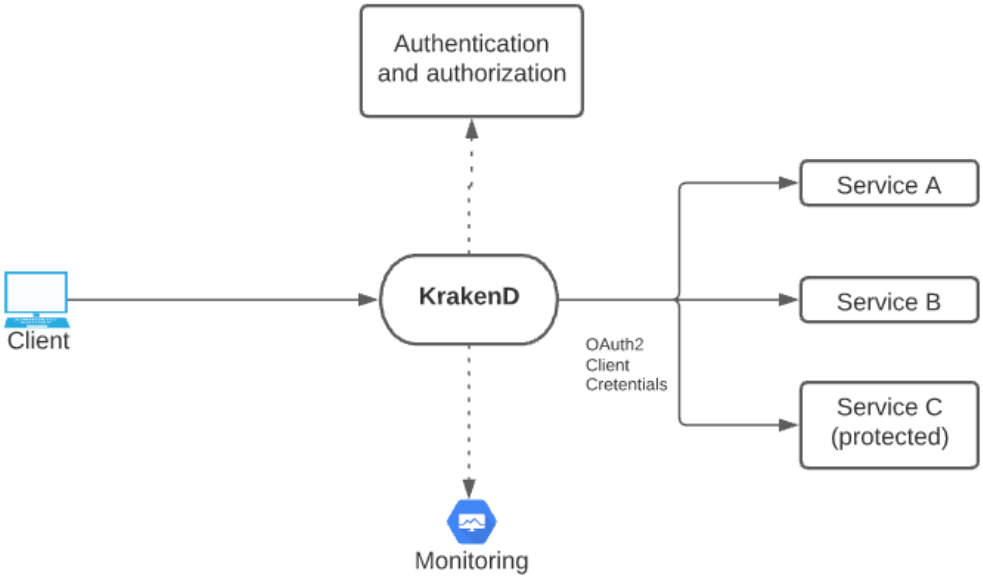
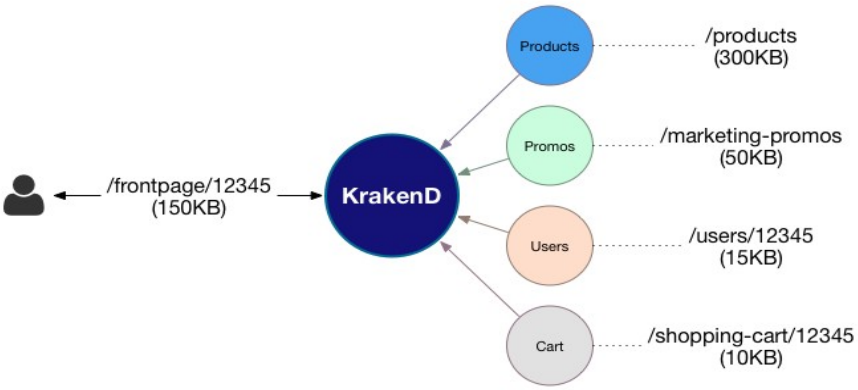
Arsitektur KrakenD



KrakenD berfungsi sebagai jembatan atau penghubung antara server penyedia API dengan klien. Fungsi utamanya yaitu sebagai agregator/penggabung dari beberapa *microservice* menjadi *single endpoint*, misalnya dengan mengambil data *product*, *users*, *promo*, dan *shopping cart* dari sumber penyedia API yang berbeda kemudian ditampilkan dalam sebuah *endpoint* saja yaitu /frontpage (ilustrasi pada Gambar 1.1). Selain itu, KrakenD juga dapat melakukan : *transform*, *filter*, *decode*, *throttle*, *auth* dan sebagainya.



Gambar 1.1

Dengan ini maka klien tidak terlibat langsung dalam penerapan *backend*. Setiap kali *backend* mengubah kontraknya, kontrak API untuk klien tetap sama dan gateway diperbarui melalui perubahan konfigurasi sederhana.

