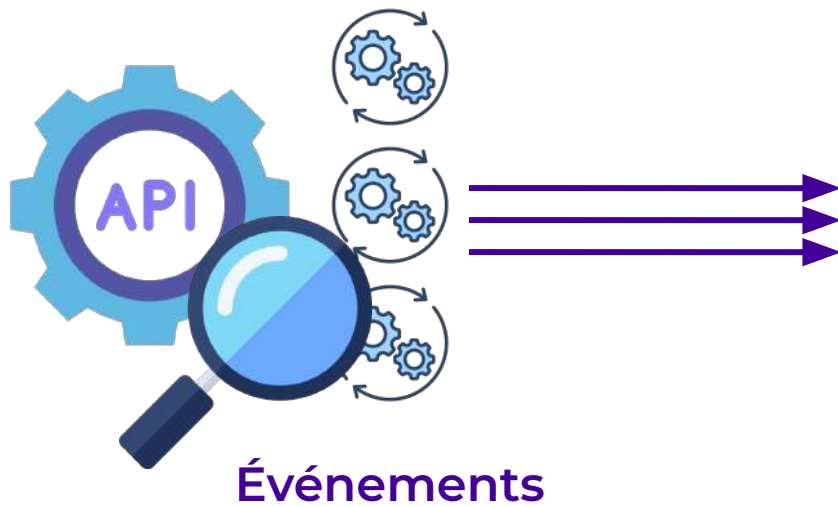




## Les logs



## Les logs



Logs

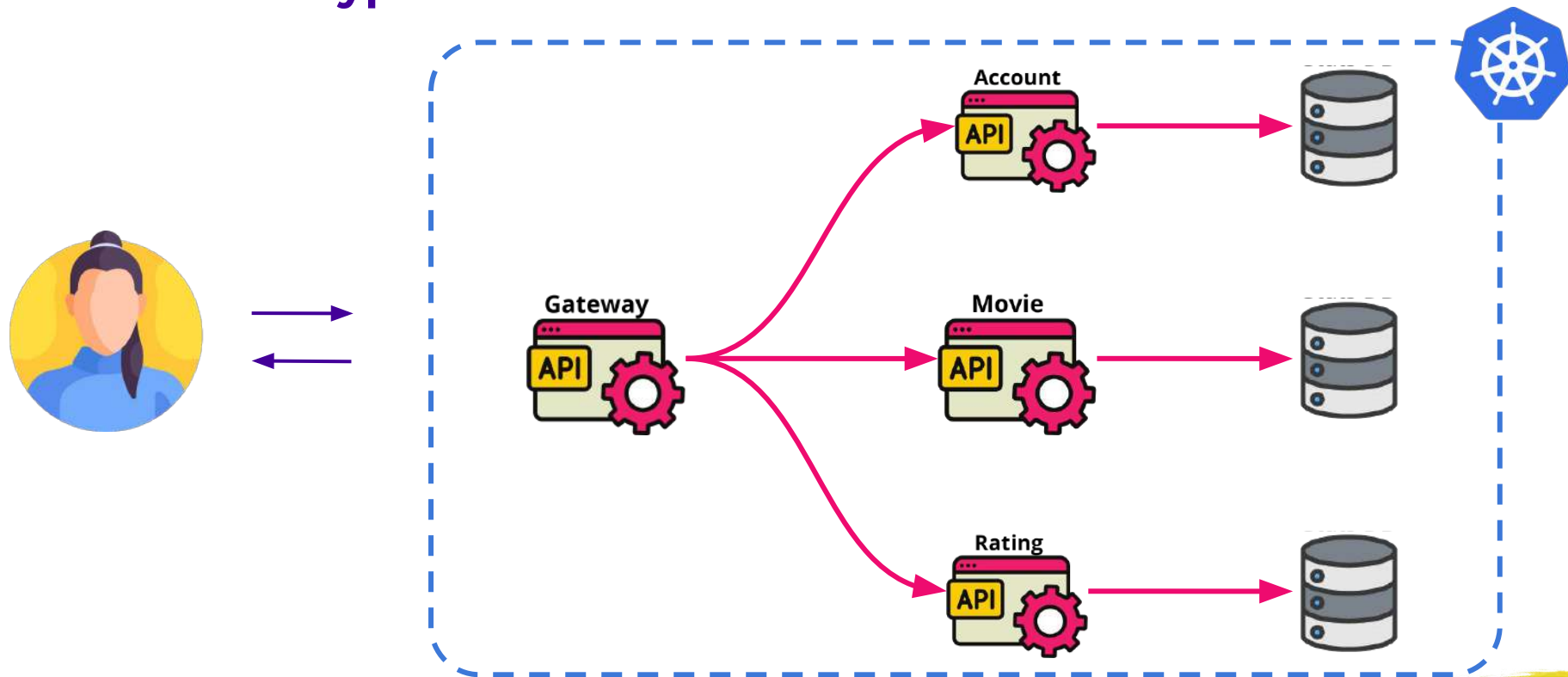
“



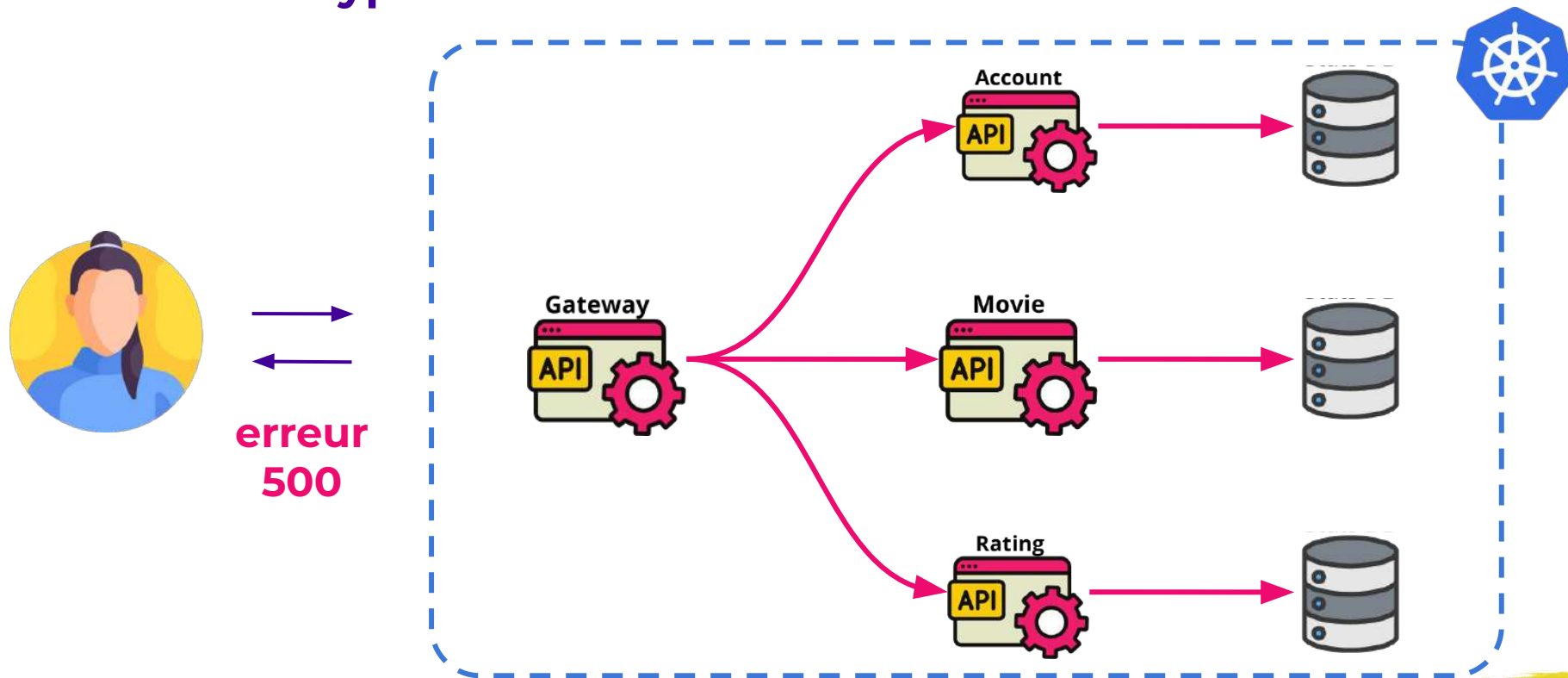
**C'est un bon début, mais les  
logs ont aussi leurs limites**

”

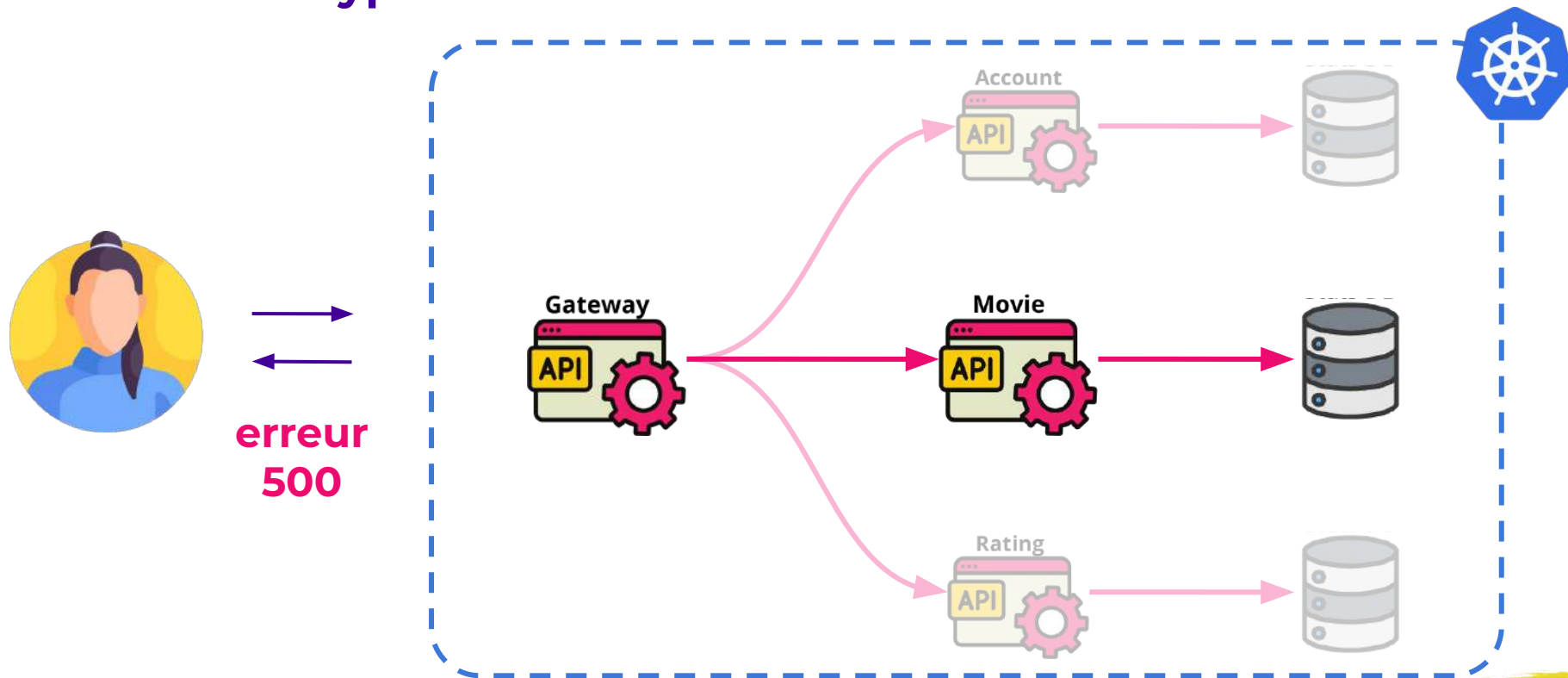
## Un dernier type de données : les traces



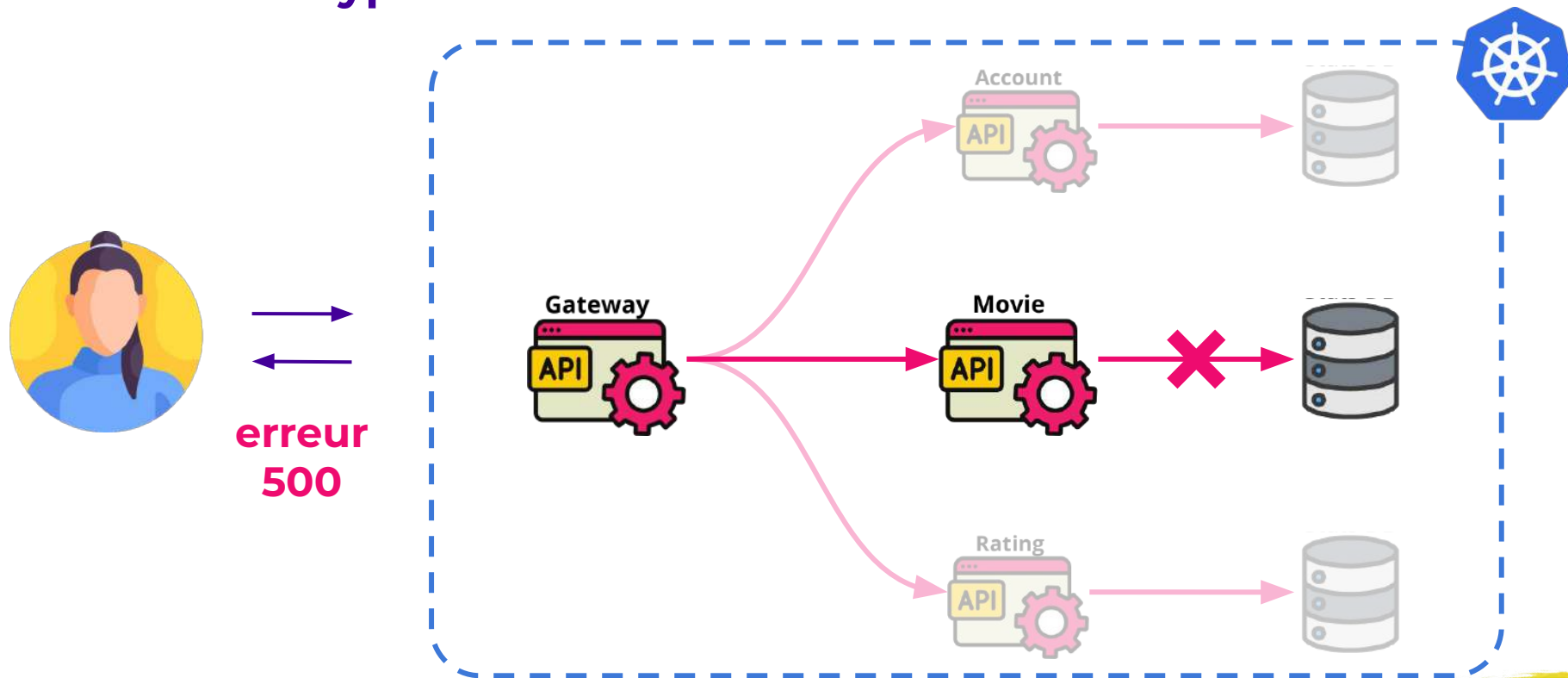
## Un dernier type de données : les traces



## Un dernier type de données : les traces



## Un dernier type de données : les traces



# FlameGraph

 GET /movie/categories/action ✓

50 ms

 GET movie-api ✓

30 ms

 DAO findAllByCategory ✓

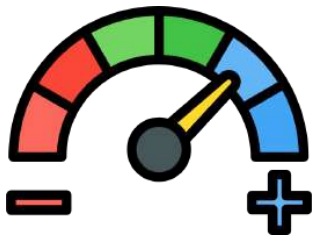
20 ms

 SELECT \* FROM movie  
WHERE category = 'action' ✗

10 ms



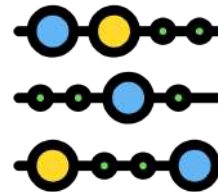
## 3 données



Métriques

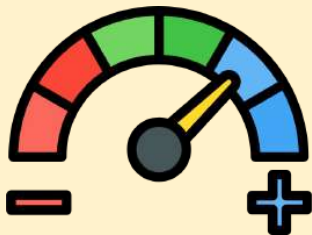


Logs



Traces

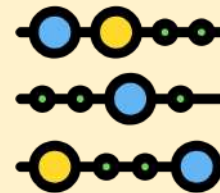
## 3 données



Métriques



Logs



Traces

**Signaux  
d'observabilité**

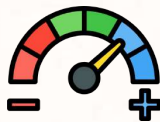
# En résumé

## Signaux d'observabilité

L'observabilité va plus loin que le monitoring, elle permet de **comprendre nos systèmes**.



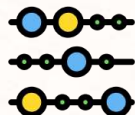
Elle est fondée sur **3 signaux**:



**Les métriques** : savoir quand



**Les logs** : savoir quoi



**Les traces** : savoir comment

“



**Comment obtenir  
ces signaux ?**

”

Comment obtenir ces données ?

## Comment on obtient nos signaux

Pour nos logs



Comment obtenir ces données ?

## Comment on obtient nos signaux

Pour nos logs

LOG4J  
SLF4J



fluentd logstash

Pour nos métriques



MICROMETER



Prometheus

Comment obtenir ces données ?

## Comment on obtient nos signaux

Pour nos logs



Pour nos métriques



Pour nos traces



Prometheus

Comment obtenir ces données ?

## Un écosystème fragmenté





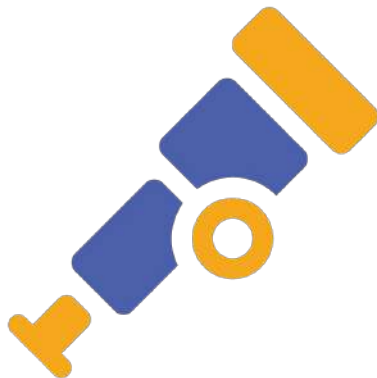
“



**L'USB-C de l'observabilité**

”

“

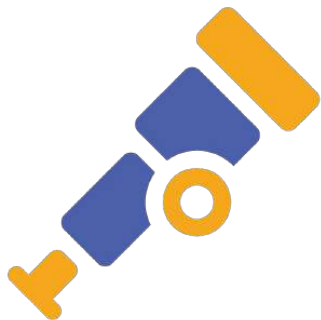


**OpenTelemetry**

”

Comment obtenir ces données ?

## OpenTelemetry



OpenTelemetry

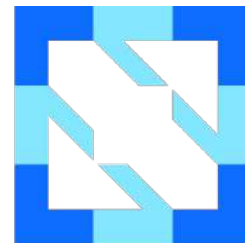
- ✓ Standard commun open-source
- ✓ Génération & Collecte
- ✓ Métriques + Logs + Traces
- ✓ De manière interopérable

Comment obtenir ces données ?

## OpenTelemetry



OpenTelemetry

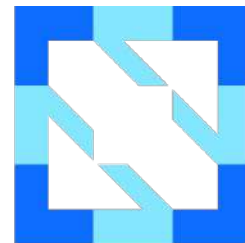


**CLOUD NATIVE**  
COMPUTING FOUNDATION

Comment obtenir ces données ?

## OpenTelemetry

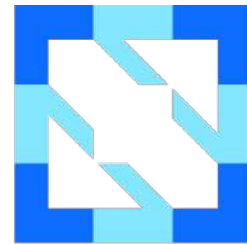
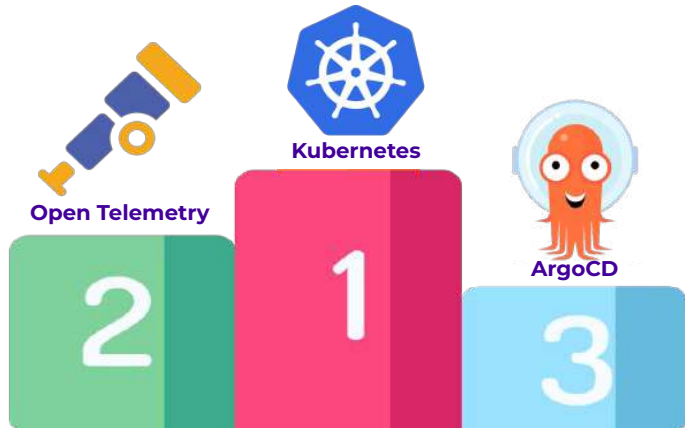
- ➔ Communauté open source
- ➔ Fondée en 2015
- ➔ +270k contributeurs
- ➔ +200 projets



**CLOUD NATIVE**  
COMPUTING FOUNDATION

Comment obtenir ces données ?

