Article
Lorenzo Scalese · Juil 11, 2022 13m de lecture

Open Exchange

Comment configurer un miroir de manière programmatique

Historique

Version	Date	Changements
V1	2022-02-08	Version initiale
V1.1	2022-04-06	Génération de certificats avec le fichier sh au lieu de pki-script Utilisation de variables d'environnement dans des fichiers de configuration

Salut, communauté,

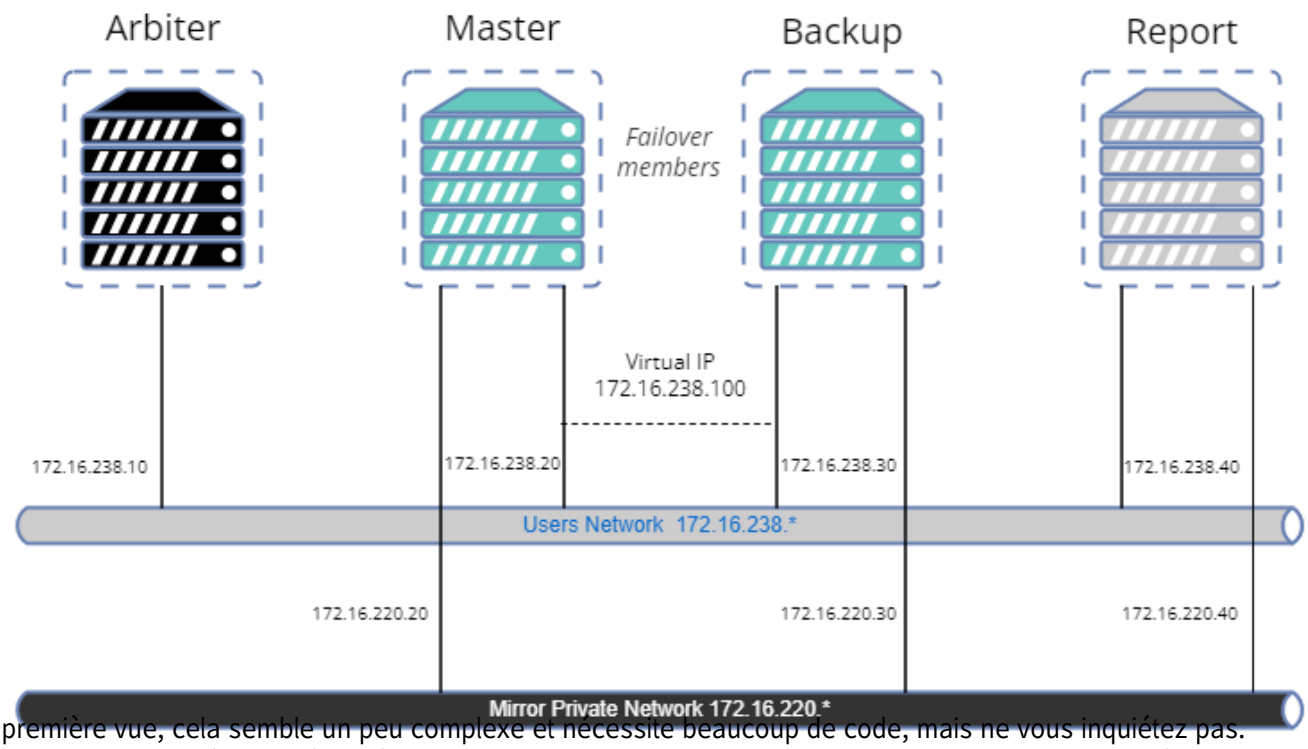
Avez-vous déjà mis en place un environnement miroir ? Dispose-t-il d'un réseau privé, d'une adresse IP virtuelle et d'une configuration SSL ?
Après avoir fait cela plusieurs fois, je me suis rendu compte que c'est long, et qu'il y a beaucoup d'actions manuelles nécessaires pour générer des certificats et configurer chaque instance IRIS.
C'est une vraie casse-tête pour les personnes qui ont souvent à le faire.

