Article
David E Nelson · Apr 5, 2019 6m read

Docker for Windows and the InterSystems IRIS Data Platform

Now that the InterSystems IRIS Data Platform Community Edition is available on Docker Hub (https://hub.docker.com//intersystems-iris-data-platform), it seems like a great time to try InterSystems IRIS in a container. For some time already, the community edition has been available in the cloud, for example on AWS (https://community.intersystems.com/post/free-iris-community-edition-aws), but maybe it would be nice to try it locally as well. Fellow Windows users are no doubt used to eye rolling, being told " YMMV ", etc., whenever they mention using Docker for Windows. Sometimes we are even told that we should really consider running Docker inside Ubuntu virtual machines. Ugh

Now Docker and Windows, especially when using Linux-based containers (the only kind supported by InterSystems IRIS), are not an ideal match. For more on why this is so, see: https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/linux-containers). Nevertheless, with some patience, a thirst for adventure, and a couple of extra steps it seems to be possible to run InterSystems IRIS in a Docker container on Windows. This configuration, while emphatically not supported for production, seems to work well enough to try out many InterSystems IRIS features. For example, I have created namespaces and databases, loaded data, created productions, and even executed some SPARK queries.

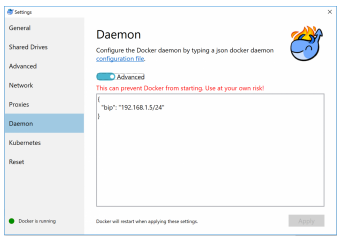
In the following, I will list the steps that I took to launch the InterSystems IRIS Data Platform Community Edition. I will show both a Docker command-line approach as well as the more convenient compose file approach. Fair warning: I am no Docker expert and we are talking about Docker for Windows here, so YMMV. ;-)

Setup

1. Windows 10 Pro
2. Docker Desktop Community Edition, Version 0.0.3 (31259) Channel: stable, available here: https://hub.docker.com/editions/community/docker-ce-desktop-windows

Some Docker Configuration

1. I changed the " bip " on my Daemon settings to prevent network conflicts.



2. I set my C:\drive as a shared drive. This will allow the container to access files on C:\ 

Digression on Named Volumes and Bind Mounts

Containers are ephemeral. We start them. We stop them. We delete them. Often, however, we would like data generated and used by the container to persist. Docker provides two approaches for persisted container data: volumes and bind mounts. InterSystems IRIS containers include the durable %SYS feature for data generated and used by the data platform. On Docker for Windows, durable %SYS must use a Docker volume. When using Docker on other operating systems, for example Docker for the Mac or Docker for Ubuntu, durable %SYS can use either a bind mount or a volume.

Volumes are entirely managed by Docker. Unlike bind mounts, they do not correspond directly to any file or directory on the file system. We must use Docker commands to create, delete, and inspect them. The fact that we cannot tune or use advanced products and tools to manage the data in Docker volumes, is a big drawback for using Docker on Windows. Clearly, it is not a production environment.

Creating a Named Volume

My next step was to create a named volume at the Windows command prompt:

C:\>docker volume create durable

To see the details for this volume I used the docker volume inspect command. When I want to delete it, I will use the docker volume rm command.

Launching the Container

At last, I was ready to launch the container. Here is the docker run command:

C:\>docker run –it –d –p 51773:51773 –p 52773:52773 –-name iris –-volume durable:/durable –-env ISCDATADIRECTORY=/durable/irissys store/intersystems/iris:2019.1.0.510.0-community

A little about the options passed to the run command:

Option	Description
it	Roughly, means run in interactive mode and provide access via a terminal
d	Means run in detached mode
p	Maps ports inside the container to ports outside the container on the host. We will use these ports from Windows to access InterSystems IRIS. Note that if you have IRIS installed locally, it is likely using ports 51773 and 52773 and you will get port conflict if you try to map these ports

volume	Maps the named volume created above to a directory inside the container.
env	Sets environment variables. In this case we are setting the location of the durable %SYS to be our durable directory inside the container.

I verified that all was well with the container using the following command:

C:\>docker container ls

It listed all the running containers on my system. The first time I ran the command, it listed my iris container ' s status as " starting " . When I ran it again after a little time had passed, it listed the container ' s status as " healthy " .

Accessing InterSystems IRIS

With my container running and healthy, the next step was to access the data platform. I did this first by launching the Management Portal using the usual url: http://localhost:52773/csp/sys/UtilHome.csp

The page asked for a username/password. I entered SuperUser/SYS and then I was directed to a password change page. I then had full access to the Management Portal.

Next, I tried out command line access. To do this, I first launched a bash shell inside the container by entering the following command in the Windows command prompt:

C:\>docker exec –it iris bash

This opened the bash shell. From the bash shell, I opened an IRIS terminal session with the following:

>iris session iris

I logged in using the username/password set above.

Stopping the Container

Finally, to stop the container I entered the following at the Windows command prompt:

C:\>docker container stop iris

Switching to Compose

Using the Docker run command with all those options to type is a bit tedious. Fortunately, Docker supports placing the options into a compose file and then launching a container using docker-compose commands to read the options from the file and to launch the container appropriately.

To use this approach, first create a Windows directory. I created a directory called iriscommunity. Note that the name of the directory becomes the name of the Docker project. Inside iriscommunity, I created another directory called local. My compose file will create a bind mount using this directory and I can use it to pass data and code between my Windows file system and my container. I then placed my compose file, named docker-compose.yml inside iriscommunity. So, the directory structure looks like this

iriscommunity

local

docker-compose.yml

Here is the contents of my docker-compose.yml. Hopefully, the comments are enough to explain what it is doing.

```
version: '3.2'

services:
  iris:
    image: store/intersystems/iris:2019.1.0.510.0-community

    container_name: iris-community

    ports:
    # 51773 is the superserver default port
    - "51773:51773"
    # 52773 is the webserver/management portal port
    - "52773:52773"

    volumes:
    # Maps directory in containr, named durable, to named volume defined below
    - durable:/durable
    # Mounts a local directory for passing in files and test scripts
    - ./local:/Samples/local

    environment:
    # Uses ISC_DATA_DIRECTORY to place the durable %SYS in the named volume
    - ISC_DATA_DIRECTORY=/durable/irissys

    # Creates the named docker volume
volumes:
  durable:
```

Launching with docker-compose

To launch the container, I opened a Windows command prompt in my iriscommunity directory and entered the following:

C:\iriscommunity>docker-compose up

This time I did not set the detach option, so I see a stream of output from docker. Note that the named volume created in this case will be iriscommunitydurable so it will not conflict with the earlier volume.

To stop the container, I use the following command:

C:\iriscommunity>docker-compose down

#Beginner #Cloud #Containerization #Docker #Microsoft Windows #InterSystems IRIS