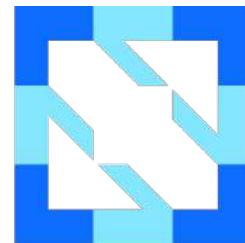
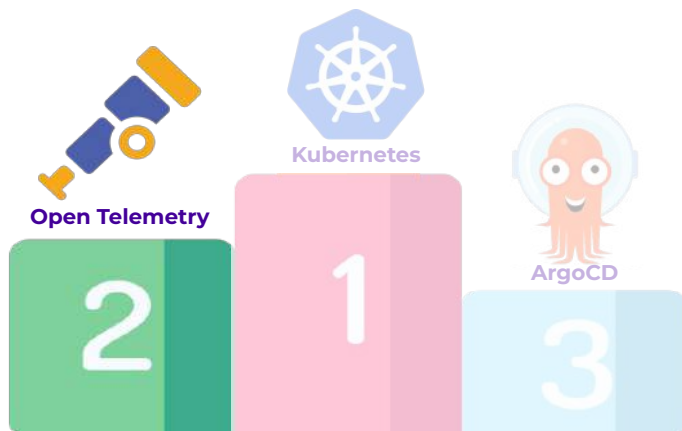
## OpenTelemetry



**CLOUD NATIVE**  
COMPUTING FOUNDATION

Comment obtenir ces données ?

## OpenTelemetry



**CLOUD NATIVE**  
**COMPUTING FOUNDATION**

“

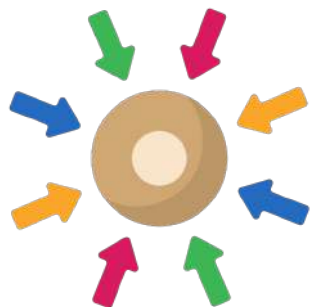


**Concrètement, c'est quoi ?**

”



## Concrètement



Spécifications



Documentation



Structure des signaux



Propagation, corrélation...



Protocoles utilisés

...



API Open Telemetry

## Concrètement



### Implémentations

→ *instrumenter nos applications*

### API Open Telemetry

- ✓ SDKs
- ✓ Bibliothèques
- ✓ Agents
- ✓ Collecteur

**Comment on fait ?**

## Comment on fait ?



Implémentations

API Open Telemetry



SDKs



Librairies



Agents



## Comment on fait ?



```
pom.xml

<dependency>
  <groupId>io.opentelemetry</groupId>
  <artifactId>opentelemetry-api</artifactId>
</dependency>
<dependency>
  <groupId>io.opentelemetry</groupId>
  <artifactId>opentelemetry-sdk</artifactId>
</dependency>
```

API Open Telemetry



Un SDK Java...



## Comment on fait ?

**Solution simple**

**Limiter l'impact sur le code**

**Ne pas avoir 15 000 outils**

**Open Source**

**API Open Telemetry**



**Un SDK Java...**



## Comment on fait ?

✗ Solution simple

Limiter l'impact sur le code

Ne pas avoir 15 000 outils

Open Source

API Open Telemetry



Un SDK Java...



## Comment on fait ?

- ✗ Solution simple
  - ✗ Limiter l'impact sur le code
- Ne pas avoir 15 000 outils
- Open Source

API Open Telemetry



Un SDK Java...





## Comment on fait ?



```
pom.xml

<dependency>
  <groupId>io.opentelemetry.instrumentation</groupId>
  <artifactId>opentelemetry-spring-boot-starter</artifactId>
</dependency>
```

### API Open Telemetry



Un SDK Java...



**Un starter Spring Boot !**

## Comment on fait ?

**Solution simple**

**Limiter l'impact sur le code**

**Ne pas avoir 15 000 outils**

**Open Source**

**API Open Telemetry**



**Un SDK Java...**



**Un starter Spring Boot !**

## Comment on fait ?



### Solution simple

Limiter l'impact sur le code

Ne pas avoir 15 000 outils

Open Source

### API Open Telemetry



Un SDK Java...



Un starter Spring Boot !

## Comment on fait ?



**Solution simple**



**Limiter l'impact sur le code**

**Ne pas avoir 15 000 outils**

**Open Source**

**API Open Telemetry**



**Un SDK Java...**



**Un starter Spring Boot !**

## Comment on fait ?



Agent



App Java

### API Open Telemetry



- ➔ Un SDK Java...
- ➔ Un starter Spring Boot !
- ➔ Un agent JVM !!



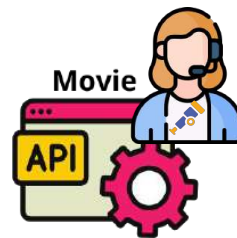
## Utiliser l'agent Java (recommandé)



JAR



Agent



Application  
Auto-Instrumentée

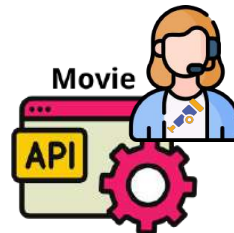
## Utiliser l'agent Java (recommandé)

**Solution simple**

**Limiter l'impact sur le code**

**Ne pas avoir 15 000 outils**

**Open Source**



**Application  
Auto-Instrumentée**

## Utiliser l'agent Java (recommandé)

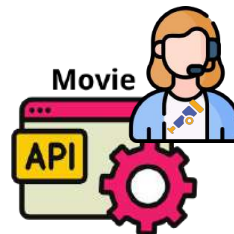


**Solution simple**

**Limiter l'impact sur le code**

**Ne pas avoir 15 000 outils**

**Open Source**



**Application  
Auto-Instrumentée**

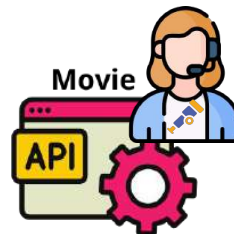


## Utiliser l'agent Java (recommandé)

- ✓ Solution simple
- ✓ Limiter l'impact sur le code

Ne pas avoir 15 000 outils

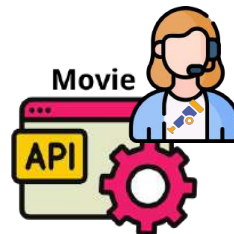
Open Source



Application  
Auto-Instrumentée

## Utiliser l'agent Java (recommandé)

- ✓ Solution simple
- ✓ Limiter l'impact sur le code
- ✓ Ne pas avoir 15 000 outils
- ✓ Open Source



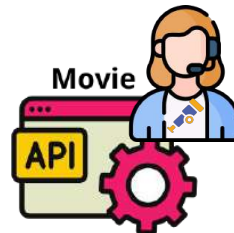
Application  
Auto-Instrumentée

## Utiliser l'agent Java (recommandé)



### Overhead:

CPU,  
Mémoire,  
Latence,  
Réseau,  
Démarrage



Application  
Auto-Instrumentée

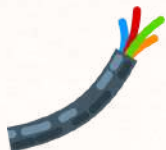
# En résumé



Signaux d'observabilité



OpenTelemetry



De nombreuses solutions existent



**OpenTelemetry** devient LE standard pour unifier la collecte de **tous les signaux**

Des **implémentations** sont disponibles pour instrumenter nos applications



En **Java**, il est recommandé d'utiliser l'**agent JVM**



OpenTelemetry **ne gère pas** le stockage de la donnée collectée



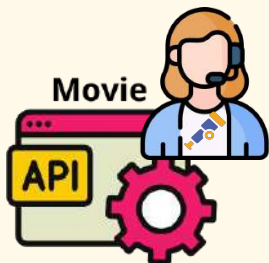
“

## Démo

Ajoutons l'agent Java OpenTelemetry

”

# On en est où?



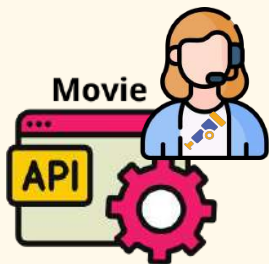
Application  
auto-Instrumentée



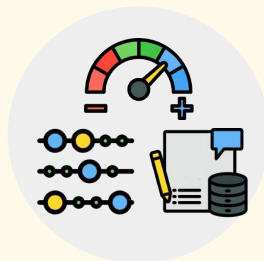
Signaux



# On en est où?



Application  
auto-Instrumentée



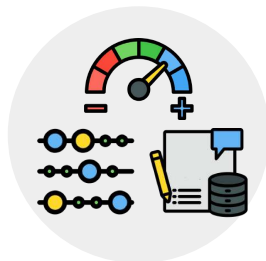
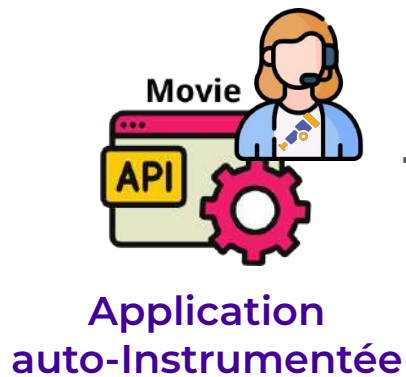
Signaux



**Exploiter les données**



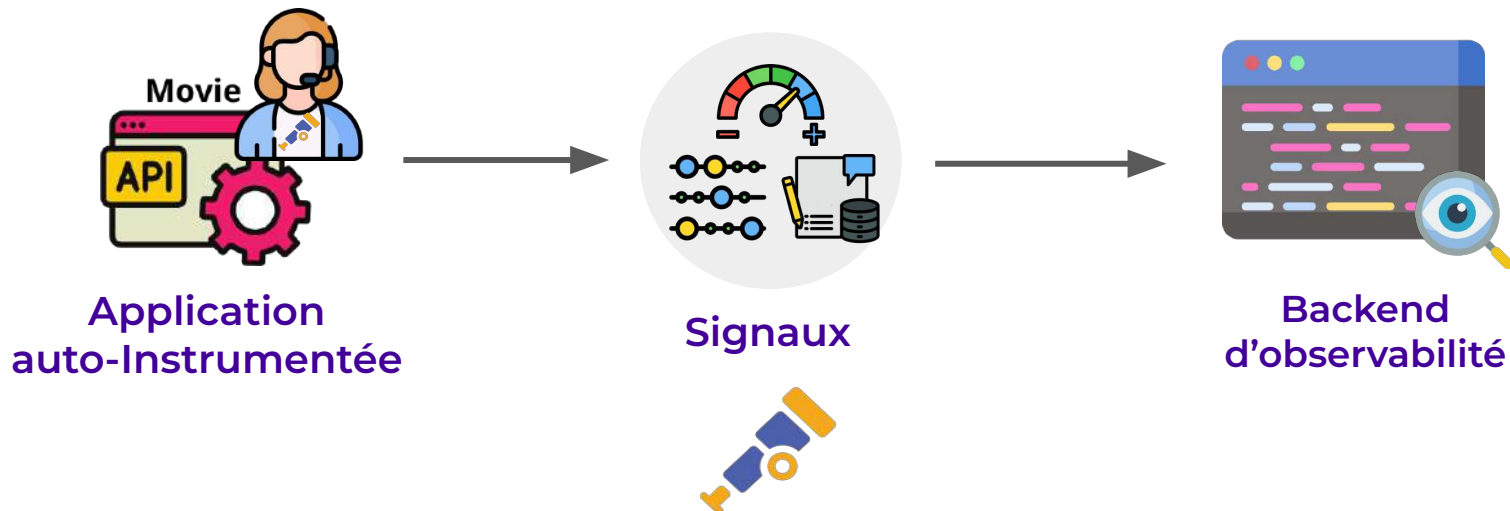
# Récapitulons



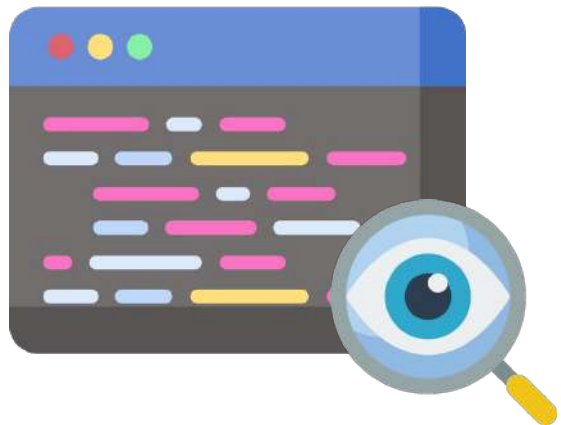
Signaux



# Récapitulons



## Backend d'Observabilité



**Stockage**



**Visualisation**



# Ecosystème

## Stockage



## Visualisation



Exploiter les données

# Ecosystème



“



Les APMs

”

“



# Les APMs

## Stockage unifié + Visualisation

”

# Ecosystème



**new relic**



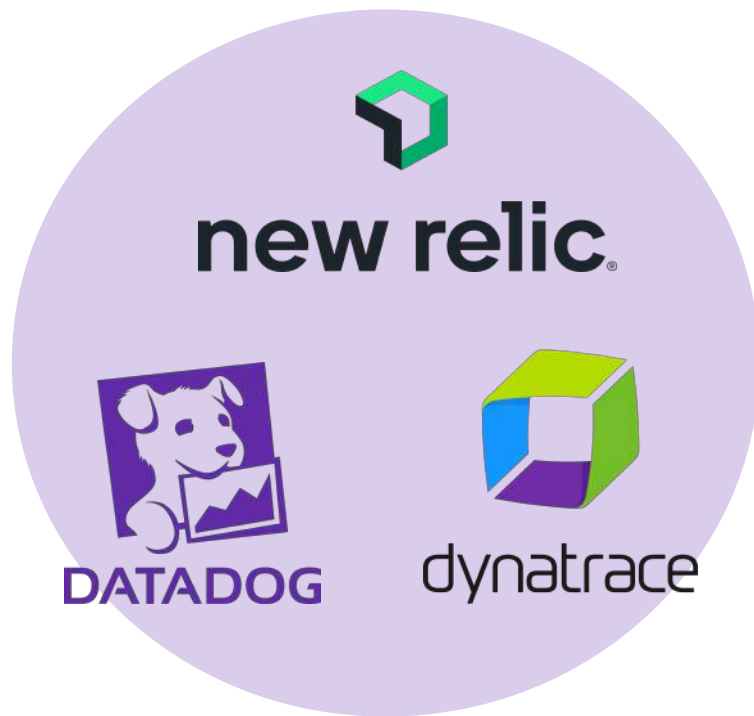
**DATADOG**



**dynatrace**



## Ecosystème



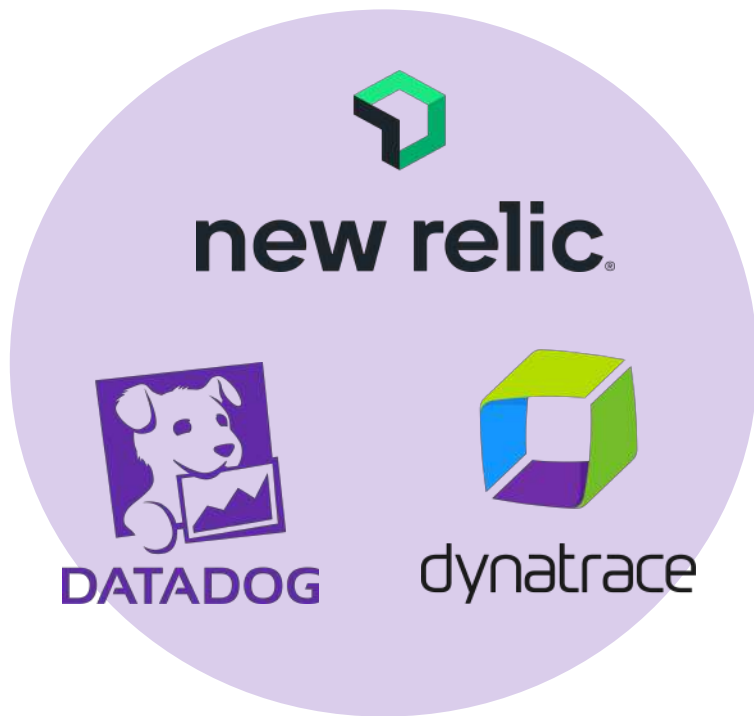
**Solution simple**

**Limiter l'impact sur notre code**

**Ne pas avoir 15 000 outils**

**Open Source**

## Ecosystème



- ✓ Solution simple
  - ✓ Limiter l'impact sur notre code
  - ✓ Ne pas avoir 15 000 outils
- Open Source

## Ecosystème



- ✓ Solution simple
- ✓ Limiter l'impact sur notre code
- ✓ Ne pas avoir 15 000 outils
- ✗ Open Source

“



SigNoz

”

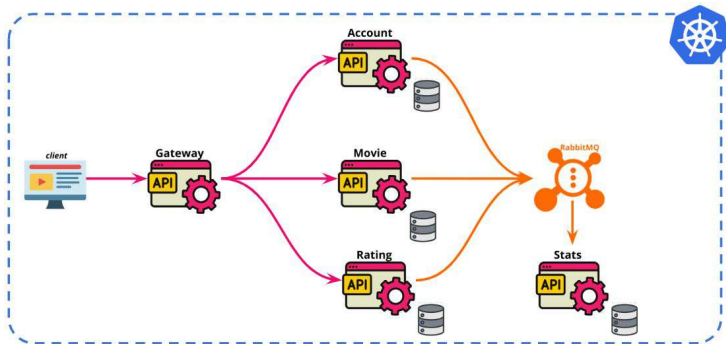
## On a choisi ici de partir avec



- ✓ **Open-Source**
- ✓ **Tout en un**
- ✓ **OpenTelemetry-first**

Exploiter les données

## Mise en place de Signoz



+



SigNoz

## Signoz sur notre cluster

```
.env  
  
OTEL_LOGS_EXPORTER=otlp  
OTEL_TRACES_EXPORTER=otlp  
OTEL_METRICS_EXPORTER=otlp  
OTEL_RESOURCE_ATTRIBUTES="service.name=movie-api"  
OTEL_EXPORTER_OTLP_ENDPOINT=?
```



**Signoz**

“



**C'est parti !**

”



# En résumé



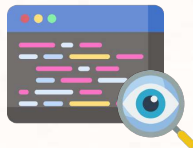
Signaux d'observabilité



OpenTelemetry



Backend d'observabilité



Pour exploiter nos signaux, on a besoin d'un **Backend d'observabilité**



Pleins de solutions complémentaires **pour stocker** des signaux ou **les visualiser**



Le tout en un : **les APMs**



Ils ingèrent **tous nos signaux** pour les visualiser

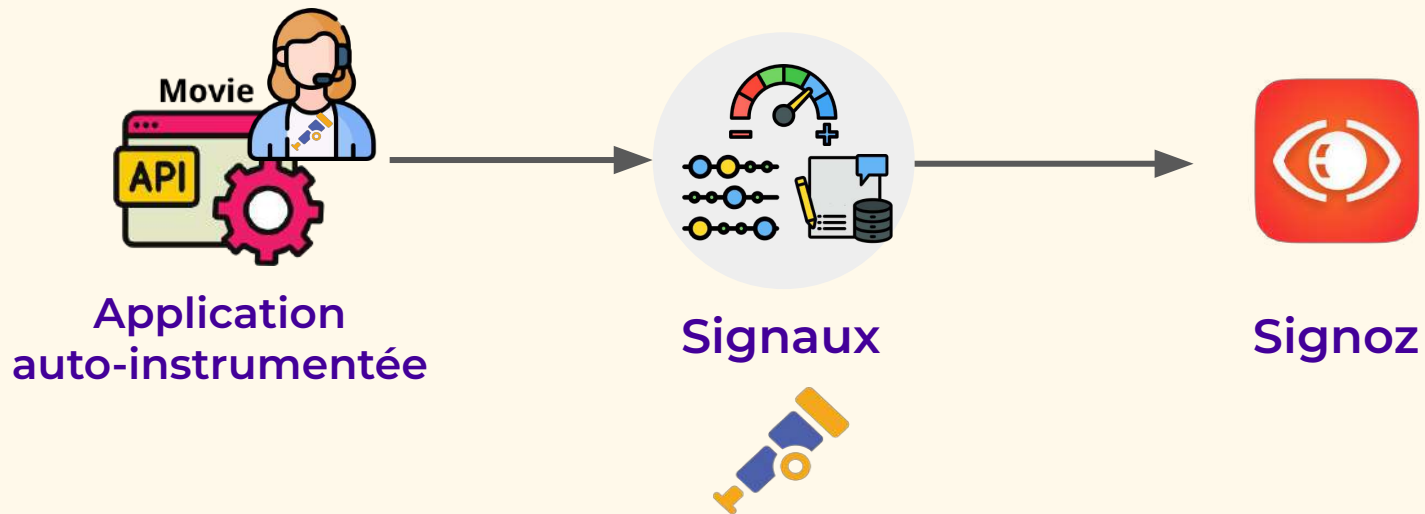


Des solutions **open source** voient le jour



C'est le cas de **Signoz**

# On en est où?



“

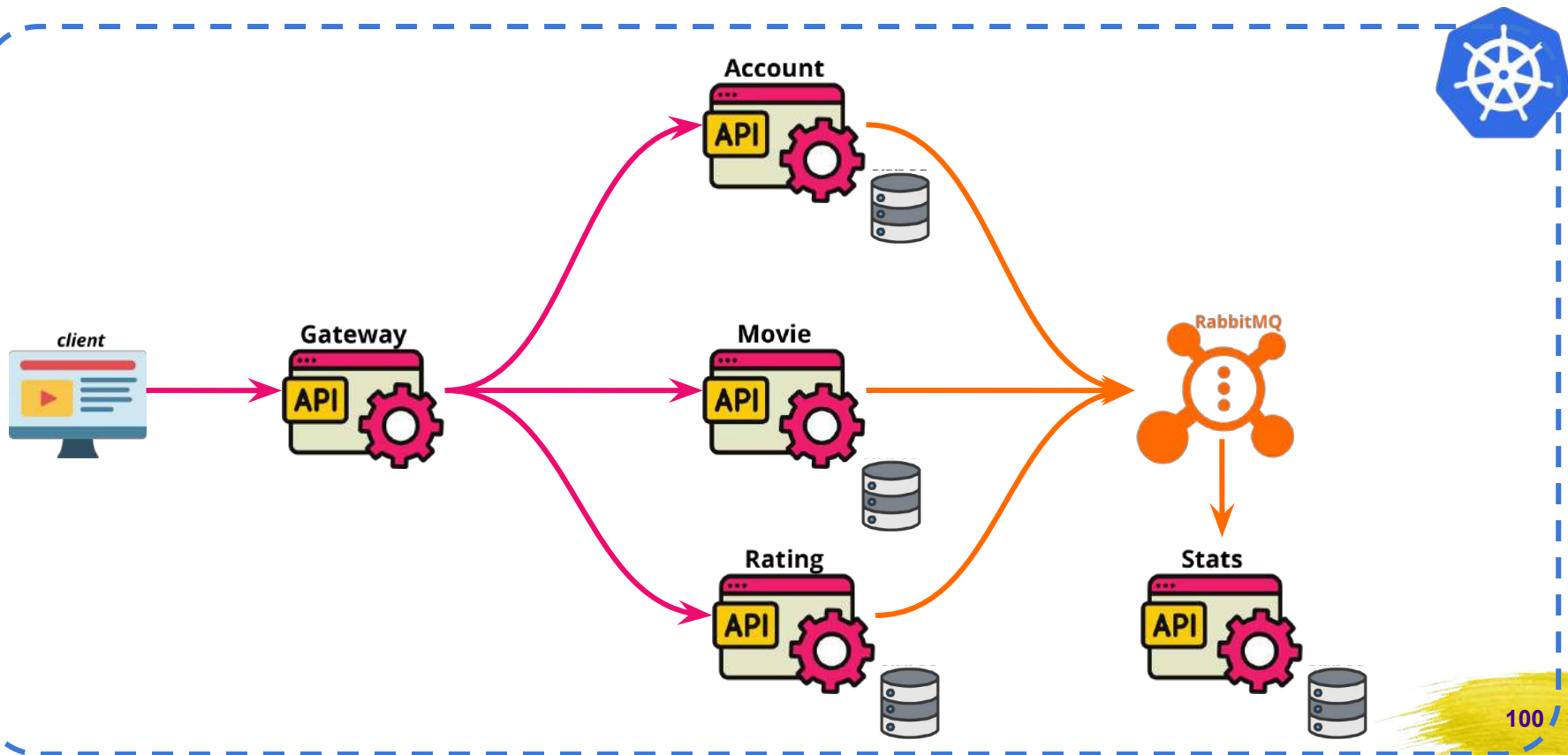


**On avait pas une infra aussi ?**

”