Support File, Struktur, dan Konfigurasi File KrakenD

Format file konfigurasi secara default adalah JSON, KrakenD juga dapat mengurai format lainnya, yaitu:

.json
.toml
.yaml
.yml
.properties
.props
.prop
.hcl

Namun format file yang direkomendasikan adalah JSON. Mengapa?

- Menggunakan UI : apabila ingin mengkonfigurasi file melalui KrakenDesigner, input dan outputnya selalu json.
- Fleksibel : dengan menggunakan json, kita dapat membagi file konfigurasi menjadi beberapa bagian atau menggunakan variabel di dalam konfigurasi.
- Dokumentasi : seluruh dokumentasi pada *website* KrakenD menggunakan format json.

Struktur file JSON

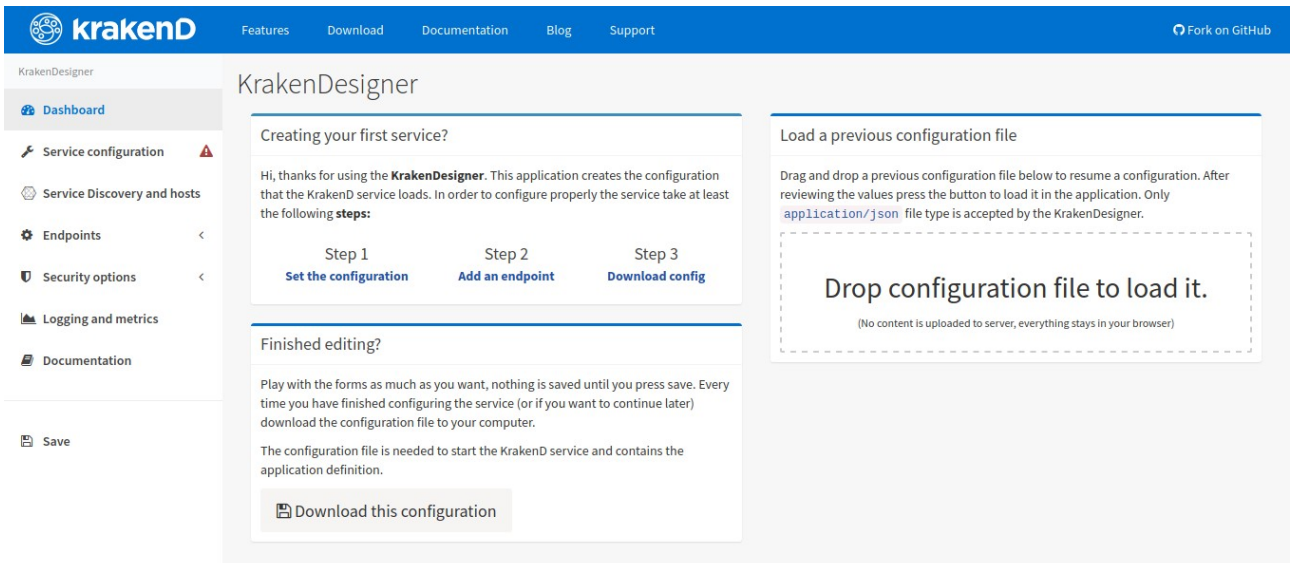
```
{
    "version": 2,
    "endpoints": [],
    "extra_config": {}
}
```

Keterangan :

- version, merupakan versi KrakenD saat ini, yaitu versi ke 2.
- endpoints, baris objek endpoint dari gateway, backend, dan konfigurasi terkait
- extra_config, tempat untuk konfigurasi tambahan, seperti menambahkan *authorization*, *rate limit*, dll.

