LOCA SERVICE
au degré de vos envies

PROCEDURE KUZZLE

PROJET : KUZZLE
DATE : 16/04/2024

Sommaire

I. Installation globale de Kuzzle

a. *Prérequis*

Le système d'exploitation utilisé dans cette procédure est debian 12. Il est aussi possible d'installer Kuzzle à partir de Windows et Ubuntu. Ci-dessous, il est vu les prérequis avant d'installer Kuzzle, le frond-end et l'interface administrateur. Il suffit de faire les étapes dans l'ordre.

Il faut tout d'abord agir sur certains paramètres :

- Se placer en tant qu'utilisateur root : su …
- Augmenter le nombre limite de fichiers que le système peut surveiller pour les modifications :
  sudo sysctl -w fs.inotify.max_user_watches=524288

De nombreuses dépendances vont être nécessaires dans cette procédure :

- software-properties-common : apt install software-properties-common
- Installer les outils essentiels : sudo apt install build-essential
- curl : apt install curl
- nvm :
  curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
  source ~/.bashrc
  Redémarrer tous les terminaux ouverts
- Installer nodejs et npm : nvm install 20
- Installer make : sudo apt install make
- Typescript : npm install -g typescript
- Vue CLI : npm install -g @vue/cli
- Kourou : npm install -g kourou
- Github : apt install git
- Node-gyp : npm install -g node-gyp
- Python 2.7.18 :
  curl -O https://www.python.org/ftp/python/2.7.18/Python-2.7.18.tgz
  tar -xzvf Python-2.7.18.tgz
  cd Python-2.7.18
  ./configure --enable-optimizations
  make
  sudo make install
  sudo update-alternatives --install /usr/bin/python python /usr/local/bin/python2.7 1
  sudo update-alternatives --config python
- Python 3 est normalement déjà installé et configuré.
- Docker : https://docs.docker.com/engine/install/debian/

b. *Installer le back-end Kuzzle*

Afin d'installer Kuzzle, il est nécessaire de créer un fichier dédié au projet : mkdir nom_du_projet.
Il faut ensuite se placer dans le fichier créé : cd nom_du_projet.

Initialiser une nouvelle application avec kourou : kourou app:scaffold playground, un fichier « playground » est créé, il faut alors se placer dans le répertoire « playground ».
Installer kuzzle : npm install kuzzle cd
Lancer ElasticSearch et Redis : kourou app : start-services
Lancer Kuzzle : nohup npx nodemon --exec node -r ts-node/register app.ts

Pour vérifier que kuzzle fonctionne : http://localhost:7512

c. *Installer l'interface administrateur de Kuzzle*

L'interface administrateur de Kuzzle permet de gérer facilement les droits des utilisateurs, créer de nouveaux documents … C'est un bon moyen d'obtenir une vue d'ensemble de Kuzzle.

Créer un nouveau répertoire de projet séparé du backend kuzzle, et se positionner dedans.

1. Cloner le répertorie : git clone https://github.com/kuzzleio/kuzzle-admin-console
2. Dans le répertoire du projet : supprimer le fichier package-lock.json
3. Rétrograder nvm : nvm install 14 puis nvm alias default 14
3. Installer les fichiers dépendances npm : npm install
4. Lancer la config : npm run build
5. Installer un lancer de serveur http : npm install -g http-server
6. Lancer le serveur http en arrière-plan : http-server dist &

L'erreur courante est une mauvaise configuration de python. Pour le vérifier, lancer la commande : python --version , si la version 2.7.18 s'affiche, python est convenablement installé. Sinon, il faut recommencer l'installation.

Pour accéder à l'interface : http://localhost:8080

Si l'application Kuzzle est démarrée sur le port 7512, alors il doit être possible de s'y connecter en entrant l'IP de la machine qui héberge le kuzzle dans hostname. La version de Kuzzle est la V2.

Ne pas oublier de rebasculer sur nvm 20 avec la commande : nvm use 20
Pour vérifier quelle version est en cours : nvm current

II. Élever le lien http de Kuzzle en https sécurisé

La plupart des applications extérieures comme Actility vont nécessiter un lien https afin de créer une connexion. On va alors utiliser nginx en tant que reverse proxy. Kuzzle est hébergé sur le port 7512, nginx va créé un lien https sur un autre port que le 7512, et sa configuration va lui permettre de rediriger les flux http et ssl vers le port 7512 de Kuzzle. Nous allons utiliser certbot et Let's Encrypt afin de générer un certificat ssl sécurisé.

a. *Installation de nginx et certbot & génération du certificat https*

Installer nginx : apt install nginx

Ajouter le dépôt Certbot : sudo add-apt-repository ppa:certbot/certbot

Installer certbot : sudo apt install certbot python3-certbot-nginx -y

Attention, pour la suite, il faut que le port http 80 de la machine soit ouvert au web !

