Artículo
Luis Angel Pére... · 23 nov, 2022 Lectura de 13 min

Open Exchange

Cómo configurar un mirror mediante programacion

Antecedentes

Versión	Fecha	Cambios
V1	08/02/2022	Lanzamiento Inicial
V1.1	06/04/2022	Generación de certificados con un archivo sh en vez de un pki-script Uso de variables de entorno en los archivos de configuración

¡Hola Comunidad!

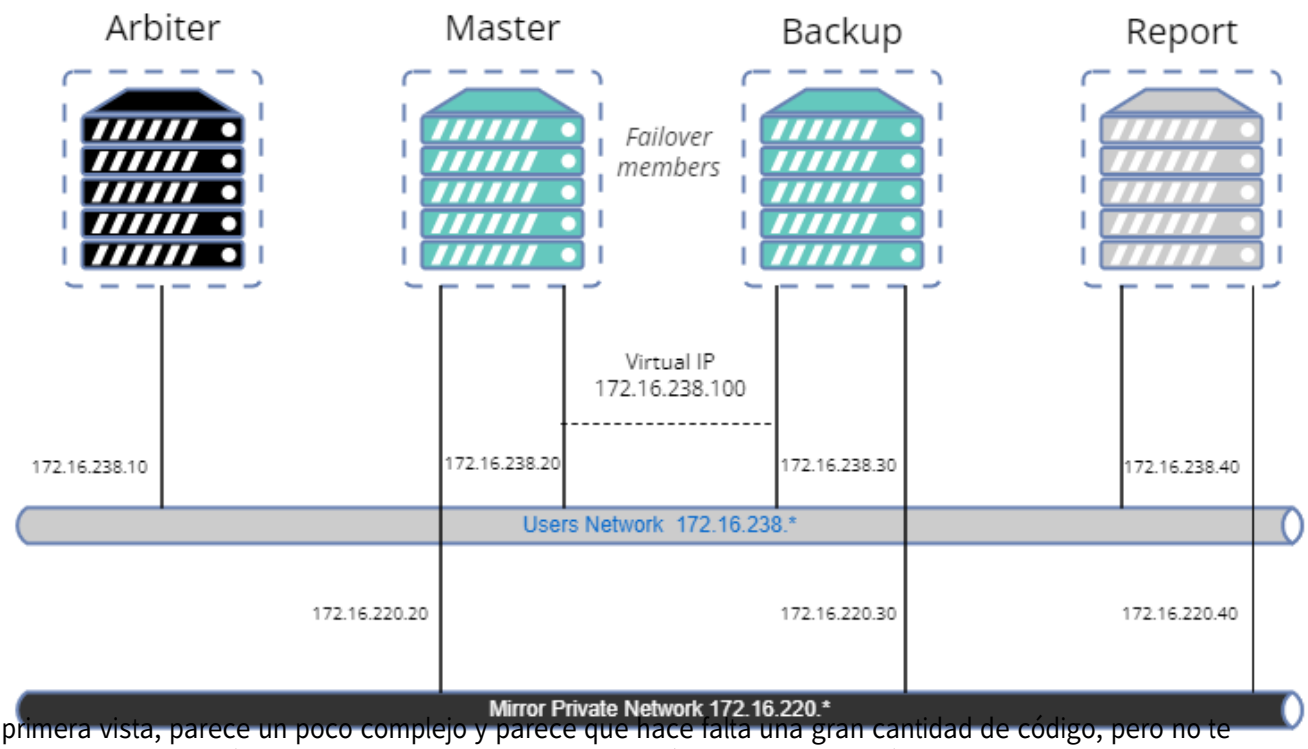
¿Ya habéis configurado un entorno en mirror? ¿Tenéis una red privada, una dirección IP virtual y una configuración SSL? Después de hacer esto un par de veces, me di cuenta de que es muy largo, y hay muchos pasos que hay que realizar manualmente para generar certificados y configurar cada instancia de IRIS. Es un dolor de cabeza para cualquiera que tenga que hacer esto a menudo.

Por ejemplo, un equipo de control de calidad podría necesitar un nuevo entorno por cada nueva versión de la aplicación que tenga que probar mientras que el equipo de soporte puede necesitar crear un entorno para reproducir un problema complejo.

Definitivamente, necesitamos herramientas para crearlos rápidamente.

En este artículo crearemos una muestra para configurar un *mirror* con:

- Arbiter.
- Primary.
- Miembro failover del backup.
- Miembro asíncrono de lectura-escritura de informes.
- Configuración SSL para transferencias de *journal* entre nodos.
- Red privada para el *mirror*.
- Dirección IP virtual.
- Una base de datos en mirror.