Par exemple, une équipe d'assurance qualité peut avoir besoin de créer un nouvel environnement pour chaque nouvelle version d'application à tester. L'équipe de support peut avoir besoin de créer un environnement pour reproduire un problème complex.

Il faut absolument des outils pour les créer rapidement.

Dans cet article, nous allons créer un exemple pour configurer un miroir avec :

- Arbitre.
- Membre primaire.
- Membre de secours en cas de panne.
- Membre asynchrone de rapport en lecture-écriture.
- Configuration SSL pour les transferts de journaux entre les nœuds.
- Réseau privé pour le miroir.
- Adresse IP virtuelle.
- Une base de données en miroir.



À première vue, cela semble un peu complexe et nécessite beaucoup de code, mais ne vous inquiétez pas. Il existe des bibliothèques hébergées sur OpenExchange pour effectuer facilement la plupart des opérations.

L'objectif de cet article est de fournir un exemple de manière à l'adapter à vos besoins, mais il ne s'agit pas d'un guide des meilleures pratiques en matière de sécurité.
Donc, créons notre sample.

Outils et bibliothèques

- config-api: Cette bibliothèque sera utilisée pour configurer IRIS. Elle supporte la configuration du mirroring depuis la version 1.1.0. Nous ne donnerons pas une description détaillée de l'utilisation de cette bibliothèque. Un ensemble d'articles existe déjà. ici. En bref, config-api sera utilisé pour créer des fichiers de configuration (format JSON) et les charger facilement.

- ZPM.

- Docker.
- OpenSSL.

Page Github

Vous pouvez trouver tous les fichiers de ressources nécessaires sur iris-mirroring-samples repository.

Préparation de votre système

Clonez le référentiel existant :

```
clone git https://github.com/lscalese/iris-mirroring-samples
cd iris-mirroring-samples
```

Si vous préférez créer un échantillon à partir de zéro, au lieu de cloner le référentiel, créez simplement un nouveau répertoire avec des sous-répertoires : backup, et config-files. Télécharger irissession.sh :

```
mkdir -p iris-mirroring-samples/backup iris-mirroring-samples/config-files
cd  iris-mirroring-samples
wget -O session.sh https://raw.githubusercontent.com/lscalese/iris-mirroring-
samples/master/session.sh
```

Pour éviter le problème "permission denied" plus tard, nous devons créer le groupe irisowner, l'utilisateur irisowner, et changer le groupe du répertoire backup en irisowner

```
sudo useradd --uid 51773 --user-group irisowner
sudo groupmod --gid 51773 irisowner
sudo chgrp irisowner ./backup
```

Ce répertoire sera utilisé comme volume pour partager une sauvegarde de la base de données après avoir configuré le premier membre miroir avec les autres nœuds.

Obtention d'une licence IRIS

La mise en miroir n'est pas disponible avec l'édition communautaire d'IRIS.
Si vous ne disposez pas encore d'une licence conteneur IRIS valide, connectez-vous au Centre de réponse mondial Worldwide Response Center (WRC) avec vos informations d'identification.
Cliquez sur "Actions" --> "Online distribtion" ("Actions" --> "Distribution en ligne"), puis sur le bouton "Evaluations" et sélectionnez "Evaluation License" ("Licence d'évaluation") ; remplissez le formulaire.
Copiez votre fichier de licence iris.key dans ce répertoire.

Connexion au Registre des conteneurs d'Intersystems

Pour des raisons pratiques, nous utilisons le Registre des conteneurs d'Intersystems (Intersystems Containers Registry) (ICR) pour extraire les images docker. Si vous ne connaissez pas votre login/mot de passe docker, connectez-vous simplement à SSO.UI.User.ApplicationTokens.cls avec vos informations d'identification WRC, et vous pourrez récupérer votre Token ICR.

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

Création de la base de données myappdata et d'un mapping global

Nous ne créons pas vraiment la base de données myappdata maintenant mais préparons une configuration pour la créer au moment de la construction l'image.
Pour cela, nous créons juste un simple fichier au format JSON ;
La bibliothèque config-api sera utilisée pour le charger dans les instances IRIS.

