Article
Lorenzo Scalese · Jul 21, 2022 11m read

ECP With Docker

Hi community,

This is the third article in the series about initializing IRIS instances with Docker. This time, we will focus on **E** nterprise **C**ache **P**rotocol (ECP).

In a very simplified way, ECP allows configuring some IRIS instances as application servers and others as data servers. Detailed technical information can be found in the official documentation.

This article aims to describe:

- How to script the initialization of a data server, and how to script the initialization of one or more application servers.
- How to establish an encrypted connection between these nodes with Docker.

To do this, we typically use some of the tools we have already seen in previous web gateway, and mirroring articles describing such instruments as OpenSSL, envsubst, and Config-API.

Requirements

ECP is not available with the IRIS Community version. Therefore, a World Response Center access is required to download a container license and to connect to the containers.intersystems.com registry.

Preparing the system

The system must share some local files with the containers. It is necessary to create certain users and groups to avoid the "access denied" error.

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

If you don ' t have the license " iris.key " yet, download it from WRC, and add it in your home directory.

Retrieve the sample repository

All the files you need are available on a public repository except the license " iris.key " , so start by cloning it:

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

SSL Certificates

In order to encrypt communications between the application servers and the data server, we need SSL certificates. A ready-to-use script ("gen-certificates.sh") is available. However, feel free to modify it so that the certificate settings are consistent with your location, company, etc.

Execute:

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

The generated certificates are now in the "./certificates" directory.

File	Container	Description
./certificates/CAServer.cer	Application server and data server	Authority server certificate
./certificates/appserver.cer	Application server	Certificate for IRIS application server instance
./certificates/appserver.key	Application server	Related private key
./certificates/dataserver.cer	Data server	Certificate for IRIS data server instance
./certificates/dataserver.key	Data server	Related private key

Build the image

First of all, log in to the Intersystems docker registry. The base image will be downloaded from the registry during the build:

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

If you don't know your token, log on to https://containers.intersystems.com/ with your WRC account.

During this build, we will add some software utilities to the IRIS base image:

- **gettext-base**: it will allow us to substitute environment variables in our configuration files with the "envsubst" command.
- **iputils-arping**: it is required in case we want to mirror the data server.
- **ZPM**: ObjectScript package manager.

Dockerfile:

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

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

FROM $IMAGE

USER root

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

```
USER ${ISC_PACKAGE_MGRUSER}

WORKDIR /home/irisowner/demo

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

There is nothing special in this Dockerfile, except the last line. It configures the IRIS data server instance to accept up to 3 application servers. Beware, this configuration requires a restart of IRIS. We assign the value of this parameter during the build to avoid having to script a restart later.

Start the build:

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

The configuration files

For the configuration of the IRIS instances (application servers and data server) we use JSON config-api file format. You will notice that these files contain environment variables "${variablename}". Their values are defined in the "environment" sections of the “ docker-compose.yml ” file that we will see later in this document. These variables will be substituted just before loading the files using the "envsubst" utility.

Data server

For the data server, we will:

- Enable the ECP service and define the list of authorized clients (application servers).
- Create the "SSL %ECPServer" configuration necessary for encrypting communications.
- Create a database "myappdata". This will be used as a remote database from the application servers.

(data-serer.json)[https://github.com/lscalese/ecp-with-docker/blob/master/config-files/dat...

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

```
        "SSLECPServer":1
    },
    "SYS.Databases":{
        "/usr/irissys/mgr/myappdata/" : {}
    },
    "Databases":{
        "myappdata" : {
            "Directory" : "/usr/irissys/mgr/myappdata/"
        }
    }
}
```

This configuration file is loaded at the start of the data server container by the "initdatasrv.sh" script. All application servers that are connected to the data server must be trusted. This script will automatically validate all connections within 100 seconds to limit manual actions in the administration portal. Of course, this can be improved to enhance security.

Application server

For the application servers, we will:

- Enable the ECP service.
- Create the SSL configuration "%ECPClient" required for communication encryption.
- Configure the connection information to the data server.
- Create the configuration of the remote database "myappdata".
- Create a global mapping "demo.*" in the "USER" namespace to the "myappdata" database. This will allow us to test the operation of ECP later.

app-server.json:

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

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

The configuration file is loaded at the start of an application server container by the script "initappsrv.sh ".

Starting the containers

Now, we can start the containers:

- 2 application servers.
- 1 data server.

To do this, run:

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

See the docker-compose file for details:

```
# Variables are defined in .env file
# to show the resolved docker-compose file, execute
# docker-compose config

version: '3.7'

services:
  ecp-demo-data-server:
    build: .
    image: ecp-demo
    container_name: ecp-demo-data-server
    hostname: data-server
    networks:
      app_net:
    environment:
      # List of allowed ECP clients (application server).
      - CLIENT_SYSTEMS=ecp-with-docker_ecp-demo-app-server_1;ecp-with-docker_ecp-demo-
app-server_2;ecp-with-docker_ecp-demo-app-server_3
      # Path authority server certificate
      - CA_ROOT=/certificates/CA_Server.cer
      # Path to data server certificate
      - CA_SERVER=/certificates/data_server.cer
      # Path to private key of the data server certificate
      - CA_PRIVATE_KEY=/certificates/data_server.key
      # Path to Config-API file to initiliaze this IRIS instance
      - IRIS_CONFIGAPI_FILE=/home/irisowner/demo/data-server.json
    ports:
```

```
    - "81:52773"
  volumes:
    # Post start script - data server initilization.
    - ./init_datasrv.sh:/home/irisowner/demo/init_datasrv.sh
    # Mount certificates (see gen-certificates.sh to generate certificates)
    - ./certificates/app_server.cer:/certificates/data_server.cer
    - ./certificates/app_server.key:/certificates/data_server.key
    - ./certificates/CA_Server.cer:/certificates/CA_Server.cer
    # Mount config file
    - ./config-files/data-server.json:/home/irisowner/demo/data-server.json
    # IRIS License
    - ~/iris.key:/usr/irissys/mgr/iris.key
  command: -a /home/irisowner/demo/init_datasrv.sh


