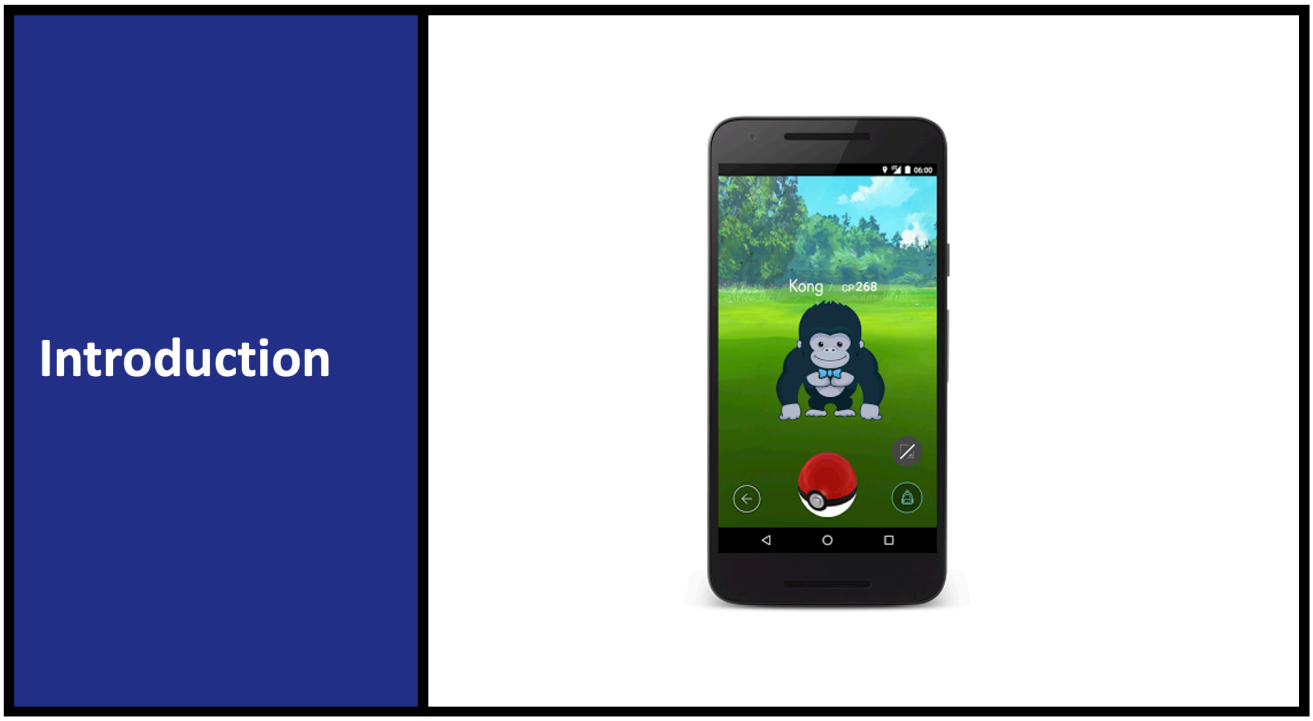
Article
Guillaume Rongier · Fév 28, 2022 38m de lecture

Open Exchange

IAM (InterSystems API Manager), le chemin vers la réussite



Cet article contient le matériel, les exemples, les exercices pour apprendre les concepts de base de IAM.

Toutes les ressources sont disponibles sur ce git : https://github.com/grongierisc/iam-training.

Les solutions sont dans la branche training.

Cet article couvre les sujets suivants :

1. Introduction

1.1. Qu'est-ce que IAM ?

IAM signifie InterSystems API Manager, il est basé sur **Kong Enterprise Edition**.

Cela signifie que vous avez accès, en plus de l'édition Kong Open Source, au :

- Portail gestionnaire
- Portail des développeurs
- Plugin avancé
    - Oauth2
    - Mise en cache
    - ...

1.2. Qu'est-ce que la gestion d"API ?

La gestion des API est le processus de création et de publication d'interfaces de programmation d'applications (API) Web, d'application de leurs politiques d'utilisation, de contrôle de l'accès, d'entretien de la communauté d'abonnés, de collecte et d'analyse des statistiques d'utilisation et de production de rapports sur les performances. Les composants de gestion des API fournissent des mécanismes et des outils pour soutenir la communauté des développeurs et des abonnés.

1.3. Portail IAM

Kong et IAM sont conçus comme des API d'abord, ce qui signifie que tout ce qui est fait dans Kong/IAM peut être fait par des appels REST ou le portail du gestionnaire.

Au cours de cet article, tous les exemples/exercices présenteront les deux de cette façon :
Portail IAM API Rest

1.4. Flux de cet article

Le but de cet article est d'utiliser IAM comme proxy d'une API rest IRIS.

La définition de cette API rest peut être trouvée ici :

```
http://localhost:52773/swagger-ui/index.html#/
```

ou ici

```
https://github.com/grongierisc/iam-training/blob/training/misc/spec.yml
```

Commencez cet article avec la branche principale.

A la fin de l'article, vous devriez avoir le même résultat que la branche d'entraînement.

2. Installation

2.1. Que devez-vous installer ?

- Git
- Docker (si vous utilisez Windows, assurez-vous de configurer votre installation Docker pour utiliser les « conteneurs Linux »).
- Docker Compose
- Visual Studio Code + l'extension VSCode InterSystems ObjectScript
- Fichier de licence IRIS IAM InterSystems activé.
- Image Docker IAM

2.2. Comment IAM fonctionne avec IRIS

Au démarrage de Kong/IAM, le conteneur vérifie la licence Kong/IAM avec un appel curl.

Le point de terminaison de cet appel est une API REST sur le conteneur IRIS.

Info : la licence Kong est incorporée dans celle d'IRIS.

2.3. Configuration

Clonez Git ce dépôt.

```
git clone https://github.com/grongierisc/iam-training
```

Exécutez l'API rest initiale :

```
docker-compose up
```

Testez-la :

```
http://localhost:52773/swagger-ui/index.html#/
```

Identifiant/mot de passe : SuperUser/SYS

2.4. Installer IAM

2.4.1. Image Iris

Vous devez d'abord passer de l'édition communautaire à une édition sous licence.