Creation du ficher config-files/simple-config.json

```
{
   "Defaults":{
       "DBDATADIR" : "${MGRDIR}myappdata/",
       "DBDATANAME" : "MYAPPDATA"

   },
   "SYS.Databases":{
       "${DBDATADIR}" : {}
```

```
   },
   "Databases":{
       "${DBDATANAME}" : {
           "Directory" : "${DBDATADIR}"
       }
   },
   "MapGlobals":{
       "USER": [{
           "Name" : "demo.*",
           "Database" : "${DBDATANAME}"
       }]
   },
   "Security.Services" : {
       "%Service_Mirror" : {                    /* Activer le service miroir sur ce
tte instance */
           "Enabled" : true
       }
   }
}
```

Ce fichier de configuration vous permet de créer une nouvelle base de données avec les paramètres par défaut et de faire un global mapping demo.* dans l'espace de noms USER.

Pour plus d'informations sur les capacités du fichier de configuration config-api, consultez l'article ou la page github

Dockerfile

Le Dockerfile est basé sur le modèle existant docker template, mais nous devons faire quelques changements pour créer un répertoire de travail, installer les outils pour utiliser l'IP virtuelle, installer ZPM, etc…

Notre image IRIS est la même pour chaque membre du miroir. Le miroir sera mis en place sur le conteneur en commençant par la configuration correcte selon son rôle (primary, backup, ou report async r\w). Voir les commentaires sur le Dockerfile ci-dessous :

```
ARG IMAGE=containers.intersystems.com/intersystems/iris:2021.1.0.215.0
# Il n'est pas nécessaire de télécharger l'image depuis WRC. Elle sera tirée de l'ICR
 au moment de la construction.

FROM $IMAGE

USER root

COPY session.sh /
COPY iris.key /usr/irissys/mgr/iris.key

# /opt/demo sera notre répertoire de travail utilisé pour stocker nos fichiers de con
figuration et autres fichiers d'installation.
# Installez iputils-
arping pour avoir une commande arping.  Nécessaire pour configurer l'IP virtuelle.
# Téléchargez la dernière version de ZPM (ZPM est inclus uniquement dans l'édition co
mmunautaire).
RUN mkdir /opt/demo && \
    chown ${ISC_PACKAGE_MGRUSER}:${ISC_PACKAGE_IRISGROUP} /opt/demo && \
    chmod 666 /usr/irissys/mgr/iris.key && \
    apt-get update && apt-get install iputils-arping gettext-base && \
    wget -O /opt/demo/zpm.xml https://pm.community.intersystems.com/packages/zpm/late
```

```
st/installer

USER ${ISC_PACKAGE_MGRUSER}

WORKDIR /opt/demo

# Définissez le rôle du miroir par défaut comme assistant.
# La valeur sera remplacée dans le fichier docker-compose au moment de l'exécution.
ARG IRIS_MIRROR_ROLE=master

# Copiez le contenu du répertoire config-files dans /opt/demo.
# Actuellement, nous n'avons créé qu'un simple-
config pour configurer notre base de données et le mapping global.
# Plus tard dans cet article, nous ajouterons d'autres fichiers de configuration pour
 mettre en place le miroir.
ADD config-files .

SHELL [ "/session.sh" ]

# Installez ZPM
# Utilisez ZPM pour installer config-api
# Chargez le fichier simple-config.json avec config-api pour :
# - créer la base de données "myappdata",
# - ajouter un mapping global dans l'espace de nom "USER" pour la globale "demo.*" su
r la base de données "myappdata".
# Fondamentalement, le point d'entrée pour installer votre application ObjectScript e
st ici.
# Pour cet exemple, nous allons charger simple-
config.json pour créer une base de données simple et un mapping global.
RUN \
Do $SYSTEM.OBJ.Load("/opt/demo/zpm.xml", "ck") \
zpm "install config-api" \
Set sc = ##class(Api.Config.Services.Loader).Load("/opt/demo/simple-config.json")

# Copiez le script d'initialisation du miroir.
COPY init_mirror.sh /
```

Construction de l'image IRIS

Le Dockerfile est prêt ; nous pouvons construire l'image :

```
docker build --no-cache --tag mirror-demo:latest .
```

Cette image sera utilisée pour exécuter les nœuds primary, backup et de report async.