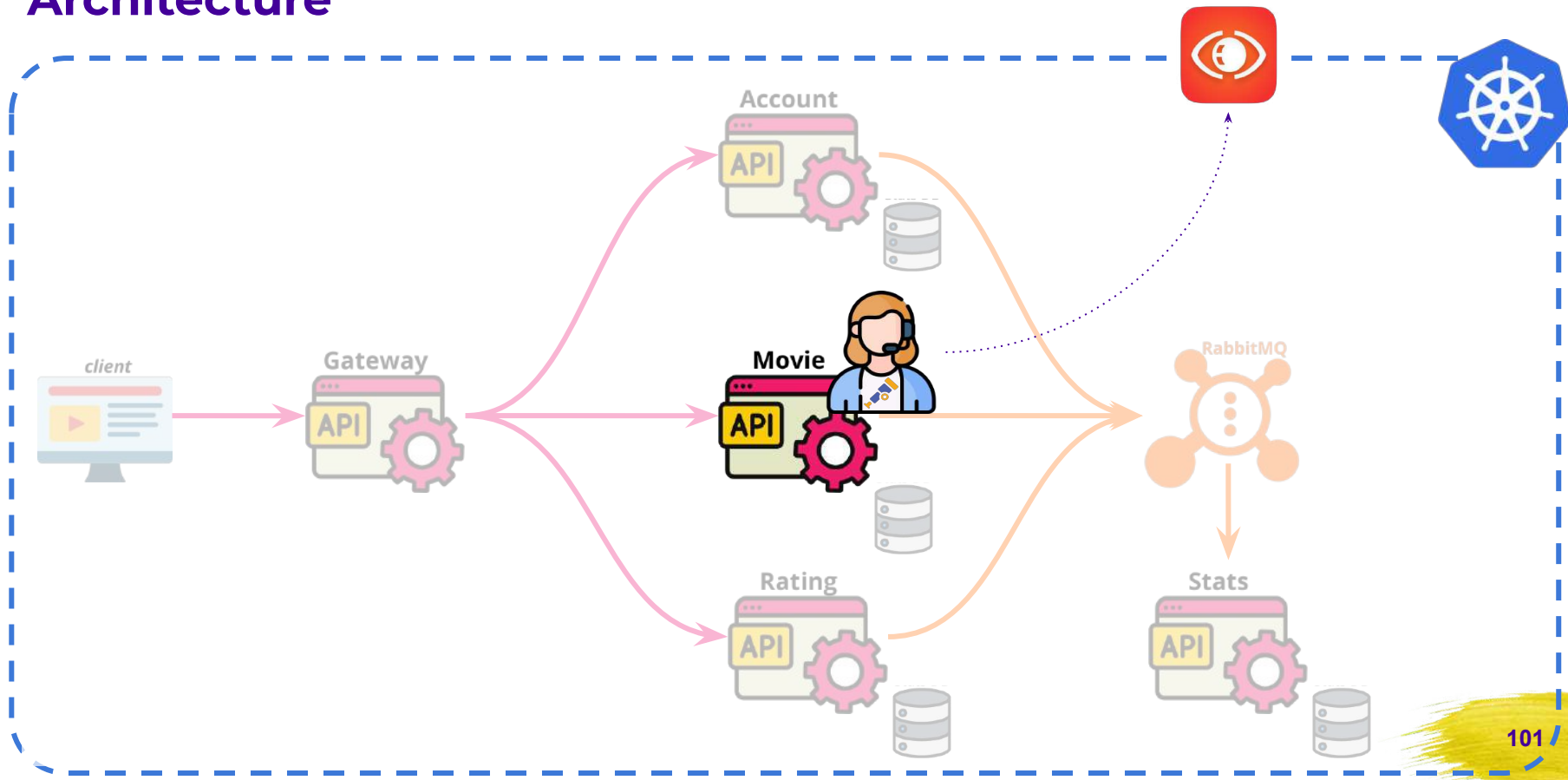
On avait pas une infra aussi ?

## Architecture

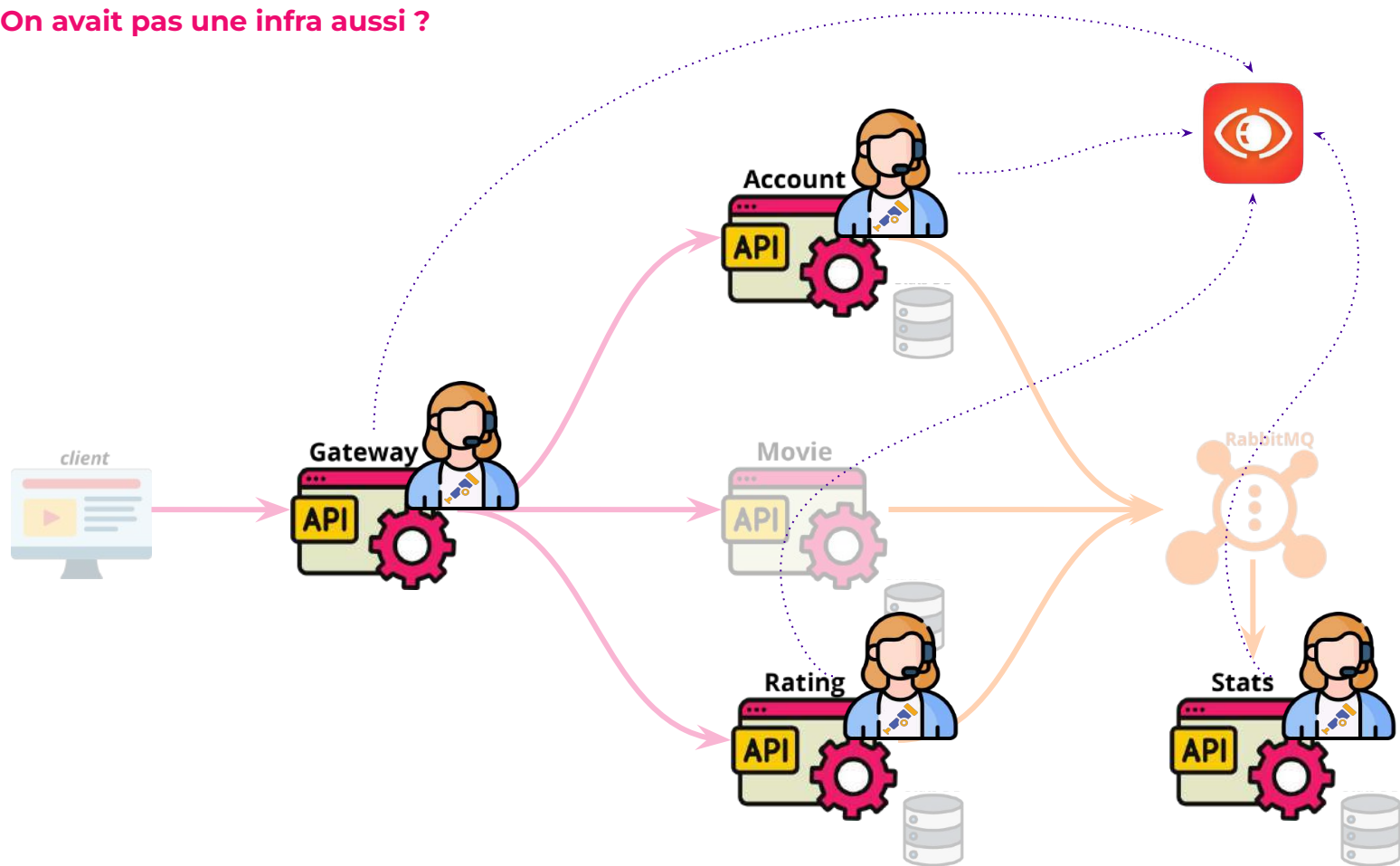


On avait pas une infra aussi ?

## Architecture



On avait pas une infra aussi ?



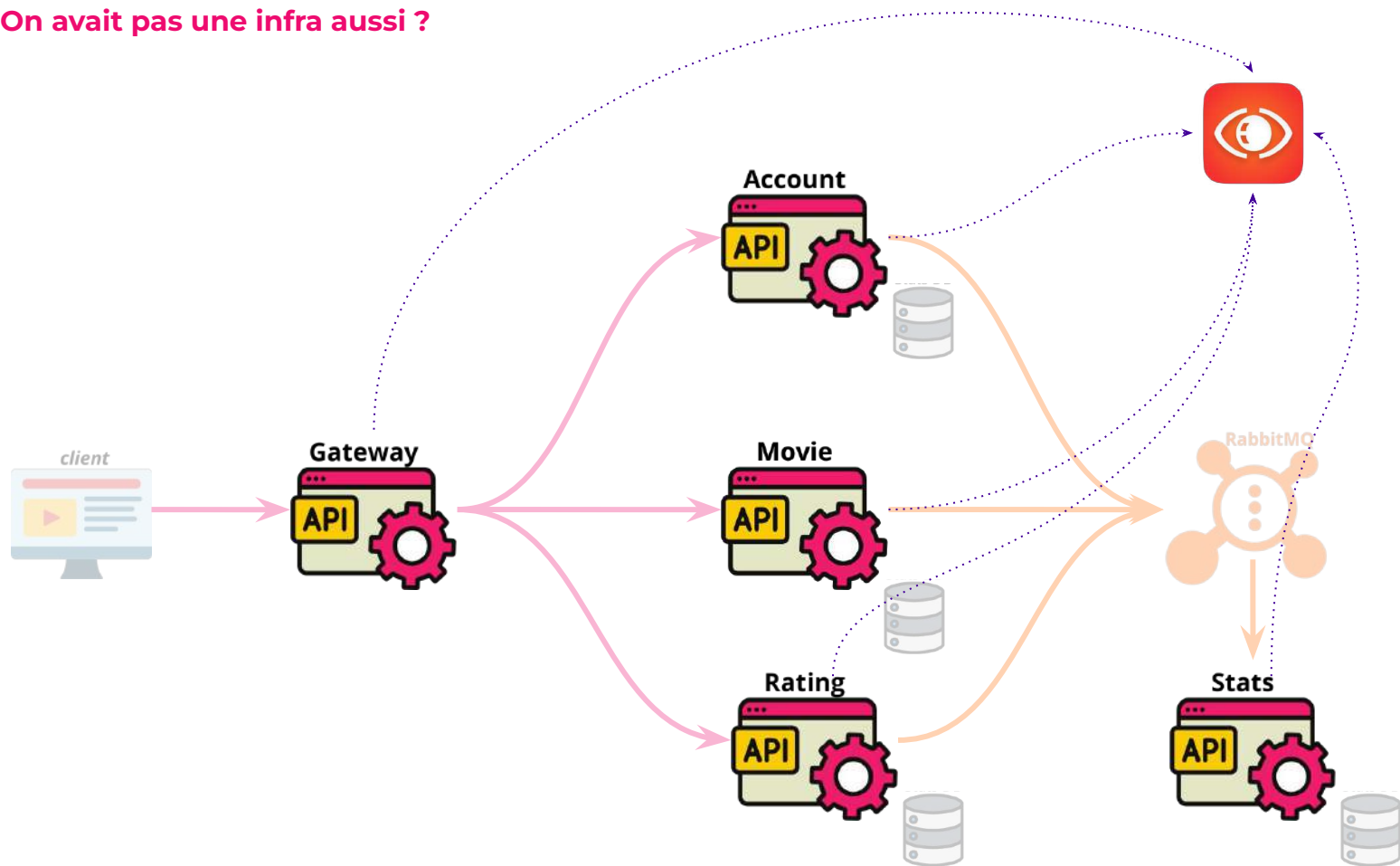
“



Démo

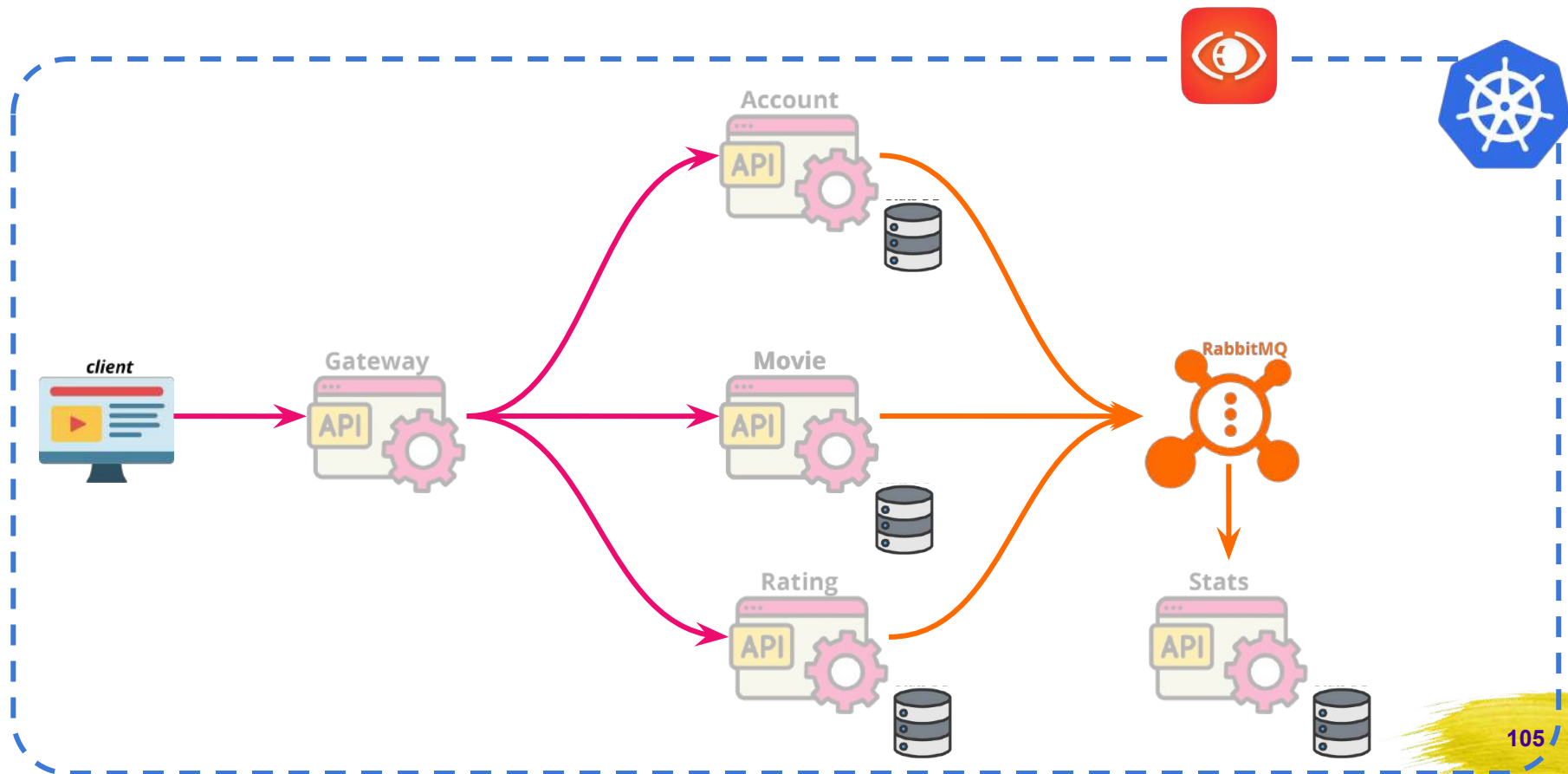
”

On avait pas une infra aussi ?

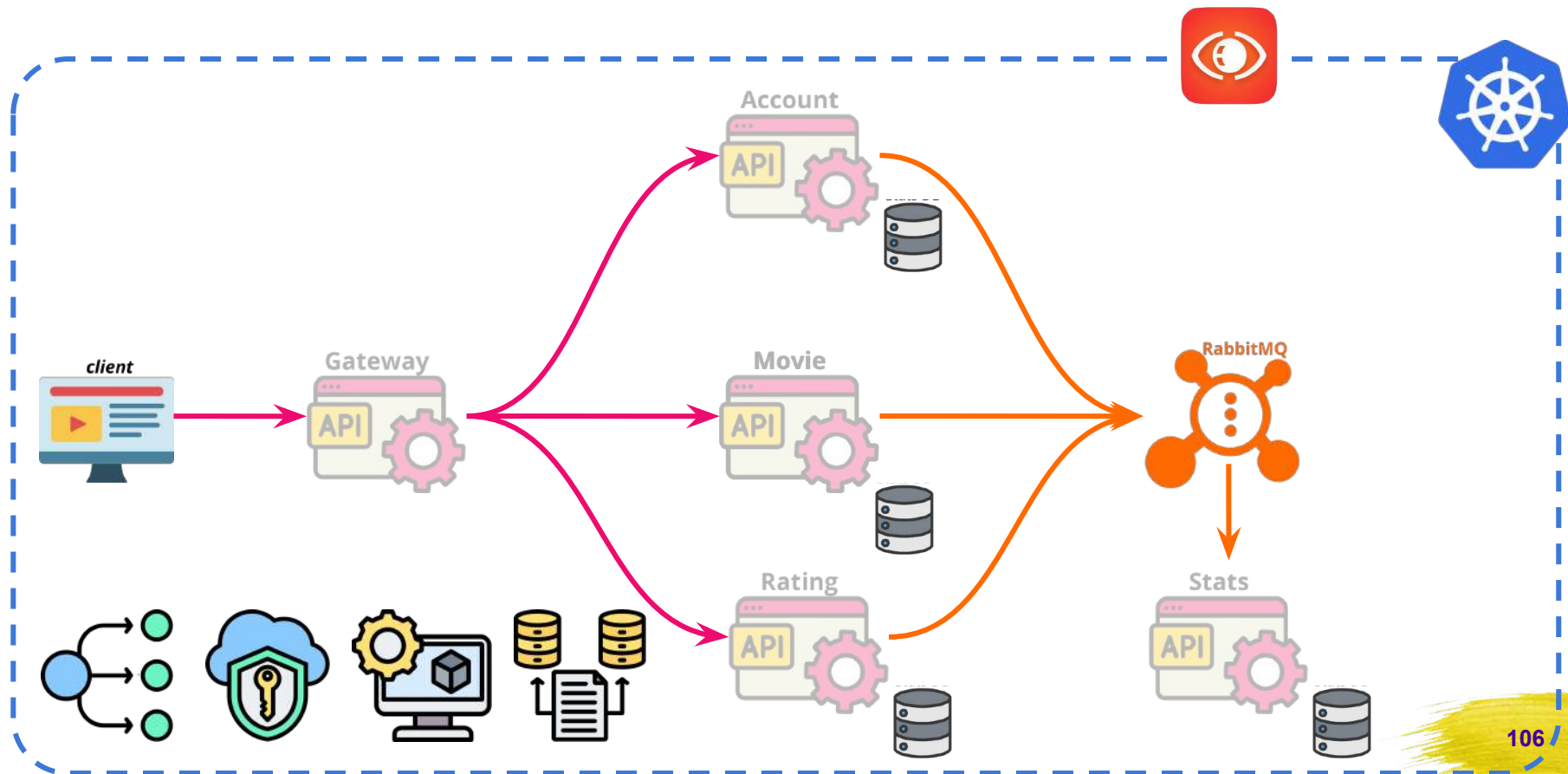




On avait pas une infra aussi ?



