Article
Lorenzo Scalese · Juil 15, 2022 12m de lecture

ECP Avec Docker

Salut la communauté,

Ceci est le troisième article de la série traitant sur l ' initialisation d ' instances IRIS avec Docker. Cette fois, on s ' intéresse à *E*nterprise *C*ache *P*rotocol (ECP).

De manière très simplifiée, ECP permet de configurer certaines instances IRIS avec le rôle de serveur d ' applications et d ' autres avec le rôle de serveur de données. Vous trouverez toutes les informations techniques détaillées dans la documentation officielle.

Le but de cet article est de décrire comment nous pouvons scripter l'initialisation d ' un serveur de données, d ' un ou plusieurs serveurs d ' applications et d ' établir une connexion cryptée entre ces nœuds avec Docker. Pour cela, nous utilisons certains outils déjà rencontrés dans les articles précédents traitant de la mise en miroir la webgateway tel que OpenSSL, envsubst et Config-API.

Prérequis

ECP n ' est pas disponible dans la version IRIS Community. Il est donc nécessaire d ' avoir un accès au World Response Center afin de pouvoir télécharger une licence " container " ainsi que pour se connecter au registry containers.intersystems.com.

Préparation du système

Le système doit partager certains fichiers locaux avec les containers. Il est donc nécessaire de créer certains utilisateurs et groupes afin d ' éviter l ' erreur " access denied " .

```
sudo useradd --uid 51773 --user-group irisowner
sudo useradd --uid 52773 --user-group irisuser
sudo groupmod --gid 51773 irisowner
sudo groupmod --gid 52773 irisuser
```bash

Si vous n'avez pas encore votre licence "iris.key", téléchargez-
là sur le WRC ensuite mettez-là dans votre répertoire home.

## Récupérer le repository d'exemple

Tous les fichiers dont vous avez besoin sont disponible sur un [repository public](ht
tps://github.com/lscalese/ecp-with-docker), commencez par le cloner:

```bash
git clone https://github.com/lscalese/ecp-with-docker.git
cd ecp-with-docker
```

Certificats SSL

Afin de crypter les communications entre les serveurs d ' applications et le serveur de données, nous avons besoin de certificats SSL. Le script " gen-certificates.sh " prêt à l ' emploi est disponible, toutefois, n ' hésitez pas à le modifier pour que les paramètres des certificats soient cohérents avec votre localisation, compagnie, etc…

Exécutez:

```
sh ./gen-certificates.sh
```

Les certificats générés sont dans le répertoire " ./certificates " .

Fichier	Container	Description
./certificates/CAServer.cer	Serveur d ' applications et serveur de données	Authority server certificate
./certificates/appserver.cer	Serveur d ' applications	Certificat pour l ' instance IRIS
./certificates/appserver.key	Serveur d ' applications	Clé privée associée
./certificates/dataserver.cer	Serveur de données	Certificat pour l ' instance IRIS
./certificates/dataserver.key	Serveur de données	Clé privée associée

Construire l ' image

Avant toute chose, loggez-vous au docker registry d ' Intersystems. Lors de la construction, l ' image de base sera téléchargée sur le registry:

docker login -u="YourWRCLogin" -p="YourICRToken" containers.intersystems.com

Si vous ne connaissez pas votre token, vous pouvez le récupérer sur cette page https://containers.intersystems.com en vous connectant avec votre compte WRC.

Lors de cette construction, nous allons ajouter à l ' image de base de IRIS quelques logiciels utilitaires:

- gettext-base : qui permettra de substituer les variables d ' environnements dans nos fichiers de configuration avec la commande " envsubst " .
- iputils-arping : qui est requis dans le cas ou l ' on souhaite mettre le serveur de données en miroir.
- ZPM: gestionnaire de paquet ObjectScript.

