First Look: Docker Containers and InterSystems IRIS

Version 2018.1
2018-01-31

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

First Look: Docker Containers and InterSystems IRIS
InterSystems Version 2018.1 2018-01-31

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

First Look: Docker Containers and InterSystems IRIS

This First Look guide is intended to introduce you to the fundamentals of using Docker containers with InterSystems IRIS by giving you a focused overview and a basic, hands-on example. You will learn the purpose, importance, and benefits of Docker containers, as well as the specifics of how InterSystems implements them. You will then deploy InterSystems IRIS in a container, make changes to the instance, and persist those changes.

For the full documentation on Docker containers, see the *InterSystems Cloud Manager Guide*, particularly the "ICM Overview" chapter and the appendix on "Running InterSystems IRIS in Docker Containers".

1 Why Docker Containers are Important

Containers package applications into platform-independent, fully portable runtime solutions, with all dependencies satisfied and isolated. Docker containers, specifically, are ubiquitous: they can be found in public and private clouds and are supported on virtual machines (VMs) and bare metal. Docker has penetrated to the extent that all major public cloud Infrastructure as a Service (IaaS) providers support specific container services; in this way, organizations can reduce system administration costs by using Docker containers and letting the cloud provider handle the infrastructure.

Containers bring all of the following benefits:

- Containers are fully portable and provide isolation from the host environment by default. An application within a container is packaged with only the elements needed to run it and make it accessible to the required connections, services, and interfaces. You can run a container from an image, make changes during runtime, and save the container with all your changes as a new image.
- Containers are very efficient. A Docker container runs a single specified process that takes no more memory than any other executable.
- Containers are a means of cleanly moving an application between environments—for example, from development to test, then from test to production—thereby reducing interdepartmental conflicts. For example, developers can focus on their latest code and libraries while operations engineers concentrate on the overall solution infrastructure.
- Containers provide the agility and repeatability needed to revolutionize the way many organizations respond to business and technology needs. Containers clearly separate the application provisioning process, including the build phase, from the run process, and allow an organization to adopt a uniform application delivery approach.
- Containers can cleanly partition code and data, providing full separation of concerns and allowing applications to be easily deployed and upgraded.

Containers are poised to become a natural building block for applications, promoting application delivery and deployment approaches that are simpler, faster, more repeatable, and more robust.

2 How InterSystems IRIS Uses Containers

Because a Docker container packages only the elements needed to run a containerized application and executes the application natively, it provides standard, well-understood application configuration, behavior, and access. If you are experienced with InterSystems IRIS running on Linux, it doesn't matter what physical, virtual, or cloud systems and OS platforms your Linux-based InterSystems IRIS containers are running on; you interact with them all in the same way, just as you would with traditional InterSystems IRIS instances running on Linux systems.

The following describes different aspects of how InterSystems IRIS uses containers.

- *Images* — A *container image* is the executable package, while a container is a runtime *instance* of an image — what the image becomes in memory when executed. In this sense an image and a container are like any other software that exists in executable form; the image are like any other software that exists in executable form; the image is the executable and the container is the running software that results from executing the image. InterSystems provides InterSystems IRIS Docker images available worldwide from a repository.

  In the example below, you will pull an InterSystems IRIS Docker image from a repository. That image defines everything required for whatever is to be executed in the container, such as the base runtime environment (InterSystems containers are based on Ubuntu 16.04 LTS) and environment variables.

- *isc-main* — The *isc-main* program enables InterSystems IRIS and other products to satisfy the requirements of applications running in Docker containers. For example, the main process started by the command **docker run** is required to block (that is, wait) until its work is complete, but InterSystems IRIS is usually started with the **ccontrol start** command, which does not run as a blocking process. The **isc-main** program solves this by starting InterSystems IRIS and then continuing to run as the blocking entrypoint application.

  You can also leverage **isc-main** to add scripts and applications to your container, using an extensive set of options to, for example, run a script to query a database after InterSystems IRIS has started. A full list of those options is available in the section "The isc-main Program" in the *InterSystems Cloud Manager Guide*.

- *Durable %SYS* — Durable %SYS enables persistent storage of instance-specific data when InterSystems IRIS is run in a container. Because a containerized application is isolated from the host environment, it does not write persistent data; whatever it writes inside the container is lost when the container is removed and replaced by a new container. Therefore, an important aspect of containerized application deployment is arranging for data to be stored outside of the container and made available to other and future containers.

  For example, if you want to upgrade an existing InterSystems IRIS instance by running an upgraded image in a new container, the durable %SYS feature stores the instance-specific data–such as user definitions, audit records, and the log, journal, and WIJ files–on an external file system, which is mounted as a volume within the container. In effect, instance-specific data exists outside the container, available for later use.

3 Creating a Container from the InterSystems IRIS Image

Now that you have had an introduction to containers, this section will walk you through a simple, hands-on exercise. In this example, you will:

- Run a container from an InterSystems IRIS image
- Change the instance by adding a global to the database
- Save the changed instance as a new image
- Run a container from the new image

- Query for the global that you added

Because this example is intended to be brief, it does not delve into details about, for example, settings and security considerations. In production systems, there are many things you will need to do differently. The resources in the last section offer a more complete picture of using containers with InterSystems IRIS.

