記事
Toshihiko Minamoto · 2022年6月2日 15m read

Open Exchange

プログラムでミラーをセットアップする方法

コミュニティの皆さん、こんにちは。

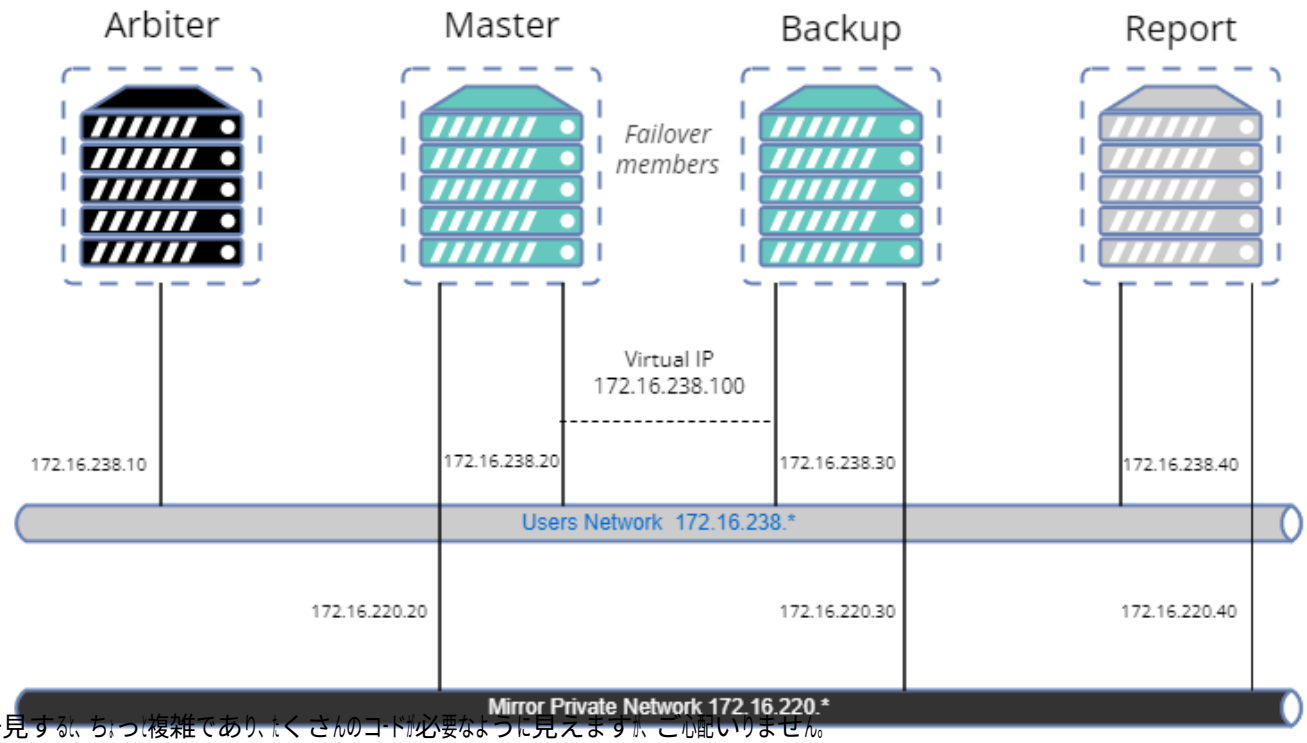
今までにミラーリング環境をセットアップされたことはありますか？ プライベートネットワーク、仮想 IP アドレス、および SSL 構成を設定したことはありますか？ この作業を何度か繰り返す、証明書の生成 IRIS インスタンスの構成など、たくさんの手動による作業が必要で、時間がかかる作業であることに気づきました。
頻繁にこの作業を行わなければならない方にとっては、面倒な作業です。

たとえば、品質管理チームでは、新しいバージョンのアプリケーションをテストするたびに、新しい環境を作成しなければならないでしょう。
サポートチームであれば、複雑な問題を再現する環境を作成しなくてはならないかもしれません。

となれば、こういった環境を素早く作成するツールが絶対に必要です。

この記事では、以下を使用するミラー環境のセットアップ例を紹介します。

- アービター
- プライマリ
- バックアップフェイルオーバーメンバー
- 読み書きレポート非同期メンバー
- ノード間でジャーナルを転送するための SSL 構成
- ミラー用のプライベートネットワーク
- 仮想 IP アドレス
- ミラーリングされたデータベース



