



Formation sur les API de uprodit.com



www.comwork.io



idriss.neumann@comwork.io



Comwork.io SASU

128 rue de la Boétie 75008 Paris SIRET : 83875798700014

Comwork

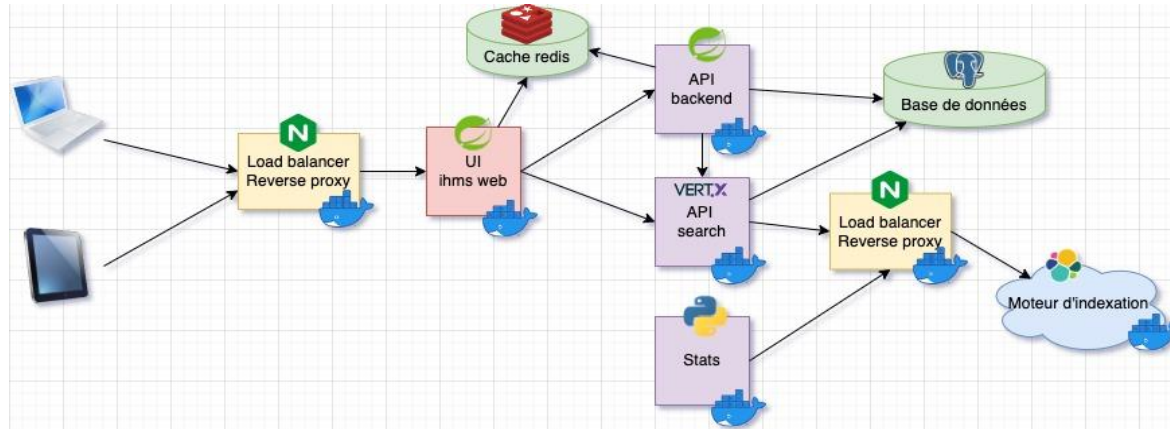
*Uprod***it**











Au programme

- ❖ Architecture
 - ❖ Principe du RESTful
 - ❖ Contrat d'interfaces WADL
 - ❖ Authentification des API
 - Authentification ws-cxf-ext
 - Authentification d'un utilisateur
 - via user/mot de passe
 - via token
- ❖ Quelques exemples d'API
 - API de recherche de profils
 - API v2 :
 - Pourquoi ?
 - profils, images et réalisations
 - ❖ Introduction à l'observabilité
 - Id de corrélation
 - Kibana
 - Imalive API
 - ❖ Conclusion / questions

Architecture de uprodit.com



Légende

	PostgreSQL - Système de gestion de base de données relationnelles
	Elasticsearch - moteur d'indexation / base de données noSQL document
	Vert.x - Framework Java pour microservices asynchrones
	Spring - Framework Java pour applications webs synchrones
	Nginx - reverse proxy et load balancer (répartition de charge + chiffrement TLS)
	Docker - conteneurisation des composants (portabilité et scalabilité)
	Python - langage de programmation utilisée pour la data et les statistiques
	Redis - base de données NoSQL in-memory clef/valeur utilisée comme cache

Principe du RESTful

REST pour "*REpresentational State Transfert*".

Il s'agit d'un style d'architecture basé sur des services web utilisant les technologies suivantes :

- ❖ Le format des ressources (données métier comme les dossiers de commandes ou autres) grâce aux sémantiques des langages de description de données tels que XML ou JSON.

- ❖ Quelques directives du protocole HTTP qui permettent d'identifier les opérations à exécuter par le service appelé. Il s'agit des directives suivantes :
 - **POST** pour la création d'une ressource en base ;
 - **PUT** pour la modification d'une ressource en base ;
 - **PATCH** pour la modification partielle d'une ressource en base ;
 - **GET** pour la récupération de ressources archivées en base ;
 - **DELETE** pour la suppression de ressources persistées en base.