Konfigurasi dasar KrakenD

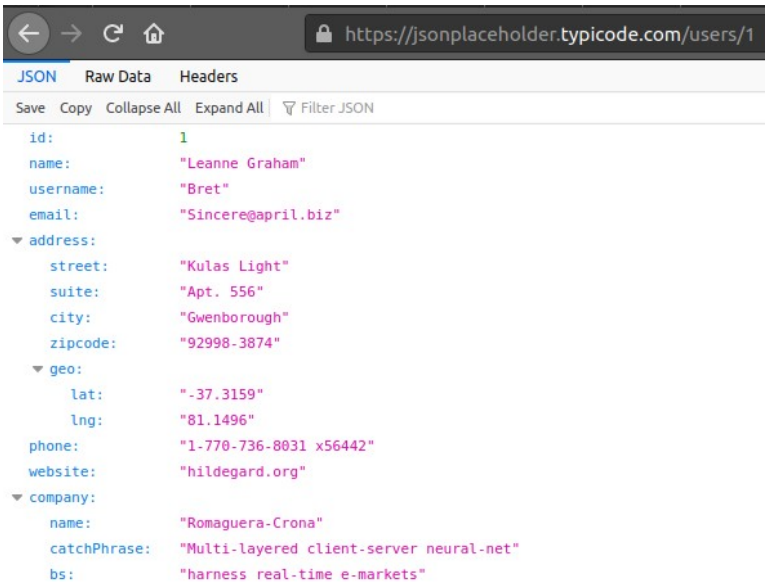
KrakenD menyediakan sebuah UI yaitu KrakenDesigner untuk membuat konfigurasi filenya, sehingga tidak perlu melakukan pemrograman dan file konfigurasi bisa langsung digunakan. Berikut tampilan dari KrakenDesigner yang dapat diakses melalui www.designer.krakend.io/.



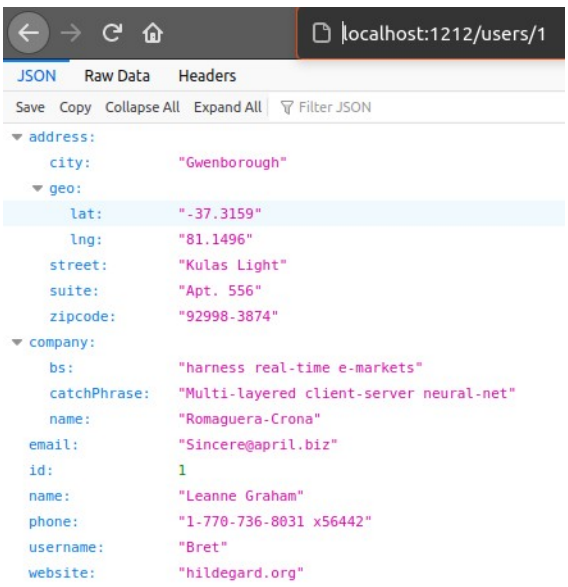
Dibawah ini adalah contoh konfigurasi dasar file json melalui KrakenDesigner.

```
{
  "version": 2,
  "extra_config": {},
  "timeout": "3000ms",
  "cache_ttl": "300s",
  "output_encoding": "json",
  "name": "tessssss",
  "port": 1212,
  "endpoints": [
    {
      "endpoint": "/users/{id}",
      "method": "GET",
      "output_encoding": "json",
      "extra_config": {},
      "backend": [
        {
          "url_pattern": "/users/{id}",
          "encoding": "json",
          "sd": "static",
          "method": "GET",
          "extra_config": {},
          "host": [
            "https://jsonplaceholder.typicode.com/"
          ],
          "disable_host_sanitize": false
        }
      ]
    }
  ]
}
```

Konfigurasi diatas digunakan untuk menampilkan data dari www.jsonplaceholder.typicode.com/ (merupakan penyedia *fake* API gratis yang bisa digunakan untuk melakukan *testing* dan *prototyping*) kemudian ditampilkan di localhost yang beralamat localhost:1212/users/{id}. Parameter {id} diganti dengan id user yang ingin diambil berdasarkan data yang ada di *website* jsonplaceholder. Berikut hasilnya :