一見すると、ちょっと複雑であり、たくさんのコードが必要なように見えますが、ご心配いりません。

OpenExchange には、ほとんどの操作を簡単に実行できるライブラリがあります。

この記事では、プロセスをニーズに適合させる方法を例示することを目的としていますが、セキュリティ事項においてはベストプラクティスガイドではありません。では、ここで使用するサンプルを作成しましょう。

ツールライブラリ

- PKI-script:
  公開鍵基盤 (PKI) は、自己署名された証明書を生成して権限サーバーを得られるようにする、IRIS に統合された機能です。 Pete Greskoff の素晴らしい記事 による、PKI Script ライブラリは、すべての操作をプログラムで実行し、管理ポータルでのあらゆる手動操作を回避することを目標としています。
  このライブラリには、ミラーリング用のユーティリティメソッドが含まれますが、証明書がすでにある場合は、PKI-Script の代わりにその証明書を使用できます。

- config-api: このライブラリは、IRIS の構成で使用されます。ミラーリング構成 バージョン 1.1.0 以来サポートされています。このライブラリの使用方法については詳しく説明しません。これについてはいくつかの記事がすでに公開されていますので、 こちら をご覧ください。
  手短に言えば、config-api は IRIS テンプレート構成ファイル (JSON 形式) の作成み込みを簡単に行うために使用されます。

- ZPM

- Docker

GitHub ページ

すべての必要なリソースは、 iris-mirroring-samples リポジトリ にあります。

システムの準備

既存のリポジトリをクローンします。

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

リポジトリをクローンする代わりにサンプルを新規作成する場合は、新しいディレクトリ、backup と config-files というサブカテゴリを作成ます。 irissession.sh をダウンロードします。

```
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
```

後で「アクセス拒否」の問題が発生しないように、irisowner グループに irisowner ユーザーを作成、バックアップディレクトリのグループを irisowner に変更する必要があります。

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

このディレクトリは、最初のミラーメンバーをセットアップした後に、他のノードとデータベースバックアップを共有するためのボリュームとして使用されます。

IRIS ライセンスを取得する

IRIS Community エディションでは、ミラーリングを使用できません。有効な IRIS コンテナーライセンスをまだお持ちでない場合は、資格情報を使用して Worldwide Response Center (WRC) にお問い合わせください。「アクション」-->「オンライン配布」、次に「評価」ボタンをクリックし、「評価ライセンス」を選択してフォームに入力します。ライセンスファイルの iris.key をこのディレクトリにコピーします。

InterSystems Containers Registry にログインする

更に、InterSystems Containers Registry (ICR) を使用して、Docker イメージをプルします。Docker ログイン \ パスワードが不明な場合は、WRC 資格情報を使用して SSO.UI.User.ApplicationTokens.cls にアクセスすると、ICR トークンを取得できます。

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

myappdata データベースとグローバルマッピングを作成する

ここでは実際には myappdata データベースの作成を行いませんが、Docker ビルド時にそれを作成するように構成を準備します。そのためには、JSON 形式で単純なファイルを作成します。IRIS インスタンスに読み込むには config-api ライブラリが使用されます。

config-files/simple-config.json ファイルを作成します。

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

   },
   "SYS.Databases":{
       "${DBDATADIR}" : {}
   },
   "Databases":{
       "${DBDATANAME}" : {
           "Directory" : "${DBDATADIR}"
       }
   },
   "MapGlobals":{
       "USER": [{
           "Name" : "demo.*",
           "Database" : "${DBDATANAME}"
       }]
   },
   "Security.Services" : {
       "%Service_Mirror" : {                    /* ?????????????????????? */
           "Enabled" : true
       }
   }
}
```

この構成ｲﾙを使用するﾋ、デフｫﾙﾄの設定で新しいデ-タベ-スを作成、USER ﾈ-ムスペ-スに
demo.* ﾋいうｸﾞﾛ-ﾊﾞﾙマｯﾋﾟﾝｸﾞを指定でｷます。

config-api 構成ｲﾙの機能に関するｾ詳細については、関連する 記事 ﾏたは GitHub ﾍﾟｼﾞ をご覧ください。

Docker ﾌｧｲﾙ

Docker ﾌｧｲﾙは既存の Docker ﾃﾝﾌﾟﾚｰﾄ を基ﾋしていますｶﾞ、作業ﾃｨﾚｸﾄﾘの作成想 IP
を使用するﾀめのツ-ﾙのインスト-ﾙ、ZPM のインスト-ﾙなどを行うように変更する必要ｶﾞあります。

IRIS ｲﾒｰｼは、すべてのミラ-メンﾊﾞ-で同一です。ミラ-ﾘﾝｸﾞは、ﾛｰﾙ(最初のメンﾊﾞ-、ﾌｪｲﾙｵ
-ﾊﾞ-ﾊﾞｯｸｱｯﾌﾟ、ﾏたは読み書ｷﾚﾎﾟｰﾄ)に応じて正しい構成使って開始するコンテナにセットｱｯﾌﾟさ
れます。以ｳの Dockerfile のコメントをご覧ください。