Principe du RESTful

- ❖ Quelques codes retours du protocole HTTP qui permettent de détecter la bonne exécution d'un webservice REST ou bien d'identifier le type d'erreur rencontrée :
 - **200** : bonne réception d'une ressource (GET)
 - **201** : bonne persistance d'une ressource (PUT/POST)
 - **202** : demande prise en compte pour un traitement asynchrone
 - **400** : paramètres de la requête invalide (GET/PUT/POST/DELETE)
 - **401** : problème d'authentification du consommateur (GET/PUT/POST/DELETE)
 - **403** : problèmes de droits d'accès (GET/PUT/POST/DELETE)
 - **404** : ressource inexistante (GET/PUT/POST/DELETE)
 - **500** : erreur technique (GET/PUT/POST/DELETE)

- ❖ L'URI possède une signification importante pour l'appel d'un service : celle de permettre, par une simple lecture visuelle, d'identifier le service auquel on fait appel, voire dans le cas d'un appel de type GET, la ressource à récupérer.

Contrats d'interface WADL

Les fournisseurs de services sont les suivants :

- ❖ API : <https://api.uprodit.com>
- ❖ API SEARCH: <https://search.uprodit.com>

Vous pouvez télécharger les fichiers WADL (web application description language) pour générer vos objets et interfaces grâce au plugin wadl2java

```
ineumann ~ $ curl https://api.uprodit.com/wadl.xml -o api_wadl.xml
% Total    % Received % Xferd Average Speed   Time    Time     Time  Current
           Dload Upload Total   Spent  Left  Speed
100 162k    100 162k    0    0 189k    0 --:--:-- --:--:-- --:--:-- 191k
ineumann ~ $ curl https://search.uprodit.com/wadl.xml -o search_wadl.xml
% Total    % Received % Xferd Average Speed   Time    Time     Time  Current
           Dload Upload Total   Spent  Left  Speed
100 19115    100 19115    0    0 28913    0 --:--:-- --:--:-- --:--:-- 29272
```

Authentification des API

La quasi-totalité des webservices sont authentifiés par application (il faut faire une demande d'ajout d'un **appid** aux équipes de uprodit.com pour pouvoir y accéder).

La solution mise en oeuvre : <https://ws-cxf-ext.github.io/ws-cxf-ext/>

Pour celles et ceux qui veulent utiliser la partie client de cette solution :

<https://gitlab.comwork.io/oss/ws-cxf-ext/-/blob/master/docs/getting-started.md#declaring-clients>

Pour les autres consommateurs, passer les paramètres suivants dans un paramètre header "Authorization":

- ❖ **auth_consumer_key** : chiffage hmac / sha1 de l'environnement avec l'appid ;
- ❖ **auth_callback** : url du webservice avec paramètres ;
- ❖ **auth_nonce** : token généré aléatoirement (UUID.randomUUID().toString()) chiffré en hmac / sha1 via l'appid ;
- ❖ **auth_token** : token généré aléatoirement (pas la même que auth_nonce)
- ❖ **auth_signature** : concaténation de l'uri et du token (auth_token) chiffage via la l'appid ;
- ❖ **auth_timestamp** : timestamp
- ❖ **auth_signature_method** : toujours "HMAC-SHA1"

Ces paramètres sont concaténés sous la forme :

```
Auth auth_consumer_key=valeur&auth_callback=valeur&...
```

Authentification des API

La quasi-totalité des webservices sont authentifiés par application (il faut faire une demande d'ajout d'un **appid** aux équipes de uprodit.com pour pouvoir y accéder).

La solution mise en oeuvre est la **signature HMAC** avec : <https://ws-cxf-ext.github.io/ws-cxf-ext/>

Pour celles et ceux qui veulent utiliser la partie client de cette solution :

<https://gitlab.comwork.io/oss/ws-cxf-ext/-/blob/master/docs/getting-started.md#declaring-clients>

Pour les autres consommateurs, passer les paramètres suivants dans un paramètre header "Authorization":

- ❖ **auth_consumer_key** : chiffrement hmac / sha1 de l'environnement avec l'appid ;
- ❖ **auth_callback** : url du webservice avec paramètres ;
- ❖ **auth_nonce** : token généré aléatoirement (UUID.randomUUID().toString()) chiffré en hmac / sha1 via l'appid ;
- ❖ **auth_token** : token généré aléatoirement (pas le même que auth_nonce)
- ❖ **auth_signature** : concaténation de l'uri et du token (auth_token) chiffrage via la l'appid ;
- ❖ **auth_timestamp** : timestamp
- ❖ **auth_signature_method** : toujours "HMAC-SHA1"

Ces paramètres sont concaténés sous la forme :

```
Auth auth_consumer_key=valeur&auth_callback=valeur&...
```


Authentification des API

La quasi-totalité des webservices sont authentifiés par application (il faut faire une demande d'ajout d'un **appid** aux équipes de uprodit.com pour pouvoir y accéder).