Gambar 1.2a Tampilan di *website* jsonplaceholder



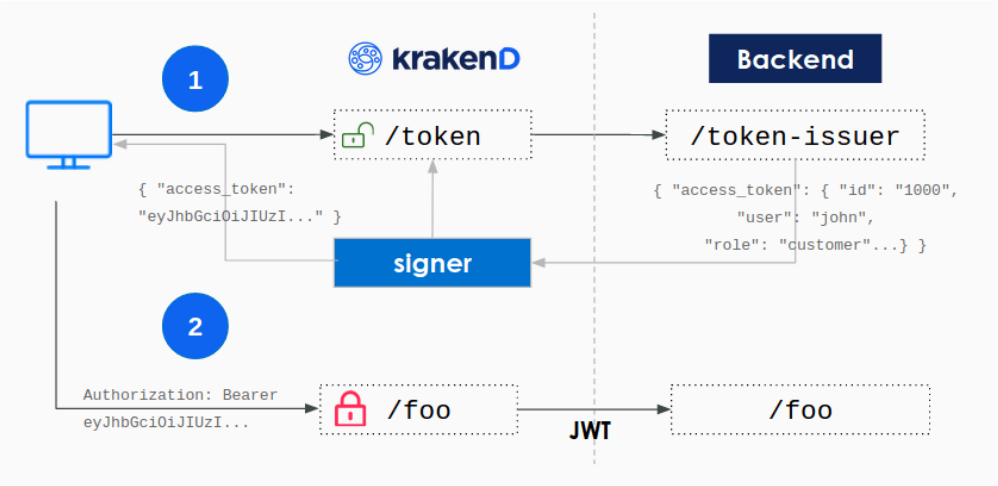
Gambar 1.2b Tampilan di *localhost*

Autentifikasi dan otorisasi

KrakenD menerapkan JSON Web Token (JWT) yaitu standar industri untuk mewakili klaim secara aman antara dua pihak. JWT adalah objek JSON yang dienkode yang berisi pasangan nilai kunci dari atribut yang ditandatangani oleh otoritas tepercaya. Saat JWT melindungi *endpoint* tertentu, permintaan ke gateway API harus memberikan token. Verifikasi token dilakukan di setiap permintaan, termasuk pemeriksaan tanda tangan dan secara opsional jaminan bahwa penerbit, peran, dan audiens cukup untuk mengaksesnya.

KrakenD menerapkan model *JWT Signing* dan JWT *Validation* untuk melindungi *endpoint* dari pengguna yang tidak diinginkan yang tidak berhak menggunakan informasi serta untuk memperkuat keamanan. Perbedaan utama antara JWT Signing dengan JWT Validation terletak pada *key* yang digunakan dimana JWT Singing menggunakan key yang diperoleh oleh *backend* lokal (bersifat pribadi), sedangkan JWT Validation menggunakan *public key* yang diperoleh dari penyedia server identitas seperti Auth0, Azure AD, Google Firebase, Keycloak, dll.

- *Sign tokens*, ketika belum memiliki server identitas (seperti aplikasi monolitik klasik dengan endpoint /login) KrakenD dapat menangani penandatanganan token dengan kunci pribadi. Prosesnya adalah sebagai berikut:



Endpoint "/foo" terproteksi oleh JWT Signing, maka diperlukan sebuah token untuk mengaksesnya. Pertama, user perlu mengakses endpoint "/token" yang akan mengenerate sebuah token dari *local backend*, kemudian diperoleh sebuah *access token.* Kedua, token yang diperoleh digunakan untuk memvalidasi *endpoint* "/foo", apabila token tersebut valid maka data dari endpoit tersebut ditampilkan. Sebagai catatan, e*ndpoint* "/token" dan "/token-issuer" bersifat lokal yang perlu dimiliki oleh *local backend* sebelum menerapkan JWT Signing. Dalam mengklaim token diperlukan *software* pihak ketiga seperti Postman.