The .env file

Les fichiers de configuration JSON et docker-compose utilisent des variables d'environnement.
Leurs valeurs sont stockées dans un fichier nommé .env, pour ce sample notre fichier env est :

```
APP_NET_SUBNET=172.16.238.0/24
MIRROR_NET_SUBNET=172.16.220.0/24

IRIS_HOST=172.16.238.100
```

```
IRIS_PORT=1972
IRIS_VIRTUAL_IP=172.16.238.100

ARBITER_IP=172.16.238.10

MASTER_APP_NET_IP=172.16.238.20
MASTER_MIRROR_NET_IP=172.16.220.20

BACKUP_APP_NET_IP=172.16.238.30
BACKUP_MIRROR_NET_IP=172.16.220.30

REPORT_APP_NET_IP=172.16.238.40
REPORT_MIRROR_NET_IP=172.16.220.40
```

Préparation du fichier de configuration du premier membre du miroir

La bibliothèque config-api permet de configurer un miroir, nous devons donc créer un fichier de configuration dédié au premier membre du miroir : config-files/mirror-master.json

Pour plus de commodité, les commentaires sont situés directement dans le JSON. Vous pouvez télécharger le fichier [mirror-master.json sans commentaire ici] (https://raw.githubusercontent.com/lscalese/iris-mirroring-samples/master...).

```
{
    "Security.Services" : {
        "%Service_Mirror" : {
            "Enabled" : true
        }
    },
    "SYS.MirrorMaster" : {
        "Demo" : {
            "Config" : {
                "Name" : "Demo",                                /* Le nom de notre mi
roir */
                "SystemName" : "master",                        /* Le nom de cette in
stance dans le miroir */
                "UseSSL" : true,
                "ArbiterNode" : "${ARBITER_IP}|2188",           /* L'adresse IP et po
rt du nœud arbitre */
                "VirtualAddress" : "${IRIS_VIRTUAL_IP}/24",     /* L'adresse IP virtu
elle */
                "VirtualAddressInterface" : "eth0",             /* L'interface réseau
 utilisée pour l'adresse IP virtuelle. */
                "MirrorAddress": "${MASTER_MIRROR_NET_IP}",     /* L'adresse IP de ce
 noeud dans le réseau miroir privé */
                "AgentAddress": "${MASTER_APP_NET_IP}"          /* L'adresse IP de ce
 nœud (l'agent est installé sur la même machine) */
            },
            "Databases" : [{                                    /* La liste des bases
 de données à ajouter au miroir */
                "Directory" : "/usr/irissys/mgr/myappdata/",
                "MirrorDBName" : "MYAPPDATA"
            }],
            "SSLInfo" : {                                       /* SSL Configuration
*/
                "CAFile" : "/certificates/CA_Server.cer",
                "CertificateFile" : "/certificates/master_server.cer",
```

```
                "PrivateKeyFile" : "/certificates/master_server.key",
                "PrivateKeyPassword" : "",
                "PrivateKeyType" : "2"
            }
        }
    }
}
```

Préparer le fichier de configuration du membre de basculement

Créer un fichier de configuration pour le membre backup (basculement) config-files/mirror-backup.json.

