Artigo
Danusa Calixto · Jul. 27, 2022 12min de leitura

Open Exchange

Como configurar um "Mirror" programaticamente

Histórico

Versão	Data	Atualizações
V1	2022-02-08	Início
V1.1	2022-04-06	Geração de certificados com arquivo sh em vez de pki-script Usando variáveis de ambiente em arquivos de configuração

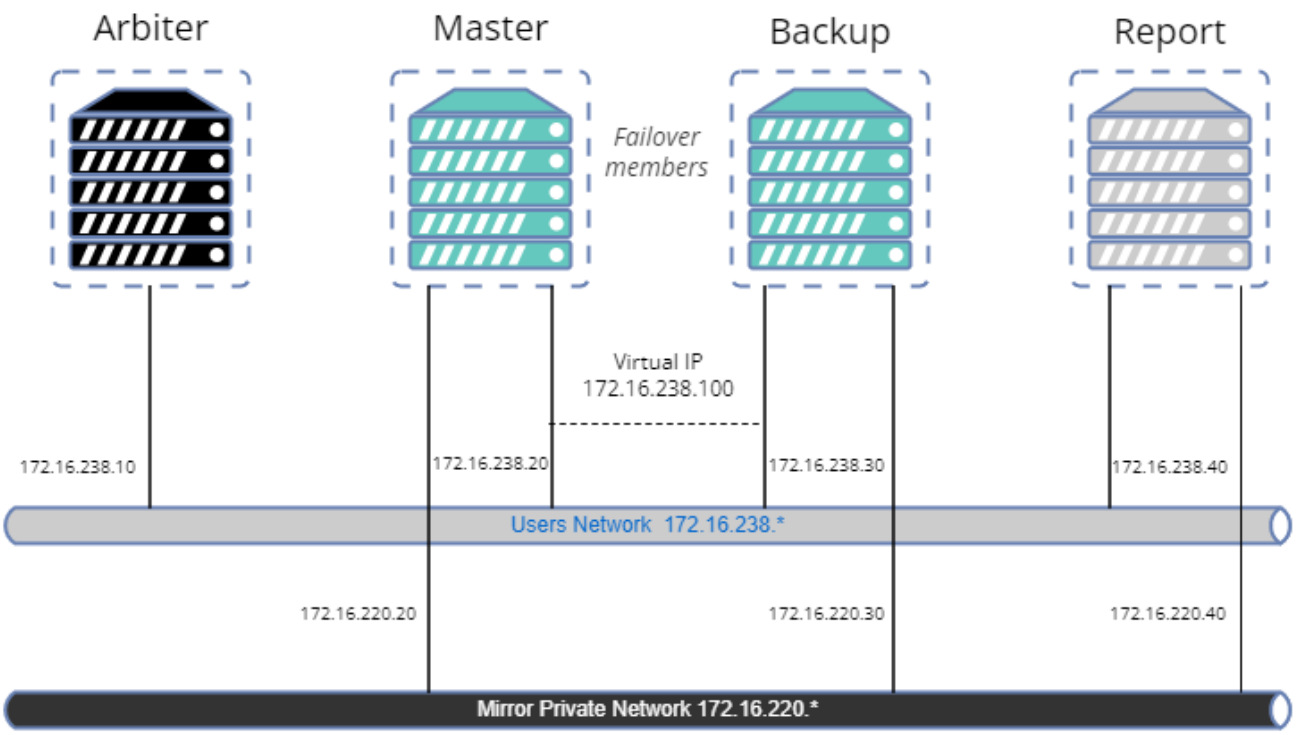
Olá Comunidade,

Vocês já configuraram um ambiente espelhado? Tem rede privada, IP virtual, e configuração SSL?
Após fazer isso algumas vezes, eu me dei conta que isso é longo, e há várias ações manuais obrigatórias para gerar os certificados e configurar cada instancia IRIS.
É uma dor no pescoço para quem tem que fazer isso muitas vezes.

Por exemplo, um time de Quality Assurance pode precisar criar um novo ambiente para cada nova versão da aplicação para testar. O time de suporte pode exigir a criação de um ambiente para reproduzir um problema complexo.

Nós definitivamente precisamos de ferramentas para criá-los rapidamente.

Neste artigo iremos criar um exemplo para configurar um Mirror com:
- Arbitro
- Primário
- Membro de backup
- Membro assíncrono de relatorio (leitura e escrita)
- Configuração SSL para transferencia de Journal entre os nós
- Rede privada para o Mirror
- Endereço IP virtual
- Um banco de dados espelhado



À primeira vista, parece um pouco complexo e parece que precisa de muito código, mas não se preocupe. Existem bibliotecas hospedadas no OpenExchange para realizar facilmente a maioria das operações.