```
ARG IMAGE=containers.intersystems.com/intersystems/iris:2021.1.0.215.0
# WRC ?????????????????????? ?????????? ICR ????????
FROM $IMAGE

USER root

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

# /opt/demo ??????????????????????????????????????????
# arping ???????????iputils-arping ??????????  ?? IP ????????
# ??? ZPM ???????????????ZPM ? Communty ???????????????
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 && \
   wget -O /opt/demo/zpm.xml https://pm.community.intersystems.com/packages/zpm/lates
t/installer

USER ${ISC_PACKAGE_MGRUSER}

WORKDIR /opt/demo

# ????????????? master ???????
# ?????????? docker-compose ????????????????????????????????????????????
ARG IRIS_MIRROR_ROLE=master

# config-files ?????????? /opt/demo ????????
# ????????????????????????????????? simple-config ??????????
# ????????????????????????????????????????
ADD config-files .

SHELL [ "/session.sh" ]

# ZPM ??????????
# ZPM ??????config-api ? pki-script ??????????
# ???????????config-api ???? simple-config.json ???????????
#  - ?myappdata???????????
#  - ?myappdata?????????????demo.*???USER?????????????????????
# ?????????ObjectScript ??????????????????????????????
```

```
# ?????????simple-config.json ????????????????????????????????
RUN \
Do $SYSTEM.OBJ.Load("/opt/demo/zpm.xml", "ck") \
zpm "install config-api" \
zpm "install pki-script" \
Set sc = ##class(Api.Config.Services.Loader).Load("/opt/demo/simple-config.json")

# ????????????????????
COPY init_mirror.sh /

# ??????????????????????????????
# init_mirror.sh ??????????????????????
CMD ["-a", "/init_mirror.sh"]
```

IRIS イメージをビルドする

Dockerfile の準備ができたので、イメージをビルドできます。

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

このイメージは、プライマリ、バックアップ、およびレポートノードの実行に使用されます。

最初のミラーメンバー構成ファイルを準備する

config-api ライブラリではミラーの構成機能であるため、最初のミラーメンバー専用の構成ファイル config-files/mirror-master.json を作成する必要があります。

便宜、コメントは直接 JSON に記述されています。 コメントなしの mirror-master.json はこちら からダウンロードできます。 すべての IP アドレスは、Docker-compose ファイルのあるノードにそれぞれ割り当てられます。

```
{
   "Defaults":{                                    /* ????????????? */
       "MirrorName" : "Demo",                      /* ???? */
       "ArbiterNode" : "172.16.238.10|2188",       /* ????????? IP ???????? */
       "VirtualAddress" : "172.16.238.100/24",     /* ?? IP ???? */
       "VirtualAddressInterface" : "eth0",         /* ?? IP ?????????????????????? *
/
       "MirrorAddress" : "172.16.220.20",          /* ?????????????????????? IP ????
*/
       "AgentAddress" : "172.16.238.20",           /* ?????? IP ?????Agent ??????????
?????????? */
       "SystemName" : "master",                    /* ?????????????? */
       "DBDir" : "${MGRDIR}myappdata/",            /* Demo ????????????????????? */
       "DBName" : "MYAPPDATA"                      /* ??????????? */
   },
   "SYS.MirrorMaster" : {
       "${MirrorName}" : {
           "Config" : {
               "Name" : "${MirrorName}",
               "SystemName" : "${SystemName}",
               "UseSSL" : true,
               "ArbiterNode" : "${ArbiterNode}",
               "VirtualAddress" : "${VirtualAddress}",
```