Pour ce faire, vous devez configurer votre accès à InterSystems Container Registry pour télécharger les images IRIS à accès limité.

Jetez un œil à la Présentation d'InterSystems Container Registry dans la Communauté des développeurs.

- Connectez-vous à https://containers.intersystems.com/ en utilisant vos informations d'identification du WRC et obtenez un *jeton*.

- Configurez la connexion docker sur votre ordinateur :

```
docker login -u="user" -p="token" containers.intersystems.com
```

- Obtenez l'image IRIS InterSystems :

```
docker pull containers.intersystems.com/intersystems/irishealth:2020.4.0.524.0
```

2.4.2. Image IAM

Dans la distribution logicielle WRC :

- Composants > Téléchargez le fichier *IAM-1.5.0.9-4.tar.gz*, décompressez & untar puis chargez l'image :

```
docker load -i iam_image.tar
```

2.4.3. Mettre le fichier docker à jour

Remplacer l'édition communautaire d'IRIS par une édition sous licence.

- containers.intersystems.com/intersystems/irishealth:2020.4.0.524.0
- ajoutez iris.key dans le dossier key

Modifiez le dockerfile pour ajouter par-dessus cette partie

```
ARG IMAGE=containers.intersystems.com/intersystems/irishealth:2020.4.0.524.0
# Première étape
FROM $IMAGE as iris-iam
COPY key/iris.key /usr/irissys/mgr/iris.key
COPY iris-iam.script /tmp/iris-iam.script
RUN iris start IRIS \
&& iris session IRIS < /tmp/iris-iam.script \
&& iris stop IRIS quietly

# Seconde étape
FROM iris-iam
```

Cette partie va créer un dockerfile à plusieurs étapes.

- la première étape consiste à permettre à IRIS de servir la licence IAM.
- la deuxième étape est pour la compliation de l'API REST

Créez un nouveau fichier iris-iam.script pour compiler une nouvelle image IRIS afin d'activer le point de terminaison et l'utilisateur IAM.

```
zn "%SYS"
write "Création d'une application Web ...",!
set webName = "/api/iam"
set webProperties("Enabled") = 1
set status = ##class(Security.Applications).Modify(webName, .webProperties)
```

```
write:'status $system.Status.DisplayError(status)
write "Web application "_webName_" a été mis à jour !",!

set userProperties("Enabled") = 1
set userName = "IAM"
Do ##class(Security.Users).Modify(userName,.userProperties)
write "User "_userName_" a été mis à jour !",!
halt
```

2.4.4. Mettre le docker-compose à jour

Metez le fichier docker-compose à jour vers :

- db
  - base de données postgres pour IAM
- iam-migration
  - amorcer la base de données
- iam
  - instance IAM réelle
- un volume pour les données persistantes

Ajoutez cette partie à la fin du fichier docker-compose.

```
  iam-migrations:
    image: intersystems/iam:1.5.0.9-4
    command: kong migrations bootstrap up
    depends_on:
      - db
    environment:
      KONG_DATABASE: postgres
      KONG_PG_DATABASE: ${KONG_PG_DATABASE:-iam}
      KONG_PG_HOST: db
      KONG_PG_PASSWORD: ${KONG_PG_PASSWORD:-iam}
      KONG_PG_USER: ${KONG_PG_USER:-iam}
      KONG_CASSANDRA_CONTACT_POINTS: db
      KONG_PLUGINS: bundled,jwt-crafter
      ISC_IRIS_URL: IAM:${IRIS_PASSWORD}@iris:52773/api/iam/license
    restart: on-failure
    links:
      - db:db
  iam:
    image: intersystems/iam:1.5.0.9-4
    depends_on:
      - db
    environment:
      KONG_ADMIN_ACCESS_LOG: /dev/stdout
      KONG_ADMIN_ERROR_LOG: /dev/stderr
      KONG_ADMIN_LISTEN: '0.0.0.0:8001'
      KONG_ANONYMOUS_REPORTS: 'off'
      KONG_CASSANDRA_CONTACT_POINTS: db
      KONG_DATABASE: postgres
      KONG_PG_DATABASE: ${KONG_PG_DATABASE:-iam}
      KONG_PG_HOST: db
      KONG_PG_PASSWORD: ${KONG_PG_PASSWORD:-iam}
      KONG_PG_USER: ${KONG_PG_USER:-iam}
      KONG_PROXY_ACCESS_LOG: /dev/stdout
```

```
      KONG_PROXY_ERROR_LOG: /dev/stderr
      KONG_PORTAL: 'on'
      KONG_PORTAL_GUI_PROTOCOL: http
      KONG_PORTAL_GUI_HOST: '127.0.0.1:8003'
      KONG_ADMIN_GUI_URL: http://localhost:8002
      KONG_PLUGINS: bundled
      ISC_IRIS_URL: IAM:${IRIS_PASSWORD}@iris:52773/api/iam/license
    volumes:
      - ./iam:/iam
    links:
      - db:db
    ports:
      - target: 8000
        published: 8000
        protocol: tcp
      - target: 8001
        published: 8001
        protocol: tcp
      - target: 8002
        published: 8002
        protocol: tcp
      - target: 8003
        published: 8003
        protocol: tcp
      - target: 8004
        published: 8004
        protocol: tcp
      - target: 8443
        published: 8443
        protocol: tcp
      - target: 8444
        published: 8444
        protocol: tcp
      - target: 8445
        published: 8445
        protocol: tcp
    restart: on-failure
  db:
    image: postgres:9.6
    environment:
      POSTGRES_DB: ${KONG_PG_DATABASE:-iam}
      POSTGRES_PASSWORD: ${KONG_PG_PASSWORD:-iam}
      POSTGRES_USER: ${KONG_PG_USER:-iam}
    volumes:
      - 'pgdata:/var/lib/postgresql/data'
    healthcheck:
      test: ["CMD", "pg_isready", "-U", "${KONG_PG_USER:-iam}"]
      interval: 30s
      timeout: 30s
      retries: 3
    restart: on-failure
    stdin_open: true
    tty: true
volumes:
  pgdata:
```

Ajoutez le fichier .env dans le dossier racine :

```
IRIS_PASSWORD=SYS
```

NB : Voici la définition des ports Kong :