Obtenir le certificat SSL Let's Encrypt : certbot --nginx -d NOM_DOMAINE
Il faut ensuite suivre les étapes indiquées.

Il est ensuite possible de fermer le port 80 au web dans le firewall.

b. *Configuration de nginx*

Il faut ensuite configurer nginx. Pour cela, rendez-vous dans le répertoire : etc/nginx/sites-available

Créer un fichier qui porte un nom en rapport avec le site avec nano : nano NOM_FICHIER
Le fichier s'ouvre, voici un exemple de configuration, copier/coller puis remplacer les mots en majuscule.

```
server {
    listen PORT_NGINX ssl;
    server_name kuzzle.loca-service.com;

    ssl_certificate /etc/letsencrypt/live/kuzzle.loca-service.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/kuzzle.loca-service.com/privkey.pem;

    error_page 497 https://$host:$server_port$request_uri;

    location / {
        proxy_pass http://localhost:PORT_KUZZLE;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Remplacer PORT_NGINX pour le port attribué au https. Si des applications extérieures au réseau local doivent accéder au site, il faut ouvrir le port sur le firewall.
Remplacer PORT_KUZZLE par le port qu'utilise Kuzzle.
Il faut que les deux ports soient différents, ou bien nginx va sans cesse rediriger les requêtes vers lui.

Copier la configuration dans sites-enabled : sudo ln -sf /etc/nginx/sites-available/kuzzle /etc/nginx/sites-enabled/
Redémarrer nginx : systemctl restart nginx
S'il y a une erreur à cette étape, nginx est peut-être déjà en cours sur le port souhaité.
Afin de couper les services nginx : sudo systemctl stop nginx
Vérifier qu'il n'y a pas de service nginx sur le port utilisé : lsof -i :PORT
S'il y a d'autre processus que nginx : kill PID
Après ces étapes, il est normalement possible de redémarrer les services nginx sans erreur.

III. Développer une application Kuzzle

a. Conseils

Développer le backend se fait sur le fichier app.ts, il s'agit d'un fichier de projet en typescript.
Au cours du développement il sera nécessaire de redémarrer plusieurs fois l'application kuzzle pour appliquer les changements. Le moyen le plus simple est de tuer le processus de l'application avec la commande : kill $(lsof -t -i:7512). Il faut ensuite sauvegarder le fichier app.ts puis redémarrer l'application avec la commande nohup npx nodemon --exec node -r ts-node/register app.ts.

Afin de faciliter la programmation, il faut installer l'interface graphique de debian 12 et installer visual studio code. Il vaut mieux se connecter en utilisant l'application connexion Bureau à distance et en tant qu'utilisateur root pour avoir tous les droits si Kuzzle a été créé en tant que root.

ElasticSearch est le processus de stockage et de gestion des données, si son processus est coupé sans sauvegarde, toutes les données sont perdues.

Si les logs ne s'affichent pas sur le terminal, ceux-ci sont conservés dans le fichier nohup.out dans le répertoire de kuzzle. Il faut alors créer un nouveau terminal, entrer en root, et lancer la commande : tail -f nohup. Les logs s'afficheront alors sur ce nouveau terminal.

b. Lire et décoder une requête https

Kuzzle possède un gestionnaire d'évènements https/http, mais aussi pour d'autre méthodes de communication ou types d'évènements comme le MQTT, API, Web socket …
HTTP | Framework | Core | Kuzzle Documentation v2

On utilise la fonction hook pour lire un évènement, ainsi, pour observer le contenu d'une trame https, la fonction est :

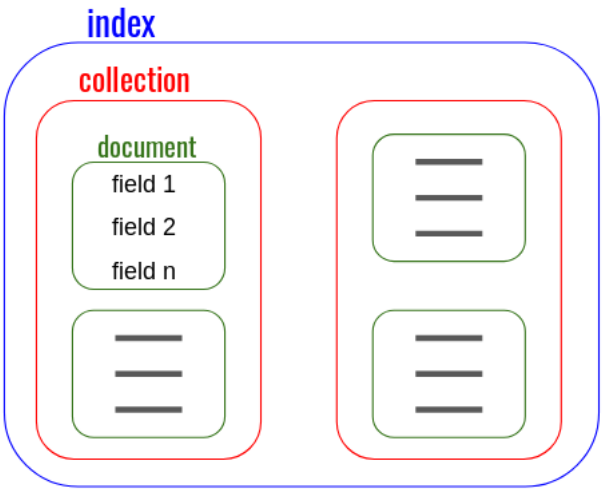
```
app.hook.register('protocol:http:beforeParsingPayload', async (request) => {
    console.log('La payload HTTP entrante avant décodage:', request);
});
```

La variable request correspond au contenu de la requête https, il est alors possible de la manipuler afin d'en extraire les données utiles.

c. *Éléments de base*

Kuzzle possède de nombreuses fonctions qui simplifient la création de son backend, c'est là tout l'intérêt de kuzzle.

Nous allons d'abord voire comment se décompose la sauvegarde de data dans Kuzzle.



Un index contient des collections, qui contiennent des documents qui contiennent des champs.
Pour exemple, l'index pourrait correspondre à un lieu, comme le siège de Loca Service, la collection, au type de data recueillie, comme une température, l'humidité … Et les documents contiennent les informations relatives à une payload. Comme le nom du capteur, sa payload, son identifiant, son emplacement …

Un exemple de code serait de créer au démarrage de Kuzzle un index et une collection, avant d'enregistrer des données dans la collection.