Le fichier est fort ressemblant au fichier de configuration du membre primary:

```
{
    "Security.Services" : {
        "%Service_Mirror" : {
            "Enabled" : true
        }
    },
    "SYS.MirrorFailOver" : {
        "Demo" : {                                          /* Le miroir à rejoindre
*/
            "Config": {
                "Name" : "Demo",
                "SystemName" : "backup",                    /* Le nom de cette instan
ce dans le miroir */
                "InstanceName" : "IRIS",                    /* Le nom de l'instance I
RIS du premier membre du miroir */
                "AgentAddress" : "${MASTER_APP_NET_IP}",    /* L'adresse IP de l'agen
t du premier membre du miroir */
                "AgentPort" : "2188",
                "AsyncMember" : false,
                "AsyncMemberType" : ""
            },
            "Databases" : [{                                /* DB dans le miroir */
                 "Directory" : "/usr/irissys/mgr/myappdata/"
            }],
            "LocalInfo" : {
                "VirtualAddressInterface" : "eth0",         /* L'interface réseau uti
lisée pour l'adresse IP virtuelle */
                "MirrorAddress": "${BACKUP_MIRROR_NET_IP}"  /* L'adresse IP de ce noe
ud dans le réseau miroir privé */
            },
            "SSLInfo" : {
                "CAFile" : "/certificates/CA_Server.cer",
                "CertificateFile" : "/certificates/backup_server.cer",
                "PrivateKeyFile" : "/certificates/backup_server.key",
                "PrivateKeyPassword" : "",
                "PrivateKeyType" : "2"
            }
        }
    }
}
```

Préparation du fichier de configuration des membres asynchrones en lecture-écriture

Il est assez similaire au fichier de configuration de basculement backup. Les différences sont les valeurs de AsyncMember, AsyncMemberType, et MirrorAddress.
Créez le fichier ./config-files/mirror-report.json :

```
{
    "Security.Services" : {
        "%Service_Mirror" : {
            "Enabled" : true
        }
    },
    "SYS.MirrorFailOver" : {
        "Demo" : {
            "Config": {
                "Name" : "Demo",
                "SystemName" : "report",
                "InstanceName" : "IRIS",
                "AgentAddress" : "${MASTER_APP_NET_IP}",
                "AgentPort" : "2188",
                "AsyncMember" : true,
                "AsyncMemberType" : "rw"
            },
            "Databases" : [{
                 "Directory" : "/usr/irissys/mgr/myappdata/"
            }],
            "LocalInfo" : {
                "VirtualAddressInterface" : "eth0",
                "MirrorAddress": "${REPORT_MIRROR_NET_IP}"
            },
            "SSLInfo" : {
                "CAFile" : "/certificates/CA_Server.cer",
                "CertificateFile" : "/certificates/report_server.cer",
                "PrivateKeyFile" : "/certificates/report_server.key",
                "PrivateKeyPassword" : "",
                "PrivateKeyType" : "2"
            }
        }
    }
}
```

Génération des certificats et configuration des nœuds IRIS et

Tous les fichiers de configuration sont prêts !

Maintenant nous devons ajouter un script pour générer des certificats afin de sécuriser la communication entre chaque nœud. Un script prêt à l'emploi est disponible dans le repository gen-certificates.sh

```
# sudo est nécessaire en raison de l'utilisation de chown, chgrp chmod.
sudo ./gen-certificates.sh
```