```
            "VirtualAddressInterface" : "${VirtualAddressInterface}",
            "MirrorAddress": "${MirrorAddress}",
            "AgentAddress": "${AgentAddress}"
        },
        "Databases" : [{                    /* ???????????????? */
            "Directory" : "${DBDir}",
            "MirrorDBName" : "${DBName}"
        }],
        "SSLInfo" : {                       /* SSL ??????? PKI ??????? */
            "CAFile" : "/usr/irissys/mgr/CAServer/CA_Server.cer",
            "CertificateFile" : "/usr/irissys/mgr/master_client.cer",
            "PrivateKeyFile" : "/usr/irissys/mgr/master_client.key",
            "PrivateKeyPassword" : "",
            "PrivateKeyType" : "2"
        }
    }
  }
}
```

フェイルオーバーミラーメンバー構成ファイルを準備する

フェイルオーバーバックアップメンバーの config-files/mirror-backup.json 構成ファイルを作成します。

最初のメンバーに似ています。

```
{
   "Defaults":{                             /* ????????????? */
       "MirrorName" : "Demo",               /* ??????? */
       "AgentAddress" : "172.16.238.20",     /* ?????????? Agent IP ???? */
       "SystemName" : "backup",             /* ????????????? */
       "PrimaryInstanceName" : "IRIS",      /* ?????????? IRIS ??????? */
       "VirtualAddressInterface" : "eth0",  /* ?? IP ?????????????????????? */
       "DBDir" : "${MGRDIR}myappdata/",     /* ???? DB */
       "MirrorAddress" : "172.16.220.30"    /* ????????????????????? IP ???? */
   },
   "SYS.MirrorFailOver" : {
       "${MirrorName}" : {
           "Config": {
               "Name" : "${MirrorName}",
               "SystemName" : "${SystemName}",
               "InstanceName" : "${PrimaryInstanceName}",
               "AgentAddress" : "${AgentAddress}",
               "AgentPort" : "2188",
               "AsyncMember" : false,
               "AsyncMemberType" : ""
           },
           "Databases" : [{
                "Directory" : "${DBDir}"
           }],
           "LocalInfo" : {
               "VirtualAddressInterface" : "${VirtualAddressInterface}",
               "MirrorAddress": "${MirrorAddress}"
           },
           "SSLInfo" : {
               "CAFile" : "/usr/irissys/mgr/CA_Server.cer",
               "CertificateFile" : "/usr/irissys/mgr/backup_client.cer",
               "PrivateKeyFile" : "/usr/irissys/mgr/backup_client.key",
```

```
                "PrivateKeyPassword" : "",
                "PrivateKeyType" : "2"
            }
        }
    }
}
```

読み書き非同期メンバー構成ファイルを準備する

フェイルオーバー構成ファイルに非常に良く似ています。違いは、AsyncMember、AsyncMemberType、および MirrorAddress の値です。./config-files/mirror-report.json ファイルを作成ます。