```
// Démarrage de l'application Kuzzle
app.start().then(async () => {
    console.log('Application Kuzzle démarrée avec succès.');

    // Création de l'index et de la collection si ils n'existent pas déjà
    if (!(await app.sdk.index.exists('iot-data'))) {
        console.log('L\'index "iot-data" n\'existe pas, création en cours...');

        await app.sdk.index.create('iot-data');
        await app.sdk.collection.create('iot-data', 'measures', {
            /*mappings: measuresMappings*/
        });

        console.log('Index "iot-data" créé avec succès.');
    } else {
        console.log('L\'index "iot-data" existe déjà.');
    }
});
```

Dans cet exemple, au démarrage de Kuzzle, est créé l'index « iot-data » et la collection « measures » dans cet index. Pour ajouter des données à la collection, on peut faire intervenir à un moment dans le programme cet exemple de fonction :

```
// Créer un document pour la température
const temperatureDoc = await app.sdk.document.create('iot-data', 'measures', {
    type: 'temperature',
    deviceId: deviceName,
    temperature: { degree: temp }
});
```

Dans cette fonction, on créer le document qui va contenir un type de mesure, un identifiant et la température en elle-même.

IV. Création d'un Front-End

a. Création du projet

Vue.js est utilisé pour le développement d'applications web. Il permet de coder son projet en une multitude de langage dont typescript. La programmation entre le backend et le frontend de kuzzle est donc très similaire.

Premièrement il faut créer un répertoire de projet dédié. Une fois dans ce répertoire, créer un nouveau projet avec la commande : vue create nom-du-projet .
Se déplacer dans le répertoire nouvellement créé portant le nom-du-projet. Pour compiler le projet, il faut lancer la commande npm run build, et pour exécuter l'application, il faut lancer la commande npm run serve.

- mkdir nom-projet
- vue create nom-du-projet : lancer en configuration manuelle, cocher typescript
- npm install kuzzle-sdk
- npm run build
- npm run serve

Le développement de l'application se passe dans le fichier component, qui contient normalement le projet HelloWorld.vue, il est possible de créer un nouveau document en .vue dans ce répertoire afin de développer un nouveau projet. Il faut veiller à appeler ce document dans le fichier App.vue.

Lorsqu'une modification est apportée, il faut relancer les commandes npm run build puis npm run serve.

b. Éléments de base

On peut retrouver plusieurs éléments de différents langages de programmation au sein d'un même fichier vue. Pour créer un script en typescript il faut utiliser la balise :
<script lang="ts"> PROGRAMME </script>
Dans cette balise il sera possible de développer les actions liées à Kuzzle.

La première chose à faire est d'importer le SDK de Kuzzle, puis de se connecter à l'instance de Kuzzle.

```
import { Kuzzle, WebSocket } from 'kuzzle-sdk';
const kuzzle = new Kuzzle(new WebSocket('192.168.14.102',{port: 7512} ))
```

Pour la suite il est alors possible de récupérer des informations de documents en utilisant la liste des fonctions fournies par Kuzzle.