Pour configurer chaque nœud, initmirror.sh sera exécuté au démarrage des conteneurs. Il sera configuré plus tard dans docker-compose.yml dans la section commande command : ["-a", "/initmirror.sh"] :

```
#!/bin/bash
```

```
# Base de données utilisée pour tester le miroir.
DATABASE=/usr/irissys/mgr/myappdata

# Répertoire contenant mes données d'application sauvegardées par l'assistant pour le
s restaurer sur les autres nœuds et en faire un miroir.
BACKUP_FOLDER=/opt/backup

# Fichier de configuration miroir au format json config-api pour le nœud d'assistant.
MASTER_CONFIG=/opt/demo/mirror-master.json

# Fichier de configuration miroir au format json config-
api pour le nœud de sauvegarde.
BACKUP_CONFIG=/opt/demo/mirror-backup.json

# Fichier de configuration miroir au format json config-
api pour le nœud asynchrone de rapport.
REPORT_CONFIG=/opt/demo/mirror-report.json

# Nom du miroir...
MIRROR_NAME=DEMO

# Liste des membres du miroir.
MIRROR_MEMBERS=BACKUP,REPORT

# Chargement de la configuration du miroir en utilisant config-
api avec le fichier /opt/demo/simple-config.json.
# Démarrage d'une tâche Job pour accepter automatiquement d'autres membres nommés "ba
ckup" et "report" pour rejoindre le miroir (éviter la validation manuelle dans la ges
tion du portail).
master() {
rm -rf $BACKUP_FOLDER/IRIS.DAT
envsubst < ${MASTER_CONFIG} > ${MASTER_CONFIG}.resolved
iris session $ISC_PACKAGE_INSTANCENAME -U %SYS <<- END
Set sc = ##class(Api.Config.Services.Loader).Load("${MASTER_CONFIG}.resolved")
Set ^log.mirrorconfig(\$i(^log.mirrorconfig)) = \$SYSTEM.Status.GetOneErrorText(sc)
Job ##class(Api.Config.Services.SYS.MirrorMaster).AuthorizeNewMembers("${MIRROR_MEMBE
RS}","${MIRROR_NAME}",600)
Hang 2
Halt
END
}

make_backup() {
iris session $ISC_PACKAGE_INSTANCENAME -U %SYS "##class(SYS.Database).DismountDatabas
e(\"${DATABASE}\")"
md5sum ${DATABASE}/IRIS.DAT
cp ${DATABASE}/IRIS.DAT ${BACKUP_FOLDER}/IRIS.TMP
mv ${BACKUP_FOLDER}/IRIS.TMP ${BACKUP_FOLDER}/IRIS.DAT
chmod 777 ${BACKUP_FOLDER}/IRIS.DAT
iris session $ISC_PACKAGE_INSTANCENAME -U %SYS "##class(SYS.Database).MountDatabase(\
"${DATABASE}\")"
}

# Restauration de la base de données miroir "myappdata".  Cette restauration est effe
ctuée sur le noeud "backup" et "report".
restore_backup() {
sleep 5
while [ ! -f $BACKUP_FOLDER/IRIS.DAT ]; do sleep 1; done
sleep 2
```

```
iris session $ISC_PACKAGE_INSTANCENAME -U %SYS "##class(SYS.Database).DismountDatabas
e(\"${DATABASE}\")"
cp $BACKUP_FOLDER/IRIS.DAT $DATABASE/IRIS.DAT
md5sum $DATABASE/IRIS.DAT
iris session $ISC_PACKAGE_INSTANCENAME -U %SYS "##class(SYS.Database).MountDatabase(\
"${DATABASE}\")"
}

# Configuration du membre "backup"
#  - Chargement du fichier de configuration /opt/demo/mirror-
backup.json si cette instance est le backup ou
#    /opt/demo/mirror-
report.json si cette instance est le rapport (nœud miroir asynchrone R\W).
other_node() {
sleep 5
envsubst < $1 > $1.resolved
iris session $ISC_PACKAGE_INSTANCENAME -U %SYS <<- END
Set sc = ##class(Api.Config.Services.Loader).Load("$1.resolved")
Halt
END
}

if [ "$IRIS_MIRROR_ROLE" == "master" ]
then
  master
  make_backup
elif [ "$IRIS_MIRROR_ROLE" == "backup" ]
then
  restore_backup
  other_node $BACKUP_CONFIG
else
  restore_backup
  other_node $REPORT_CONFIG
fi

exit 0
```