On avait pas une infra aussi ?



On avait pas une infra aussi ?

## Comment faire pour mon infra ?



**Pleins de composants à surveiller  
sur notre infra**



**Chacun présente des enjeux  
différents**

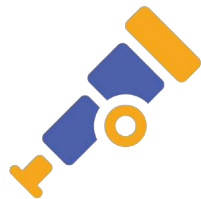


**Et des formats de données différents**



On avait pas une infra aussi ?

## Le collecteur



OpenTelemetry

COLLECTOR

On avait pas une infra aussi ?

## Le collecteur

Implémenté en Go



Prêt à être déployé



Collecte nos ressources

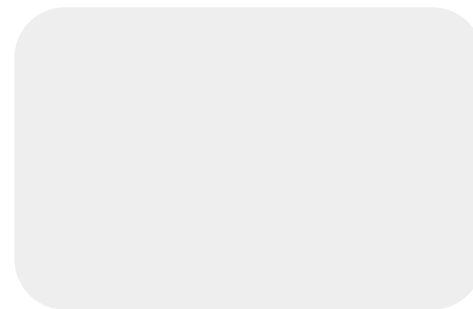
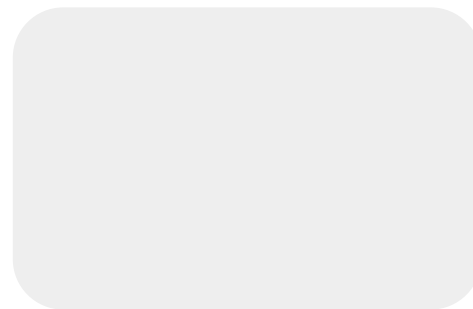
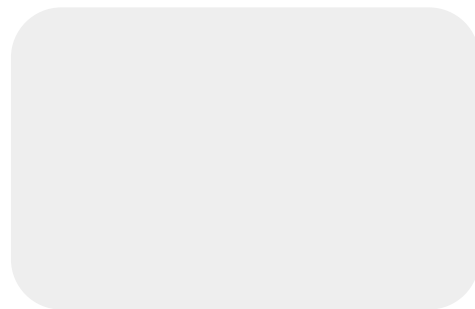
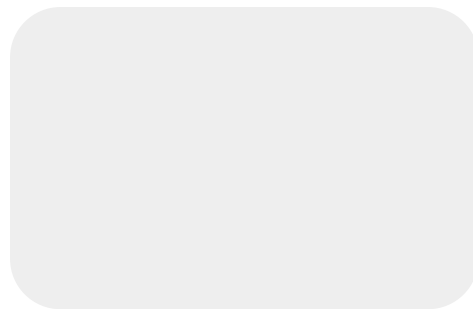
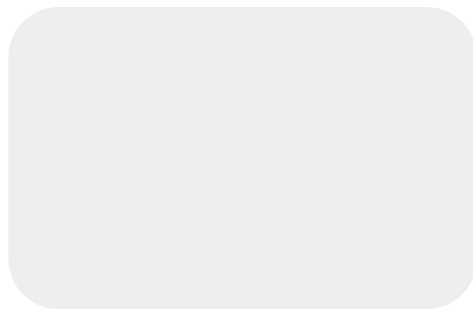
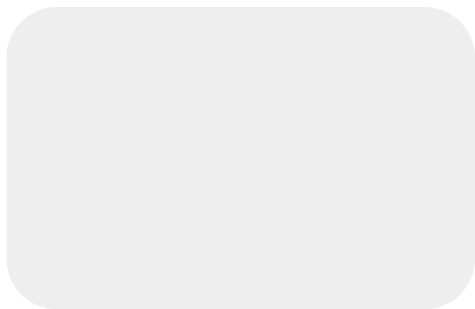


Entièrement configurable



On avait pas une infra aussi ?

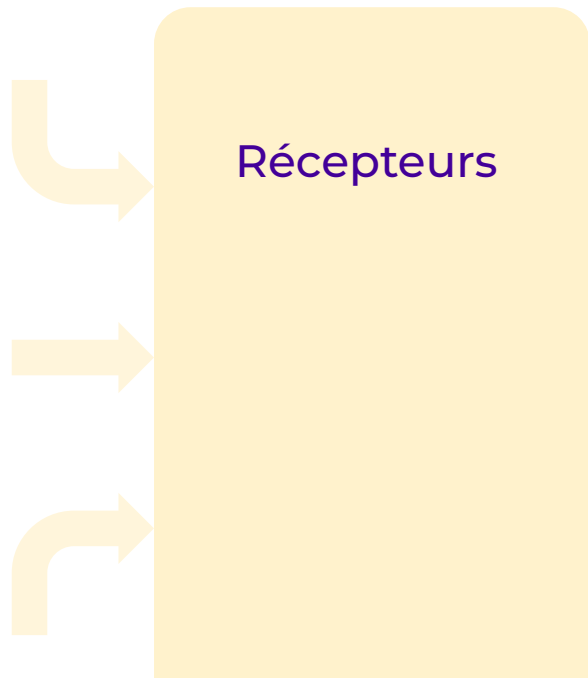
## Le collecteur



**COLLECTOR**

On avait pas une infra aussi ?

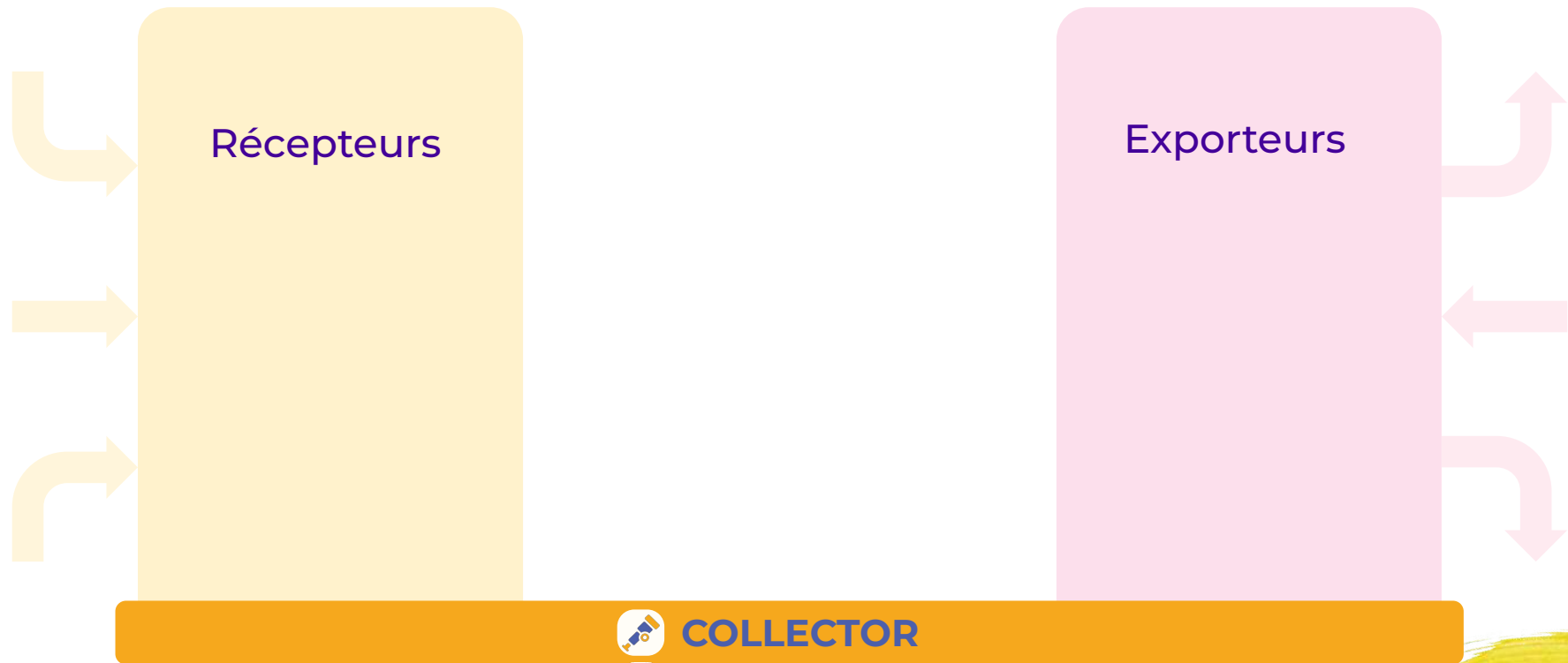
## Le collecteur



**COLLECTOR**

On avait pas une infra aussi ?

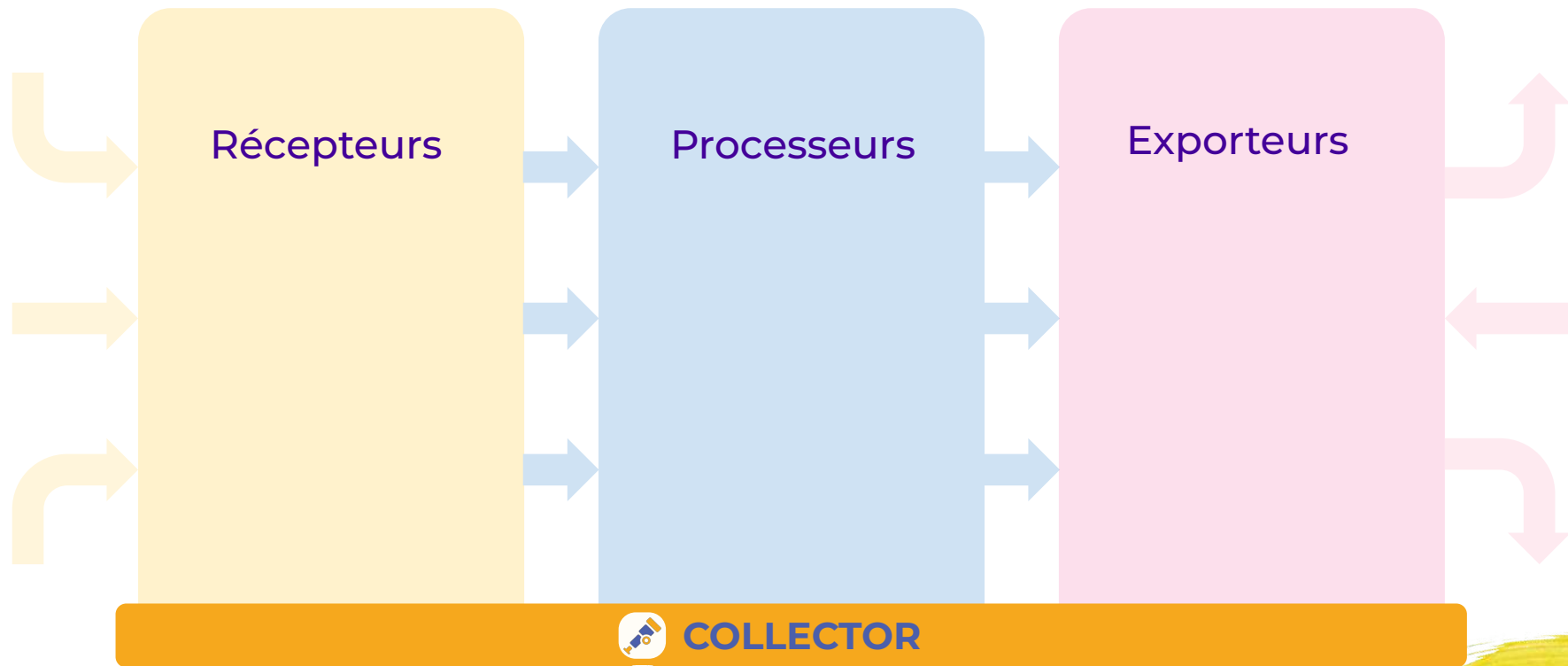
## Le collecteur





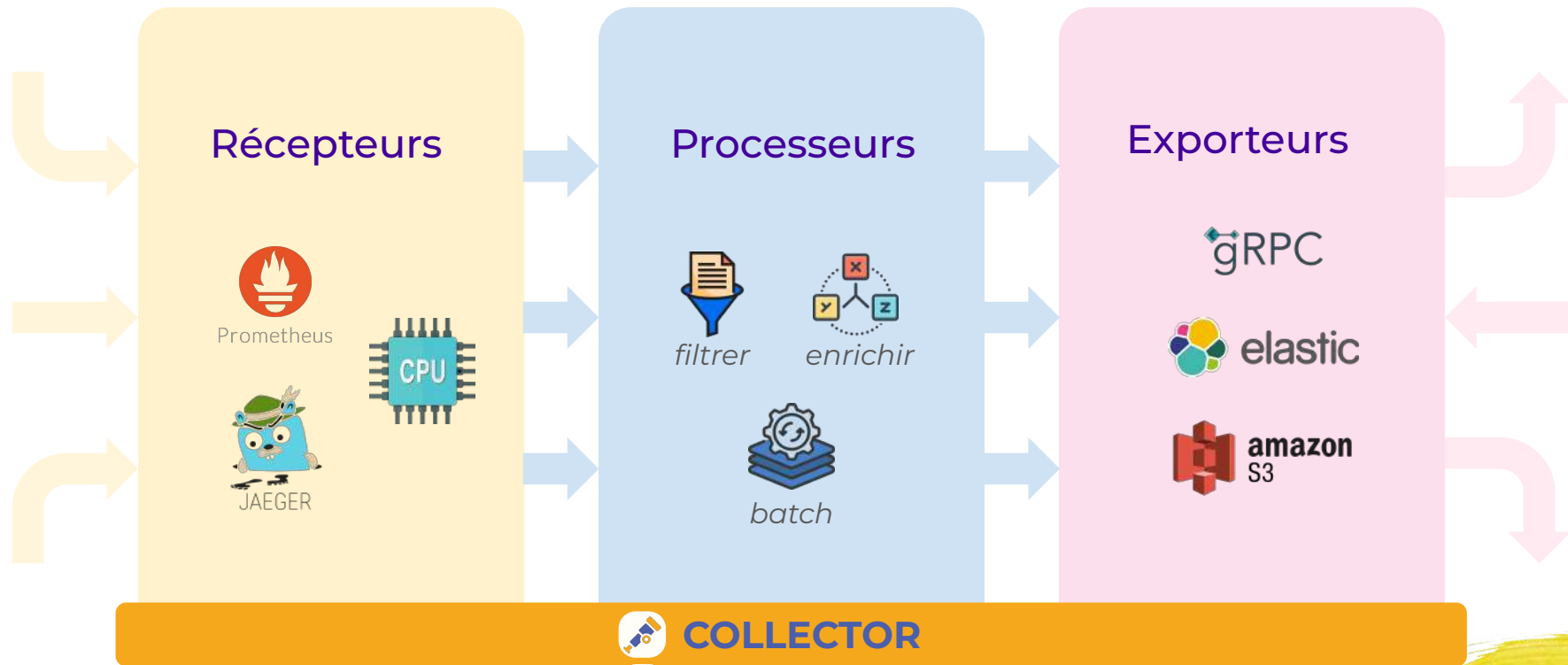
On avait pas une infra aussi ?

## Le collecteur



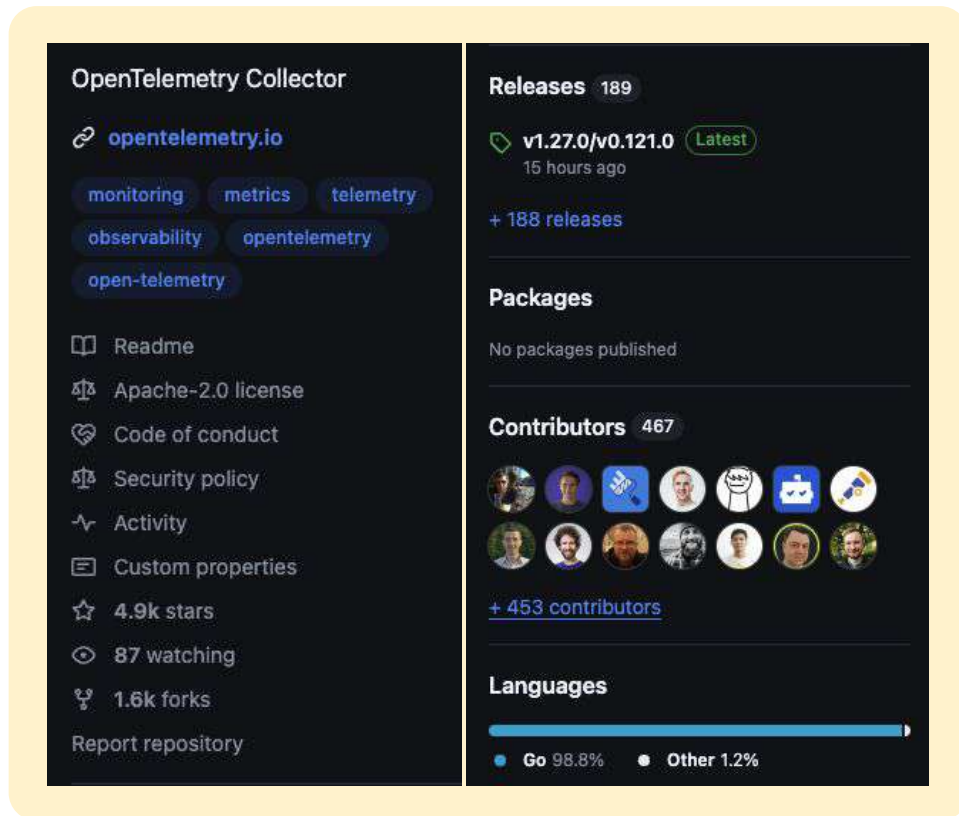
On avait pas une infra aussi ?

## Le collecteur



On avait pas une infra aussi ?

## Le collecteur



The screenshot displays the GitHub repository for the OpenTelemetry Collector. The left sidebar contains links to the repository's documentation and resources, including the README, license, code of conduct, security policy, activity, custom properties, star count (4.9k), watching count (87), fork count (1.6k), and a link to report a repository issue. The main content area is divided into several sections: 'Releases' showing the latest version (v1.27.0/v0.121.0) and a link to view all 189 releases; 'Packages' indicating no packages are published; 'Contributors' showing a list of 467 contributors; and 'Languages' showing a bar chart with Go at 98.8% and Other at 1.2%.