Port	Protocole	Description
:8000	HTTP	Prend le trafic HTTP entrant des Consommateurs, et le transmet aux Services en amont.
:8443	HTTPS	Reçoit le trafic HTTPS entrant des consommateurs et le transmet aux services en amont.
:8001	HTTP	API d'administration. Écoute les appels de la ligne de commande via HTTP.
:8444	HTTPS	API d'administration. Écoute les appels de la ligne de commande via HTTPS.
:8002	HTTP	Kong Manager (GUI). Écoute le trafic HTTP.
:8445	HTTPS	Kong Manager (GUI). Écoute le trafic HTTPS.
:8003	HTTP	Portail des développeurs. Écoute le trafic HTTP, en supposant que Portail des développeurs est activé.
:8446	HTTPS	Portail des développeurs. Écoute le trafic HTTPS, en supposant que Portail des développeurs est activé.
:8004	HTTP	Portail des développeurs /écoute le trafic HTTP, en supposant que le portail des développeurs est activé.
:8447	HTTPS	Portail des développeurs /écoute le trafic HTTPS, en supposant que le portail des développeurs est activé.

2.4.5. Option : add IRISPASSWARD comme .env

Pour des raisons de facilité d'utilisation (et peut-être de sécurité), vous pouvez utiliser le fichier .env dans le dockerfile IRIS.

Pour ce faire, modifiez le docker-compose avec ceci dans la partie service iris :

```
    build:
      context: .
      dockerfile: dockerfile
      args:
        - IRIS_PASSWORD=${IRIS_PASSWORD}
```

Et le dockerfile (deuxième ou première étape de la compilation) :

```
ARG IRIS_PASSWORD
RUN echo "${IRIS_PASSWORD}" > /tmp/password.txt && /usr/irissys/dev/Container/changeP
assword.sh /tmp/password.txt
```

2.4.6. Essayez !

```
docker-compose -f "docker-compose.yml" up -d --build
```

3. Premier service/route

Vous vous souvenez comment fonctionne Kong/IAM ?

Ici, nous compilerons :

- un service
  - pour notre API crud
- une route
  - pour accéder à ce service

3.1. Créer un service

Portail IAM

API Rest

```
# Créer un servicecurl -i -X POST \--url http://localhost:8001/services/ \--data 'name=crud' \--data 'url=http://iris:52773/crud/'
```

Par défaut, tout est accessible pour un utilisateur non authentifié.

3.2. Créer une route

Portail IAM

API Rest

```
# Créer une routecurl -i -X POST \--url http://localhost:8001/services/crud/routes \--data 'name=crud-route' \--data 'paths=/persons/*' \--data 'strip_path=false'
```

Que voyons-nous ici, pour créer une route nous avons besoin de :

- son nom de service
- un chemin ou les RegEx sont autorisées

3.3. Essayez !

API originale

```
# Obsolète


curl –i --location --request GET 'http://localhost:52773/crud/persons/all' \
--header 'Authorization: Basic U3VwZXJVc2VyOlNZUw=='
```

API proxy

```
# KONG


curl –i --location --request GET 'http://localhost:8000/persons/all' \
```

```
--header 'Authorization: Basic U3VwZXJVc2VyOlNZUw=='
```

Ce que nous voyons ici :

- Rien de nouveau du côté hérité.
- Du côté kong :
  - Nous changeons le port
  - Le chemin correspond à la route
  - Nous devons toujours nous authentifier

4. Ensuite, allez plus loin avec le plugin

Pour aller plus loin, nous allons essayer d'auto-authentifier Kong au point de terminaison IRIS.

Pour ce faire, nous utiliserons un plugin, resquest-transformer.

4.1. Ajouter un plugin au service

Portail IAM

API Rest

```
# Créer un plugincurl -i -X POST \--url ht
tp://localhost:8001/services/crud/plugins
\--data 'name=request-transformer' \--data
 'config.add.headers=Authorization:Basic U
3VwZXJVc2VyOlNZUw==' \--data 'config.repla
ce.headers=Authorization:Basic U3VwZXJVc2V
yOlNZUw==='
```

4.2. Essayez !

```
# Obsolète

**API originale**
curl –i --location --request GET 'http://localhost:52773/crud/persons/all'
```

API proxy

```
# KONG


curl –i --location --request GET 'http://localhost:8000/persons/all'
```

Ce que nous voyons ici :

- Erreur 401 sur l'API originale
- Nous atteignons les données sans authentification

5. Troisièmement, ajoutez votre propre authentification

Ce que nous voulons réaliser ici est d'ajouter notre propre authentification sans aucun dérangement de l'API originale.

5.1. Ajouter des consommateurs

Portail IAM

API Rest

```
# Ajouter un consommateur anonymecurl -i -X POST \--url http://localhost:8001/consumers/ \--data "username=anonymous" \--data "custom_id=anonymous"
# Ajouter un utilisateur consommateurcurl -i -X POST \--url http://localhost:8001/consumers/ \--data "username=user" \--data "custom_id=user"
```

5.2. Ajouter un plugin d'authentification basique

Portail IAM

API Rest

```
# Activer l'authentification de base pour le servicecurl -i -X POST http://localhost:8001/routes/crud-route/plugins \--data "name=basic-auth" \--data "config.anonymous=5cc8dee4-066d-492e-b2f8-bd77eb0a4c86" \--data "config.hide_credentials=false"
```

Où :

- config.anonymous = uuid du consommateur anonyme

5.3. Ajouter un plugin ACL

Portail IAM

API Rest

```
# Activer ACLcurl -i -X POST http://localhost:8001/routes/crud-route/plugins \--data "name=acl" \--data "config.whitelist=user"
```

5.4. Configurer USER avec ACL et des identifiants

Portail IAM

API Rest

```
# Ajouter un groupe consommateurcurl -i -X POST \--url http://localhost:8001/consumers/user/acls \--data "group=user"# Add consumer credentialscurl -i -X POST http://localhost:8001/consumers/user/basic-auth \--data "username=user" \--data "password=user"
```

5.5. Essayez !

API originale

```
# Obsolète

curl –i --location --request GET 'http://localhost:52773/crud/persons/all' \
```

```
--header 'Authorization:Basic dXNlcjp1c2Vy'
```