La solution mise en oeuvre : <https://ws-cxf-ext.github.io/ws-cxf-ext/>

Pour celles et ceux qui veulent utiliser la partie client de cette solution :

<https://gitlab.comwork.io/oss/ws-cxf-ext/-/blob/master/docs/getting-started.md#declaring-clients>

Pour les autres consommateurs, passer les paramètres suivants dans un paramètre header "Authorization":

- ❖ **auth_consumer_key** : chiffrement hmac / sha1 de l'environnement avec l'appid ;
- ❖ **auth_callback** : url du webservice avec paramètres ;
- ❖ **auth_nonce** : token généré aléatoirement (UUID.randomUUID().toString()) chiffré en hmac / sha1 via l'appid ;
- ❖ **auth_token** : token généré aléatoirement (pas la même que auth_nonce)
- ❖ **auth_signature** : concaténation de l'uri et du token (auth_token) chiffrage via la l'appid ;
- ❖ **auth_timestamp** : timestamp
- ❖ **auth_signature_method** : toujours "HMAC-SHA1"

Ces paramètres sont concaténés sous la forme :

```
Auth auth_consumer_key=valeur&auth_callback=valeur&...
```

Authentification des API

Il en fin également possible de générer le header via l'api [/v1/authheader](#):

```
$ curl "https://api.uprodit.com/v1/authheader" -d
'{"appid":"XXXXXXX","env":"YYYY","uri":"https://api.uprodit.com/v1/search/all?startIndex=0&maxResults=10&usecase=perso"}' -H "Content-Type:
application/json" 2>/dev/null | jq .
{
  "authorization": "Auth
?auth_signature=0ch4ZZB0t%2FWI1FLbTF6ODqs2xfc%3D&auth_nonce=3c9pqK7i%2BBY%2B%2BLsXxF5%2BsdD1MUk%3D&auth_callback=%2Fv1%2Fs
earch%2Fall%3FstartIndex%3D0%26maxResults%3D10%26usecase%3Dperso&auth_timestamp=1653840105914&auth_token=6f4703db-79b4-495e-b1d1-
bdc0fcfe3d77&auth_signature_method=HMAC-SHA1&auth_consumer_key=IQS5cRkibAZB7Sofg7vuLlbrpwA%3D"
}
```

Il faudra passer cette valeur dans le header **Authorization** du webservice que l'on souhaite appeler (qui correspond à l'uri passé dans le body). Cette solution est adaptée pour faire des tests rapides mais n'est pas recommandée en production car vous serez obligé de faire une double quantité d'appels (car pour chaque appel la signature est différente). Il vaut mieux avoir localement le code qui vous permet de générer la signature à partir de l'appid.

Authentification des API

Il en fin également possible de générer le header via l'api [/v1/authheader](#):

```
$ curl "https://api.uprodit.com/v1/authheader" -d
'{"appid":"XXXXXXX","env":"YYYY","uri":"https://api.uprodit.com/v1/search/all?startIndex=0&maxResults=10&usecase=perso"}' -H "Content-Type:
application/json" 2>/dev/null| jq .
{
  "authorization": "Auth
?auth_signature=0ch4ZZB0t%2FWI1FLbTF6ODqs2xfc%3D&auth_nonce=3c9pqK7i%2BBY%2B%2BLsXxF5%2BsdD1MUk%3D&auth_callback=%2Fv1%2Fs
earch%2Fall%3FstartIndex%3D0%26maxResults%3D10%26usecase%3Dperso&auth_timestamp=1653840105914&auth_token=6f4703db-79b4-495e-b1d1-
bdc0fcfe3d77&auth_signature_method=HMAC-SHA1&auth_consumer_key=IQS5cRkibAZB7Sofg7vuLlbrpwA%3D"
}
```

Il faudra passer cette valeur dans le header **Authorization** du webservice que l'on souhaite appeler (qui correspond à l'uri passé dans le body). Cette solution est adaptée pour faire des tests rapides mais n'est pas recommandée en production car vous serez obligé de faire une double quantité d'appels (car pour chaque appel la signature est différente). Il vaut mieux avoir localement le code qui vous permet de générer la signature à partir de l'appid.