Dockerfile

```
ARG IMAGE=containers.intersystems.com/intersystems/iris:2022.2.0.281.0

# Don't need to download the image from WRC. It will be pulled from ICR at build time
.

FROM $IMAGE

USER root

# Install iputils-
arping to have an arping command.  It's required to configure Virtual IP.
# Download the latest ZPM version (ZPM is included only with community edition).
RUN apt-get update && apt-get install iputils-arping gettext-base && \
   rm -rf /var/lib/apt/lists/*

USER ${ISC_PACKAGE_MGRUSER}

WORKDIR /home/irisowner/demo
```

```
RUN --mount=type=bind,src=.,dst=. \
   iris start IRIS && \
       iris session IRIS < iris.script && \
   iris stop IRIS quietly
```

Il n ' y a rien de particulier dans ce Dockerfile, excepté la dernière ligne. Celle-ci configure l ' instance IRIS serveur de données pour accepter jusqu ' à 3 serveurs d ' applications. Attention, cette configuration nécessite un redémarrage de IRIS. Nous affectons la valeur de ce paramètre pendant la construction afin d ' éviter de devoir scripter un redémarrage plus tard.

Démarrez la construction:

```
docker-compose build –no-cache
```

Les fichiers de configurations

Pour la configuration des instances IRIS (serveurs d ' applications et serveur de données) nous utilisons des fichiers format JSON config-api. Vous remarquerez que ceux-ci contiennent des variables d ' environnements " ${variablename} " . Leurs valeurs sont définies dans les sections " environment " du fichier docker-compose.yml que nous verrons plus tard dans ce document. Ces variables seront substituées juste avant le chargement des fichiers à l ' aide de l ' utilitaire " envsubst " .

Serveur de données

Pour le serveur de données, nous allons:

- Activer le service ECP et définir la liste des clients autorisés (serveurs d ' applications).
- Créer la configuration " SSL %ECPServer " nécessaire pour le cryptage des communications.
- Créer une base de données " myappdata " . Celle-ci sera utilisée comme base de données distante depuis les applications serveurs.

data-server.json

```
{
   "Security.Services" : {
       "%Service_ECP" : {
           "Enabled" : true,
           "ClientSystems":"${CLIENT_SYSTEMS}",
           "AutheEnabled":"1024"
       }
   },
   "Security.SSLConfigs": {
       "%ECPServer": {
           "CAFile": "${CA_ROOT}",
           "CertificateFile": "${CA_SERVER}",
           "Name": "%ECPServer",
           "PrivateKeyFile": "${CA_PRIVATE_KEY}",
           "Type": "1",
           "VerifyPeer": 3
       }
   },
   "Security.System": {
       "SSLECPServer":1
```

```
    },
    "SYS.Databases":{
        "/usr/irissys/mgr/myappdata/" : {}
    },
    "Databases":{
        "myappdata" : {
            "Directory" : "/usr/irissys/mgr/myappdata/"
        }
    }
}
```

Ce fichier de configuration est chargé au démarrage du container serveur de données par le script " initdatasrv.sh " . Tous les serveurs d ' applications qui se connectent au serveur de données doivent être approuvés, ce script validera automatiquement toutes les connexions dans un délai de 100 secondes afin de limiter les actions manuelles dans le portail d ' administration. Bien sûr, cela peut être amélioré pour renforcer la sécurité.

Serveur d ' applications

Pour les serveurs d ' applications, nous allons:

- Activer le service ECP.
- Créer la configuration SSL " %ECPClient " nécessaire pour le cryptage des communications.
- Configurer les informations de connexion au serveur de données.
- Créer la configuration de la base de données distante " myapp "
- Créer un mapping de globale " demo.* " dans le namespace " USER " a destination la base de données " myappdata " . Ceci nous permettra de tester le fonctionnement de ECP plus tard.