**OpenTelemetry Collector**

[opentelemetry.io](https://opentelemetry.io)

monitoring metrics telemetry observability opentelemetry open-telemetry

Readme  
Apache-2.0 license  
Code of conduct  
Security policy  
Activity  
Custom properties  
4.9k stars  
87 watching  
1.6k forks  
Report repository

**Releases** 189

v1.27.0/v0.121.0 **Latest**  
15 hours ago

+ 188 releases

**Packages**

No packages published

**Contributors** 467

+ 453 contributors

**Languages**

Go 98.8% Other 1.2%

“



- ✓ **Solution simple**
- ✓ **Limiter l'impact sur notre code**
- ✓ **Ne pas avoir 15 000 outils**
- ✓ **Open Source**

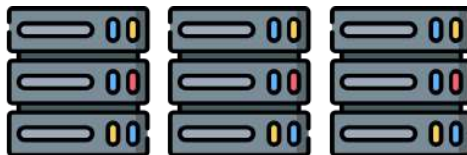
”

# Configurer notre collecteur

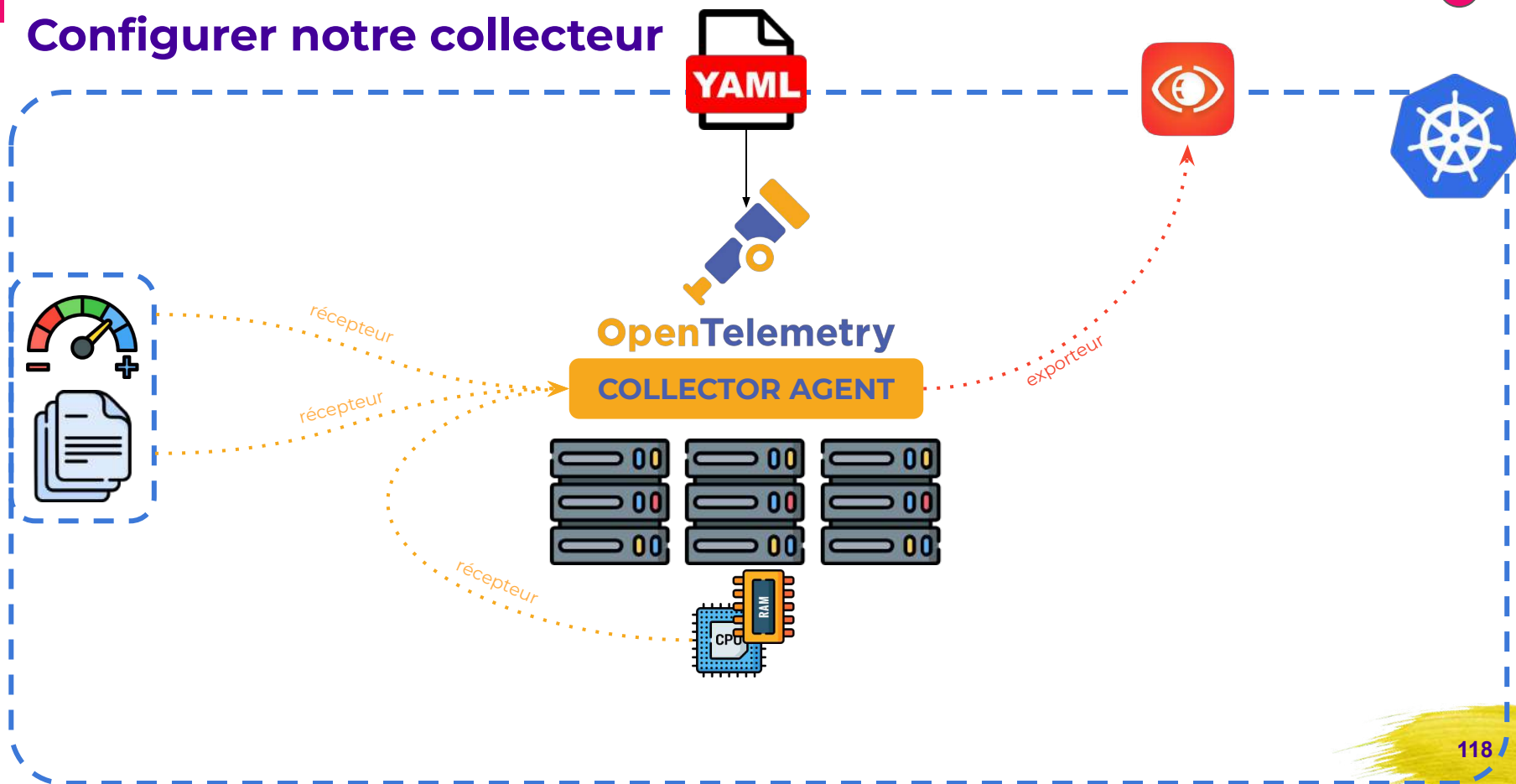


OpenTelemetry

COLLECTOR AGENT



## Configurer notre collecteur



“

Démo

”

# En résumé



Signaux d'observabilité



OpenTelemetry



Backend d'observabilité



Collecter l'infrastructure



On a **instrumenté** nos applications pour en collecter les signaux



Pour tout le reste on peut utiliser **le collecteur OpenTelemetry**



C'est un composant open-source **prêt à être déployé** sur notre infra



# En résumé



Signaux d'observabilité



OpenTelemetry



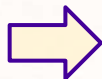
Backend d'observabilité



Collecter l'infrastructure



Il est **hautement configurable** pour s'adapter à nos besoins



Une ressource = un **récepteur**

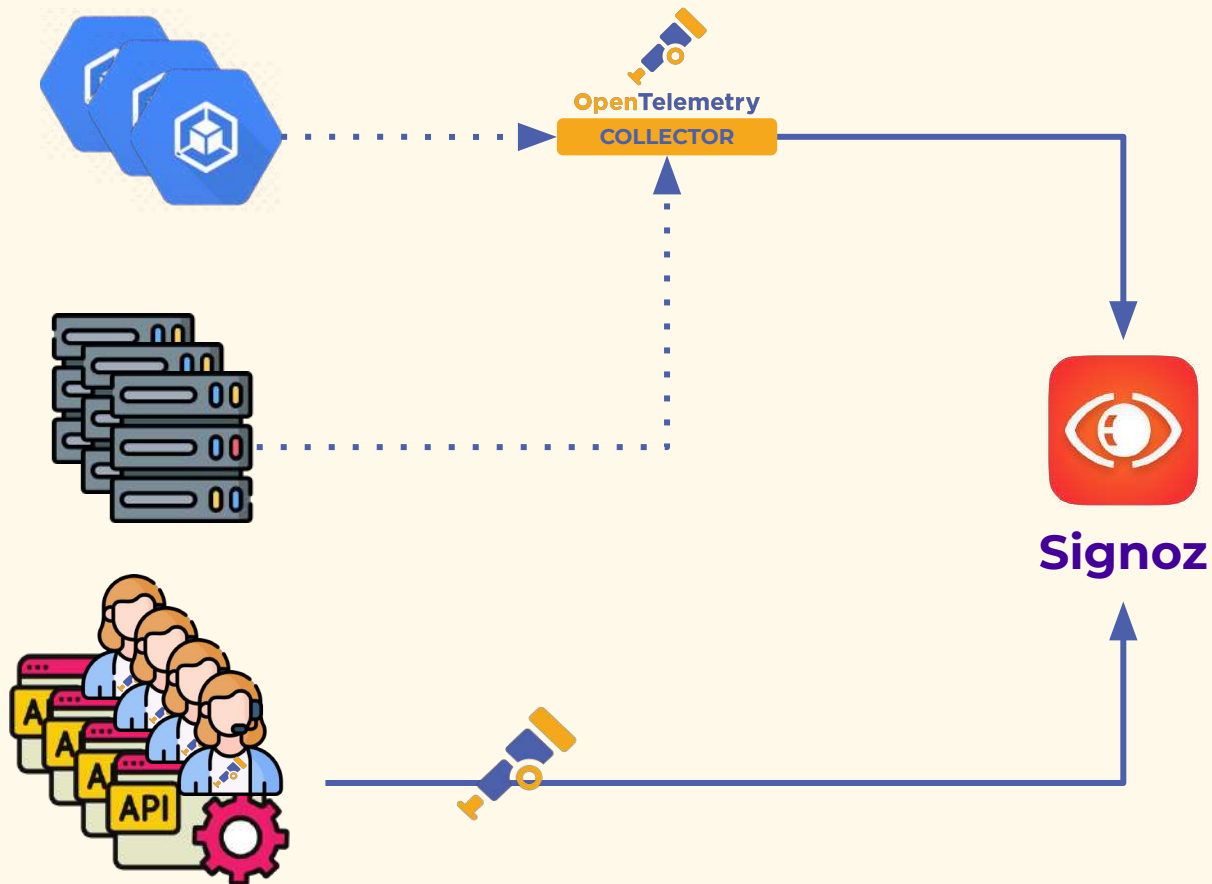


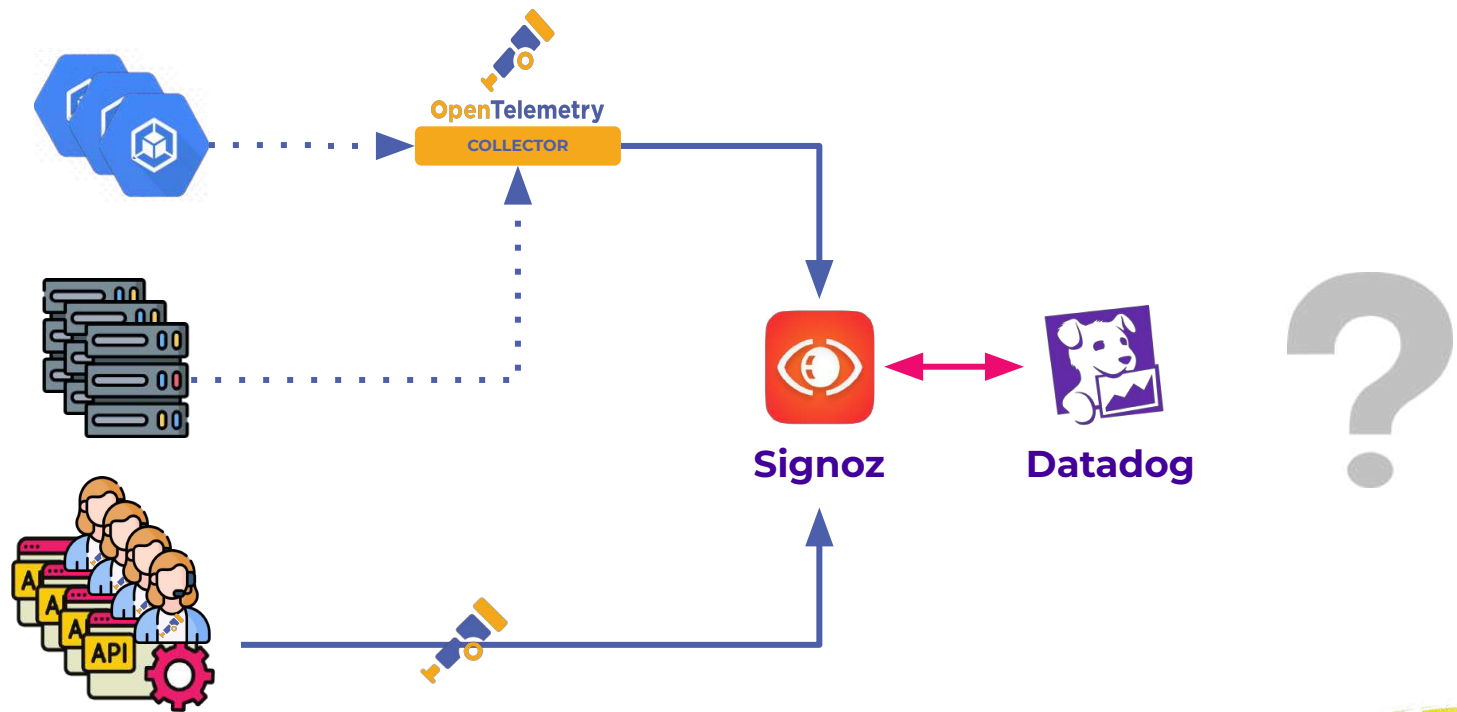
Une modification = un **processeur**



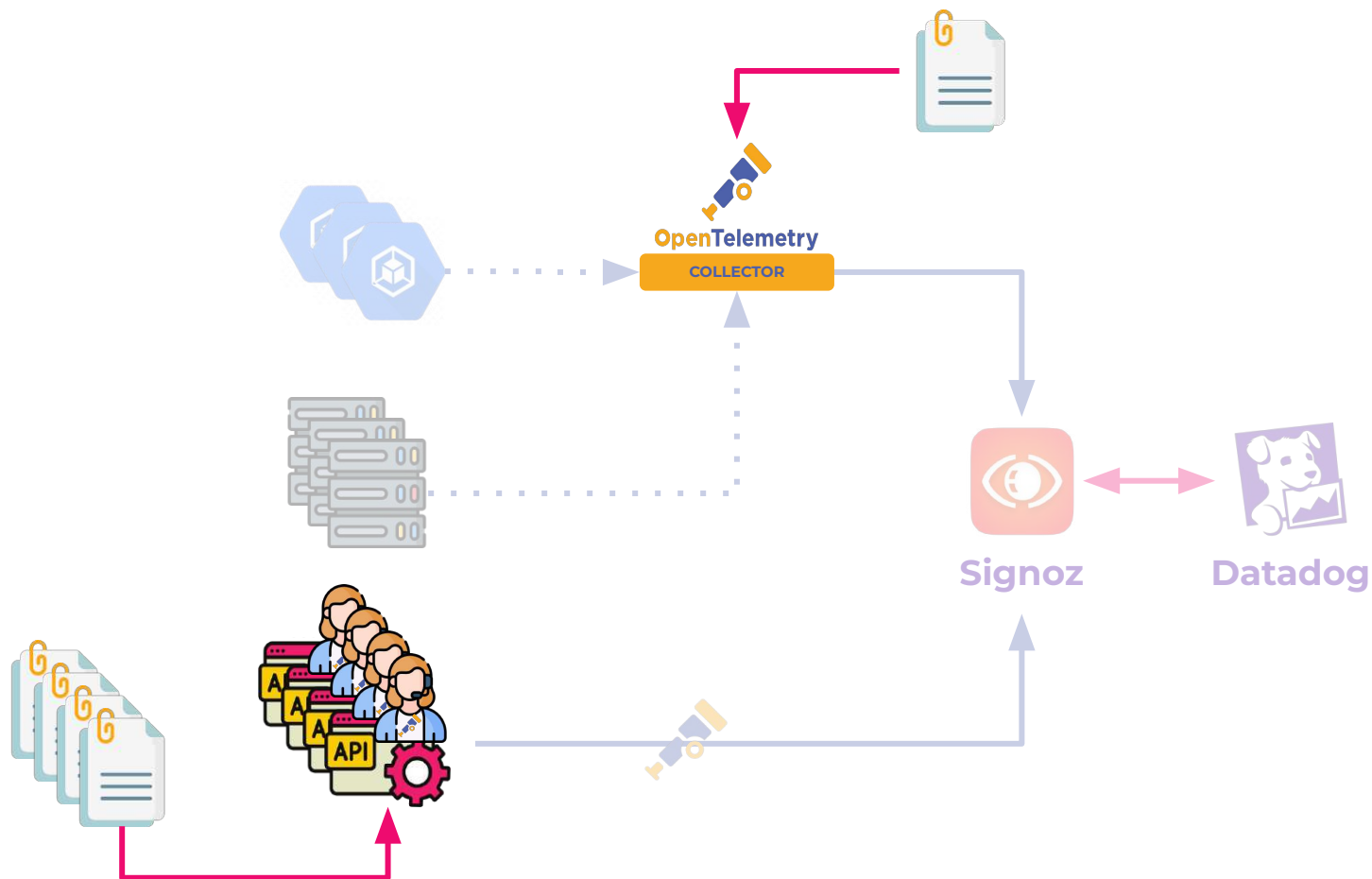
Un envoi = un **exporteur**

# On en est où?





## État des lieux

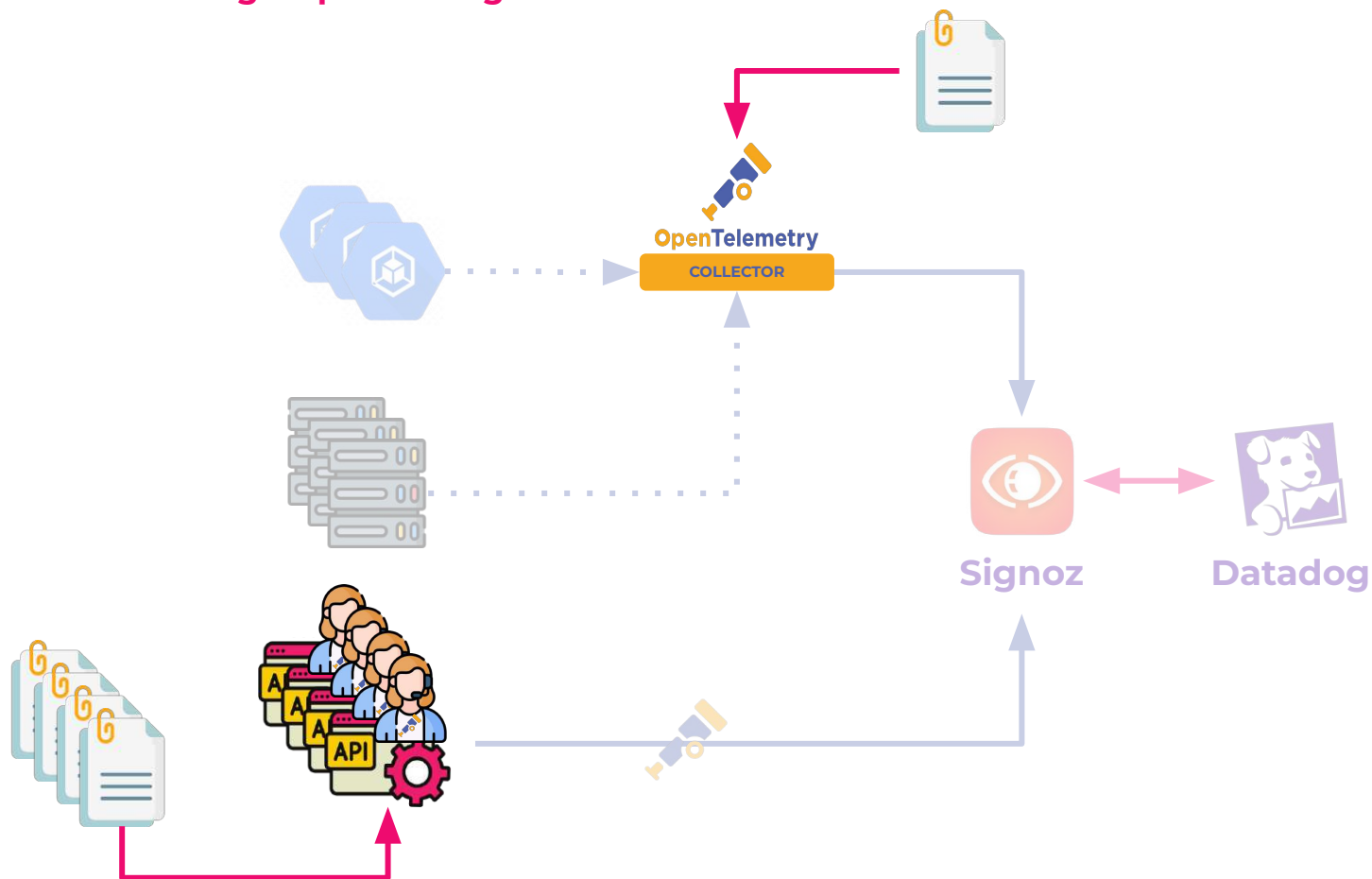


“

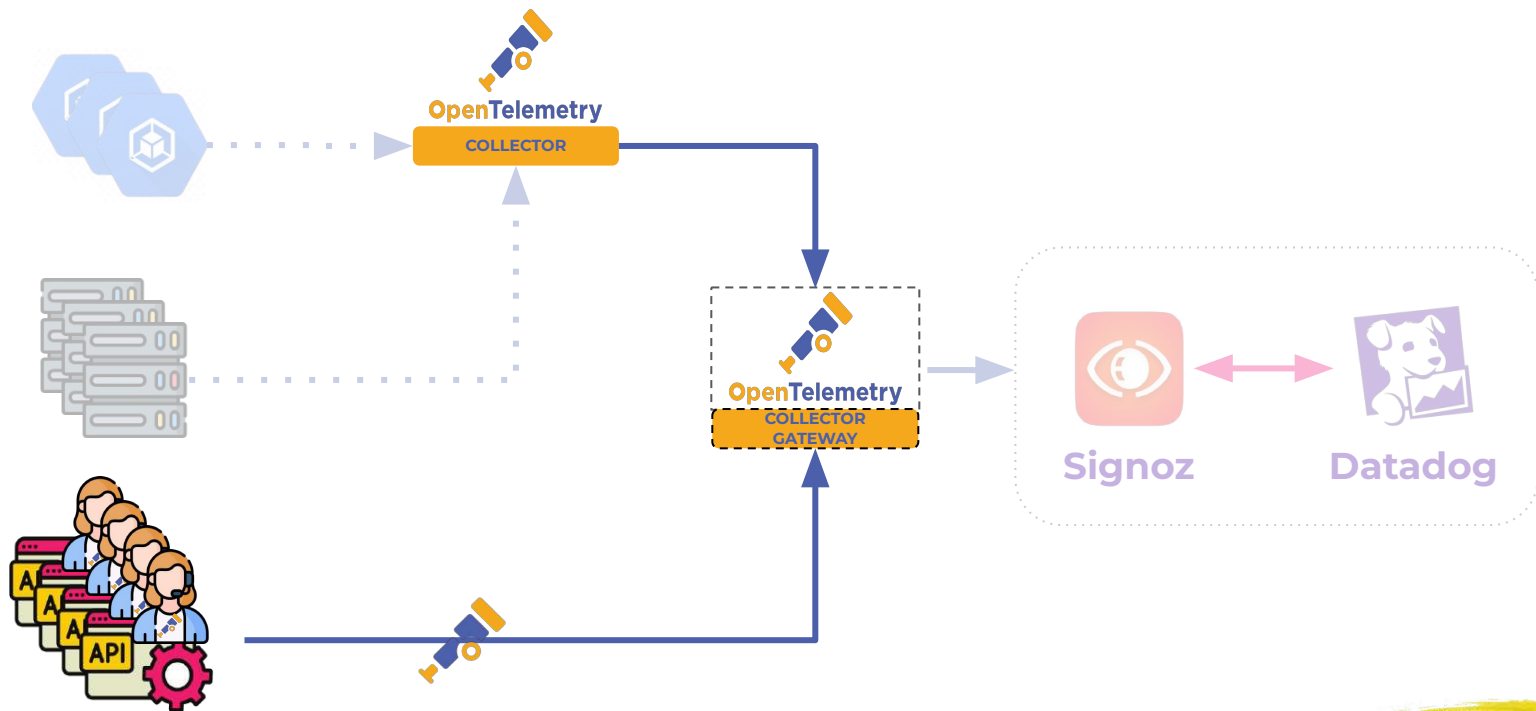
**Comment regrouper nos  
signaux ?**

”