- *Validate tokens* yang dikeluarkan oleh pihak ketiga atau middleware penandatanganan JWT, memastikan integritas dan klaim yang tepat. Contoh konfigurasi dasarnya yaitu sebagai berikut :

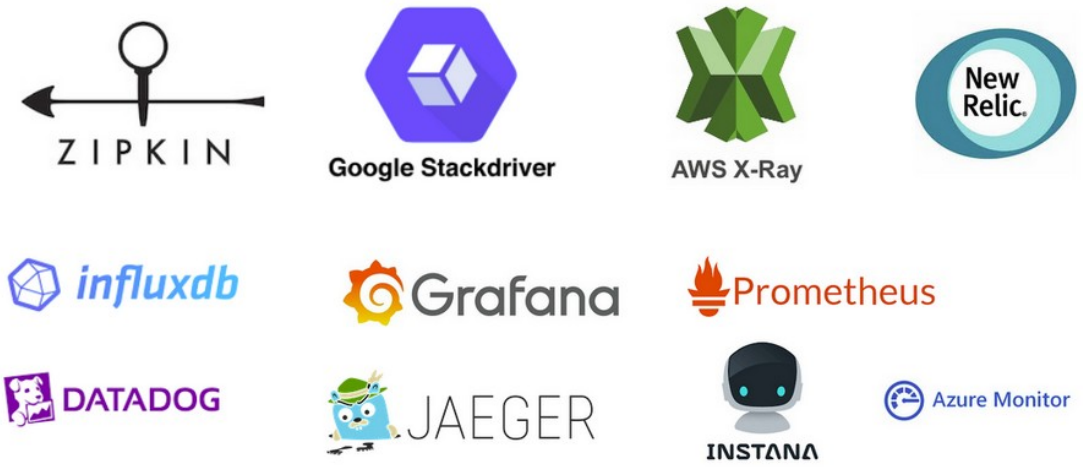
```
{
    "endpoint": "/protected/resource",
    "extra_config": {
        "github.com/devopsfaith/krakend-jose/validator": {
            "alg": "RS256",
            "audience": ["http://api.example.com"],
            "roles_key": "http://api.example.com/custom/roles",
            "roles": ["user", "admin"],
            "jwk-url": "https://albert-test.auth0.com/.well-known/jwks.json"
        }
    },
    "backend": [
        {
        "url_pattern": "/"
        }
    ]
}
```

Konfigurasi diatas membuat endpoint "/protected/resource" ketika diakses tidak dapat menampilkan data (*Unauthorized*) karena telah terproteksi oleh *JWT Validation.* Untuk mengaksesnya diperlukan sebuah token untuk mengklaim *endpoint* tersebut yang diperoleh

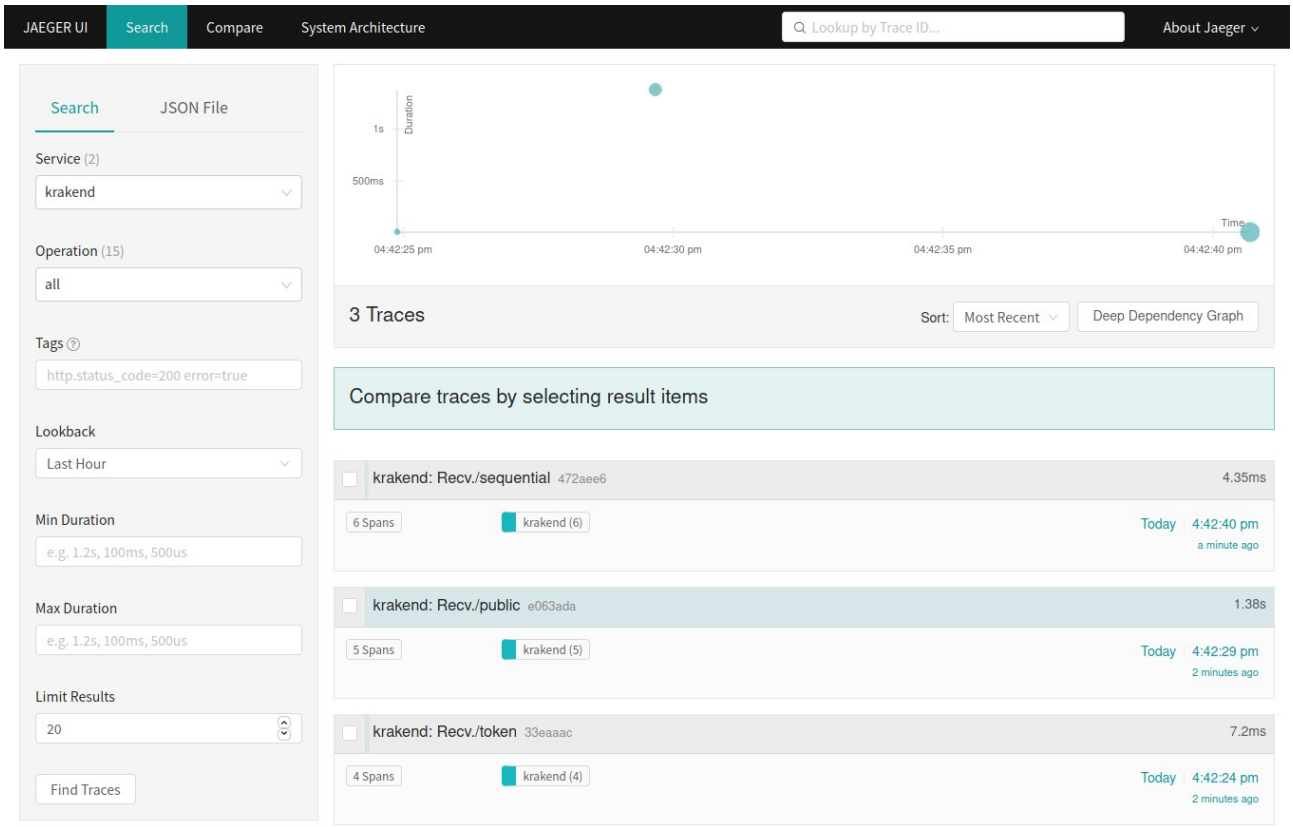
dengan mengenerate token dari public key yang disediakan oleh server identitas Auth0. Kemudian token tersebut divalidasi (untuk memvalidasinya diperlukan software lain seperti Postman), jika token tersebut valid maka data dalam endpoint "/protected/resource" ditampilkan.