A primera vista, parece un poco complejo y parece que hace falta una gran cantidad de código, pero no te preocupes. Hay librerías en OpenExchange para realizar fácilmente la mayoría de las operaciones.

El propósito de este artículo es ofrecer un ejemplo de cómo adaptar el proceso a vuestras necesidades, pero no es una guía de prácticas recomendadas en materia de seguridad.

Así que vamos a crear nuestra muestra.

Herramientas y librerías

- config-api: Esta librería se utilizará para configurar IRIS. Es compatible con la configuración del *mirroring* desde la versión 1.1.0. No vamos a dar una descripción detallada de cómo utilizar esta librería. Ya hay varios artículos aquí. En resumen, config-api se utilizará para crear archivos de configuración de plantillas IRIS (formato JSON) y cargarlos fácilmente.
- ZPM.
- Docker.
- OpenSSL.

Página de Github

Podéis encontrar todos los archivos de recursos necesarios en el repositorio iris-mirroring-samples.

Preparación del sistema

Clonad el repositorio existente:

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

Si preferís crear una muestra desde cero, en vez de clonar el repositorio, simplemente cread un nuevo directorio

con subdirectorios: backup, y config-files. Descargad 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
```

Para evitar la incidencia "permiso rechazado" más tarde, tenemos que crear el grupo irisowner, el usuario irisowner, y cambiar el grupo del directorio del *backup* a irisowner

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

Este directorio se utilizará como volumen para compartir una copia de seguridad de la base de datos después de que se configure el primer miembro *mirror* con los otros nodos.

Obtención de una licencia de IRIS

Mirroring no está disponible con la Edición Community de IRIS. Si aún no tenéis una licencia válida para el contenedor de IRIS, conectaos al Centro de Soporte Internacional (WRC) con vuestras credenciales. Haced clic en "Actions" --> "Online distribtion", después en el botón "Evaluations" y seleccionad "Evaluation License". Completad el formulario. Copiad vuestro archivo de licencia iris.key en este directorio.

Inicio de sesión en el Registro de Contenedores de Intersystems

Por comodidad, utilizamos Intersystems Containers Registry (ICR) para extraer imágenes de Docker. Si no sabéis vuestro nombre de usuario\contraseña de Docker, conectaos a SSO.UI.User.ApplicationTokens.cls con vuestras credenciales del Centro de Soporte Internacional (WRC), y podréis recuperar vuestro Token ICR.

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

Creación de la base de datos myappdata y un mapeo de globales

De momento no vamos a crear la base de datos myappdata, únicamente estamos preparando la configuración para crearla al momento de crear el Docker. Para ello, simplemente creamos un archivo sencillo utilizando el formato JSON. La librería config-api se utilizará para cargarlo en las instancias de IRIS.

Cread el archivo 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" : {                       /* Enable the mirror service on thi
s instance */
            "Enabled" : true
        }
    }
}
```

Este archivo de configuración permite crear una nueva base de datos con la configuración predeterminada y hacer un mapeo del global demo.* en el *namespace* USER.

Para más información sobre las funciones del archivo de configuración config-api consulta el artículo, relacionado o la página de github.

El archivo Docker

El archivo Docker se basa en la plantilla existente de Docker, pero necesitamos hacer algunos cambios para crear un directorio de trabajo, instalar las herramientas para el uso de la IP virtual, instalar ZPM, etc···

Nuestra imagen IRIS es la misma para cada miembro *mirror*. El *mirroring* se establecerá en el contenedor empezando con la configuración correcta dependiendo de su función (primer miembro, backup de respaldo/failover o informe de lectura-escritura). Observa los comentarios en el Dockerfile:

```
ARG IMAGE=containers.intersystems.com/intersystems/iris:2021.1.0.215.0
# No es necesario descargar la imagen desde WRC, se hará automáticamente desde ICR cu
ando se despliegue el contenedor.

FROM $IMAGE

USER root

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

# /opt/demo será nuestro directorio de trabajo y en el que almacenaremos nuestros arc
hivos de configuración así como otros archivos de instalación.
# Instalamos iputils-arping para hacer uso del comando arping. Es necesario para conf
igurar una IP Virtual.
# Descargamos la última versión de ZPM (o IPM, incluida en las versiones a partir de
la 2023.1).
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

# Configuramos el rol del mirror por defecto a master.
# Se sobreescribirá en el archivo docker-compose en el momento de la ejecución (maste
r para la primera instancia, backup, y report)
ARG IRIS_MIRROR_ROLE=master

# Copiamos el contenido del directorio de archivos de configuración en /opt/demo.
# Únicamente hemos creado una configuración simple de nuestra base de datos y los map
eos de globales.
# Posteriormente en este mismo artículo incluiremos otros archivos de configuración p
ara desplegar el mirror.
ADD config-files .

SHELL [ "/session.sh" ]

# Instalamos el ZPM (no será necesario para versiones a partir de 2023.1)
# Usamos ZPM para instalar config-api
# Cargamos el archivo de configuración simple-config.json con config-api para:
#  - crear la base de datos "myappdata",
#  - añadirmos un mapeo de globales en el namespace "USER" para los globales "demo.*"
 a la base de datos "myappdata".
# Basicamente, el punto de entrada para instalar tu aplicación de ObjectScript es est
e.
# Para este ejemplo cargaremos simple-
config.json para crear una base de datos simple y un mapeo de globals.
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")

# Copiamos el script de arranque del mirror.
COPY init_mirror.sh /
```