**API proxy **

```
# KONG

curl –i --location --request GET 'http://localhost:8000/persons/all' \
--header 'Authorization:Basic dXNlcjp1c2Vy'
```

6. Exercice, limitation du débit

1. Activer un utilisateur non authentifié
2. Limiter le débit à 2 appels par minutes pour un utilisateur non authentifié

6.1. Solution

1. Activer un utilisateur non authentifié

Portail IAM

API Rest

```
# Ajouter un groupe consommateurcurl -i -X
 POST \--url http://localhost:8001/consume
rs/anonymous/acls \--data "group=user"
```

2. Limiter le débit à 2 appels par minutes pour un utilisateur non authentifié

Portail IAM

API Rest

```
# Ajouter une limitation du débit consomma
teurcurl -i -X POST \--url http://localhos
t:8001/consumers/anonymous/plugins \--data
 "name=rate-limiting" \--data "config.limi
t_by=consumer" \--data "config.minute=2"
```

7. Portail des développeurs

7.1. Vue d'ensemble

Le Portail des développeurs Kong fournit :

- une source unique de vérité pour tous les développeurs
- une gestion intuitive du contenu de la documentation
- une intégration simplifiée des développeurs
- un contrôle d'accès basé sur les rôles (RBAC)

7.2. Activez-le !

Portail IAM

API Rest

```
curl -X PATCH http://localhost:8001/worksp
aces/default --data "config.portal=true"
```

7.3. Ajoutez votre première spécification

Portail IAM

API Rest

```
curl -X POST http://localhost:8001/default/files -F "path=specs/iam-
training.yml" -F "contents=@misc/spec.yml"
```

7.4. Essayez !

```
http://localhost:8003/default/documentation/iam-training
```

Que s'est-il passé ?

Comment résoudre ce problème ?

7.5. Exercice

1. Ajouter un plugin CORS à la route

7.5.1. Solution

Portail IAM

API Rest

```
# Activer CORScurl -i -X POST http://local
host:8001/routes/crud-
route/plugins \--data "name=cors"
```

8. Portail développeur, partie deux, authentification

8.1. Activer l'authentification de base

Portail IAM

Configuration de la session (JSON)

```
{    "cookie_secure": false,    "cookie_na
me": "portal_session",    "secret": "SYS",
    "storage": "kong"}
```

L'authentification est maintenant activée pour le portail des développeurs.

8.2. Limiter l'accès

Par défaut, tout est accessible pour un utilisateur non authentifié.

Nous pouvons créer un rôle pour limiter certains accès.

Par exemple, limitons l'accès à la documentation de notre API CRUD.

8.2.1. Créer un rôle

Portail IAM

API Rest

```
# Activer le rôlecurl -i -X POST http://lo
calhost:8001/default/developers/roles \--d
ata "name=dev"
```

8.2.2. Ajouter un rôle à la spec

Portail IAM

API Rest

```
# Activer le rôle

curl 'http://localhost:8001/default/files/specs/iam-training.yml' -X PATCH -H 'Accept
: application/json, text/plain, */*'  --compressed -H 'Content-Type: application/json
;charset=utf-8' -H 'Origin: http://localhost:8002' -H 'Referer: http://localhost:8002
/default/portal/permissions/' --data-raw $'{"contents":"x-headmatter:\\n  readable_by
:\\n    - dev\\nswagger: \'2.0\'\\ninfo:\\n  title: InterSystems IRIS REST CRUD demo\
\n  description: Demo of a simple rest API on IRIS\\n  version: \'0.1\'\\n  contact:\
\n    email: apiteam@swagger.io\\n  license:\\n    name: Apache 2.0\\n    url: \'http
://www.apache.org/licenses/LICENSE-2.0.html\'\\nhost: \'localhost:8000\'\\nbasePath:
/\\nschemes:\\n  - http\\nsecurityDefinitions:\\n  basicAuth:\\n    type: basic\\nsec
urity:\\n  - basicAuth: []\\npaths:\\n  /:\\n    get:\\n      description: \' Persons
REST general information \'\\n      summary: \' Server Info \'\\n      operationId: G
etInfo\\n      x-ISC_CORS: true\\n      x-ISC_ServiceMethod: GetInfo\\n      response
s:\\n        \'200\':\\n          description: (Expected Result)\\n          schema:\
\n            type: object\\n            properties:\\n              version:\\n
          type: string\\n        default:\\n          description: (Unexpected Error
)\\n  /persons/all:\\n    get:\\n      description: \' Retreive all the records of Sa
mple.Person \'\\n      summary: \' Get all records of Person class \'\\n      operati
onId: GetAllPersons\\n      x-ISC_ServiceMethod: GetAllPersons\\n      responses:\\n
       \'200\':\\n          description: (Expected Result)\\n          schema:\\n
        type: array\\n            items:\\n              $ref: \'#/definitions/Person
\'\\n        default:\\n          description: (Unexpected Error)\\n  \'/persons/{id}
\':\\n    get:\\n      description: \' Return one record fo Sample.Person \'\\n
summary: \' GET method to return JSON for a given person id\'\\n      operationId: Ge
tPerson\\n      x-ISC_ServiceMethod: GetPerson\\n      parameters:\\n        - name:
id\\n          in: path\\n          required: true\\n          type: string\\n      r
esponses:\\n        \'200\':\\n          description: (Expected Result)\\n          s
chema:\\n            $ref: \'#/definitions/Person\'\\n        default:\\n          de
scription: (Unexpected Error)\\n    put:\\n      description: \' Update a record in S
ample.Person with id \'\\n      summary: \' Update a person with id\'\\n      operati
onId: UpdatePerson\\n      x-ISC_ServiceMethod: UpdatePerson\\n      parameters:\\n
      - name: id\\n          in: path\\n          required: true\\n          type: st
ring\\n        - name: payloadBody\\n          in: body\\n          description: Requ
est body contents\\n          required: false\\n          schema:\\n            type:
 string\\n      responses:\\n        \'200\':\\n          description: (Expected Resu
lt)\\n        default:\\n          description: (Unexpected Error)\\n    delete:\\n
    description: \' Delete a record with id in Sample.Person \'\\n      summary: \' D
elete a person with id\'\\n      operationId: DeletePerson\\n      x-ISC_ServiceMetho
d: DeletePerson\\n      parameters:\\n        - name: id\\n          in: path\\n
     required: true\\n          type: string\\n      responses:\\n        \'200\':\\n
          description: (Expected Result)\\n        default:\\n          description:
```

```
(Unexpected Error)\\n  /persons/:\\n    post:\\n      description: \' Creates a new S
ample.Person record \'\\n      summary: \' Create a person\'\\n      operationId: Cre
atePerson\\n      x-ISC_ServiceMethod: CreatePerson\\n      parameters:\\n        - n
ame: payloadBody\\n          in: body\\n          description: Request body contents\
\n          required: false\\n          schema:\\n            type: string\\n      re
sponses:\\n        \'200\':\\n          description: (Expected Result)\\n        defa
ult:\\n          description: (Unexpected Error)\\ndefinitions:\\n  Person:\\n    typ
e: object\\n    properties:\\n      Name:\\n        type: string\\n      Title:\\n
     type: string\\n      Company:\\n        type: string\\n      Phone:\\n        ty
pe: string\\n      DOB:\\n        type: string\\n        format: date-time\\n"}'
```