OAuth2 Client Credentials
Melalui OAuth2 Client Credentials, KrakenD dapat melakukan permintaan ke *authorization server* sebuah akses token untuk menjangkau suatu *resource* yang terproteksi. Dalam hal ini KrakenD bertindak sebagai klien. Ketika berhasil mengatur kredensial klien untuk backend berarti KrakenD bisa mendapatkan konten yang terproteksi, akan tetapi *endpoint* yang ditawarkan kepada pengguna akhir akan menjadi publik kecuali diproteksi dengan JWT.

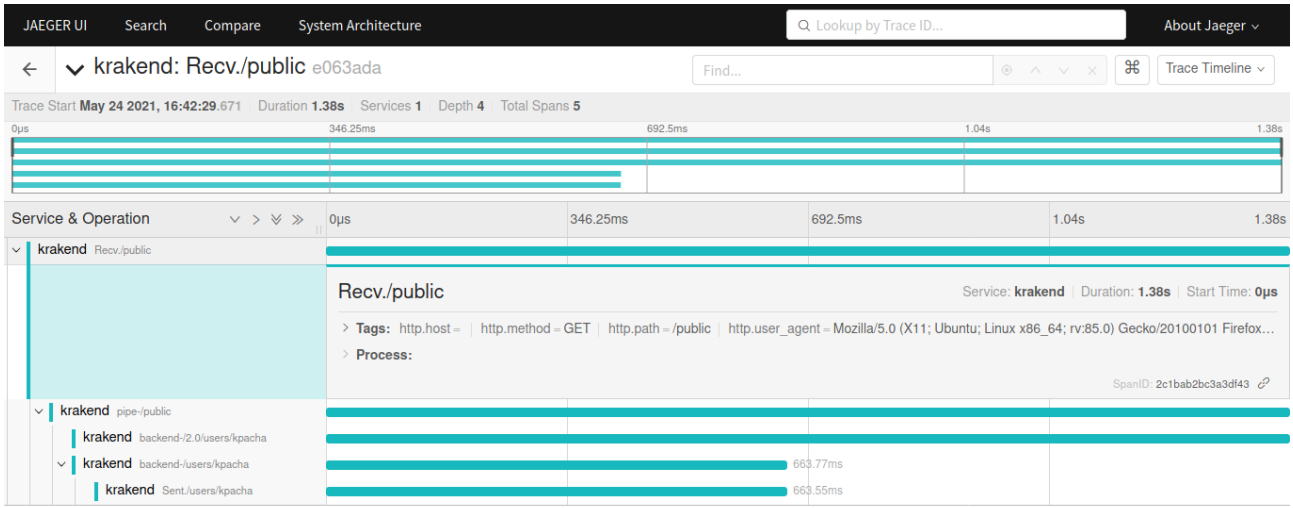
Monitoring
Kunci keberhasilan dalam skenario arsitektur layanan mikro terdistribusi adalah pengamatan dan jaringan. Maka diperlukan suatu tools yang mampu mendeteksi akar penyebab masalah, pemantauan dan detail dari transaksi terdistribusi yang berbeda, serta pengoptimalan kinerja dan latensi. Dalam hal ini, KrakenD mendukung beberapa tools untuk melakukan monitoring.