Creación de la imagen IRIS

El Dockerfile está listo, podemos crear la imagen:

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

Esta imagen se utilizará para ejecutar los nodos primarios, los de copias de seguridad y los de informes.

El archivo .env

Los archivos de configuración JSON y docker-compose utilizan variables de entorno. Sus valores se almacenan en un archivo llamado .env. Para este ejemplo, nuestro archivo env es:

```
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
```

Preparación del archivo de configuración del primer miembro del mirror

La librería config-api permite configurar un *mirror*, por lo que debemos crear un archivo de configuración específico para el primer miembro *mirror* config-files/mirror-master.json

Para mayor comodidad, los comentarios se sitúan directamente en el JSON. Podéis descargar el mirror-master.json sin comentarios aquí.

```
{
    "Security.Services" : {
        "%Service_Mirror" : {
            "Enabled" : true
        }
    },
    "SYS.MirrorMaster" : {
        "Demo" : {
            "Config" : {
                "Name" : "Demo",                                /* El nombre de nuest
ro mirror */
                "SystemName" : "master",                        /* El nombre de esta
instancia en el mirror */
                "UseSSL" : true,
                "ArbiterNode" : "${ARBITER_IP}|2188",           /* Dirección IP y pue
rto para el nodo del arbiter */
                "VirtualAddress" : "${IRIS_VIRTUAL_IP}/24",     /* Dirección IP Virtu
al IP */
                "VirtualAddressInterface" : "eth0",             /* Interfaz de red us
ada para la dirección IP Virtual. */
                "MirrorAddress": "${MASTER_MIRROR_NET_IP}",     /* Dirección IP de es
te nodo en la red privada del mirror */
                "AgentAddress": "${MASTER_APP_NET_IP}"          /* Dirección IP de es
te nodo (Agent está instalado en la misma máquina) */
            },
            "Databases" : [{                                    /* Lista de bases de
datos añadidas al mirror */
                "Directory" : "/usr/irissys/mgr/myappdata/",
                "MirrorDBName" : "MYAPPDATA"
            }],
            "SSLInfo" : {                                       /* Configuración SSL
*/
                "CAFile" : "/certificates/CA_Server.cer",
                "CertificateFile" : "/certificates/master_server.cer",
                "PrivateKeyFile" : "/certificates/master_server.key",
                "PrivateKeyPassword" : "",
```

```
                "PrivateKeyType" : "2"
            }
        }
    }
}
```

Preparación del archivo de configuración del miembro failover

Creamos un archivo de configuración para los miembros de backup de respaldo (failover) config-files/mirror-backup.json.

Se parece al primer miembro del mirror:

```
{
    "Security.Services" : {
        "%Service_Mirror" : {
            "Enabled" : true
        }
    },
    "SYS.MirrorFailOver" : {
        "Demo" : {                                          /* Datos del mirror al qu
e se va a unir */
            "Config": {
                "Name" : "Demo",
                "SystemName" : "backup",                    /* Nombre de esta instanc
ia en el mirror */
                "InstanceName" : "IRIS",                    /* Nombre de la instancia
 de IRIS del primer miembro del mirror */
                "AgentAddress" : "${MASTER_APP_NET_IP}",    /* Dirección IP del Agent
 del primer miembro del mirror */
                "AgentPort" : "2188",
                "AsyncMember" : false,
                "AsyncMemberType" : ""
            },
            "Databases" : [{                                /* Base de datos en mirro
r */
                 "Directory" : "/usr/irissys/mgr/myappdata/"
            }],
            "LocalInfo" : {
                "VirtualAddressInterface" : "eth0",         /* Interfaz de red usada
por la dirección IP Virtual. */
                "MirrorAddress": "${BACKUP_MIRROR_NET_IP}"  /* Dirección IP de este n
odo en la red privada del mirror */
            },
            "SSLInfo" : {
                "CAFile" : "/certificates/CA_Server.cer",
                "CertificateFile" : "/certificates/backup_server.cer",
                "PrivateKeyFile" : "/certificates/backup_server.key",
                "PrivateKeyPassword" : "",
                "PrivateKeyType" : "2"
            }
        }
    }
}
```