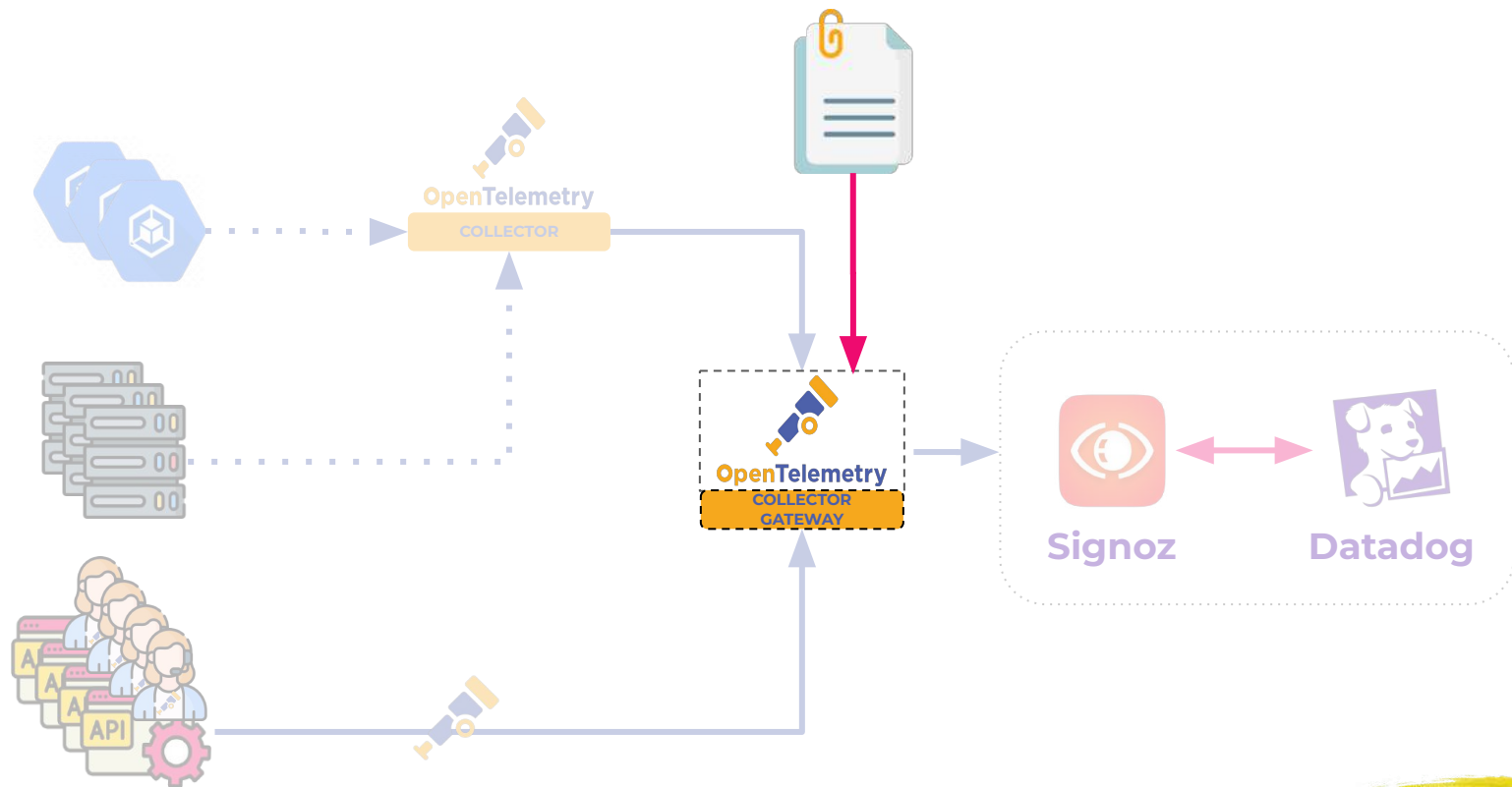
## Comment regrouper nos signaux ?



## Comment regrouper nos signaux

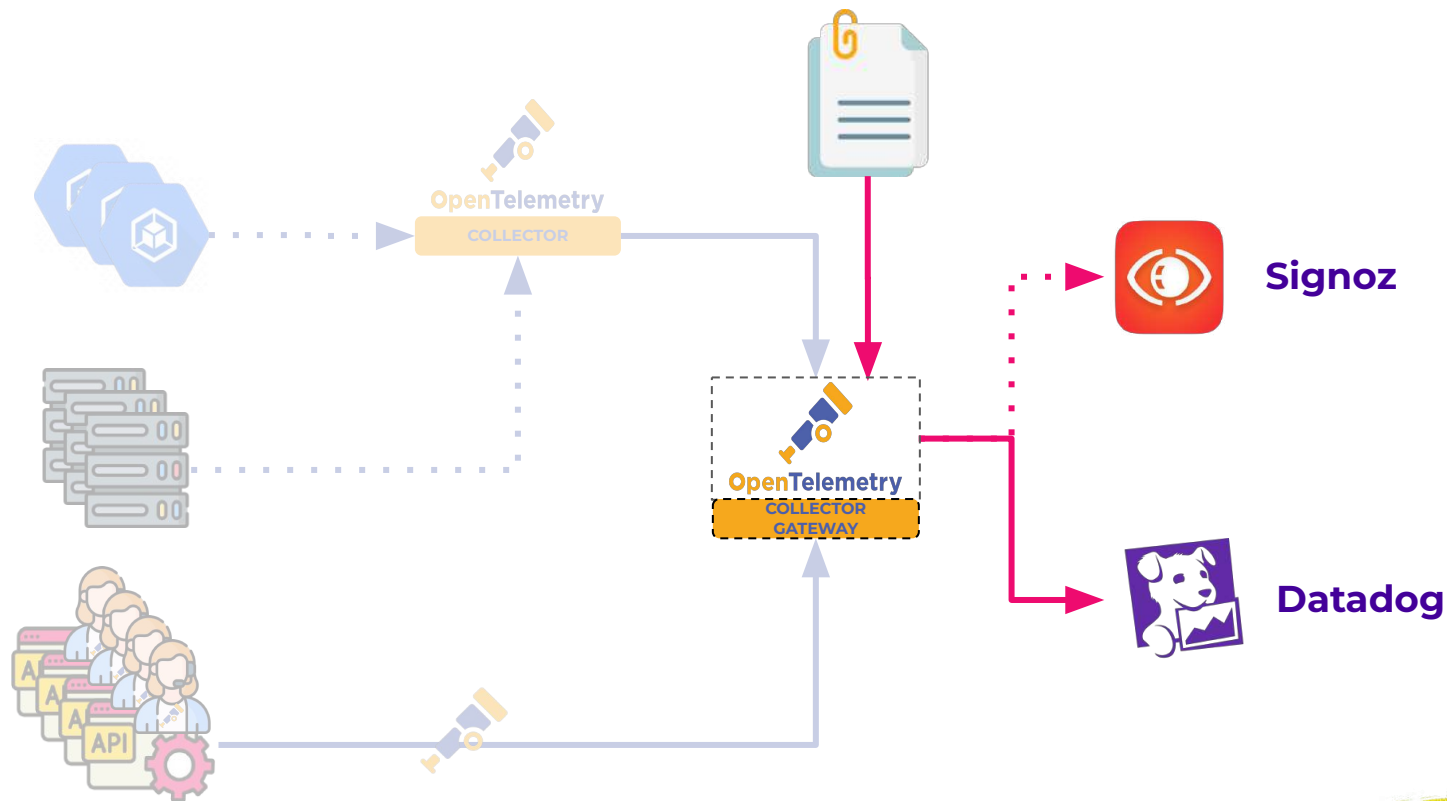


## Comment regrouper nos signaux

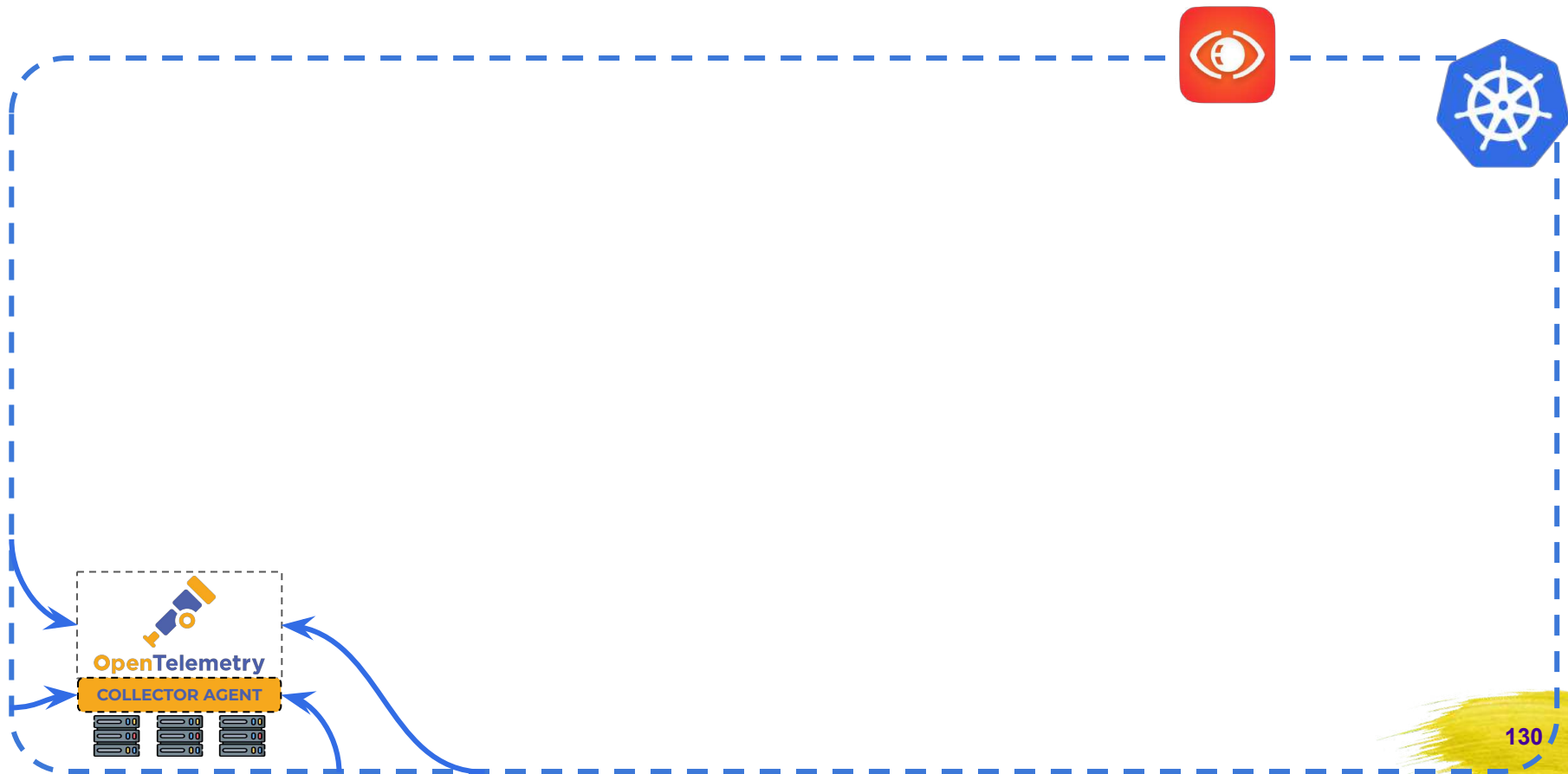




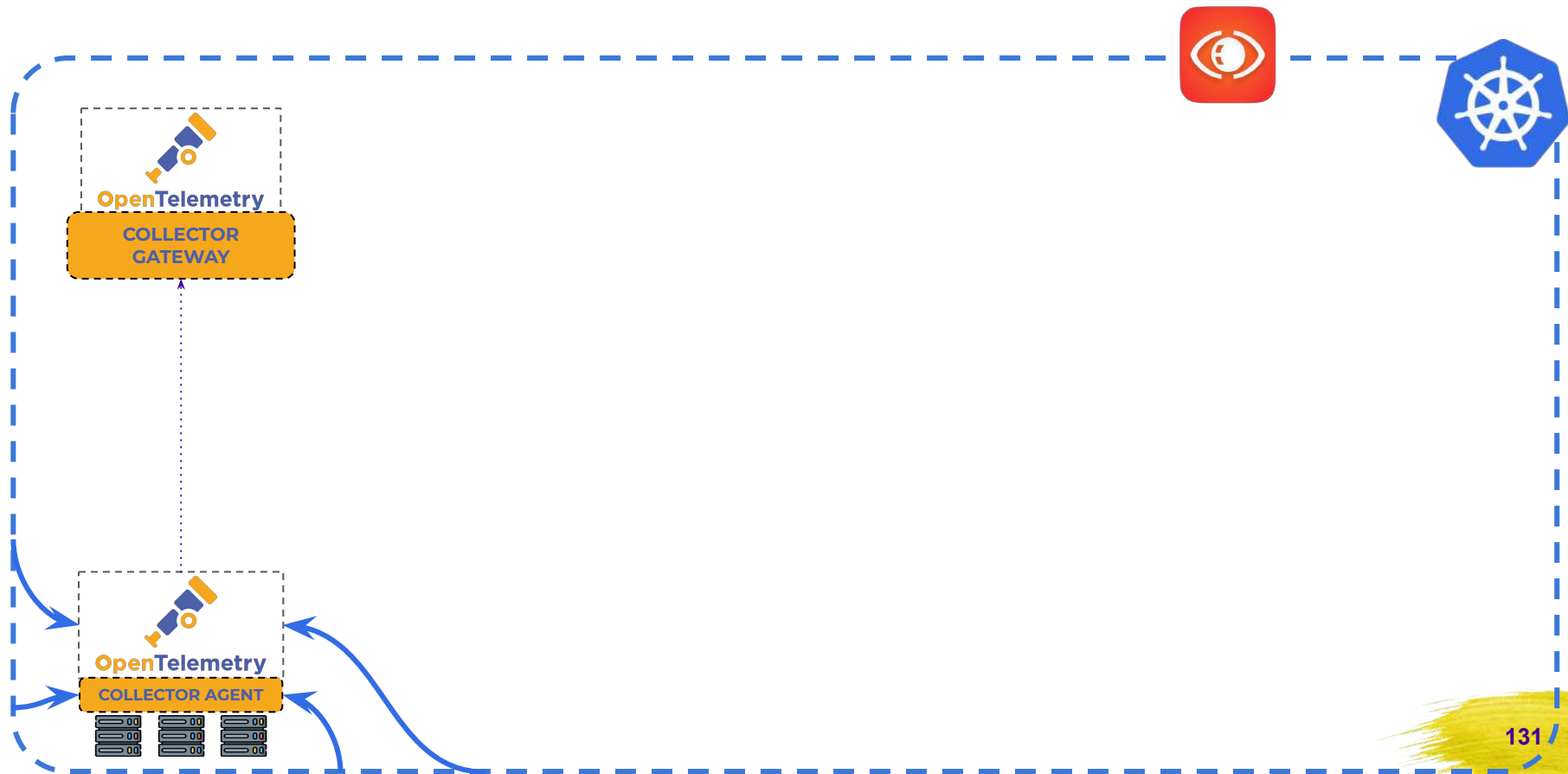
## Comment regrouper nos signaux



## Mon infra maintenant

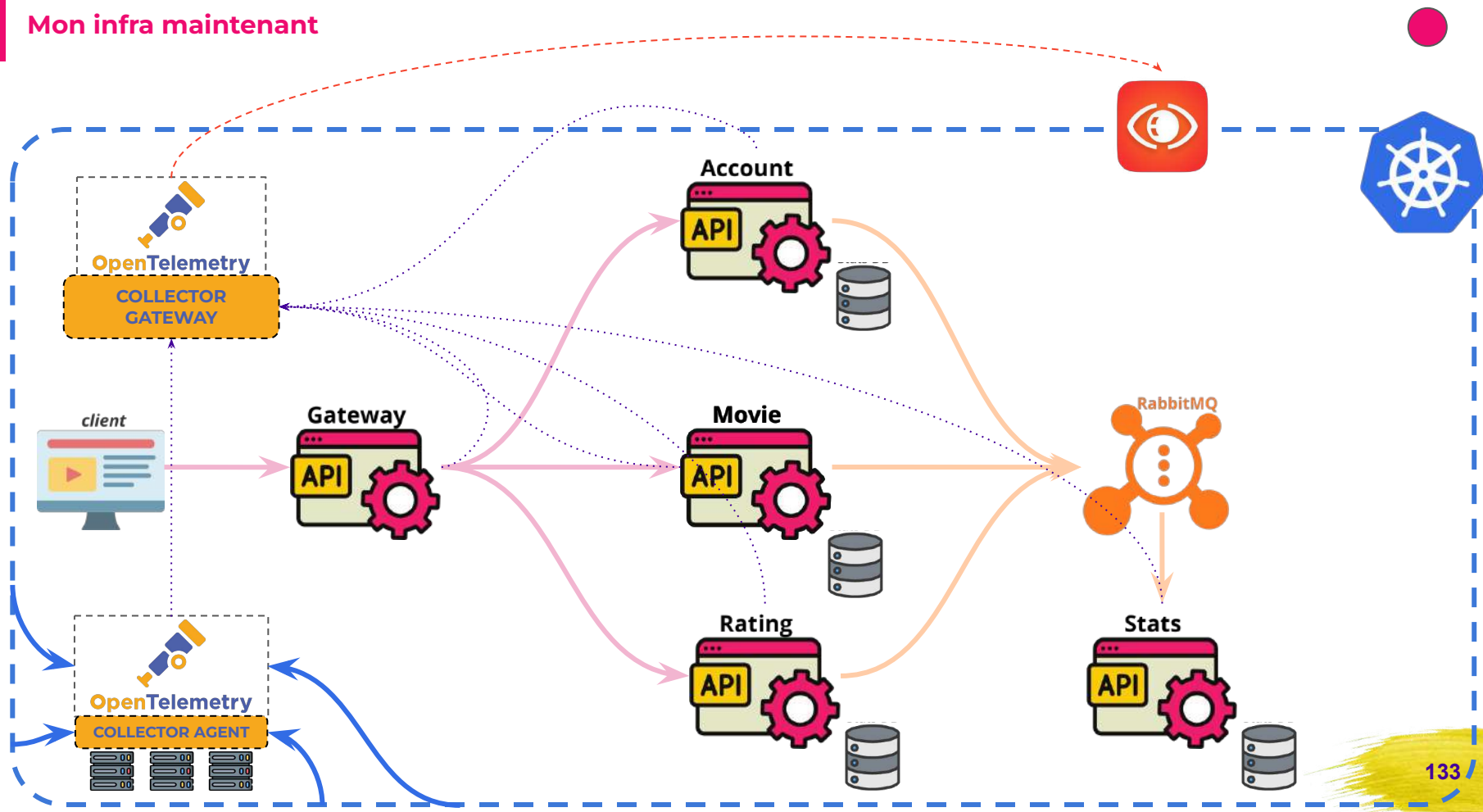


## Mon infra maintenant





## Mon infra maintenant





Maintenant, on peut façonner nos données d'observabilité comme on veut ici !