ecp-demo-app-server:
  image: ecp-demo
  networks:
    app_net:
  environment:
    # Hostname or IP of the data server.
    - DATASERVER_IP=data-server
    - DATASERVER_NAME=data-server
    - DATASERVER_PORT=1972
    # Path authority server certificate
    - CA_ROOT=/certificates/CA_Server.cer
    - CA_CLIENT=/certificates/app_server.cer
    - CA_PRIVATE_KEY=/certificates/app_server.key
    - IRIS_CONFIGAPI_FILE=/home/irisowner/demo/app-server.json
  ports:
    - 52773
  volumes:
    # Post start script - application server initilization.
    - ./init_appsrv.sh:/home/irisowner/demo/init_appsrv.sh
    # Mount certificates
    - ./certificates/CA_Server.cer:/certificates/CA_Server.cer
    # Path to private key of the data server certificate
    - ./certificates/app_server.cer:/certificates/app_server.cer
    # Path to private key of the data server certificate
    - ./certificates/app_server.key:/certificates/app_server.key
    # Path to Config-API file to initiliaze this IRIS instance
    - ./config-files/app-server.json:/home/irisowner/demo/app-server.json
    # IRIS License
    - ~/iris.key:/usr/irissys/mgr/iris.key
  command: -a /home/irisowner/demo/init_appsrv.sh
networks:
app_net:
  ipam:
    driver: default
    config:
      # APP_NET_SUBNET variable is defined in .env file
      - subnet: "${APP_NET_SUBNET}"
```

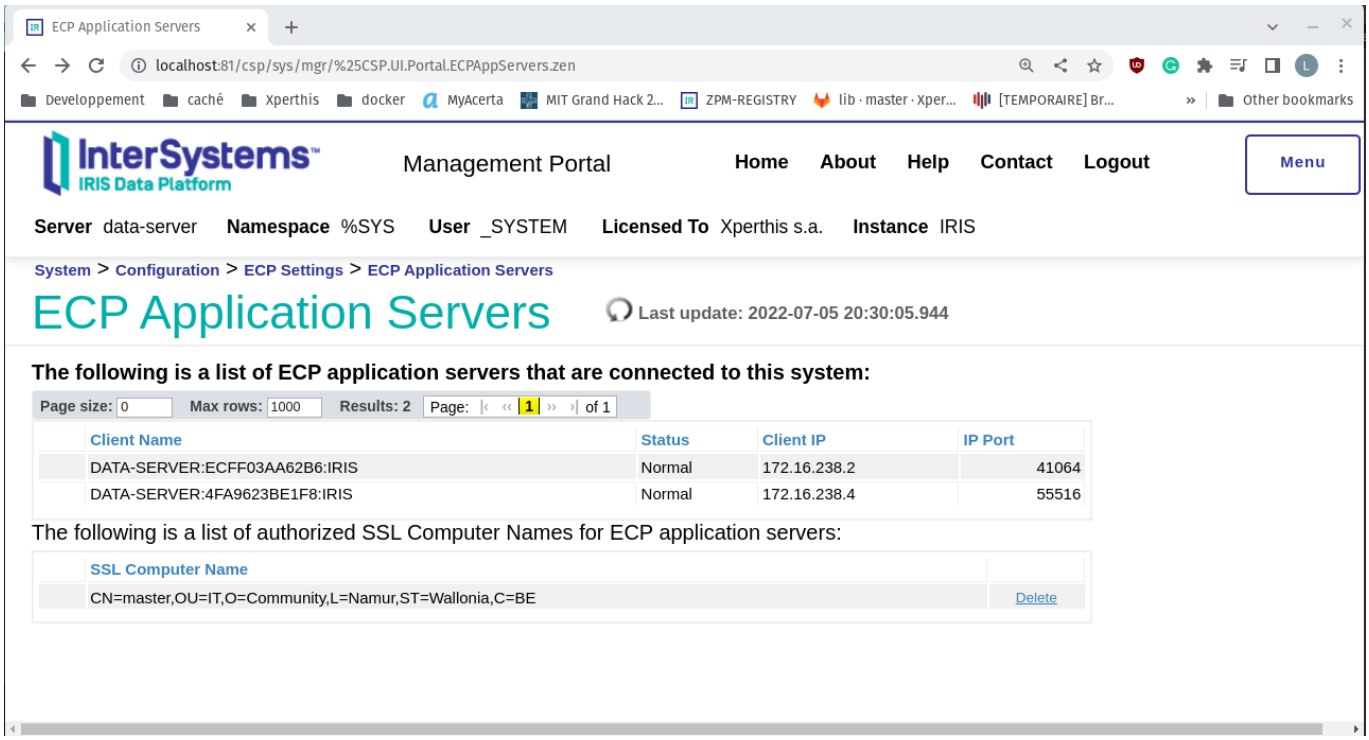
Let's test it!

Access to the data server administration portal

The containers have been started. Let's check the status from the data server.

Port 52773 is mapped to local port 81, so you can access it with this address
http://localhost:81/csp/sys/utilhome.csp

Log in with the default login\password, and then go to System -> Configuration -> ECP Params. Click on "ECP Application Servers". If everything works fine, you should see 2 application servers with the status "Normal". The structure of the client name is "data server name":"application server hostname":"IRIS instance name". In our case, we have not set the application server hostnames, so we have auto-generated hostnames.



Accessing the application server administration portal

To connect to the application servers ' administration portal, firstly, you need to get the port number. Since we used the "--scale" option, we could not set the ports in the docker-compose file. So you have to retrieve them with the command docker ps:

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

In this example, the ports are:

- 49170 for the first application server http://localhost:49170/csp/sys/utilhome.csp
- 49169 for the second application server http://localhost:49169/csp/sys/utilhome.csp

Read/write test on the remote database

Let's perform some read/write tests in the terminal.
Open an IRIS terminal on the first application server:

```
docker exec -it ecp-with-docker_ecp-demo-app-server_1 iris session iris
Set ^demo.ecp=$zdt($h,3,1) _ “ write from the first application server.”
```

Now open a terminal on the second application server:

```
docker exec -it ecp-with-docker_ecp-demo-app-server_2 iris session iris
Set ^demo.ecp(2)=$zdt($h,3,1) _ " write from the second application server."
zwrite ^demo.ecp
```

You should see the responses from both servers:

```
^demo.ecp(1)="2022-07-05 23:05:10 write from the first application server."
^demo.ecp(2)="2022-07-05 23:07:44 write from the second application server."
```

Finally, open an IRIS terminal on the data server and perform a read of the global demo.ecp:

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

^["^^/usr/irissys/mgr/myappdata/"]demo.ecp(1)="2022-07-05 23:05:10 write from the fir
st application server."
^["^^/usr/irissys/mgr/myappdata/"]demo.ecp(2)="2022-07-05 23:07:44 write from the sec
ond application server."
```

That's all for today. I hope you have enjoyed this article. Do not hesitate to leave your comments.

#Deployment #DevOps #InterSystems IRIS

Source URL:https://community.intersystems.com/post/ecp-docker