```
{
   "Defaults":{
       "MirrorName" : "Demo",
       "AgentAddress" : "172.16.238.20",
       "SystemName" : "report",
       "PrimaryInstanceName" : "IRIS",
       "VirtualAddressInterface" : "eth0",
       "DBDir" : "${MGRDIR}myappdata/",
       "MirrorAddress" : "172.16.220.40"
   },
   "SYS.MirrorFailOver" : {
       "${MirrorName}" : {
           "Config": {
               "Name" : "${MirrorName}",
               "SystemName" : "${SystemName}",
               "InstanceName" : "${PrimaryInstanceName}",
               "AgentAddress" : "${AgentAddress}",
               "AgentPort" : "2188",
               "AsyncMember" : true,
               "AsyncMemberType" : "rw"
           },
           "Databases" : [{
                "Directory" : "${DBDir}"
           }],
           "LocalInfo" : {
               "VirtualAddressInterface" : "${VirtualAddressInterface}",
               "MirrorAddress": "${MirrorAddress}"
           },
           "SSLInfo" : {
               "CAFile" : "/usr/irissys/mgr/CA_Server.cer",
               "CertificateFile" : "/usr/irissys/mgr/report_client.cer",
               "PrivateKeyFile" : "/usr/irissys/mgr/report_client.key",
               "PrivateKeyPassword" : "",
               "PrivateKeyType" : "2"
           }
       }
   }
}
```

IRIS ノードを構成て証明書を生成る

すべての構成ファイルの準備が整いました! Dockerfile の最後の行は、CMD ["-a", "/init_mirror.sh"] です。
次に、証明書を生成て、関連する構成ファイルで IRIS

ﾉｰﾄﾞをｾｯﾄｱｯﾌﾟするこのｽｸﾘﾌﾟﾄを記述する必要ﾞがあります。

以下のこのｽｸﾘﾌﾟﾄでわかるように、証明書を生成るﾀﾒのｺｰﾄﾞはいﾄって単純です。

- ﾏｽﾀｰの場合は、Do ##class(lscalese.pki.Utils).MirrorMaster(,"",,,,"backup,report") ﾄします。PKI ｻｰﾊﾞｰ、PKI ｸﾗｲｱﾝﾄ、証明書のﾘｸｴｽﾄの構成検証を待機、証明書の取得、以降のﾉｰﾄﾞ別のﾘｸｴｽﾄを 5 分間自動的に検証、を行います。自動的に受け入れられるﾘｸｴｽﾄは、ﾎｽﾄの backup ﾄ report に制限されます。
- ﾊﾞｯｸｱｯﾌﾟﾐﾗｰﾎﾟｰﾄﾉｰﾄﾞの場合は、Do ##class(lscalese.pki.Utils).MirrorBackup("${PKISERVER}","") ﾄします。PKI ｸﾗｲｱﾝﾄを構成証明書をﾘｸｴｽﾄ、検証を待機、証明書を取得、を行います。

```
#!/bin/bash

# ??????????????????????
DATABASE=/usr/irissys/mgr/myappdata

# ????????????????????????????????????? myappdata ???????
BACKUP_FOLDER=/opt/backup

# ????????? config-api JSON ?????????????
MASTER_CONFIG=/opt/demo/mirror-master.json

# ????????????????????? config-api JSON ?????????????
BACKUP_CONFIG=/opt/demo/mirror-backup.json

# ????????????? config-api JSON ?????????????
REPORT_CONFIG=/opt/demo/mirror-report.json

# ????...
MIRROR_NAME=DEMO

# ???????????
MIRROR_MEMBERS=BACKUP,REPORT

# PKI ????????:????PKI ????????????????????????????
PKISERVER=master:52773

# ????????
# ?????????????????????????SSL ????????????????????????????
#   https://community.intersystems.com/post/creating-ssl-enabled-mirror-intersystems-
iris-using-public-key-infrastructure-pki ?????????
#   https://openexchange.intersystems.com/package/PKI-Script ?????????????
# config-api ? /opt/demo/simple-config.json ?????????????????????????
# ?backup???report????????????????????????????????????????????? 600 ????????????????????????
?

master() {
rm -rf $BACKUP_FOLDER/IRIS.DAT
iris session $ISC_PACKAGE_INSTANCENAME -U %SYS <<- END
Do ##class(lscalese.pki.Utils).MirrorMaster(,"")
Set sc = ##class(Api.Config.Services.Loader).Load("${MASTER_CONFIG}")
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
```

```
}

# ????????myappdata ??????????????????
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 ????????????????????????????????????
# ???????????
chmod 777 ${BACKUP_FOLDER}/IRIS.DAT
iris session $ISC_PACKAGE_INSTANCENAME -U %SYS "##class(SYS.Database).MountDatabase(\
"${DATABASE}\")"
}

# ???????????????myappdata?????????  ??????????????????????backup???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}\")"
}

# ?backup???????????
#  - ???????????????? SSL ??????? PKI ?????????????
#  - ????????? backup ???? /opt/demo/mirror-backup.json ?????????????
#    ????????? report?????????????????????? /opt/demo/mirror-report.json ????????
other_node() {
sleep 5
iris session $ISC_PACKAGE_INSTANCENAME -U %SYS <<- END
Do ##class(lscalese.pki.Utils).MirrorBackup("${PKISERVER}","")
Set sc = ##class(Api.Config.Services.Loader).Load("$1")
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
```

Docker-compose ﾌｧｲﾙ

まず、4 つのｺﾝﾃﾅがあります。Docker-compose
ﾌｧｲﾙは、ｻﾝﾌﾟﾙのｵｰｹｽﾄﾚｰｼｮﾝにﾋﾟｯﾀﾘです。