app-server.json

```
{
    "Security.Services" : {
        "%Service_ECP" : {
            "Enabled" : true
        }
    },
    "Security.SSLConfigs": {
        "%ECPClient": {
            "CAFile": "${CA_ROOT}",
            "CertificateFile": "${CA_CLIENT}",
            "Name": "%ECPClient",
            "PrivateKeyFile": "${CA_PRIVATE_KEY}",
            "Type": "0"
        }
    },
    "ECPServers" : {
        "${DATASERVER_NAME}" : {
            "Name" : "${DATASERVER_NAME}",
            "Address" : "${DATASERVER_IP}",
            "Port" : "${DATASERVER_PORT}",
            "SSLConfig" : "1"
        }
    },
    "Databases": {
        "myappdata" : {
            "Directory" : "/usr/irissys/mgr/myappdata/",
```

```
            "Name" : "${REMOTE_DB_NAME}",
            "Server" : "${DATASERVER_NAME}"
        }
    },
    "MapGlobals":{
        "USER": [{
            "Name" : "demo.*",
            "Database" : "myappdata"
        }]
    }
}
```

Le fichier de configuration est chargé au démarrage d ' un container serveur d ' applications par le script " initappsrv.sh " .

Démarrer les containers

Nous pouvons maintenant démarrer les containers:

- 2 serveurs d ' applications.
- 1 serveur de données.

Pour cela, exécutez:

```
docker-compose up –scale ecp-demo-app-server=2
```

Voir le fichier docker-compose pour les détails:

Docker-compose

```
# Les variables sont définies dans  le fichier .env file
# Pour afficher le fichier de configuration résolu, exécutez :
# docker-compose config

version: '3.7'

services:
  ecp-demo-data-server:
   build: .
   image: ecp-demo
   container_name: ecp-demo-data-server
   hostname: data-server
   networks:
     app_net:
   environment:
     # Liste des serveurs d'applications autorisés.
     - CLIENT_SYSTEMS=ecp-with-docker_ecp-demo-app-server_1;ecp-with-docker_ecp-demo-
app-server_2;ecp-with-docker_ecp-demo-app-server_3
     # Chemin vers le certificat du serveur d'autorité.
     - CA_ROOT=/certificates/CA_Server.cer
     # Chemin vers le certificat du serveur de données
     - CA_SERVER=/certificates/data_server.cer
     # Chemin vers la clé privée du certificat du serveur de données
```