Authentification des API

Exemple scénario d'authentification en utilisant curl et jq:

```
ineumann ~ $ authorization=$(curl "https://api.uprodit.com/v1/authheader" -d
'{"appid":"challenge_uprodit","env":"production","uri":"https://api.uprodit.com/v2/profile/personal/en/51"}' -H "Content-Type: application/json" 2>/dev/nulljq
.authorization -r)
ineumann ~ $ curl -H "Authorization: ${authorization}" "https://api.uprodit.com/v2/profile/personal/en/51" 2>/dev/null | jq .
{
  "id": 51,
  "name": "N",
  "surname": "Idriss",
  "anonymousDenomination": "Idriss N.",
  "denomination": "Idriss N.",
  "hobbies": "Dev|Ops",
  "about": "Engineer specialized in Information Systems, DevOps and SRE consultant at Shippeo and CTO of uprodit.com. \n\nMy favorite areas : tech lead JEE / Java,
DevOps, enterprise architecture.",
}
```

Authentification des API

Certaines API demandent également une authentification de l'utilisateur pour afficher certaines données. Pour ces API, il faut soit passer ces deux headers (méthode déconseillée dans une application SPA ou mobile):

- ❖ **x-uprodit-username:** email de l'utilisateur
- ❖ **x-uprodit-password:** mot de passe de l'utilisateur

```
$ authorization=$(curl "https://api.uprodit.com/v1/authheader" -d
'{"appid":"challenge_uprodit","env":"production","uri":"https://api.uprodit.com/v1/user/1"}' -H "Content-Type: application/json"
2>/dev/null | jq .authorization -r)
$ curl -H "Authorization: ${authorization}" -H 'x-uprodit-username: changeit@changeit.com' -H 'x-uprodit-password: changeit'
"https://api.uprodit.com/v1/user/1" 2>/dev/null | jq .
{
  "id": 1,
  "email": "changeit@changeit.com",
  "password": "{bcrypt}xxxxxxx",
  "confirm_password": null,
  "active": 1,
  "authorizations": [
    {
      "user_id": 1,
      "role": {
        "id": 3,
        "all_profiles": 0,
```

Authentification des API

Soit le header **x-uprodit-token** avec comme valeur un token valable 30j généré avec l'endpoint [/v1/token](#) de la façon suivante:

```
ineumann $ authorization=$(curl "https://api.uprodit.com/v1/authheader" -d
'{"appid":"challenge_uprodit","env":"production","uri":"https://api.uprodit.com/v1/token"}' -H "Content-Type: application/json" 2>/dev/nulljq
.authorization -r)
ineumann $ token=$(curl -X POST https://api.uprodit.com/v1/token -H "Authorization: ${authorization}" -H "Content-Type: application/json" -d
'{"username":"changeit@changeit.com","password":"changeit"}' 2>/dev/nulljq .token -r)
ineumann $ authorization=$(curl "https://api.uprodit.com/v1/authheader" -d
'{"appid":"challenge_uprodit","env":"production","uri":"https://api.uprodit.com/v1/user/1"}' -H "Content-Type: application/json" 2>/dev/nulljq
.authorization -r)
ineumann $ curl -H "Authorization: ${authorization}" -H "x-uprodit-token: ${token}" "https://api.uprodit.com/v1/user/1" 2>/dev/nulljq .
{
  "id": 1,
  "email": "changeit@changeit.com",
  "password": "{bcrypt}xxxxxxxxxxxxxxxxxxxxx",
}
```

Cette méthode est préférable pour éviter de multiplier les chances de compromission du mot de passe hors-transit. Vous pouvez garder ce token dans un cookie durant 30j puis vous reconnecter (pas encore de système de refresh token mais c'est sur la roadmap).