O objetivo deste artigo é fornecer um exemplo de como adaptar o processo às suas necessidades, mas não é um guia de práticas recomendadas em termos de segurança.
Então, vamos criar nossa amostra.

Ferramentas e bibliotecas

- config-api: Esta biblioteca será utilizada para configurar o IRIS. Ela suporta configuração de espelhamento desde a versão 1.1.0. Não iremos dar uma descrição detalhada de como usar esta biblioteca. Já existe uma série de artigos. here. Resumindo, config-api será usada para criar um modelo de arquivos de configuração IRIS (formato JSON) e carregá-los facilmente.
- ZPM.
- Docker.
- OpenSSL.

Github

Você pode encontrar todos os arquivos necessários em iris-mirroring-samples repository.

Prepare o seu sistema

Clone o repositorio existente:

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

Se você preferir criar uma amostra do zero, ao invés de clonar o repositório, basta criar um novo diretório com subdiretórios: backup e config-files. Baixe
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 o problema de "permissão negada" posteriormente, precisamos criar o grupo 'irisowner', o usuário 'irisowner', e alterar o grupo do diretório de backup para 'irisowner'

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

Este diretório será utilizado como volume para compartilhar um backup do banco de dados após configurar o primeiro membro espelho com os outros nós.

Obtenha uma licença IRIS

O espelhamento não está disponível com a edição da comunidade IRIS.
Se você ainda não tiver uma licença de contêiner IRIS válida, conecte-se ao Worldwide Response Center (WRC) com suas credenciais.
Clique em "Ações" --> "Distribuição online", depois no botão "Avaliações" e selecione "Licença de avaliação";
Preencha o formulário.
Copie seu arquivo de licença iris.key para este diretório.

Login no Intersystems Containers Registry

Por motivos de conveniência, usamos o Intersystems Containers Registry (ICR) para extrair imagens do docker.
Se você não souber seu login\password do docker, basta conectar-se a SSO.UI.User.ApplicationTokens.cls com suas credenciais WRC, e você pode recuperar seu Token ICR.

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

Crie o banco de dados myappdata e o mapeamento de global

Nós não criamos realmente o banco de dados myappdata agora, mas preparamos uma configuração para criá-lo no momento da compilação do docker.
Para isso, basta criar um arquivo simples usando o formato JSON;
A biblioteca config-api será usada para carregá-la nas instâncias IRIS.

Crie o arquivo 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" : {                       /* Habilita o serviço de espelhamen
to nesta instancia */
            "Enabled" : true
        }
    }
}
```

Este arquivo de configuração permite que você crie um novo banco de dados com configurações padrão e faça o mapeamento global demo.* no namespace USER.

Para obter mais informações sobre os recursos do arquivo de configuração config-api, consulte o [artigo](https://community.intersystems.com/post/environment -setup-config-api) ou a página do github

O arquivo Docker

O arquivo docker é baseado no [modelo docker] existente (https://github.com/intersystems-community/objectscript-docker-template), mas precisamos fazer algumas alterações para criar um diretório de trabalho, instalar ferramentas para usar IP, instalar ZPM, etc…

Nossa imagem IRIS é a mesma para cada membro do espelho. O espelhamento será configurado no contêiner começando com a configuração correta dependendo de sua função (primeiro membro, backup ou relatório de leitura/gravação). Veja os comentários sobre o Dockerfile abaixo:

```
ARG IMAGE=containers.intersystems.com/intersystems/iris:2021.1.0.215.0
# Não precisa baixar a imagem do WRC. Ele será retirado do ICR no momento da construç
ão.

FROM $IMAGE

USER root

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

# /opt/demo será nosso diretório de trabalho usado para armazenar nossos arquivos de
configuração e outros arquivos de instalação.
# Instale o iputils-
arping para ter um comando arping. É necessário configurar o IP virtual.
# Baixe a versão mais recente do ZPM (o ZPM está incluído apenas na edição da comunid
ade).
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

# Defina a função de espelho padrão como mestre.
# Ele será substituído no arquivo docker-
compose em tempo de execução (mestre para a primeira instância, backup e relatório)
ARG IRIS_MIRROR_ROLE=master

# Copie o conteúdo do diretório config-files em /opt/demo.
# Atualmente criamos apenas uma configuração simples para configurar nosso banco de d
ados e mapeamento de global.
# Mais adiante neste artigo adicionaremos outros arquivos de configuração para config
urar o espelho.
ADD config-files .