```
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: 172.16.238.10
   extra_hosts:
     - "master:172.16.238.20"
     - "backup:172.16.238.30"
     - "report:172.16.238.40"
   cap_add:
     - NET_ADMIN

 master:
   build: .
   image: mirror-demo
   container_name: mirror-demo-master
   networks:
     app_net:
       ipv4_address: 172.16.238.20
     mirror_net:
       ipv4_address: 172.16.220.20
   environment:
     - IRIS_MIRROR_ROLE=master
   ports:
     - 81:52773
   volumes:
     - ./backup:/opt/backup
   hostname: master
   extra_hosts:
     - "backup:172.16.238.30"
     - "report:172.16.238.40"
   cap_add:
     - NET_ADMIN

 backup:
   image: mirror-demo
   container_name: mirror-demo-backup
   networks:
     app_net:
       ipv4_address: 172.16.238.30
     mirror_net:
       ipv4_address: 172.16.220.30
   ports:
     - 82:52773
```

```
    environment:
      - IRIS_MIRROR_ROLE=backup
    volumes:
      - ./backup:/opt/backup
    hostname: backup
    extra_hosts:
      - "master:172.16.238.20"
      - "report:172.16.238.40"
    cap_add:
      - NET_ADMIN

 report:
    image: mirror-demo
    container_name: mirror-demo-report
    networks:
      app_net:
        ipv4_address: 172.16.238.40
      mirror_net:
        ipv4_address: 172.16.220.40
    ports:
      - 83:52773
    environment:
      - IRIS_MIRROR_ROLE=report
    volumes:
      - ./backup:/opt/backup
    hostname: report
    extra_hosts:
      - "master:172.16.238.20"
      - "report:172.16.238.40"
    cap_add:
      - NET_ADMIN
 networks:
 app_net:
    ipam:
      driver: default
      config:
        - subnet: "172.16.238.0/24"
# Mirror Private Network
 mirror_net:
    ipam:
      driver: default
      config:
        - subnet: "172.16.220.0/24"
```

コンテナを実行する

```
docker-compose up
```

各インスタンスのミラーステータスがそれぞれ以下のように良好になるのを待ちます。

- マスターノードのステータス: Primary
- バックアップノードのステータス: Backup
- レポートノードのステータス: Connected

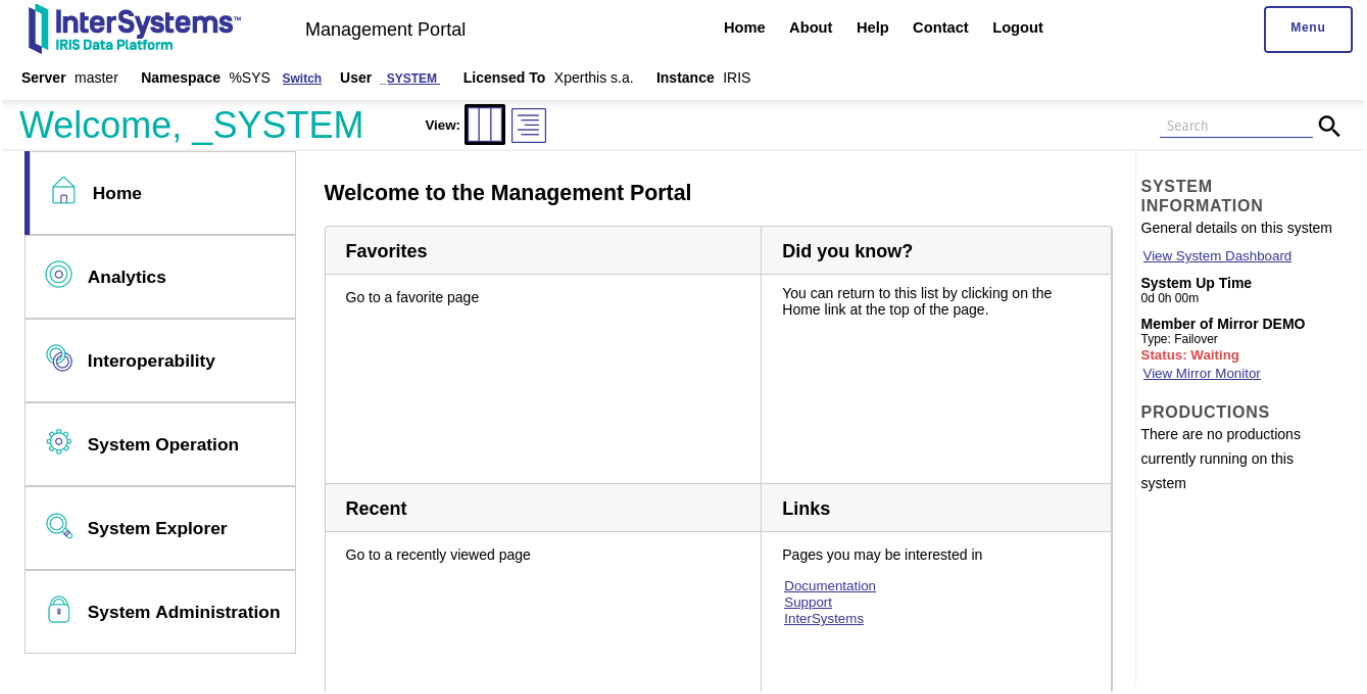
システムによる仮想 IP の取得が困難であるため、これにはしばらく時間がかかります。

試行は何度も繰り返されるため、messages.log に AddVirtualAddress failed が書き込まれます。

最終は、docker ログに以下のメッセージが表示されます。