Ce qui est important ici, c'est cette partie :

```
x-headmatter:
  readable_by:
    - dev
```

Référez-vous à cette documentation : readableby attribute

8.2.3. Essayez !

8.2.3.1. Inscrire un nouveau développeur

8.2.3.2. Approuver ce développeur

8.2.3.3. Ajouter un rôle pour ce développeur

```
curl 'http://localhost:8001/default/developers/dev@dev.com' -X PATCH --compressed -H
'Content-Type: application/json;charset=utf-8' -H 'Cache-Control: no-cache' -H 'Origi
n: http://localhost:8002' -H 'DNT: 1' -H 'Connection: keep-alive' -H 'Referer: http:/
/localhost:8002/default/portal/permissions/dev/update' -H 'Pragma: no-cache' --data-
raw '{"roles":["dev"]}'
```

8.3. Ajouter Oauth2 pour le développeur

Dans cette partie, nous allons ajouter une authentification Oauth2 pour que les développeurs puissent utiliser notre API crud de manière sécurisée.

Ce flux permettra l'auto-enregistrement du développeur et lui donnera accès à l'API crud.

8.3.1. Tout d'abord, retirez l'authentification de base

Pour ce faire, nous allons remplacer notre authentification de base par une authentification bearToken.

Désactivez d'abord notre authentification/acl de base.

Portail IAM

API Rest

```
# Désactiver le plugin ACL
```

```
curl 'http://localhost:8001/default/routes/afefe836-b9be-49a8-927a-1324a8597a9c/plugi
ns/3f2e605e-9cb6-454a-83ec-d1b1929b1d30' -X PATCH --compressed -H 'Content-Type: appl
ication/json;charset=utf-8' -H 'Cache-Control: no-cache' -H 'Origin: http://localhost
:8002' -H 'DNT: 1' -H 'Connection: keep-alive' -H 'Referer: http://localhost:8002/def
ault/plugins/acl/3f2e605e-9cb6-454a-83ec-d1b1929b1d30/update' -H 'Pragma: no-cache' -
-data-raw '{"enabled":false,"name":"acl","route":{"id":"afefe836-b9be-49a8-927a-1324a
8597a9c"},"config":{"hide_groups_header":false,"whitelist":["user","dev","crud"]}}'
```

8.3.2. Ensuite, ajoutez le plugin de demande d'inscription

Portail IAM

API Rest

```
# Créer un plugin d'enregistrement d'appli
cationcurl -i -X POST \--url http://localh
ost:8001/services/crud/plugins \--data 'na
me=application-registration' \--data 'conf
ig.auth_header_name=authorization' \--data
 'config.auto_approve=true' \--data 'confi
g.display_name=auth' \--data 'config.enabl
e_client_credentials=true'
```

8.3.3. Associez le service et la documentation

Portail IAM

API Rest

```
curl 'http://localhost:8001/default/document_objects' --compressed -H 'Content-Type:
application/json;charset=utf-8' -H 'Cache-Control: no-cache' -H 'Origin: http://local
host:8002' -H 'DNT: 1' -H 'Connection: keep-alive' -H 'Referer: http://localhost:8002
/default/services/create-documentation' -H 'Pragma: no-cache' --data-raw '{"service":
{"id":"7bcef2e6-117c-487a-aab2-c7e57a0bf61a"},"path":"specs/iam-training.yml"}'
```

8.3.3.1. Essayez !

À partir du portail des développeurs, connecté en tant que dev@dev.com, créez une nouvelle demande.

Cela vous donnera clientid et clientsecret.

Il peuvent être utilisé dans le portail développeur.

Inscrivez cette demande dans le service crud :

Obtenez un jeton :

```
curl --insecure -X POST https://localhost:8443/persons/oauth2/token \
  --data "grant_type=client_credentials" \
  --data "client_id=2TXNvDqjeVMHydJbjv9t96lWTXOKAtU8" \
  --data "client_secret=V6Vma6AtIvl04UYssz6gAxPc92eCF4KR"
```

Utilisez le jeton :

```
curl --insecure -X GET https://localhost:8443/persons/all \
  --header "authorization: Bearer u5guWaYR3BjZ1KdwuBSC6C7udCYxj5vK"
```

9. Portail de gestion sécurisé

9.1. Créer un administrateur

Comme nous avons amorcé Kong sans mot de passe de démarrage.

Nous devons créer un administrateur avant d'appliquer RBAC.

Pour cela :

- Allez dans Équipes
- Invitez un administrateur
    - Définissez l'adresse e-mail
    - Définissez le nom d'utilisateur
    - Définissez le rôle sur super administrateur
    - Invitez
- Allez dans Administrateur invité
- Affichez
- Créez un lien

9.2. Activer l'authentification de base pour Kong Manager

Pour activer cette fonctionnalité, nous devons modifier le fichier docker-compose.