Tools diatas memiliki kelebihan dan kekurangannya masing-masing. Salah satu *tools* yang bersifat *open source* adalah Jaeger. Jaeger merupakan sistem pelacakan *end-to-end* terdistribusi yang memungkinkan untuk melakukan pemantauan dan memecahkan masalah transaksi dalam sistem terdistribusi yang kompleks. Dalam melakukan proses pemantauan, diperlukan suatu eksporter yang mampu mengeksport data ke Jaeger, yaitu dengan menggunakan OpenCensus exporter. Setelah data dieksport ke Jaeger, maka akan menampilkan UI berupa *trace* seluruh proses yang telah dilakukan. Contoh tampilannya adalah sebagai berikut :



Hasil diatas menampilkan *trace* yang telah dilakukan selama satu jam terakhir. Terlihat bahwa setiap proses berjalan dengan baik tanpa adanya *error*.
Apabila salah satu *trace* di klik maka akan menampilkan informasi yang lebih detail, seperti gambar dibawah ini :



Docker Deployment

Docker adalah aplikasi open source untuk menyatukan file-file yang dibutuhkan sebuah software sehingga menjadi menjadi satu kesatuan yang lengkap dan berfungsi. Data pengaturan dan file pendukung disebut sebagai image. Selanjutnya kumpulan image digabung dalam satu wadah yang disebut Container.

KrakenD mendukung *deployment* menggunakan docker yang dapat menggabungkan beberapa layanan seperti *web service*, *fake api*, dan *monitoring tools* menjadi satu. Sehingga dapat meningkatkan keefektivitasan, karena hanya dengan sekali running saja dapat menjalankan banyak layanan sekaligus. Dalam dokumentasi KrakenD, format file yang digunakan dalam *docker deployment* adalah **.yml** dan untuk menjalaknannya menggunakan perintah **nama_file up** melalui terminal. Contohnya filenya sebagai berikut :

Misalnya nama file dibawah adalah docker-compose.yml, perintah untuk menjalankannya adalah **docker-compose up.**

```yaml
version: "3"
services:
  grafana:
    build:
      dockerfile: Dockerfile
      context: ./grafana
    ports:
      - "3003:3000"
  influxdb:
    image: influxdb:latest
    environment:
      - "INFLUXDB_DB=krakend"
      - "INFLUXDB_USER=krakend-dev"
      - "INFLUXDB_USER_PASSWORD=pas5w0rd"
      - "INFLUXDB_ADMIN_USER=admin"
      - "INFLUXDB_ADMIN_PASSWORD=supersecretpassword"
    ports:
      - "8086:8086"
  jaeger:
    image: jaegertracing/all-in-one:latest
    ports:
      - "16686:16686"
      - "14268:14268"
  fake_api:
    image: jaxgeller/lwan
    volumes:
      - ./data:/lwan/wwwroot
    ports:
      - "8000:8080"
  web:
    build:
      context: ./web
    ports:
      - "3000:3000"
```

https://www.niagahoster.co.id/blog/docker-tutorial