Fichier Docker-compose

Nous avons quatre conteneurs pour commencer. Un fichier Docker-compose est parfait pour orchestrer notre sample.

```
version: '3.7'

services:
  arbitre:
    image: containers.intersystems.com/intersystems/arbiter:2021.1.0.215.0
    init: true
    container_name: mirror-demo-arbiter
    commande:
      - /usr/local/etc/irissys/startISCAgent.sh 2188
    réseaux:
      app_net:
        ipv4_address: ${ARBITER_IP}
    extra_hosts:
      - "master:${MASTER_APP_NET_IP}"
      - "backup:${BACKUP_APP_NET_IP}"
```

```
      - "report:${REPORT_APP_NET_IP}"
    cap_add:
      - NET_ADMIN

  assistant:
    construction: .
    image: mirror-demo
    container_name: mirror-demo-master
    réseaux:
      app_net:
        ipv4_address: ${MASTER_APP_NET_IP}
      mirror_net:
        ipv4_address: ${MASTER_MIRROR_NET_IP}
    environnement:
      - IRIS_MIRROR_ROLE=master
      - WEBGATEWAY_IP=${WEBGATEWAY_IP}
      - MASTER_APP_NET_IP=${MASTER_APP_NET_IP}
      - MASTER_MIRROR_NET_IP=${MASTER_MIRROR_NET_IP}
      - ARBITER_IP=${ARBITER_IP}
      - IRIS_VIRTUAL_IP=${IRIS_VIRTUAL_IP}
    ports:
      - 81:52773
    volumes:
      - ./backup:/opt/backup
      - ./init_mirror.sh:/init_mirror.sh
      # Montage des certificats
      - ./certificates/master_server.cer:/certificates/master_server.cer
      - ./certificates/master_server.key:/certificates/master_server.key
      - ./certificates/CA_Server.cer:/certificates/CA_Server.cer
      #- ~/iris.key:/usr/irissys/mgr/iris.key
    nom de l'hôte: master
    extra_hosts:
      - "backup:${BACKUP_APP_NET_IP}"
      - "report:${REPORT_APP_NET_IP}"
    cap_add:
      - NET_ADMIN
    commande: ["-a", "/init_mirror.sh"]

  sauvegarde:
    image: mirror-demo
    container_name: mirror-demo-backup
    réseaux:
      app_net:
        ipv4_address: ${BACKUP_APP_NET_IP}
      mirror_net:
        ipv4_address: ${BACKUP_MIRROR_NET_IP}
    ports:
      - 82:52773
    environnement :
      - IRIS_MIRROR_ROLE=backup
      - WEBGATEWAY_IP=${WEBGATEWAY_IP}
      - BACKUP_MIRROR_NET_IP=${BACKUP_MIRROR_NET_IP}
      - MASTER_APP_NET_IP=${MASTER_APP_NET_IP}
      - BACKUP_APP_NET_IP=${BACKUP_APP_NET_IP}
    volumes:
      - ./backup:/opt/backup
      - ./init_mirror.sh:/init_mirror.sh
      # Montage des certificats
      - ./certificates/backup_server.cer:/certificates/backup_server.cer
```

```
      - ./certificates/backup_server.key:/certificates/backup_server.key
      - ./certificates/CA_Server.cer:/certificates/CA_Server.cer
      #- ~/iris.key:/usr/irissys/mgr/iris.key
    nom de l'hôte: backup
    extra_hosts:
      - "master:${MASTER_APP_NET_IP}"
      - "report:${REPORT_APP_NET_IP}"
    cap_add:
      - NET_ADMIN
    commande: ["-a", "/init_mirror.sh"]

  rapport :
    image: mirror-demo
    container_name: mirror-demo-report
    réseaux:
      app_net:
        ipv4_address: ${REPORT_APP_NET_IP}
      mirror_net:
        ipv4_address: ${REPORT_MIRROR_NET_IP}
    ports:
      - 83:52773
    environnement :
      - IRIS_MIRROR_ROLE=report
      - WEBGATEWAY_IP=${WEBGATEWAY_IP}
      - MASTER_APP_NET_IP=${MASTER_APP_NET_IP}
      - REPORT_MIRROR_NET_IP=${REPORT_MIRROR_NET_IP}
      - REPORT_APP_NET_IP=${REPORT_APP_NET_IP}
    volumes:
      - ./backup:/opt/backup
      - ./init_mirror.sh:/init_mirror.sh
      # Montage des certificats
      - ./certificates/report_server.cer:/certificates/report_server.cer
      - ./certificates/report_server.key:/certificates/report_server.key
      - ./certificates/CA_Server.cer:/certificates/CA_Server.cer
      #- ~/iris.key:/usr/irissys/mgr/iris.key
    nom de l'hôte: report
    extra_hosts:
      - "master:${MASTER_APP_NET_IP}"
      - "backup:${BACKUP_APP_NET_IP}"
    cap_add:
      - NET_ADMIN
    commande: ["-a", "/init_mirror.sh"]

réseaux:
  app_net:
    ipam:
      driver: default
      config:
        - subnet: "${APP_NET_SUBNET}"
  mirror_net:
    ipam:
      driver: default
      config:
        - subnet: "${MIRROR_NET_SUBNET}"
```