Ajoutez ceci au service iam, environnement

```
      KONG_ENFORCE_RBAC: 'on'
      KONG_ADMIN_GUI_AUTH: 'basic-auth'
      KONG_ADMIN_GUI_SESSION_CONF: '{"secret":"${IRIS_PASSWORD}","storage":"kong","co
okie_secure":false}'
```

Redémarrez le conteneur

```
docker-compose down && docker-compose up -d
```

Allez au lien administrateur invité :

```
http://localhost:8002/register?email=test.test%40gmail.com&username=admin&token=eyJhb
GciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2MTYzMzYzNzEsImlkIjoiY2JiZGE5Y2UtODQ3NS00M
mM2LTk4ZjItNDgwZTI4MjQ4NWNkIn0.sFeOc_5UPIr3MdlQrgyGvmvIjRFvSn3nQjo2ph8GrJA
```

9.3. Utiliser l'API Kong Admin avec RBAC

Comme RBAC est défini, nous ne pouvons plus utiliser l'api kong admin :

```
curl -s -X GET \
```

```
  --url http://localhost:8001/routes
```

Revoie cette erreur :

```
{"message":"Identifiants incorrects. Jeton ou identifiants utilisateurs requis"}
```

9.3.1. Créer un utilisateur administrateur avec un jeton

- Allez dans Équipes
- Utilisateurs RBAC
- Ajouter un nouvel utilisateur

```
curl -s -X GET \
  --url http://localhost:8001/routes \
  --header "Kong-Admin-Token: SYS"
```

10. Plug-ins

Kong est livré avec des plugins de haute qualité.

Mais et si nous avions besoin de plugins qui ne soient pas intégrés ? Si nous voulons des plugins communautaires ?

Dans ce chapitre, nous allons parler des plugins communautaires et comment les importer.

Ensuite, nous verrons comment compiler notre propre plugin.

10.1. Importer un plugin communautaire

Pour cette partie, nous allons utiliser le plugin jwt-crafter.

Ce plugin ajoute la possibilité de générer un jeton JWT au sein même de Kong, éliminant ainsi le besoin d'un service en amont effectuant la génération du jeton.

Voici le plugin :

```
https://github.com/grongierisc/kong-plugin-jwt-crafter
```

Pour installer ce plugin, comme nous utilisons la version docker, nous devons compiler une nouvelle image qui intègre le plugin.

10.1.1. Compliez une nouvelle image Kong/IAM avec le plugin communautaire

1. Créez un dossier nommé iam à la racine de ce git.
2. Créez un dockerfile dans ce nouveau dossier
3. Créez un dossier nommé plugins
    1. C'est ici que nous ajouterons tous nos plugins communautaires
4. Mettez à jour le fichier docker-compose pour activer le nouveau plug-in

Dans le dossier des plugins, git clonez notre plugin communautaire.

```
git clone https://github.com/grongierisc/kong-plugin-jwt-crafter
```

Le dockerfile doit ressembler à ceci :

```
FROM intersystems/iam:1.5.0.9-4

USER root
COPY ./plugins /custom/plugins

RUN cd /custom/plugins/kong-plugin-jwt-crafter && luarocks make

USER kong
```

Que voyons-nous dans ce dockerfile ?

Pour installer simplement un plugin communautaire, nous devons nous rendre dans son dossier racine (où se trouve le rockspec) et appeler luarocks make. C'est tout. Vous avez installé le plugin.

Pour la partie docker-compose :

1. Modifiez la balise iam iamge
    1. intersystems/iam:1.5.0.9-4 -> intersystems/iam-custom:1.5.0.9-4
2. Ajoutez un contexte de compilation

```
build:
  context: iam
  dockerfile: dockerfile
```

3. Activez le plugin dans les variables d'environnement

```
KONG_PLUGINS: 'bundled,jwt-crafter'
```

Maintenant, compilons notre nouvelle image iam :

```
docker-compose build iam
```

10.1.2. Essayez !

```
docker-compose up -d
```

Si vous allez dans plugin -> nouveau, en bas de la liste vous devriez voir le plugin jwt-crafter.

10.1.2.1. Utilisez-le !

1. Créez un nouveau service :

Portail IAM	API Rest

```
# Créer un servicecurl -i -X POST \--url h
ttp://localhost:8001/services/ \--data 'na
me=crud-persons' \--data 'url=http://iris:
52773/crud/persons/'
```

2. Créez une route

Portail IAM

API Rest

```
# Créer une routecurl -i -X POST \--url ht
tp://localhost:8001/services/crud-persons/
routes \--data 'name=crud-route-jwt' \--da
ta 'paths=/crud/persons/*' \--data 'strip_
path=true'
```

3. Réutilisez notre authentification automatique

Portail IAM

API Rest

```
# Créer un plugincurl -i -X POST \--url ht
tp://localhost:8001/services/crud-persons/
plugins \--data 'name=request-transformer'
 \--data 'config.add.headers=Authorization
:Basic U3VwZXJVc2VyOlNZUw==' \--data 'conf
ig.replace.headers=Authorization:Basic U3V
wZXJVc2VyOlNZUw=='
```

Maintenant tout est prêt. La véritable utilisation de jwt-crafter.

```
# Ajouter acl à la route
curl -i -X POST http://localhost:8001/routes/crud-route-jwt/plugins \
    --data "name=acl"  \
    --data "config.whitelist=test" \
    --data "config.hide_groups_header=false"

# Créer le service
curl -i -X POST \
  --url http://localhost:8001/services/ \
  --data 'name=jwt-login' \
  --data 'url=http://neverinvoked/'

# Créer une route
curl -i -X POST \
  --url http://localhost:8001/services/jwt-login/routes \
  --data 'name=jwt-login-route' \
  --data 'paths=/jwt/log-in'

# Activer l'authentification de base pour le service
curl -i -X POST http://localhost:8001/routes/jwt-login-route/plugins \
    --data "name=basic-auth"  \
    --data "config.hide_credentials=false"

# Activer l'authentification de base pour le service
curl -i -X POST http://localhost:8001/routes/jwt-login-route/plugins \
    --data "name=jwt-crafter"  \
```

```
    --data "config.expires_in=86400"

# Ajouter un consommateur
curl -i -X POST \
   --url http://localhost:8001/consumers/ \
   --data "username=test"

# Ajouter un groupe consommateur
curl -i -X POST \
   --url http://localhost:8001/consumers/test/acls \
   --data "group=test"

# Ajouter des identifiants consommateur
curl -i -X POST http://localhost:8001/consumers/test/basic-auth \
    --data "username=test" \
    --data "password=test"
curl -i -X POST http://localhost:8001/consumers/test/jwt \
    --data "key=test" \
    --data "algorithm=HS256"

# Plugins JWT
curl -i -X POST http://localhost:8001/routes/crud-route-jwt/plugins \
    --data "name=jwt"
```