Quelques exemples d'API

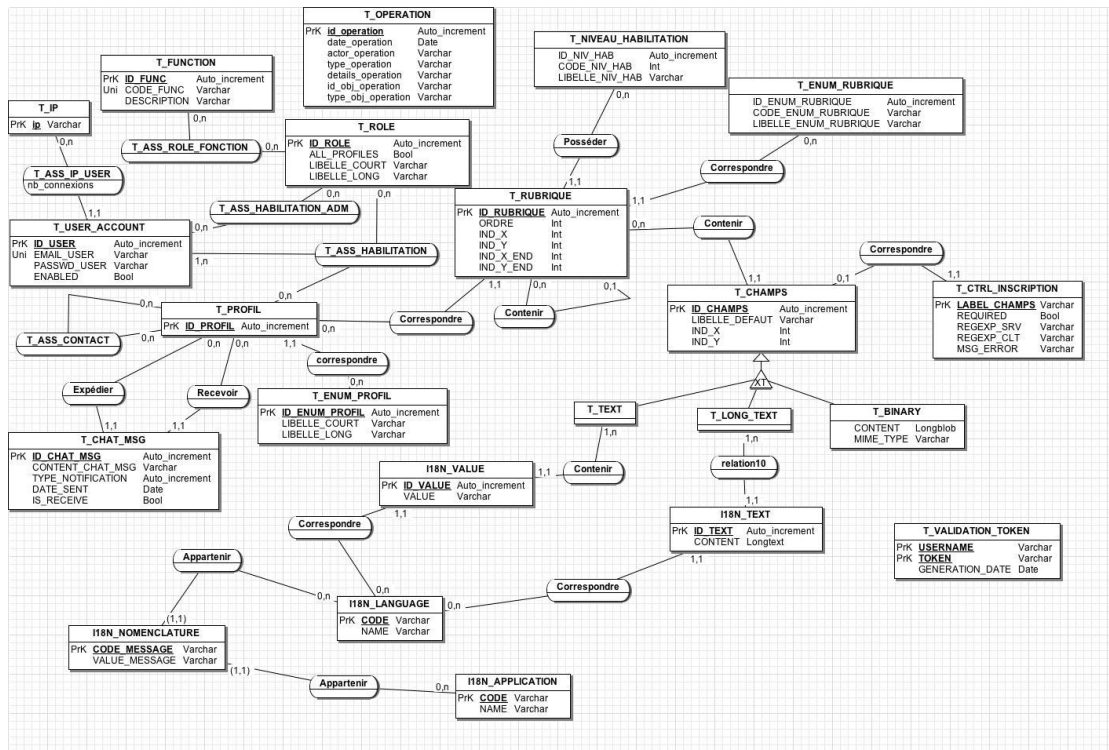
API générique de recherche des profils multi-critères:

```
ineumann ~ $ authorization=$(curl "https://api.uprodit.com/v1/authheader" -d
'{"appid":"challenge_uprodit","env":"production","uri":"https://api.uprodit.com/v1/search/all?usecase=perso&startIndex=0&maxResults=1"}' -H
"Content-Type: application/json" 2>/dev/nulljq .authorization -r)
ineumann ~ $ curl -H "Authorization: ${authorization}" "https://api.uprodit.com/v1/search/all?usecase=perso&startIndex=0&maxResults=1" 2>/dev/nulljq .
[
  {
    "id": "130150",
    "denomination": "Khalil B.",
    "gender": "man",
    "image_id": 14896397,
    "usecase": "perso",
    "profile_type": "free",
    "specialities": [
      "Filmmaker",
      "Director",
      "Producer"
    ],
    "skills": [
      {
        "name": "Montage",
        "level": "avancé",
```

API v2 sur les profils ? Pourquoi faire ?

API v2 sur les profils ? Pourquoi faire ?

Simplifier avec une couche d'abstraction un modèle de donnée très générique...