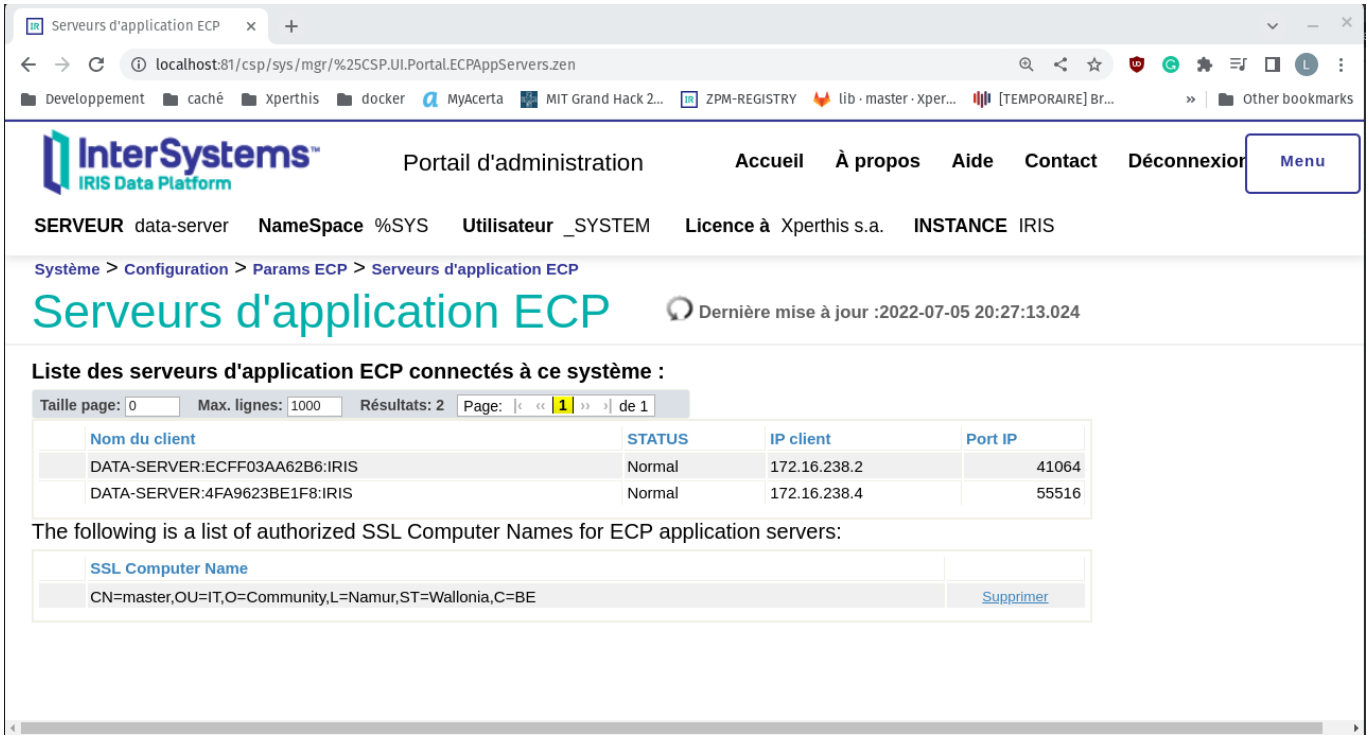
```
      - CA_PRIVATE_KEY=/certificates/data_server.key
      # Chemin vers le fichier de Configuration Config-
API utilisé pour configurer cette instance
      - IRIS_CONFIGAPI_FILE=/home/irisowner/demo/data-server.json
    ports:
      - "81:52773"
    volumes:
      # Post start script pour initialiser l'instance du serveur de données.
      - ./init_datasrv.sh:/home/irisowner/demo/init_datasrv.sh
      # Montage des certificats
      - ./certificates/app_server.cer:/certificates/data_server.cer
      - ./certificates/app_server.key:/certificates/data_server.key
      - ./certificates/CA_Server.cer:/certificates/CA_Server.cer
      # Montage du fichier de configuration Config-API
      - ./config-files/data-server.json:/home/irisowner/demo/data-server.json
      # Montage du fichier de licence IRIS
      - ./iris.key:/dur/iris.key
    command: -a /home/irisowner/demo/init_appsrv.sh


  ecp-demo-app-server:
    image: ecp-demo
    networks:
      app_net:
    environment:
      # Nom d'hôte ou adresse ip du serveur de données.
      - DATASERVER_IP=data-server
      - DATASERVER_NAME=data-server
      - DATASERVER_PORT=1972
      # Chemin vers le certificat du serveur d'autorité.
      - CA_ROOT=/certificates/CA_Server.cer
      # Chemin vers le certificat du serveur d'applications.
      - CA_CLIENT=/certificates/app_server.cer
      # Chemin vers la clé privée du certificat du serveur d'applications.
      - CA_PRIVATE_KEY=/certificates/app_server.key
      - IRIS_CONFIGAPI_FILE=/home/irisowner/demo/app-server.json
    ports:
      - 52773
    volumes:
      # Post start script pour initialiser l'instance du serveur de données.
      - ./init_appsrv.sh:/home/irisowner/demo/init_appsrv.sh
      # Montage des certificats.
      - ./certificates/CA_Server.cer:/certificates/CA_Server.cer
      - ./certificates/app_server.cer:/certificates/app_server.cer
      - ./certificates/app_server.key:/certificates/app_server.key
      # Montage du fichier de configuration Config-API.
      - ./config-files/app-server.json:/home/irisowner/demo/app-server.json
      # Montage du fichier de licence IRIS.
      - ./iris.key:/dur/iris.key
    command: -a /home/irisowner/demo/init_appsrv.sh

  networks:
  app_net:
    ipam:
      driver: default
      config:
        # APP_NET_SUBNET variable is defined in .env file
        - subnet: "${APP_NET_SUBNET}"
```

Testons-le!

Accès au portail d ' administration du serveur de données

Les containers sont démarrés, vérifions l ' état à partir du serveur de données.
Le port 52773 est mappé sur le port local 81, vous pouvez donc y accéder avec cette adresse http://localhost:81/csp/sys/utilhome.csp.