Essayez !

```
# test:test est en base64
curl -H 'Authorization: basic dGVzdDp0ZXN0' localhost:8000/jwt/log-in


curl --location --request GET 'http://localhost:8000/crud/persons/all' \
--header 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW0iOiJ0ZXN0
Iiwic3ViIjoiODJiNjcwZDgtNmY2OC00NDE5LWJiMmMtMmYxZjMxNTViN2E2Iiwicm9sIjpbInRlc3QiXSwiZ
XhwIjoxNjE2MjUyMTIwLCJpc3MiOiJ0ZXN0In0.g2jFqe0hDPumy8_gG7J3nYsuZ8KUz9SgZOecdBDhfns'
```

10.2. Créer un nouveau plugin

Ce n'est pas l'endroit pour apprendre le lua.

Mais je vous donnerai quelques astuces comme la façon de redémarrer rapidement IAM pour tester notre nouveau développement.

10.2.1. Structure de fichiers

```
kong-plugin-helloworld
??? kong
?   ??? plugins
?       ??? helloworld
?           ??? handler.lua
?           ??? schema.lua
??? kong-plugin-helloworld-0.1.0-1.rockspec
```

Par convention, les plugins kong doivent être préfixés par kong-plugin.

Dans notre exemple, le nom du plugin est helloworld.

Trois fichiers sont obligatoires :

- handler.lua : le cœur de votre plugin. Il s'agit d'une interface à mettre en œuvre, dans laquelle chaque fonction sera exécutée au moment souhaité dans le cycle de vie d'une demande/connexion.
- schema.lua : votre plugin doit probablement conserver une configuration saisie par l'utilisateur. Ce module contient le schéma de cette configuration et définit des règles sur celui-ci, afin que l'utilisateur ne puisse saisir que des valeurs de configuration valides.
- *.rockspec : Rockspec : un fichier de spécification de paquetage Un script Lua déclaratif, avec des règles sur la façon de compiler et d'empaqueter rocks *.rockspec - un fichier Lua contenant quelques tableaux.

10.2.1.1. handler.lua

L'interface du plugin vous permet de remplacer l'une des méthodes suivantes dans votre fichier handler.lua pour mettre en œuvre une logique personnalisée à divers points d'entrée du cycle de vie de Kong :

Nom de la fonction	Phase	Description
:initworker()	initworker	Exécuté au démarrage de chaque processus worker Nginx.
:certificate()	sslcertificate	Exécuté pendant la phase de service du certificat SSL de la poignée de main SSL.
:rewrite()	rewrite	Exécuté pour chaque demande à sa réception d'un client en tant que gestionnaire de la phase de réécriture. NOTE : dans cette phase, ni le Service ni le Consommateur n'ont été identifiés, donc ce gestionnaire ne sera exécuté que si le plugin a été configuré comme un plugin global !
:access()	access	Exécuté pour chaque demande d'un client et avant qu'elle ne soit transmise par proxy au service en amont.
:response()	access	Remplace à la fois headerfilter() et bodyfilter(). Exécuté après la réception de la réponse complète du service en amont, mais avant d'en envoyer une partie au client.
:headerfilter()	headerfilter	Exécuté lorsque tous les octets d'en-tête de réponse ont été reçus du service en amont.
:bodyfilter()	bodyfilter	Exécuté pour chaque fragment du corps de la réponse reçu du service en amont. Comme la réponse est renvoyée au client en continu, elle peut dépasser la taille du tampon et être envoyée par morceaux. cette méthode peut donc être appelée plusieurs fois si la réponse est importante. Consultez la documentation de lua-nginx-module pour plus de détails.
:log()	journal	Exécuté lorsque le dernier octet de réponse a été envoyé au client.

10.2.1.1.1. Exemple

```
local BasePlugin = require "kong.plugins.base_plugin"

local HelloWorldHandler = BasePlugin:extend()

function HelloWorldHandler:new()
  HelloWorldHandler.super.new(self, "helloworld")
end

function HelloWorldHandler:access(conf)
  HelloWorldHandler.super.access(self)
  if conf.say_hello then
    ngx.log(ngx.ERR, "============ Hello World! ============")
    ngx.header["Hello-World"] = "Hello World!!!"
  else
    ngx.log(ngx.ERR, "============ Bye World! ============")
    ngx.header["Hello-World"] = "Bye World!!!"
  end
end

return HelloWorldHandler
```

10.2.1.2. schema.lua

Simplement le fichier de configuration vu dans le portail.

```
return {
    no_consumer = true,
    fields = {
      say_hello = { type = "boolean", default = true },
      say_hello_body = { type = "boolean", default = true }
    }
  }
```

10.2.1.3. *.rockspec

```
package = "kong-plugin-helloworld" -- hint : renommer, doit correspondre à l'info dan
s le nom de fichier de ce rockspec !
                                   -- comme convention ; tenez-
vous en au préfixe : « kong-plugin- »
version = "0.1.0-1" -- indice : n'oubliez pas, il doit correspondre à l'info dans le
nom de fichier de ce rockspec !
-- La version « 0.1.0 » est la version du code source, le « 1 » suivant est la versio
n de ce rockspec.
-- à chaque fois que la version de la source change, le rockspec doit être réinitiali
sé à 1. La version rockspec est seulement
-- mise à jour (incrémentée) lorsque ce fichier change, mais la source reste la même.

-- TODO : C'est le nom à définir dans le paramètre « plugins » de la configuration Ko
ng.
-- Ici, nous l'extrayons du nom du paquet.
local pluginName = package:match("^kong%-plugin%-(.+)$")  -- "myPlugin"

supported_platforms = {"linux", "macosx"}
source = {
  url = "https://github.com/grongierisc/iam-training",
  branch = "master",
```

```
--  tag = "0.1.0"
-- hint: "tag" could be used to match tag in the repository
}

description = {
  summary = "Ceci est une démo pour le plugin Kong",
  homepage = "https://github.com/grongierisc/iam-training",
  license = "Apache 2.0"
}

dependencies = {
   "lua >= 5.1"
   -- other dependencies should appear here
}

build = {
  type = "builtin",
  modules = {
    ["kong.plugins."..pluginName..".handler"] = "kong/plugins/"..pluginName.."/handle
r.lua",
    ["kong.plugins."..pluginName..".schema"] = "kong/plugins/"..pluginName.."/schema.
lua",
  }
}
```