Le fichier docker-compose.yml contient beaucoup de variables d'environnement. Pour voir le fichier résolu, tapez dans le terminal :

```
docker-compose config
```

Exécuter des conteneurs

```
docker-compose up
```

Attendez que chaque instance ait un statut de miroir correct :

- nœud primary avec le statut Primary.
- nœud backup avec le statut Backup.
- noeud report avec le statut Connected.

Enfin, vous devriez voir ces messages dans les journaux de docker :

```
mirror-demo-master | 01/09/22-11:02:08:227 (684) 1 [Utility.Event] Devient le serveur
 miroir primaire
...
mirror-demo-backup | 01/09/22-11:03:06:398 (801) 0 [Utility.Event] Trouver MASTER com
me primaire, devenant sauvegarde
...
mirror-demo-report | 01/09/22-11:03:10:745 (736) 0 [Generic.Event] MirrorClient : Con
necté au primaire : MASTER (ver 4)
```

Vous pouvez aussi vérifier l'état du miroir avec le portail http://localhost:81/csp/sys/utilhome.csp

Accès aux portails

Dans Docker-compose, nous mappons les ports 81, 82 et 83 pour avoir un accès à chaque portail de gestion.
Il s'agit du login/mot de passe par défaut pour toutes les instances :

- Assistant http://localhost:81/csp/sys/utilhome.csp
- Membre de sauvegarde du basculement http://localhost:82/csp/sys/utilhome.csp
- Membre asynchrone du rapport en lecture-écriture http://localhost:83/csp/sys/utilhome.csp

Test

Vérifiez le moniteur miroir (portail de gestion ; il s'agit de l'utilisateur et du mot de passe par défaut) :
http://localhost:81/csp/sys/op/%25CSP.UI.Portal.Mirror.Monitor.zen

Vérifiez les paramètres du miroir :
http://localhost:81/csp/sys/mgr/%25CSP.UI.Portal.Mirror.EditFailover.zen?$NAMESPACE=%25SYS

Nous pouvons démarrer un test en configurant simplement une globale commençant par demo.
Rappelez-vous que nous avons configuré un mapping global demo.* sur l'espace de nom USER.

Ouvrez une session de terminal sur le serveur primaire :

```
docker exec -it mirror-demo-master irissession iris
```

```
Set ^demo.test = $zdt($h,3,1)
```

Vérifiez si les données sont disponibles sur le nœud de sauvegarde :

```
docker exec -it mirror-demo-backup irissession iris
```

```
Write ^demo.test
```

Vérifiez si les données sont disponibles sur le nœud de rapport ::

```
docker exec -it mirror-demo-report irissession iris
```

```
Write ^demo.test
```

Bien ! Nous avons un environnement miroir prêt, entièrement créé par programmation.
Pour que ce soit un peu plus complet, il faudrait ajouter une passerelle web avec https et un cryptage entre la passerelle web et IRIS, mais nous allons laisser cela pour le prochain article.

Nous espérons que cet article vous sera utile si vous décidez de créer votre propre script

Source

Le contenu de cet article est inspiré par :

- @Dmitry Maslennikov iris-mirror-with-docker
- @Evgeny Shvarov docker template intersystems-community/objectscript-docker-template
- @Pete Greskoff article creating-ssl-enabled-mirror-intersystems-iris-using-public-key-infrastructure-pki
- @Robert Cemper IRIS easy ECP workbench

#DevOps #Mise en miroir #InterSystems IRIS
Voir l'application sur InterSystems Open Exchange

URL de la source:https://fr.community.intersystems.com/post/comment-configurer-un-miroir-de-mani%C3%A8re-programmatique