SHELL [ "/session.sh" ]

# Instale o ZPM
# Use ZPM para instalar config-api
# Carregue o arquivo simple-config.json com config-api para:
#  - criar o banco de dados  "myappdata" ,
#  - adicionar o mapeamento de global no namespace "USER" para a global "demo.*" no b
anco de dados "myappdata" .
# Basicamente, o ponto de entrada para instalar seu aplicativo ObjectScript é aqui.
# Para este exemplo, carregaremos o simple-
config.json para criar um banco de dados simples e um mapeamento de 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")

# Copie o script de instalação do espelho
COPY init_mirror.sh /
```

Construa a imagem IRIS

O Dockerfile está pronto; podemos construir a imagem:

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

Essa imagem será usada para executar nós primários, de backup e de relatório.

O arquivo .env

Os arquivos de configuração JSON e a composição do docker usam variáveis de ambiente.
Seus valores são armazenados em um arquivo chamado .env, para este exemplo nosso arquivo env é:

```
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
```

Prepare o primeiro arquivo de configuração do membro espelho

A biblioteca config-api permite configurar um espelho, então temos que criar um arquivo de configuração dedicado ao primeiro membro do espelho config-files/mirror-master.json

Por conveniência, os comentários estão localizados diretamente no JSON. Você pode baixar o mirror-master.json sem comentários aqui.

```
{
    "Security.Services" : {
        "%Service_Mirror" : {
            "Enabled" : true
        }
    },
    "SYS.MirrorMaster" : {
        "Demo" : {
            "Config" : {
                "Name" : "Demo",                                /* O nome do nosso es
pelho */
                "SystemName" : "master",                        /* O nome da instanci
a no espelhamento */
                "UseSSL" : true,
                "ArbiterNode" : "${ARBITER_IP}|2188",           /* Endereço IP e Port
a do nó do Arbitro */
                "VirtualAddress" : "${IRIS_VIRTUAL_IP}/24",     /* Endereço IP virtua
l */
                "VirtualAddressInterface" : "eth0",             /* Interface de rede
usada para o endereço de IP virtual  */
                "MirrorAddress": "${MASTER_MIRROR_NET_IP}",     /* Endereço IP deste
nó na rede privada do espelho */
                "AgentAddress": "${MASTER_APP_NET_IP}"          /* Endereço IP deste
nó (o Agente é istalado na mesma máquina) */
            },
            "Databases" : [{                                    /* Lista de banco de
dados para adicionar ao espelhamento */
                "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"
            }
        }
    }
}
```

Prepare o arquivo de configuração do membro de failover

Crie um arquivo de configuração do membro de backup de failover config-files/mirror-backup.json.

Parece o primeiro membro:

```
{
    "Security.Services" : {
        "%Service_Mirror" : {
            "Enabled" : true
        }
    },
    "SYS.MirrorFailOver" : {
        "Demo" : {                                          /* Espelho para juntar */
            "Config": {
                "Name" : "Demo",
                "SystemName" : "backup",                    /* Este nome de instância
 no espelho */
                "InstanceName" : "IRIS",                    /* Nome da instancia IRIS
 do primeiro membro do espelho */
                "AgentAddress" : "${MASTER_APP_NET_IP}",    /* Endereço IP do agente
do primeiro membro espelho */
                "AgentPort" : "2188",
                "AsyncMember" : false,
                "AsyncMemberType" : ""
            },
            "Databases" : [{                                /* DB no espelho */
                 "Directory" : "/usr/irissys/mgr/myappdata/"
            }],
            "LocalInfo" : {
                "VirtualAddressInterface" : "eth0",         /* Interface de rede usad
a para o endereço IP virtual. */
                "MirrorAddress": "${BACKUP_MIRROR_NET_IP}"  /* Endereço IP deste nó n
a rede espelho privada*/
            },
            "SSLInfo" : {
                "CAFile" : "/certificates/CA_Server.cer",
                "CertificateFile" : "/certificates/backup_server.cer",
                "PrivateKeyFile" : "/certificates/backup_server.key",
                "PrivateKeyPassword" : "",
                "PrivateKeyType" : "2"
            }
        }
    }
}
```

Prepare o arquivo de configuração do membro assíncrono de leitura e gravação

É bastante semelhante ao arquivo de configuração de failover. As diferenças são os valores de AsyncMember, AsyncMemberType e MirrorAddress.

Crie o arquivo ./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"
            }
        }
    }
}
```

Gere os certificados e configure os nós IRIS

Todos os arquivos de configuração estão prontos!