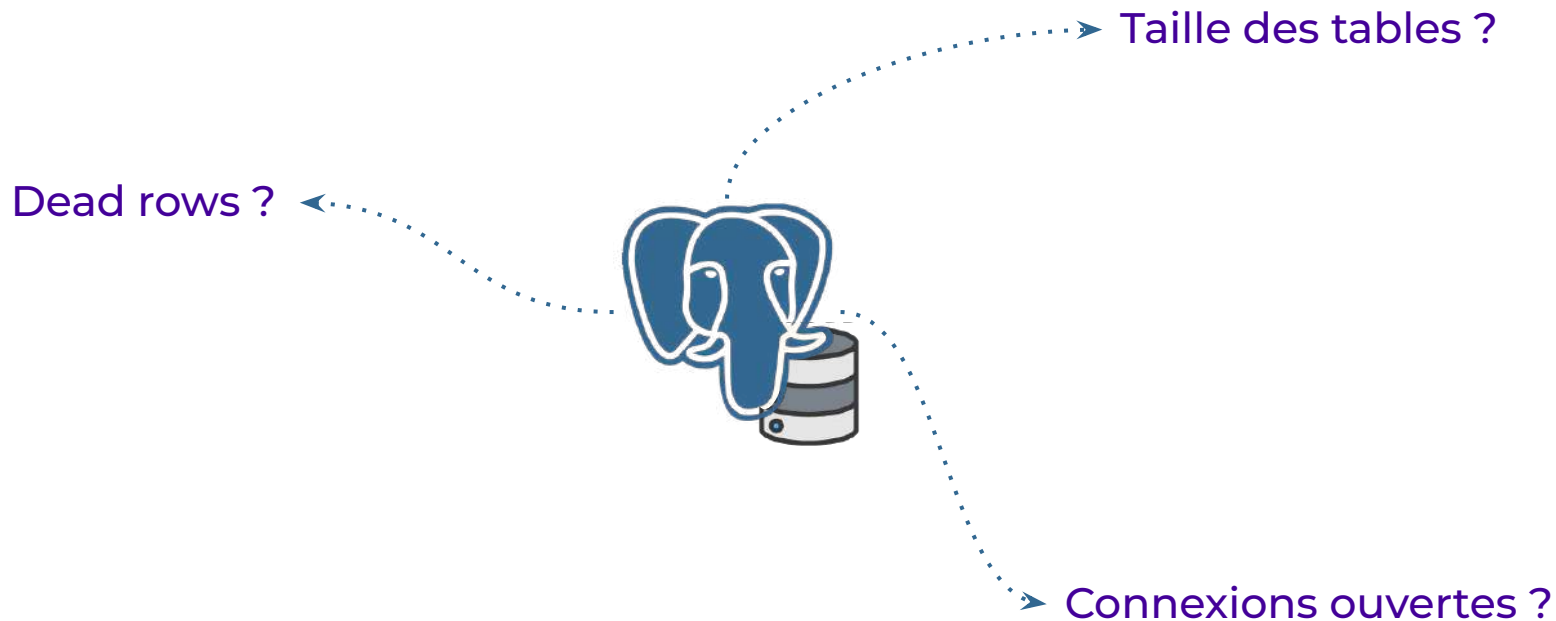
**Nos données “par défaut”**

“

Nos besoins spécifiques:  
**De nouvelles données ?**

”

## Nos bases de données





“

**Un récepteur !**

”

Besoins spécifiques

## Nos bases de données

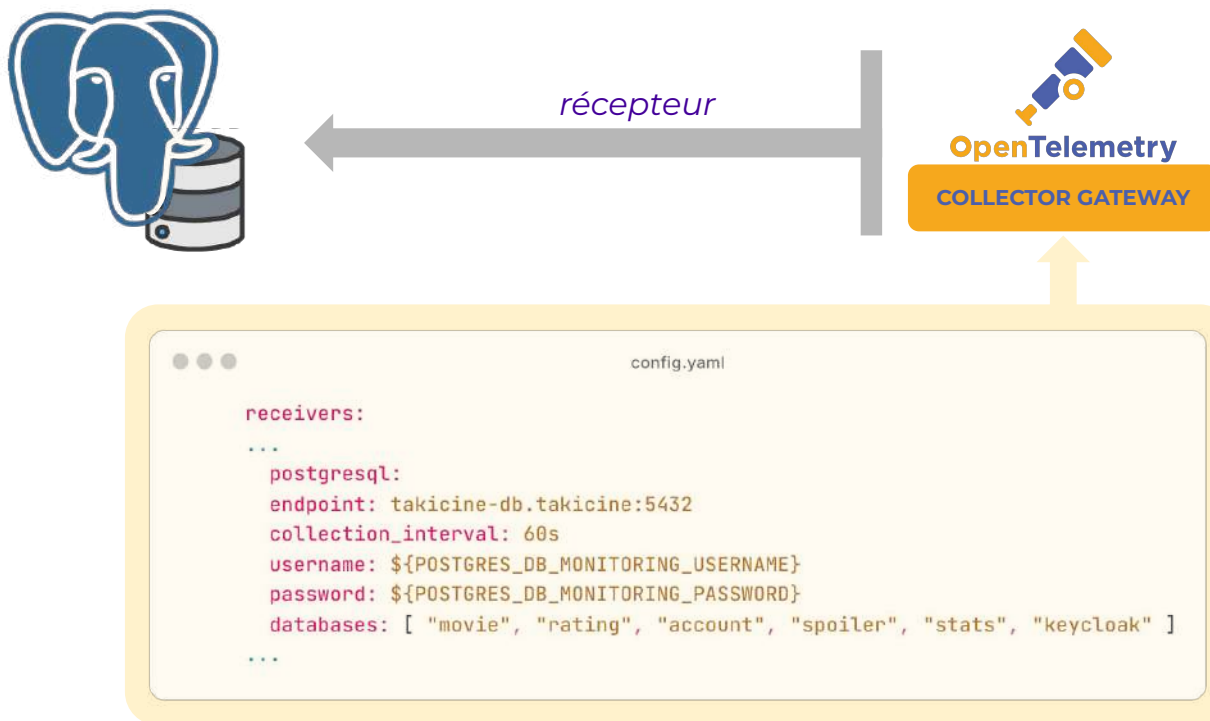


Besoins spécifiques

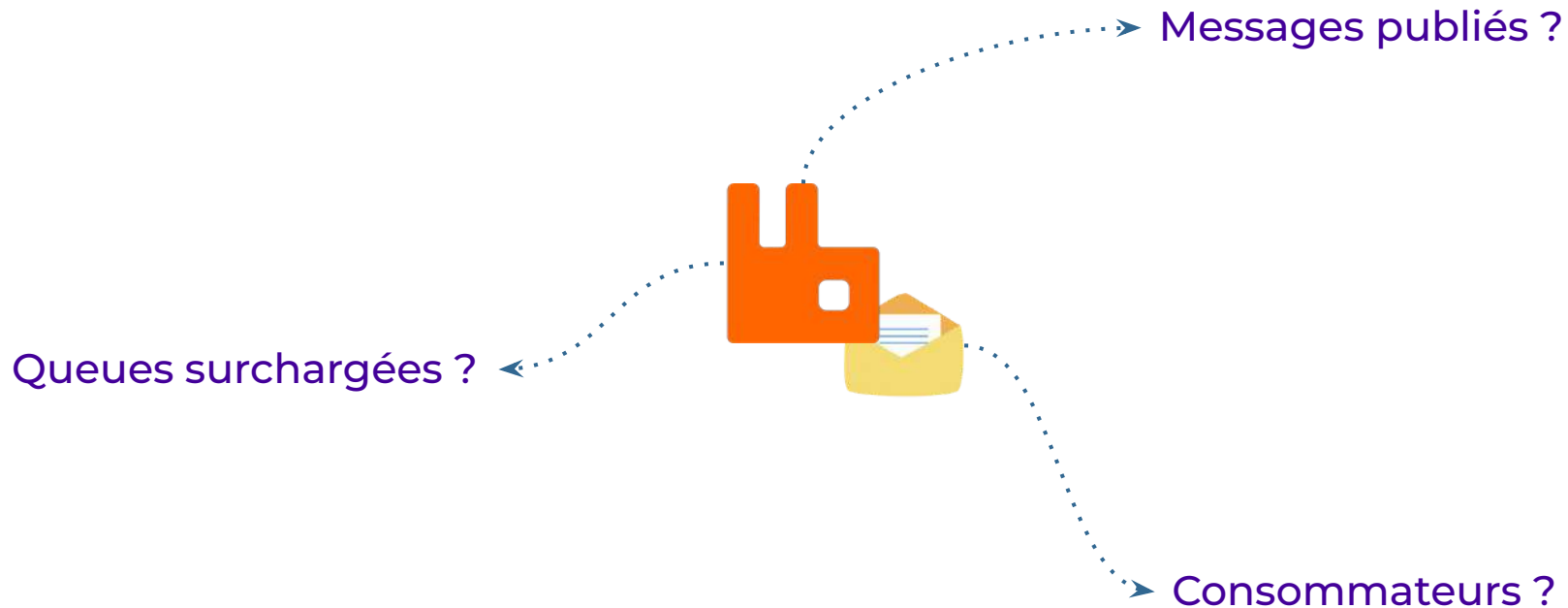
## Nos bases de données



## Nos bases de données



## Notre broker de messages



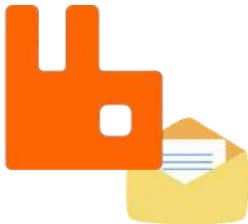
“

**Un récepteur !**

”

Besoins spécifiques

## Notre broker de messages



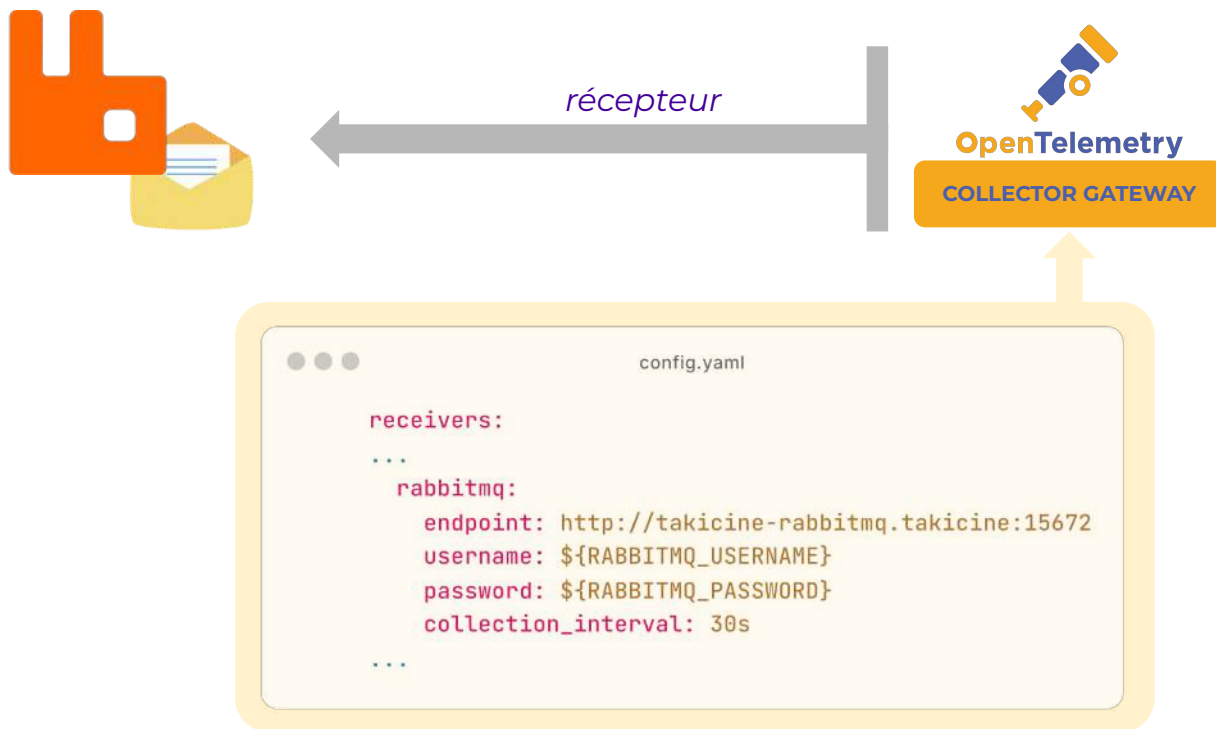
Besoins spécifiques

## Notre broker de messages

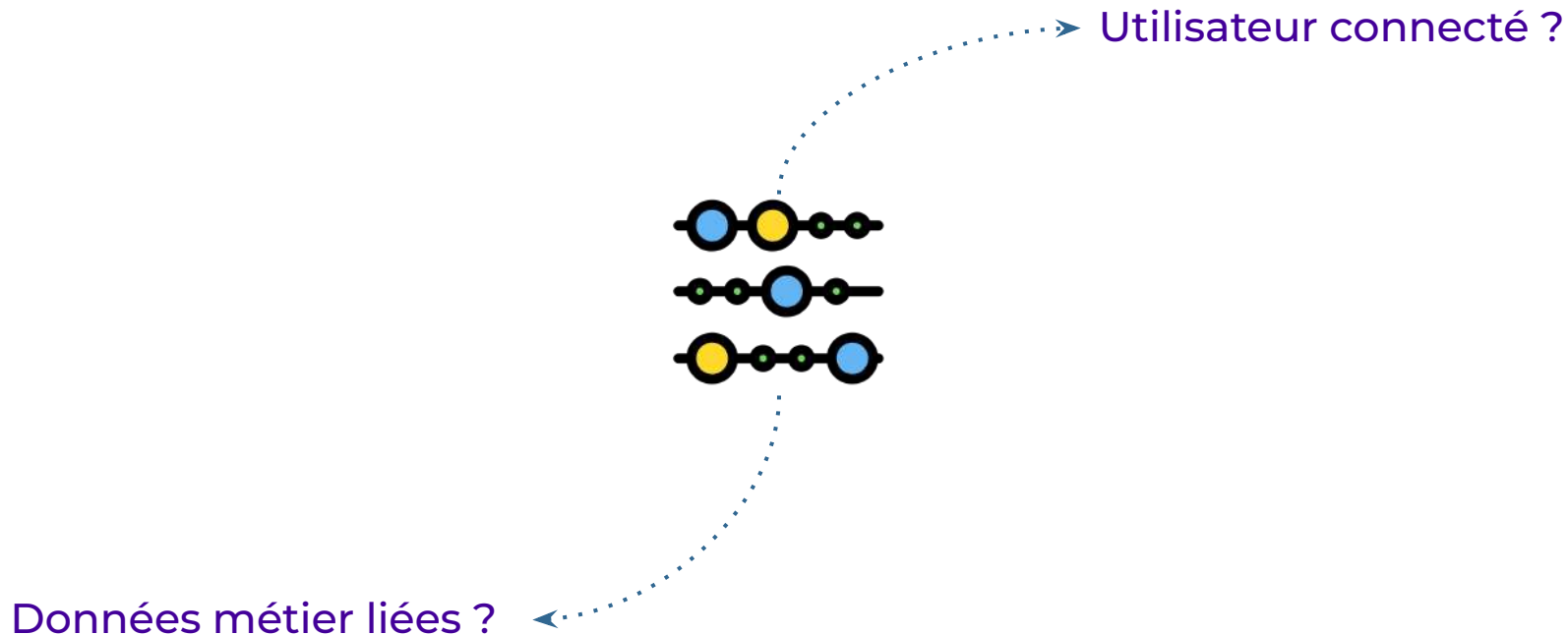




# Notre broker de messages



## Personnaliser nos signaux



“

~~Un récepteur!~~

Personnaliser l'instrumentation

”

## Besoins spécifiques

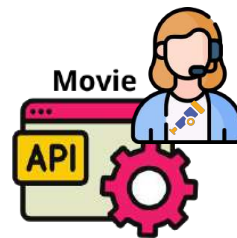
# Instrumentation personnalisée



JAR



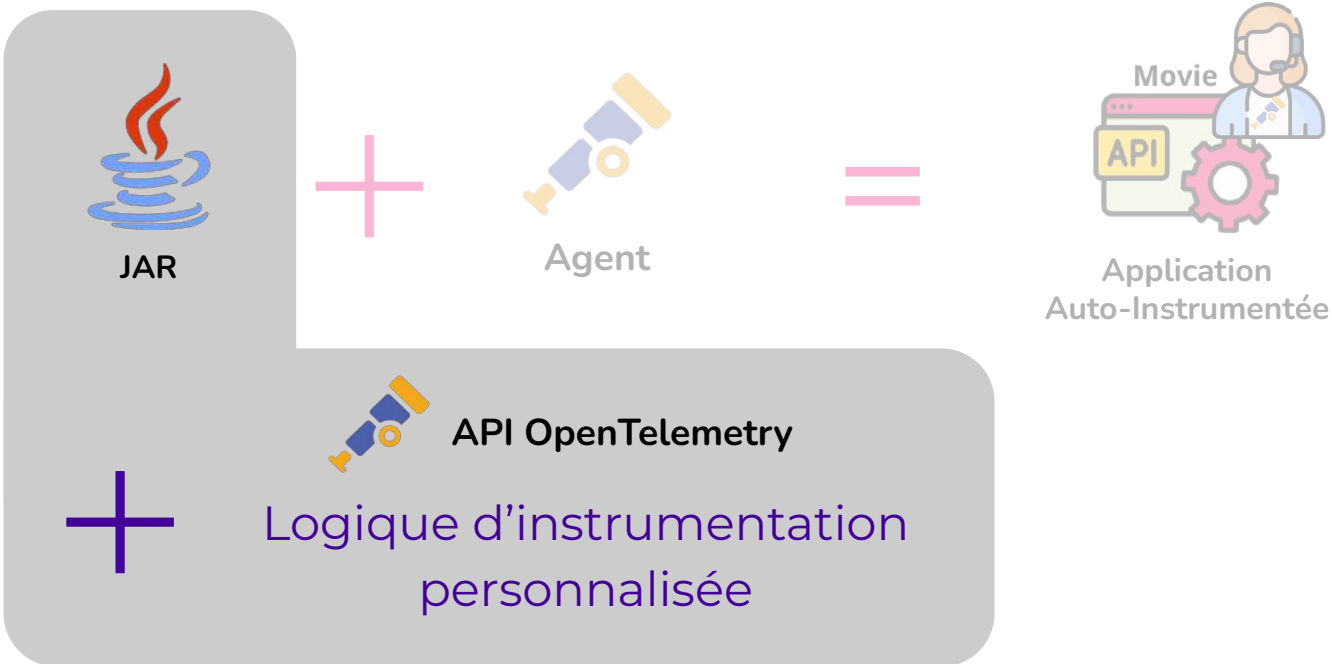
Agent



Application  
Auto-Instrumentée

Besoins spécifiques

# Instrumentation personnalisée





*pom.xml*

*ObservabilityFilter.java*

```
pom.xml

<dependencies>
  ...
  <dependency>
    <groupId>io.opentelemetry</groupId>
    <artifactId>opentelemetry-api</artifactId>
    <version>${otel.version}</version>
  </dependency>
  ...
</dependencies>
```



*pom.xml*

*ObservabilityFilter.java*

```
JwtUserIdObservabilityFilter.java

@Component
public class JwtUserIdObservabilityFilter implements Filter {

    @Override
    public void doFilter(
        HttpServletRequest servletRequest,
        HttpServletResponse servletResponse,
        FilterChain filterChain) throws IOException, ServletException {

    }
}
```



*pom.xml*

*ObservabilityFilter.java*

```
JwtUserIdObservabilityFilter.java

@Component
public class JwtUserIdObservabilityFilter implements Filter {

    @Override
    public void doFilter(
        ServletRequest servletRequest,
        ServletResponse servletResponse,
        FilterChain filterChain) throws IOException, ServletException {

        Authentication auth = SecurityContextHolder.getContext().getAuthentication();

    }
}
```





*pom.xml*

*ObservabilityFilter.java*

```
JwtUserIdObservabilityFilter.java

@Component
public class JwtUserIdObservabilityFilter implements Filter {

    @Override
    public void doFilter(
        ServletRequest servletRequest,
        ServletResponse servletResponse,
        FilterChain filterChain) throws IOException, ServletException {

        Authentication auth = SecurityContextHolder.getContext().getAuthentication();

        if (auth instanceof JwtAuthenticationToken jwtAuthToken) {
            var userId = jwtAuthToken.getToken().getSubject();
            Span.current().setAttribute("user_id", userId);
        }
    }
}
```



*pom.xml*

*ObservabilityFilter.java*

```
JwtUserIdObservabilityFilter.java

@Component
public class JwtUserIdObservabilityFilter implements Filter {

    @Override
    public void doFilter(
        ServletRequest servletRequest,
        ServletResponse servletResponse,
        FilterChain filterChain) throws IOException, ServletException {

        Authentication auth = SecurityContextHolder.getContext().getAuthentication();

        if (auth instanceof JwtAuthenticationToken jwtAuthToken) {
            var userId = jwtAuthToken.getToken().getSubject();
            Span.current().setAttribute("user_id", userId);
        }

        filterChain.doFilter(servletRequest, servletResponse);
    }
}
```



*pom.xml*

*ObservabilityFilter.java*

```
JwtUserIdObservabilityFilter.java

@Component
public class JwtUserIdObservabilityFilter implements Filter {

    @Override
    public void doFilter(
        ServletRequest servletRequest,
        ServletResponse servletResponse,
        FilterChain filterChain) throws IOException, ServletException {

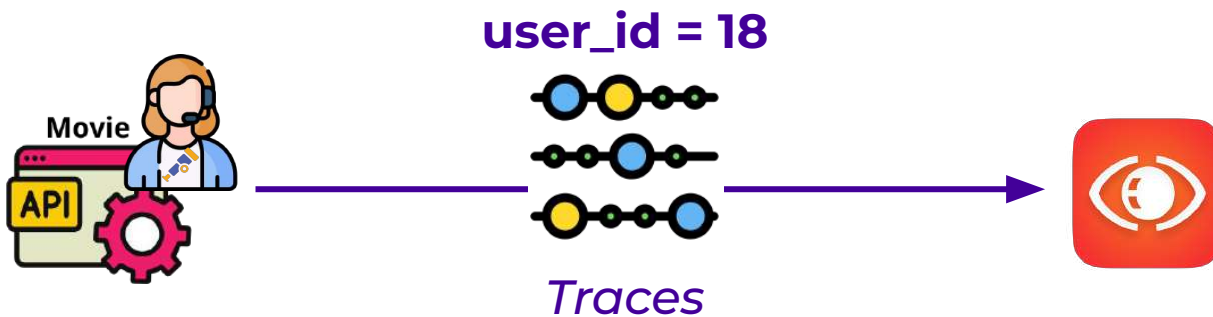
        Authentication auth = SecurityContextHolder.getContext().getAuthentication();

        if (auth instanceof JwtAuthenticationToken jwtAuthToken) {
            var userId = jwtAuthToken.getToken().getSubject();
            Span.current().setAttribute("user_id", userId);
        }

        filterChain.doFilter(servletRequest, servletResponse);
    }
}
```

Besoins spécifiques

## Instrumentation personnalisée



“



Démo

”