Quelques exemples d'API

API v2: Profil personnel (langue par défaut):

```
ineumann $ authorization=$(curl "https://api.uprodit.com/v1/authheader" -d
'{"appid":"challenge_uprodit","env":"production","uri":"https://api.uprodit.com/v2/profile/personal/130177"}' -H "Content-Type: application/json"
2>/dev/nulljq .authorization -r)
ineumann $ curl -H "Authorization: ${authorization}" "https://api.uprodit.com/v2/profile/personal/130177" 2>/dev/nulljq .
{
  "id": 130177,
  "denomination": "Youssef D.",
  "link": "https://www.uprodit.com/profile/personal/130177",
  "birthdate": 790992000000,
  "pictureId": 14897196,
  "bgPictureId": 14897198,
  "linkedin": "http://www.linkedin.com/in/youssef-dhraief-5743a7123",
  "github": "https://github.com/piximos",
  "gitlab": "https://gitlab.com/piximos",
  "nationality": "Tunisia",
  "website": "http://comwork.io",
  "specialities": [
    "DevOps / SRE / Automation engineer",
    "Fullstack & mobile developer",
  ],
  "skills": [
    {
      "name": "SpringBoot",
    },
  ],
}
```

Quelques exemples d'API

API v2: Profil personnel (choisir la langue):

```
ineumann $ authorization=$(curl "https://api.uprodit.com/v1/authheader" -d
'{"appid":"challenge_uprodit","env":"production","uri":"https://api.uprodit.com/v2/profile/personal/fr/130177"}' -H "Content-Type: application/json"
2>/dev/null|jq .authorization -r)
ineumann $ curl -H "Authorization: ${authorization}" "https://api.uprodit.com/v2/profile/personal/fr/130177" 2>/dev/null|jq .
{
  "id": 130177,
  "denomination": "Youssef D.",
  "link": "https://www.uprodit.com/profile/personal/130177",
  "languages": [
    {
      "name": "Français",
      "level": "Bien"
    },
    {
      "name": "Anglais",
      "level": "Excellent"
    },
    {
      "name": "Arabe",
      "level": "Langue maternelle"
    }
  ]
}
```

Quelques exemples d'API

API v2: Récupérer la photo de profil:

```
ineumann $ authorization=$(curl "https://api.uprodit.com/v1/authheader" -d
'{"appid":"challenge_uprodit","env":"production","uri":"https://api.uprodit.com/v2/profile/picture/f/14897196"}' -H "Content-Type: application/json"
2>/dev/nulljq .authorization -r)
ineumann $ curl -H "Authorization: ${authorization}" "https://api.uprodit.com/v2/profile/picture/f/14897196" 2>/dev/nulljq .
{
  "id": 9385485,
  "profileId": 130177,
  "fileId": 14897194,
  "searchId": 14897196,
  "x": 0,
  "y": 0,
  "direction": 0,
  "b64Content": "iVBORw0KGgoAAAANSUUEUgAAAvMAAAAMECAIAAAC9h...."
  "mimeType": "image/png",
  "usingSearchId": true
}
ineumann $ curl -H "Authorization: ${authorization}" "https://api.uprodit.com/v2/profile/picture/f/14897196" 2>/dev/nulljq .b64Content -r|base64 -d >
~/Desktop/youssef.png
```

Quelques exemples d'API

API v2: Récupérer les réalisations du profil:

```
ineumann $ authorization=$(curl "https://api.uprodit.com/v1/authheader" -d
'{"appid":"challenge_uprodit","env":"production","uri":"https://api.uprodit.com/v2/profile/realization/all?profileId=51&lng=fr&startIndex=0&maxResults=1"}' -H "Content-Type: application/json" 2>/dev/nulljq .authorization -r)
ineumann $ curl -H "Authorization: ${authorization}"
"https://api.uprodit.com/v2/profile/realization/all?profileId=51&lng=fr&startIndex=0&maxResults=1" 2>/dev/nulljq .
{
  "startIndex": 0,
  "maxResults": 1,
  "totalResults": 47,
  "results": [
    {
      "id": 9385799,
      "profileId": 51,
      "lng": "fr",
      "about": "L'ensemble de mes cours sur Linux, Java, Scrum sur la première communauté francophone des professionnels de l'informatique.",
      "type": 1,
      "name": "Mes cours sur developpez.com",
      "creationDate": 1330387200000,
      "content": "http://ineumann.developpez.com",
      "totalStars": 4
    }
  ]
}
```