Agora temos que adicionar script para gerar certificados para comunicação segura entre cada nós. Um script pronto para uso está disponível no repositório gen-certificates.sh

```
# sudo é necessário devido ao uso de chown, chgrp chmod.
sudo ./gen-certificates.sh
```

Para configurar cada nó o initmirror.sh será executado no início dos containers. Ele será configurado posteriormente em docker-compose.yml na seção de comando command: ["-a","/initmirror.sh"] :

```
#!/bin/bash

# Banco de dados usado para testar o espelho.
DATABASE=/usr/irissys/mgr/myappdata
```

```
# O diretório contém myappdata com backup feito pelo mestre para restaurar em outros
nós e fazer o espelho.
BACKUP_FOLDER=/opt/backup

# Espelhar arquivo de configuração no formato json config-api para o nó mestre.
MASTER_CONFIG=/opt/demo/mirror-master.json

# Espelhar arquivo de configuração no formato json config-api para o nó de backup.
BACKUP_CONFIG=/opt/demo/mirror-backup.json

# Espelhar arquivo de configuração no formato json config-
api para o nó assíncrono do relatório.
REPORT_CONFIG=/opt/demo/mirror-report.json

# O nome do espelho
MIRROR_NAME=DEMO

# Lista de membros do espelho .
MIRROR_MEMBERS=BACKUP,REPORT

# Realizado no mestre.
# Carregue a configuração do espelho usando config-api com o arquivo /opt/demo/simple-
config.json.
# Inicie um Job para aceitar automaticamente outros membros chamados "backup" e "repo
rt" para ingressar no espelho (evite validação de manual no gerenciamento do portal).
mestre() {
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
}

# Realizado pelo mestre, faça um 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}\")"
}

# Restaure o banco de dados espelhado "myappdata". Essa restauração é executada no nó
 "backup" e "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}\")"
}

# Configura o membro "backup"
# - Carregar arquivo de configuração /opt/demo/mirror-
backup.json se esta instância for o backup ou
# /opt/demo/mirror-
report.json se esta instância do relatório (nó espelho R\W assíncrono).
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
```

Arquivo Docker-compose

Temos quatro contêineres para começar. Um arquivo de composição do Docker é perfeito para orquestrar nossa amostra.

```
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=${REPORT_APP_NET_IP}
    volumes:
      - ./backup:/opt/backup
      - ./init_mirror.sh:/init_mirror.sh
      # Montar certificados
      - ./certificates/report_server.cer:/certificates/report_server.cer
      - ./certificates/report_server.key:/certificates/report_server.key
      - ./certificates/CA_Server.cer:/certificates/CA_Server.cer
      #- ~/iris.key:/usr/irissys/mgr/iris.key
    hostname: report
    extra_hosts:
      - "master:${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}"
```

O docker-compose.yml contém muitas variáveis de ambiente. Para ver o tipo de arquivo resolvido no terminal:

```
docker-compose config
```

Executar contêineres

```
docker-compose up
```

Aguarde para que cada instância tenha um bom status de espelhamento:

- nó mestre com status 'Primário'.
- nó de backup com status Backup.
- nó de relatório com status 'Conectado'.

Por fim, você deve ver estas mensagens nos logs do 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)
```

Você também pode verificar o status do espelho com o portal http://localhost:81/csp/sys/utilhome.csp

Acesso aos portais

No Docker-compose mapeamos as portas 81,82 e 83 para ter acesso a cada portal de gerenciamento.
Este é o login\senha padrão para todas as instâncias:

- Mestre http://localhost:81/csp/sys/utilhome.csp
- Membro de backup de failover http://localhost:82/csp/sys/utilhome.csp
- Membro assíncrono de relatório de leitura-gravação http://localhost:83/csp/sys/utilhome.csp

Teste

Verifique o monitor do espelho (porta de gerenciamento; este é o usuário e a senha padrão.):
http://localhost:81/csp/sys/op/%25CSP.UI.Portal.Mirror.Monitor.zen

Verifique as configurações do espelho:http://localhost:81/csp/sys/mgr/%25CSP.UI.Portal.Mirror.EditFailover.zen?$NAMESPACE=%25SYS

Podemos iniciar um teste simplesmente definindo um global começando por demo.
Lembre-se que configuramos um mapeamento global demo.* no namespace USER.

Abra uma sessão de terminal no servidor primário:

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

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

Verifique se os dados estão disponíveis no nó de backup:

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

Write ^demo.test
```

Check if the data is available on report node :

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

Write ^demo.test
```

Bom! Temos um ambiente espelho pronto, totalmente criado programaticamente.
Para ficar um pouco mais completo, devemos adicionar um web gateway com https e criptografia entre o web gateway e o IRIS, mas deixaremos para o próximo artigo.

Espero que este artigo seja útil para você se você decidir criar seu próprio script.

Fonte

O conteúdo deste artigo é inspirado em:

- @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 #Espelhamento #InterSystems IRIS
Confira o aplicativo relacionado no InterSystems Open Exchange