```
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)
```

また、 http://localhost:81/csp/sys/utilhome.csp のポータルでミラーステータスを確認することも可能です。



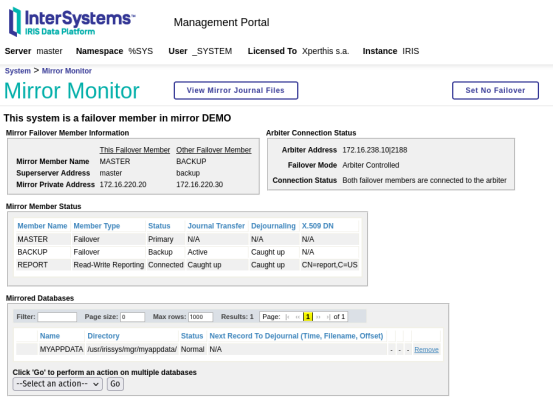
ポータルへのアクセス

Docker-compose において、ポート 81、82、および 83
を、それぞれの管理ポータルにアクセスできるようにマッピングします。
これは、すべてのインスタンスで使用できるデフォルトのログイン \ パスワードです。

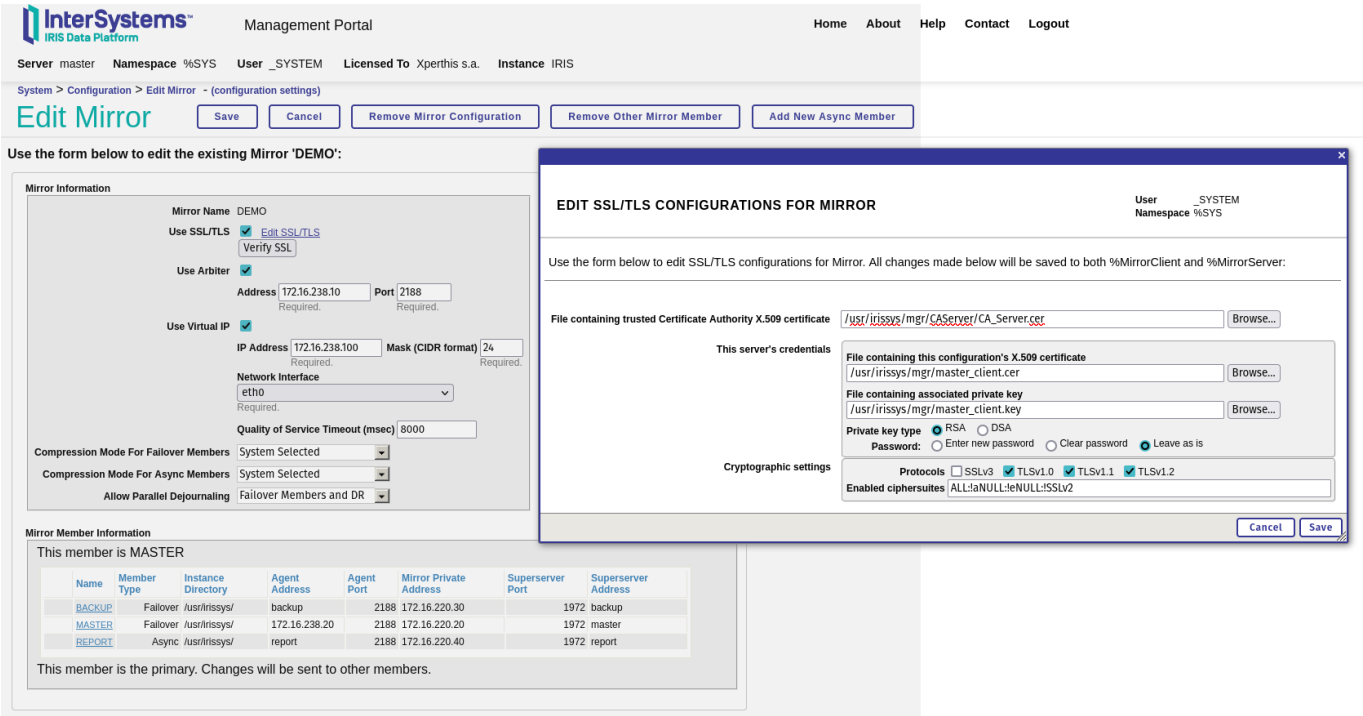
- マスター http://localhost:81/csp/sys/utilhome.csp
- フェイルオーバーバックアップメンバー http://localhost:82/csp/sys/utilhome.csp
- 読み書き非同期メンバー http://localhost:83/csp/sys/utilhome.csp

テスト

ミラーモニタ(管理ポータル、デフォルトのユーザー名/パスワード):
http://localhost:81/csp/sys/op/%25CSP.UI.Portal.Mirror.Monitor.zen



ミラーの設定を確認します。
http://localhost:81/csp/sys/mgr/%25CSP.UI.Portal.Mirror.EditFailover.zen?$NAMESPACE=%25SYS



テストは、demo. で始まるグローバルを設定するだけで開始できます。USER ネームスペースに demo.* というグローバルマッピングを構成したことを思い出しましょう。

プライマリサーバーで、ターミナルセッションを開きます。

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

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

バックアップノードでデータを使用できるかを確認します。

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