Introduction à l'observabilité

Id de corrélation en cas d'erreur ou de bug:

The screenshot shows a web browser at the URL `www.uprodit.com/profile/personal/5100`. The page displays a "RESOURCE NOT FOUND" error. The browser's developer tools are open, showing the Network tab. The list of resources includes:

- 5100
- jquery-2.1.3.min.js
- jquery-ui.min.js
- bootstrap.min.js
- jquery.inview.min.js
- jquery.dataTables.min.js
- dataTables.colReorder.js
- dataTables.colVis.js
- jquery.dataTables.custom.json-1.0.js
- jquery.dataTables.custom.html-1.0.js
- parsley.min.js
- jquery.autosubmitform-1.2.js

The "5100" resource is selected, and its headers are displayed:

- Connection: keep-alive
- Content-Encoding: gzip
- Content-Language: en-US
- Content-Type: text/html; charset=UTF-8
- Date: Sun, 29 May 2022 18:29:58 GMT
- Expires: 0
- Pragma: no-cache
- Server: nginx
- Transfer-Encoding: chunked
- X-Content-Type-Options: nosniff
- X-Frame-Options: DENY
- X-UPRODIT-CID: 92c786c7-0772-4469-b30c-d41bec03e2f6**
- X-XSS-Protection: 1; mode=block

Introduction à l'observabilité

Retrouver la root cause de l'erreur dans Kibana grâce au CID:

The screenshot displays the Elastic Kibana interface. At the top, the Elastic logo and a search bar are visible. Below the search bar, the 'Discover' tab is selected. The search query is '"92c786c7-0772-4469-b30c-d41bec03e2f6"'. The results show 1 hit. The left sidebar contains a list of available fields, including 'container.image.name', 'message', and various agent-related fields. The main content area shows a histogram of the search results and a document viewer displaying the message: "Une exception technique est survenue : d41bec03e2f6, message = HTTP 404 Not Found, type = NotFoundException, thrown: {commonElementCount: 0, localizedMessage: HTTP 4".

Introduction à l'observabilité

API "Imalive" pour récupérer les métriques des machines: <https://gitlab.comwork.io/oss/imalive>

```
ineumann $ curl "https://imalive.uprodit.com" 2>/dev/null | jq .
```