10.2.2. Compilez-le

Nous ferons la même chose qu'ici : 11.1.1. Compilez une nouvelle image Kong/IAM avec le plugin communautaire

Mais adapté à notre plugin :

Dockerfile :

```
FROM intersystems/iam:1.5.0.9-4

USER root
COPY ./plugins /custom/plugins

RUN cd /custom/plugins/kong-plugin-jwt-crafter && luarocks make
RUN cd /custom/plugins/kong-plugin-helloworld && luarocks make

#USER kong #Rester root, nous verrons pourquoi plus tard
```

Activez le plugin dans les variables d'environnement

```
KONG_PLUGINS: 'bundled,jwt-crafter,helloworld'
```

Maintenant, compilons notre nouvelle image iam :

```
docker-compose build iam
```

Puis docker-compose up et testez-le.

10.2.3. Astuces

Pour exécuter le conteneur IAM en « mode débogage », pour l'arrêter/le redémarrer facilement, modifier le dockerfile pour ajouter/supprimer le plugin et ainsi de suite.

Vous pouvez arrêter le service iam :

```
docker-compose stop iam
```

Et lancez-le en mode exécution avec un shell :

```
docker-compose run -p 8000:8000 -p 8001:8001 -p 8002:8002 iam sh
```

Dans le conteneur :

```
./docker-entrypoint.sh kong
```

Bonne programmation :)

11. CI / CD

Nous sommes proches de la fin de cet article.

Pour finir, parlons de DevOps/CI/CD. L'objectif de ce chapitre est de vous donner quelques idées sur la manière d'implémenter/scripter ci/cd pour IAM/Kong.

Comme Kong est d'abord une API, l'idée est de scripter tous les appels rest et de jouer ensuite sur chaque environnement.

La façon la plus simple de scripter les appels Rest est avec postman et son meilleur ami newman (version en ligne de commande de postman).

11.1. Créer la collection Postman

Une chose pratique avec postman est sa capacité à exécuter un script avant et après un appel rest.

Nous utiliserons cette fonctionnalité dans la plupart des cas.

11.1.1. IAM est-il démarré ?

Notre premier script va vérifier si IAM est en place et fonctionne.

```
var iam_url = pm.environment.get("iam_url");
var iam_config_port = pm.environment.get("iam_config_port");
var url = "http://" + iam_url + ":" + iam_config_port + "/";
SenReq(20);
async function SenReq(maxRequest) {
```

```
    var next_request = "end request";
    const result = await SendRequest(maxRequest);
    console.log("result:",result);
    if(result == -1)
    {
        console.error("IAM starting .... failed !!!!");
    }
}

function SendRequest(maxRequest) {
    return new Promise(resolve => {
        pm.sendRequest(url,
            function (err) {
                if (err) {
                    if (maxRequest > 1) {
                        setTimeout(function () {}, 5000);
                        console.warn("IAM not started...retry..next retry in 5 sec");
                        SendRequest(maxRequest - 1);
                    } else {
                        console.error("IAM starting .... failed");
                        resolve(-1);
                    }
                } else {
                    console.log("IAM starting .... ok");
                    resolve(1);
                }

            }
        );
    });
}
```

11.1.2. Supprimer les anciennes données

```
var iam_url=pm.environment.get("iam_url");
var iam_config_port=pm.environment.get("iam_config_port");

pm.sendRequest("http://"+iam_url+":"+iam_config_port+"/plugins", function (err, res)
{
    if (err) {
        console.log("ERROR : ",err);
    }
    else {

        var body_json=res.json();
        if(body_json.data)
        {
            for( i=0; i < body_json.data.length; i++)
            {
                // Example with a full fledged SDK Request
                route_id = body_json.data[i].id;
                const delete_route = {
                  url: "http://"+iam_url+":"+iam_config_port+"/plugins/" + route_id,
                  method: 'DELETE',
                };
                pm.sendRequest(delete_route, function(err, res){
                    console.log(err ? err : res);
                });
```

```
            }
        }
    }
});
```

Faites de même pour les routes, les services et les consommateurs.

Cet ordre est important car vous ne pouvez pas supprimer les services avec des routes.

11.1.3. Créer un service/route

Les routes dépendent des services. Pour ce type de cas, nous pouvons utiliser la fonction Test de postman pour récupérer les données :

Écran

Script

```
var id = pm.response.json().id;var name =
pm.response.json().name;pm.globals.set("se
rvice_crud_id", id);pm.globals.set("servic
e_crud_name", name);
```

Ici, nous sauvegardons de la réponse l'ID et le nom des nouveaux services.

Ensuite, nous pouvons l'utiliser dans la prochaine création de route :

Écran

Script

```
service_crud_name = pm.globals.get("servic
e_crud_name");
```

Ici, nous récupérons la variable globale « servicecrudname ».

Ensuite, utilisez-le dans l'appel réel.

Écran

Script

```
{    "paths": [        "/persons/*"    ],
   "protocols": [        "http"    ],    "
name": "crud-persons",    "strip_path": fa
lse,    "service": {        "name": "{{ser
vice_crud_name}}"        }}
```

11.1.3.1. Astuces

- le contenu peut être soit au format json, soit au format form-data
    - form-data :

- json :

Une façon simple d'obtenir le format json, allez sur le portail du gestionnaire, affichez, copiez le json :

11.2. Exécutez-le avec newman

```
docker run  --rm -v "`pwd`/ci/":"/etc/newman" \
 --network="iam-training_default" \
 -t postman/newman run "DevOps_IAM.postman_collection.json" \
 --environment="DevOps_IAM.postman_environment.json"
```

#InterSystems API Manager (IAM) #InterSystems IRIS
Voir l'application sur InterSystems Open Exchange

URL de la source:https://fr.community.intersystems.com/post/iam-intersystems-api-manager-le-chemin-vers-la-r%C3%A9ussite