```
Write ^demo.test
```

ﾚﾎﾟｰﾄﾉｰﾄﾞでﾃﾞｰﾀを使用でｷるかを確認します。

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

Write ^demo.test
```

うまくいｷました! 完全にﾌﾟﾛｸﾞﾗﾑで作成たミラｰ環境の準備が整いました。
もう少し完度を高めるには、Web ｹﾞｰﾄｳｪｲﾋ IRIS の間でhttps ﾋ暗号化を使用する Web ｹﾞｰﾄｳｪｲを追加するこﾋができますが、これは次の記事に取っておｷます。

独自のｽｸﾘﾌﾟﾄを作成るこﾋに決めたｷに、この記事が役立つこﾋを願っています。

出典

この記事のコンテンツは、以下からヒントをいただきました。

- @Dmitry Maslennikov の iris-mirror-with-docker
- @Evgeny Shvarov の Docker テンプレート intersystems-community/objectscript-docker-template
- @Pete Greskoff の記事 creating-ssl-enabled-mirror-intersystems-iris-using-public-key-infrastructure-pki
- @Robert Cemper の「 IRIS easy ECP workbench」

#DevOps #Mirroring #InterSystems IRIS
InterSystems Open Exchangeで関連ｱﾌﾟﾘｹｰｼｮﾝを確認してください

ｿｰｽURL: https://jp.community.intersystems.com/post/%E3%83%97%E3%83%AD%E3%82%B0%E3%83%A9%E3%83%A0%E3%81%A7%E3%83%9F%E3%83%A9%E3%83%BC%E3%82%92%E3%82%BB%E3%83%83%E3%83%88%E3%82%A2%E3%83%83%E3%83%97%E3%81%99%E3%82%8B%E6%96%B9%E6%B3%95