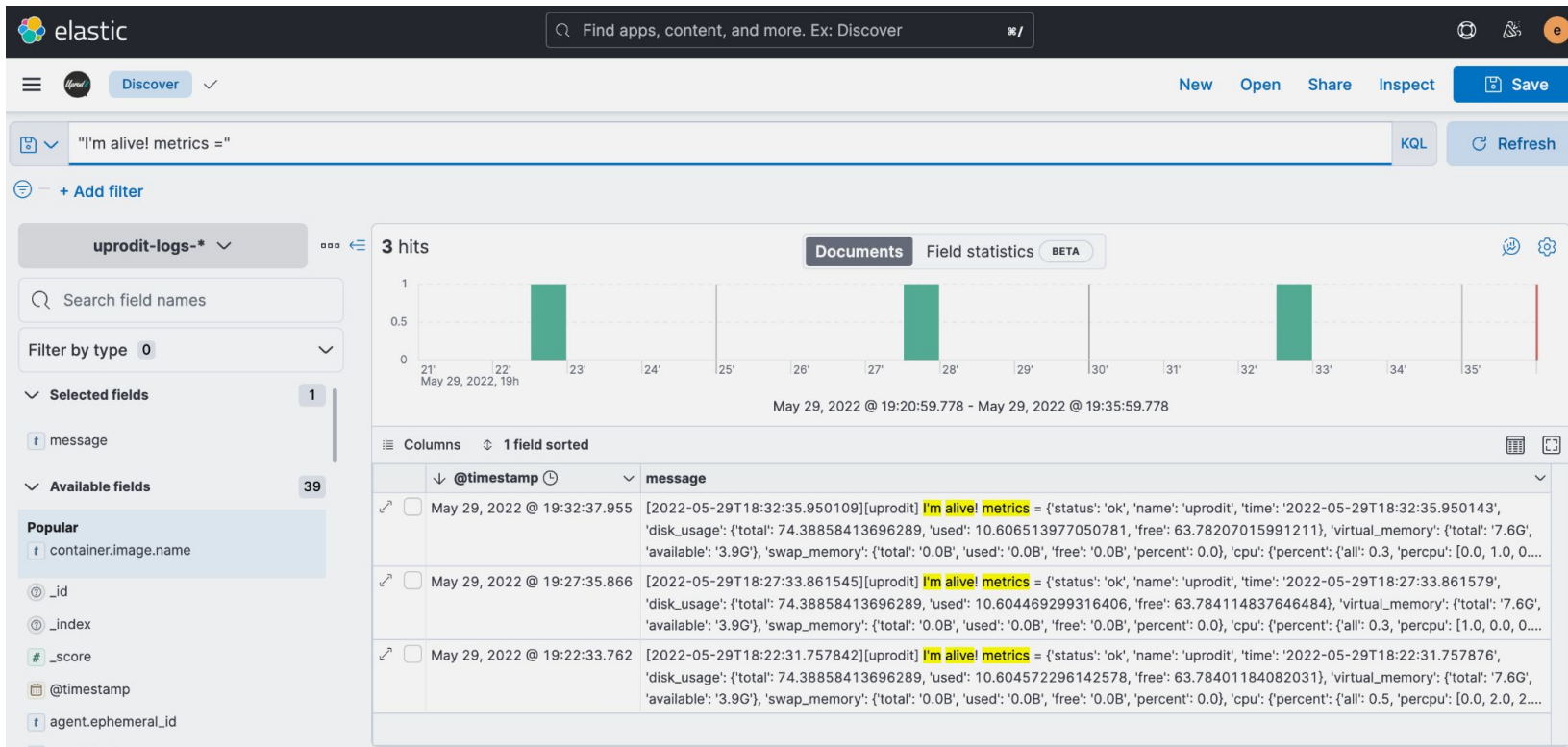
```
{
  "status": "ok",
  "time": "2022-05-29T18:37:57.562244",
  "alive": true,
  "name": "uprodit"
}
```


```
ineumann $ curl "https://imalive.uprodit.com/metrics" 2>/dev/null
```

```
{
  "status": "ok",
  "name": "uprodit",
  "time": "2022-05-29T18:38:46.081034",
  "disk_usage": {
    "total": 74.38858413696289,
    "used": 10.606399536132812,
    "free": 63.78218460083008
  },
  "virtual_memory": {
    "total": "7.6G",
    "available": "3.9G",
    "swap_memory": {
      "total": "0.0B",
      "used": "0.0B",
      "free": "0.0B",
      "percent": 0.0
    },
    "cpu": {
      "percent": {
        "all": 0.5,
        "percpu": [0.0, 1.0, 0.0, 1.0]
      },
      "count": {
        "all": 4,
        "with_logical": 4
      },
      "times": {
        "all": [
          541242.19, 2958.47, 129028.97, 50413597.07, 9045.26, 0.0, 12653.86, 35723.58, 0.0, 0.0,
          4784.58, 8301.83, 0.0, 0.0, 135619.17, 750.42, 36621.26, 12596458.92, 2375.21, 0.0, 2648.01, 9410.1, 0.0, 0.0,
          137552.66, 849.0, 36397.39, 12593065.7, 2338.34, 0.0, 3057.26, 10106.2, 0.0, 0.0, 125794.19, 800.01, 30149.79, 12622730.9, 2425.22, 0.0, 2163.99, 7905.43, 0.0, 0.0]
        ]
      }
    }
  }
}
```

Introduction à l'observabilité

Les métriques sont également périodiquement stockée dans Elasticsearch pour faire des graphs:



A decorative graphic in the top-left corner consisting of two overlapping parallelograms, one blue and one green, both slanted downwards to the right.

Conclusion / questions

Merci. Avez-vous des questions ?