Preparación del archivo de configuración del miembro en modo lectura-escritura asíncrono

Es bastante similar al archivo de configuración de failover. Las diferencias son los valores de AsyncMember, AsyncMemberType, y MirrorAddress. Creamos el archivo ./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"
            }
        }
    }
}
```

Generación de certificados y configuración de los nodos IRIS y

¡Todos los archivos de configuración están listos!

Ahora tenemos que añadir un *script* para generar certificados para asegurar la comunicación entre cada uno de los nodos. En el repositorio gen-certificates.sh hay un script listo para ser usado

```
# sudo es obligatorio debido al uso de chown, chgrp chmod.
sudo ./gen-certificates.sh
```

Para configurar cada nodo initmirror.sh se realizará al iniciar los contenedores. Se configurará posteriormente en docker-compose.yml en la sección de comandos command: ["-a", "/initmirror.sh"] :

```
#!/bin/bash

# Base de datos usada para probar el mirror.
```

```
DATABASE=/usr/irissys/mgr/myappdata

# Directorio que contiene myappdata copiada por el master para restaurar en los otros
 nodos y hacer el mirror.
BACKUP_FOLDER=/opt/backup

# Archivo de configuración del mirror en json con formato de config-
api para el nodo master.
MASTER_CONFIG=/opt/demo/mirror-master.json

# Archivo de configuración del mirror en json con formato de config-
api para el nodo de backup.
BACKUP_CONFIG=/opt/demo/mirror-backup.json

# Archivo de configuración del mirror en json con formato de config-
api para el nodo asíncrono.
REPORT_CONFIG=/opt/demo/mirror-report.json

# El nombre del mirror...
MIRROR_NAME=DEMO

# Lista de miembros del mirror.
MIRROR_MEMBERS=BACKUP,REPORT

# Ejecutado en el master.
# Carga de la configuración del mirror usando config-
api con el archivo /opt/demo/simple-config.json.
# Iniciamos un Job para auto-aceptar otros miembros llamados "backup" y "report" se u
nan al mirror (evitando la validación manual desde el portal de gestión).
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
}

# Ejecutado por el master, hacemos un backup de /usr/irissy
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}\")"
}

# Restauramos la base datos en mirror "myappdata".  Esta restauración es ejecutada en
 los nodos "backup" y "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}\")"
}

# Configuramos el miembro "backup"
#  - Cargamos el archivo de configuración /opt/demo/mirror-
backup.json si esta instancia es la de backup o
#    /opt/demo/mirror-report.json si esta instancia es la de report (nodo mirror con
lecturas/escrituras asíncronas).
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
```

Archivo Docker-compose

Tenemos cuatro contenedores para empezar. El archivo Docker-compose es perfecto para organizar nuestro ejemplo.

```
version: '3.7'

services:
  arbiter:
    image: containers.intersystems.com/intersystems/arbiter:2021.1.0.215.0
    init: true
    container_name: mirror-demo-arbiter
    command:
      - /usr/local/etc/irissys/startISCAgent.sh 2188
    networks:
      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

  master:
    build: .
    image: mirror-demo
    container_name: mirror-demo-master
    networks:
      app_net:
        ipv4_address: ${MASTER_APP_NET_IP}
      mirror_net:
        ipv4_address: ${MASTER_MIRROR_NET_IP}
    environment:
      - IRIS_MIRROR_ROLE=master
      - WEBGATEWAY_IP=${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
      # Mount certificates
      - ./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
    hostname: master
    extra_hosts:
      - "backup:${BACKUP_APP_NET_IP}"
      - "report:${REPORT_APP_NET_IP}"
    cap_add:
      - NET_ADMIN
    command: ["-a", "/init_mirror.sh"]

  backup:
    image: mirror-demo
    container_name: mirror-demo-backup
    networks:
      app_net:
        ipv4_address: ${BACKUP_APP_NET_IP}
      mirror_net:
        ipv4_address: ${BACKUP_MIRROR_NET_IP}
    ports:
      - 82:52773
    environment:
      - IRIS_MIRROR_ROLE=backup
      - WEBGATEWAY_IP=${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
      # Mount certificates
```