Connectez-vous avec le login\password par défaut ensuite allez dans Système -> Configuration -> Params ECP. Cliquez sur " Serveurs d ' application ECP " . Si tout fonctionne bien, vous devriez voir 2 serveurs d ' applications avec le statut " Normal " . La structure du nom du client est "data server name":"application server hostname":"IRIS instance name". Dans notre cas, nous n ' avons pas paramétré les hostnames des serveurs d ' applications, nous avons donc des hostnames auto-générés.



Accès au portail d ' administration des serveurs d ' applications

Pour vous connecter au portail d ' administration des serveurs d ' applications, vous devez d ' abord récupérer le numéro de port. Etant donné que nous avons utilisé l ' option " --scale " , nous n ' avons pas pu fixer les ports dans le fichier docker-compose. Il faut donc les retrouver à l ' aide de la commande " docker ps " :

```
docker ps -a
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS
          PORTS
               NAMES
a1844f38939f   ecp-demo   "/tini -- /iris-main…"   25 minutes ago   Up 25 minutes (un
healthy)   1972/tcp, 2188/tcp, 53773/tcp, 54773/tcp, 0.0.0.0:81->52773/tcp, :::81->52
773/tcp          ecp-demo-data-server
4fa9623be1f8   ecp-demo   "/tini -- /iris-main…"   25 minutes ago   Up 25 minutes (un
healthy)   1972/tcp, 2188/tcp, 53773/tcp, 54773/tcp, 0.0.0.0:49170->52773/tcp, :::491
70->52773/tcp    ecp-with-docker_ecp-demo-app-server_1
ecff03aa62b6   ecp-demo   "/tini -- /iris-main…"   25 minutes ago   Up 25 minutes (un
healthy)   1972/tcp, 2188/tcp, 53773/tcp, 54773/tcp, 0.0.0.0:49169->52773/tcp, :::491
69->52773/tcp    ecp-with-docker_ecp-demo-app-server_2
```

Dans cet exemple, les ports sont:

- 49170 pour le premier serveur d ' applications http://localhost:49170/csp/sys/utilhome.csp
- 49169 pour le second serveur d ' applications http://localhost:49169/csp/sys/utilhome.csp

Test de lecture \écriture sur la base de données distante

Effectuons maintenant quelques tests de lecture \écriture en terminal.

Ouvrons un terminal IRIS sur le premier serveur d ' applications:

```
docker exec -it ecp-with-docker_ecp-demo-app-server_1 iris session iris
Set ^demo.ecp=$zdt($h,3,1) _ “ écriture à partir du premier serveur d’applications”
```

Ouvrons maintenant un terminal sur le second serveur d ' applications:

```
docker exec -it ecp-with-docker_ecp-demo-app-server_2 iris session iris
Set ^demo.ecp(2)=$zdt($h,3,1) _ " écriture à partir du second serveur d’applications.
"
zw ^demo.ecp
```

Vous devriez visualiser les écritures effectuées depuis les deux serveurs:

```
^demo.ecp(1)="2022-07-05 23:05:10 écriture à partir du premier serveur d’applications
."
^demo.ecp(2)="2022-07-05 23:07:44 écriture à partir du second serveur d’applications.
"
```

Pour terminer, ouvrons un terminal IRIS sur le serveur de données et effectuons une lecture de la globale demo.ecp :

```
docker exec -it ecp-demo-data-server iris session iris
zw ^["^^/usr/irissys/mgr/myappdata/"]demo.ecp

^["^^/usr/irissys/mgr/myappdata/"]demo.ecp(1)="2022-07-05 23:05:10 écriture à partir
du premier serveur d’applications."
^["^^/usr/irissys/mgr/myappdata/"]demo.ecp(2)="2022-07-05 23:07:44 écriture à partir
du second serveur d’applications."
```

Voilà, c ' est tout pour aujourd ' hui, j ' espère que cet article vous a plu. N ' hésitez pas à laisser vos commentaires.

#DevOps #ECP #InterSystems IRIS

URL de la source:https://fr.community.intersystems.com/post/ecp-avec-docker