Article
Bob Binstock · May 16, 2018 6m read

Using InterSystems IRIS Containers with Docker for Windows

InterSystems supports use of the InterSystems IRIS Docker images it provides on Linux only. Rather than executing containers as native processes, as on Linux platforms, Docker for Windows creates a Linux VM running under Hyper-V, the Windows virtualizer, to host containers. These additional layers add complexity that prevents InterSystems from supporting Docker for Windows at this time.

We understand, however, that for testing and other specific purposes, you may want to run InterSystems IRIS-based containers from InterSystems under Docker for Windows. This article describes the differences between Docker for Windows and Docker for Linux that InterSystems is aware of as they apply to working with InterSystems-provided container images. Other, unanticipated issues may arise. When using InterSystems IRIS images and containers on a Windows platform, ensure that you have access to the Docker documentation for convenient reference; see in particular Getting started with Docker for Windows

Because handling by a container of external persistent storage under Docker for Windows involves both the Windows and Linux file systems and file handling, the differences noted are largely storage-related.

For general information about running InterSystems IRIS in Docker containers using image provided by InterSystems, see Running InterSystems IRIS in Docker Containers and First Look: InterSystems IRIS in Docker Containers.

Share Disk Drives

On Windows, you must give Docker access to any storage with which it interacts by sharing the disk drive on which it is located. To share one or more drives, follow these steps (you can combine this with the procedure in the previous item):

1. Right-click the Docker icon in the system tray and select **Settings ...** .
2. Choose the **Shared Drives** tab, then select the drive(s) on which the storage is located and click **Apply**. If a drive is already selected (the C drive is selected by default), clear the checkbox and click **Apply**, then select it and click **Apply** again.
3. Enter your login credentials when prompted.
4. Docker automatically restarts after applying the changes; if it does not, right-click the Docker icon in the system tray and select **Restart**.

Copy External Files Within the Container

When using Docker, it is often convenient to mount a directory in the external file system as a volume inside the container, and use that as the location for all the external files needed by the software inside the container. For example, you might mount a volume and place the InterSystems IRIS licence key, **iris.key**, and a file containing the intended InterSystems IRIS password in the external directory for access by the **--key** option of **iris-main** program and the password change script, respectively (see The iris-main Program and Changing the InterSystems IRIS Password in *Running InterSystems IRIS in Containers*). Under Docker for WIndows, however, file-handling and permissions differences sometimes prevent a file on a mounted external volume from being used properly by a program in the container. You can often overcome permissions difficulties by having the program copy the file within the container and then use the copy.

For example, the **iris-main --before** option is often used to change the password of the InterSystems IRIS instance in the container, for example:

--before "changePassword.sh /external/password.txt"

If this fails to change the password as intended on Windows, try the following:

--before "cp /external/password.txt /external/passwordcopied.txt && \
changePassword.sh /external/passwordcopied.txt"

Use Named Volumes

When numerous dynamic files are involved, any direct mounting of the Windows file system within a container is likely to lead to problems, even if individual mounting and copying of all the files were feasible. In the case of InterSystems IRIS, this applies in particular to both the durable %SYS feature for persistent storage of instance-specific data (see Durable %SYS for Persistent Instance Data in *Running InterSystems IRIS in Containers)* and journal file storage. You can overcome this problem by mounting a named volume, which is a storage volume with a mount point in the filesystem of the Linux VM hosting the containers on your system. Because the VM is file system-based, the contents of such a volume are saved to the Windows disk along with the rest of the VM, even when Docker or your system goes down.

For example, the standard way to enable durable %SYS when running an InterSystems IRIS container is to mount an external volume and use the **--env** option to set the **ISCDATADIRECTORY** environment variable to a location on that volume, for example:

docker run ... \
--volume /nethome/pmartinez/irisexternal:/external \
--env ISCDATADIRECTORY=/external/durable/

This will not work with Docker for Windows; you must instead create a named volume with the **docker volume create** command and locate the durable %SYS directory there. Additionally, you must include the top level of the durable %SYS directory, **/irissys**, in the **ISCDATADIRECTORY** specification, which is not the case on Linux. On Windows, therefore, your options would look like this:

docker volume create durable
docker run ... \
--volume durable:/durable \
--env ISCDATADIRECTORY=/durable/irissys/

To use this approach for the instance's journal files, create and mount a named volume, as in the durable %SYS example above, and then use any configuration mechanism (the **^JOURNAL** routine, the Journal Settings page in the Management Portal, or the **iris.cpf** file) to set the current and alternate journal directories to locations on the named volume. To separate the current and alternate journal directories, create and mount a named volume for each. (Note that this approach ***has not been thoroughly tested*** and that journal files under this scheme may therefore be at risk.)

To discover the Linux file system mount point of a named volume within the VM, you can use the **docker volume inspect** command, as follows:

```
docker volume inspect durabledata
[
  {
  "CreatedAt": "2018-05-04T12:11:54Z",
  "Driver": "local",
  "Labels": null,
  "Mountpoint": "/var/lib/docker/volumes/durabledata/data",
  "Name": "durabledata",
  "Options": null,
  "Scope": "local"
  }
]
```

Command Comparison

Taking all of the preceding items together, the following presents a comparison between the final **docker run** command described Run and Investigate the InterSystems IRIS-based Container in *First Look: InterSystems IRIS in Docker Containers*, which is intended to be executed on a Linux platform, and the equivalent **docker run** command using Docker for Windows.

Linux

```
$ docker run --name iris3 --detach --publish 52773:52773 \
  --volume /nethome/pmartinez/irisexternal:/external \
  --env ISCDATADIRECTORY=/external/durable \
  --env ICMSENTINELDIR=/external iris3:test --key /external/iris.key \
  --before "changePassword.sh /external/password.txt"
```

Windows

```
C:\Users\pmartinez>docker volume create durable
C:\Users\pmartinez>docker volume create journals
C:\Users\pmartinez>docker run --name iris3 --detach --publish 52773:52773 \
--volume durable:/durable\
--volume journals:/journals
--env ISCDATADIRECTORY=/durable/irissys \
--env ICMSENTINELDIR=/durable iris3:test --key /external/iris.key \
--before "cp /external/password.txt /external/passwordcopied.txt && \
changePassword.sh /durable/passwordcopied.txt"
```

If you have any information to contribute about using InterSystems-provided containers with Docker for Windows, please add as a comment here, or post your own article!

#Best Practices #Containerization #Deployment #Docker #Microsoft Windows #InterSystems IRIS

Source URL:https://community.intersystems.com/post/using-intersystems-iris-containers-docker-windows