```
      - ./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
    hostname: backup
    extra_hosts:
      - "master:${MASTER_APP_NET_IP}"
      - "report:${REPORT_APP_NET_IP}"
    cap_add:
      - NET_ADMIN
    command: ["-a", "/init_mirror.sh"]

  report:
    image: mirror-demo
    container_name: mirror-demo-report
    networks:
      app_net:
        ipv4_address: ${REPORT_APP_NET_IP}
      mirror_net:
        ipv4_address: ${REPORT_MIRROR_NET_IP}
    ports:
      - 83:52773
    environment:
      - IRIS_MIRROR_ROLE=report
      - WEBGATEWAY_IP=${WEBGATEWAY_IP}
      - MASTER_APP_NET_IP=${MASTER_APP_NET_IP}
      - REPORT_MIRROR_NET_IP=${REPORT_MIRROR_NET_IP}
      - REPORT_APP_NET_IP=${REPORT_APP_NET_IP}
    volumes:
      - ./backup:/opt/backup
      - ./init_mirror.sh:/init_mirror.sh
      # Mount certificates
      - ./certificates/report_server.cer:/certificates/report_server.cer
      - ./certificates/report_server.key:/certificates/report_server.key
      - ./certificates/CA_Server.cer:/certificates/CA_Server.cer
      #- ~/iris.key:/usr/irissys/mgr/iris.key
    hostname: report
    extra_hosts:
      - "master:${MASTER_APP_NET_IP}"
      - "backup:${BACKUP_APP_NET_IP}"
    cap_add:
      - NET_ADMIN
    command: ["-a", "/init_mirror.sh"]

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

El archivo docker-compose.yml contiene una gran cantidad de variables de entorno. Observa el tipo de archivo que devuelve en el terminal:

```
docker-compose config
```

Despliegue de los contenedores

```
docker-compose up
```

Esperamos a que cada instancia tenga su estado correcto en el *mirror*:

- nodo maestro (master) con estado Primary.
- nodo de la copia de seguridad (backup) con estado Backup.
- nodo de informes (report) con el estado Connected.

Finalmente, deberíais ver estos mensajes en los inicio de sesión de Docker:

```
mirror-demo-master | 01/09/22-11:02:08:227 (684) 1 [Utility.Event] Becoming primary m
irror server
...
mirror-demo-backup | 01/09/22-11:03:06:398 (801) 0 [Utility.Event] Found MASTER as pr
imary, becoming backup
...
mirror-demo-report | 01/09/22-11:03:10:745 (736) 0 [Generic.Event] MirrorClient: Conn
ected to primary: MASTER (ver 4)
```

También se puede verificar el estado del *mirror* mediante el portal http://localhost:81/csp/sys/utilhome.csp

Acceso a los portales

En Docker-compose mapeamos los puertos 81, 82 y 83 para tener un acceso a cada portal de administración. Esta es la contraseña de acceso predeterminada para todas las instancias:

- Maestra http://localhost:81/csp/sys/utilhome.csp
- Miembro de backup de respaldo (failover) http://localhost:82/csp/sys/utilhome.csp
- Miembro asíncrono del informe de lectura-escritura http://localhost:83/csp/sys/utilhome.csp

Prueba

Comprobad el monitor del *mirror* (portal de administración, este es el usuario y la contraseña predeterminada): http://localhost:81/csp/sys/op/%25CSP.UI.Portal.Mirror.Monitor.zen

Verifica la configuración del *mirror*: http://localhost:81/csp/sys/mgr/%25CSP.UI.Portal.Mirror.EditFailover.zen?$NAMESPACE=%25SYS

Podemos iniciar una prueba simplemente configurando un *global* que comience por demo. Recuerda que hemos configurado un mapeo del global demo.* en el *namespace* USER.

Abrid una sesión del terminal en el servidor primario:

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

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

Comprobad si los datos están disponibles en el nodo de la copia de seguridad:

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

```
Write ^demo.test
```

Comprobamos si los datos están disponibles en el nodo del informe:

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

```
Write ^demo.test
```

¡Bien! Tenemos un entorno de *mirror* listo, creado por completo programáticamente. Para rizar el rizo, deberíamos añadir un web gateway con https y encriptación entre el web gateway e IRIS, pero lo dejaremos para el próximo artículo.

Espero que este artículo os haya resultado útil si decidís crear vuestro propio *script*.

Fuentes

El contenido de este artículo está inspirado en:

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

#DevOps #Mirroring #InterSystems IRIS
Ir a la aplicación en InterSystems Open Exchange