# En résumé



Signaux d'observabilité



OpenTelemetry



Backend d'observabilité



Collecter l'infrastructure



Architecture évolutive

On peut **chaîner** des collecteurs si besoin



Un **collecteur Gateway** offre une bonne architecture d'observabilité



La configuration des données qu'on exporte est **centralisée** en amont



On peut y **manipuler tous nos signaux**



Il pourra s'occuper de collecter de **nouvelles ressources**

# En résumé



Signaux d'observabilité



OpenTelemetry



Backend d'observabilité



Collecter l'infrastructure



Architecture évolutive



Instrumentation perso



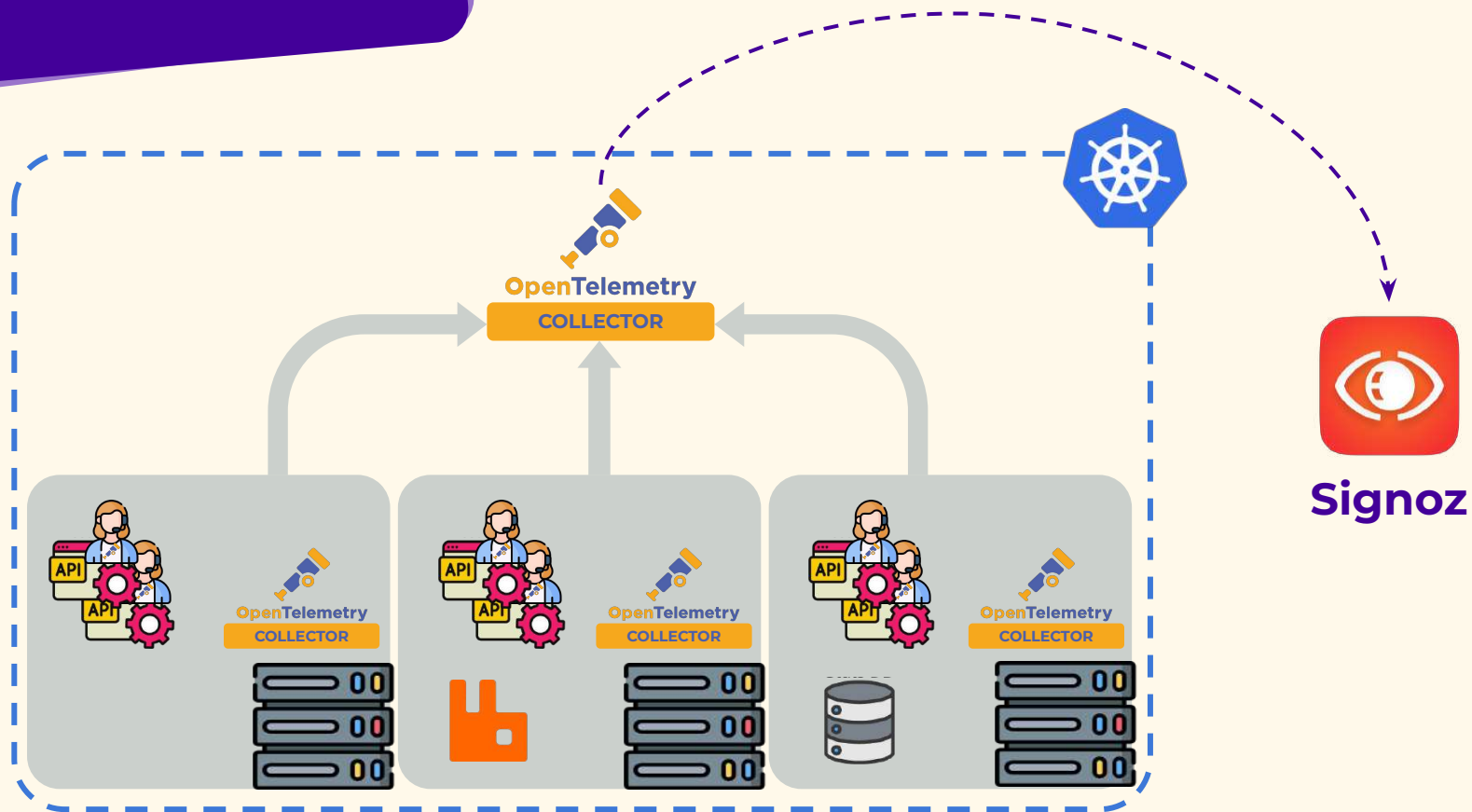
On peut **personnaliser l'instrumentation** de nos applications

Et ce, même si on utilise un agent



En utilisant **l'API OpenTelemetry** pour ajouter les comportements désirés

# On en est où?





“

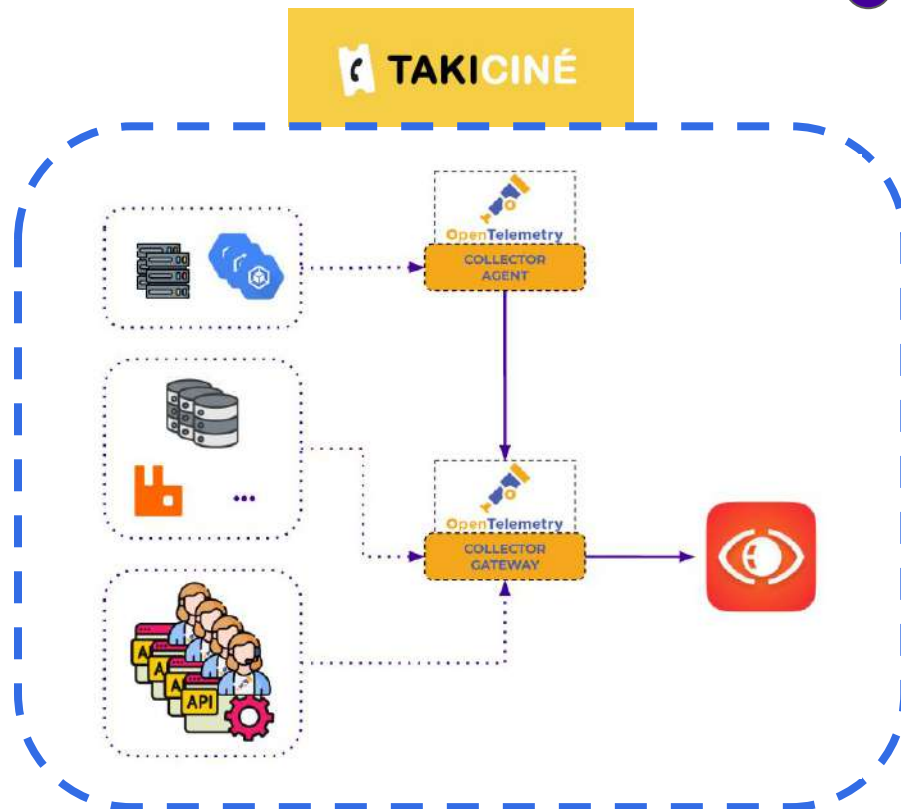


**On se met en situation réelle**

”

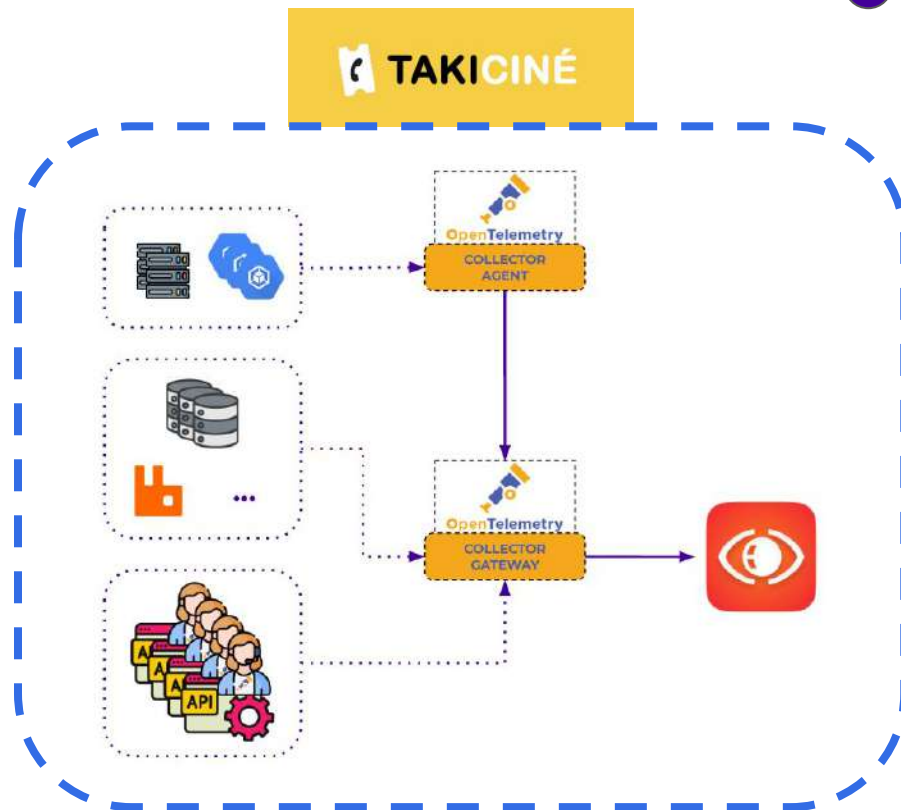
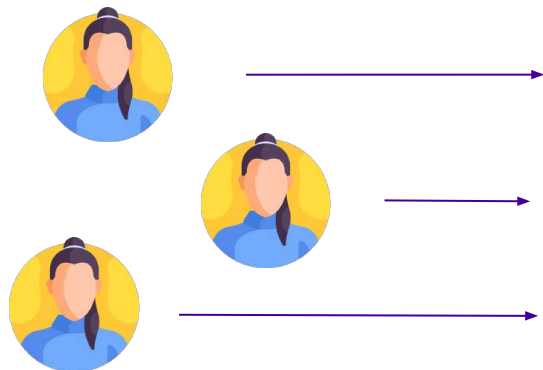
L'observabilité

On charge notre prod



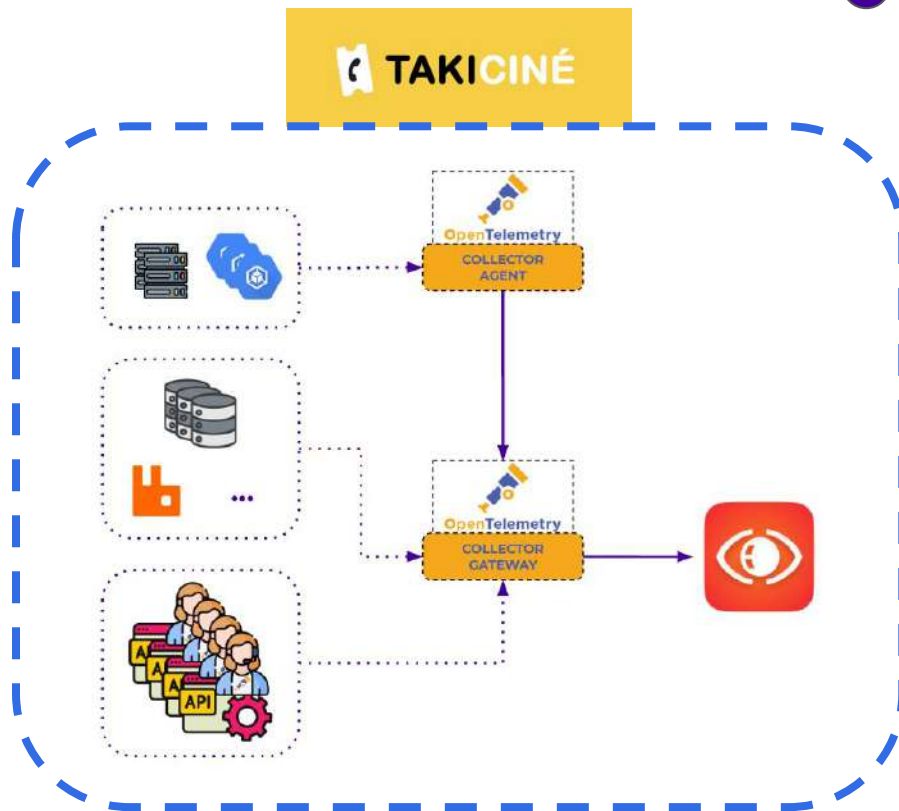
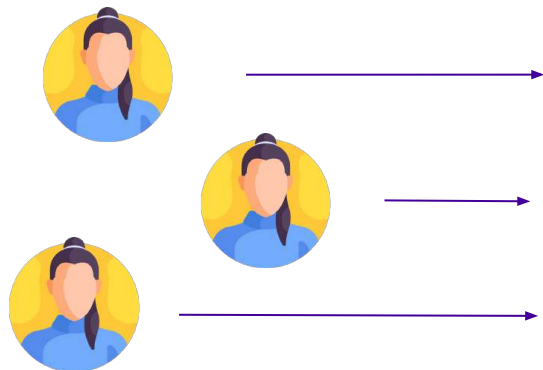
L'observabilité

## On charge notre prod



L'observabilité

On charge notre prod



“



Démo

”

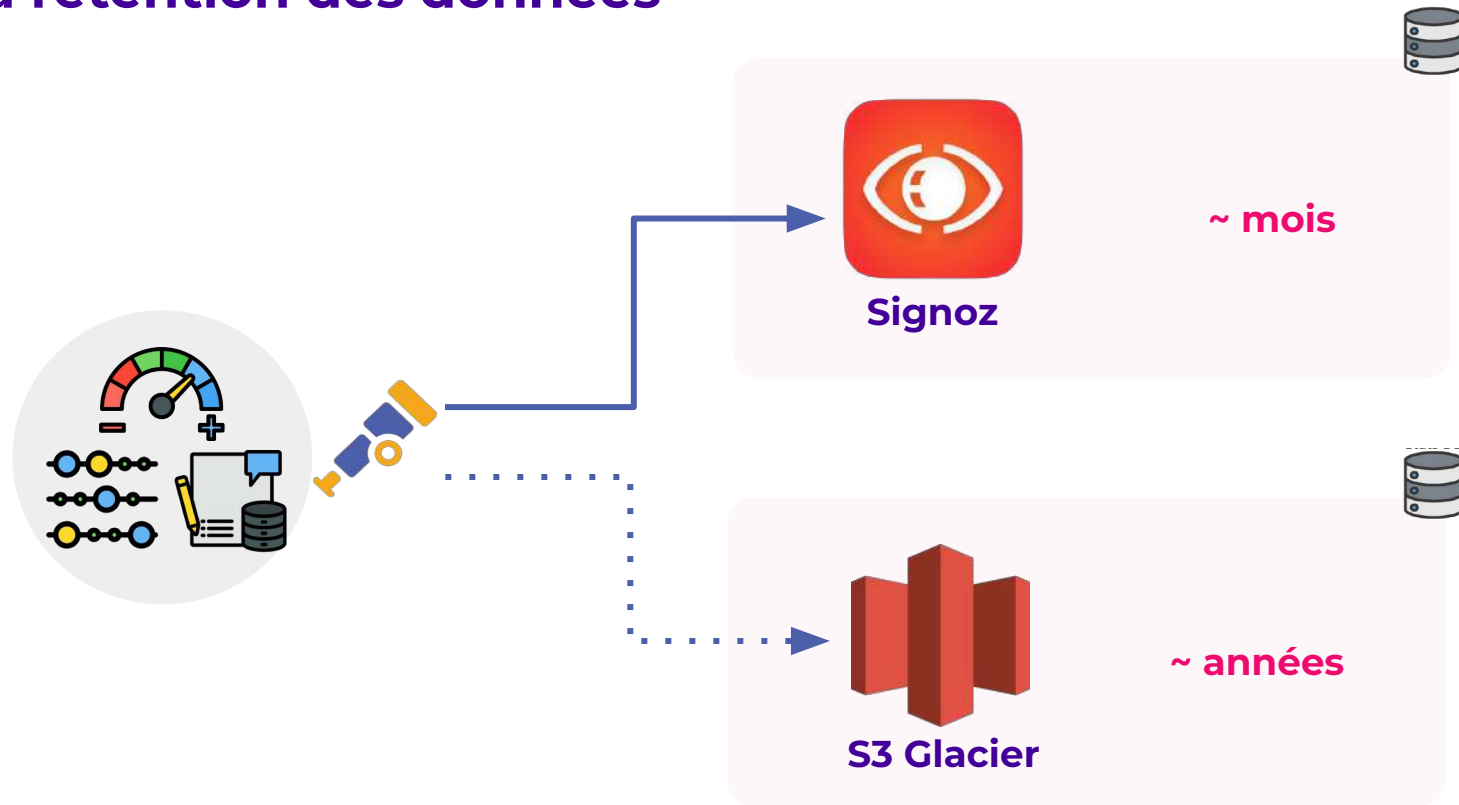
“



**Quelques tips à emporter ?**

”

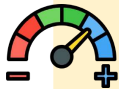
# La rétention des données



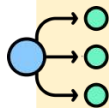
## Gérer son collecteur



Authentification



Auto-observabilité



Scalabilité



Tri des données



Limiter les alertes techniques

Proche des indices de qualité métier

Doivent susciter une action manuelle

À trier et affiner dans le temps



# Héberger son Signoz



**Signoz**



## On aime:

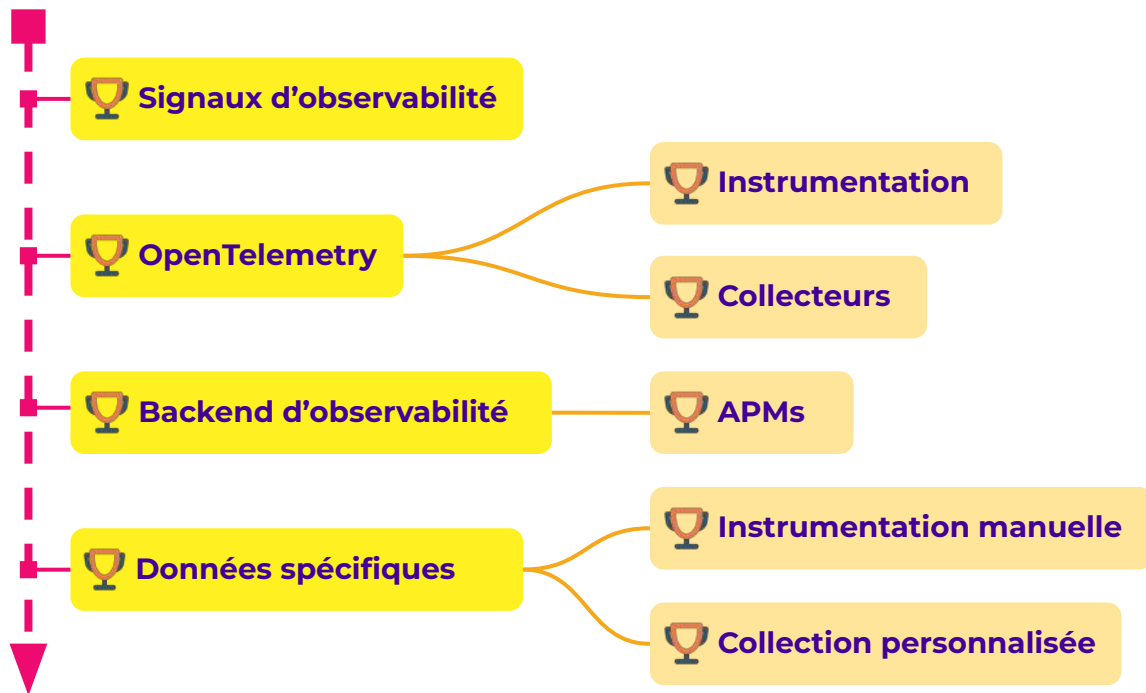
- Installations rapides
- Équipe à l'écoute
- Projet actif



## On aimerait:

- Plus de GitOps (création des admin, dashboards, rétention)
- Gestion des droits plus fine

## Tout ce qu'on a vu en 45 min



“



Les **systemes distribués**  
sont de plus en plus  
présents

”

“



Pouvoir les observer  
devient indispensable

”

“



L'écosystème de  
l'**observabilité** évolue aussi

”

“



Des outils **tout en un**

”

“



Des outils **open-source**

”



“



**Des outils accessibles**

”

“



**Monter une bonne observabilité  
est à la portée de tous.**

”

## Quelques ressources



*OpenTelemetry blog posts*

*“Evaluating OpenTelemetry’s Impact on  
Performance in Microservice Architectures”*



Envie de se lancer rapidement ?

On vous a obtenu des promos sur Signoz Cloud !



CLOUD

Premier mois gratuit  
-15% pendant 6 mois



[taki.li/observe](https://taki.li/observe)

Intéressés par les tests de charge ?

Ils ont fait nos scripts Gatling !

« 45 min pour mettre son application à  
genoux : le guide complet du test de charge »

Loïc Ortola  
Mathilde Lorrain

Vendredi 18 avril  
11:35 - 12:20





Merci



*Feedback*



**Florian Meuleman**

Devops / Backend Engineer @Takima

@florianmeuleman



**Alexandre Moray**

Lead dev fullstack @Takima

@alexandremoray

Venez vous **former avec nous** :  
**formation.takima.fr**

**& Rdv au stand !**