3.1 Basic Requirements

InterSystems IRIS is provided as a Docker image that includes everything you need. Therefore the only requirements for the Linux, macOS or Microsoft Windows system on which you launch InterSystems IRIS are that Docker CE 17.06+ is installed, with the Docker daemon running, and that the system is connected to the Internet.

It is necessary on some operating systems to change the default storage driver. For more information, see the "Docker Storage Driver" section in the *InterSystems Cloud Manager Guide*.

3.2 Identify the Docker Repository and Credentials

To download and run the InterSystems IRIS image, you need to identify the repository in which the image is located and the credentials you need to log into that repository.

InterSystems IRIS images are distributed as Docker tar archive files, available in the InterSystems Worldwide Response Center (WRC) download area. Your enterprise may have already have added these images to its Docker repository; in this case, you should get the location of the repository and the needed credentials from the appropriate IT administrator. If your enterprise has a Docker repository but has not yet added the InterSystems images, get the location of the repository and the needed credentials, obtain the tar archive files containing the InterSystems IRIS images from the WRC and add each of them to the repository using the following steps on the command line:

1. Load an image from the tar archive file:

   docker load -i *archive_file*

2. Tag the image into your repository:

   docker tag docker.intersystems.com/intersystems/*imageName***:***version your_repository/image_name***:***version*

   For example:

   ```
   docker tag docker.intersystems.com/intersystems/iris:2018.1.0.583 acme/iris:2018.1.0.583
   ```

3. Log in to your repository:

   docker login *your_repository*

   For example:

   ```
   docker login docker.acme.com
   Username: gsanchez@acme.com
   Pasword: **********
   ```

4. Push the image to your repository:

   docker push *your_repository/image_name***:***version*

   For example:

   ```
   docker push acme/iris:2018.1.0.583
   ```

3.3 Creating a Container from the InterSystems IRIS Image

Once you have an InterSystems IRIS image in your organization's repository or on your local machine, you are ready to begin the example.

1. Pull the image from your repository:

   docker pull *your_repository/image_name*:*version*

   For example:

   ```
   docker pull acme/iris:2018.1.0.583
   ```

2. Create and start a new container called `iris`:

   docker run -d --name `iris` *your_repository/image_name*:*version*

   ```
   docker run -d --name iris acme/iris:2018.1.0.583
   ```

   You can enter **docker ps** to see your container in the list with a **STATUS** of `Up`.

3. Create a Bash session in the container:

   docker exec -it `iris bash`

4. For security reasons, the predefined user accounts in the InterSystems IRIS instance in an image provided by InterSystems have random, unrecorded passwords. Before starting and connecting to the instance, change the password by entering the following:

   ```
   echo SYS > /password.isc
   $ISC_PACKAGE_INSTALLDIR/dev/Cloud/ICM/changePassword.sh /password.isc
   ```

   This sets the username to `_SYSTEM` and the password to `SYS`.

   For more information on predefined user accounts in InterSystems IRIS, see the "Important" box in the "Creating InterSystems IRIS Docker Images" section of the *InterSystems Cloud Manager Guide*; in particular, this section discusses automating the password-change process with a script.

5. Start InterSystems IRIS:

   ```
   ccontrol start IRIS
   ```

6. Connect to the instance:

   ```
   csession IRIS
   ```

7. Log in with the credentials from above:

   ```
   Username: _SYSTEM
   Password: SYS
   ```

3.4 Change the Instance and Commit the Container

1. Add a global to the database, for example:

   ```
   set ^newglobal="sample"
   ```

   Write the global to ensure that it exists:

   ```
   zw ^newglobal
   ```

2. Enter `halt` to halt the InterSystems IRIS Terminal, then `exit` to exit the shell.

3. Stop the container:

   docker stop `iris`

4. Create a new container image:

   docker commit `iris` *image_name*:*version*

   For example:

   ```
   docker commit iris my-iris:v1
   ```

   Verify that you have a new image by running:

   docker images

3.5 Run an Instance from the New Image

1. Create and start a container from the new image:

   docker run -d --name `iris2` *image_name*:*version*

   where *image_name*:*version* is the name and tag you specified in the previous step.

   For example:

   ```
   docker run -d --name iris2 my-iris:v1
   ```

2. Create a Bash session:

   ```
   docker exec -it iris2 bash
   ```

3. Log in to InterSystems IRIS:

   ```
   csession IRIS
   ```

   and enter your credentials. Note that the instance remembers your credentials.

4. Write the global that you created previously and ensure that it is correct:

   ```
   zw ^newglobal
   ```

4 More Information about Docker Containers

At this point, you are ready to continue exploring what Docker has to offer. Use the documentation and resources below to dive deeper into containers and InterSystems IRIS.

- *InterSystems Cloud Manager Guide* — Use Docker containers with InterSystems Cloud Manager (ICM) to easily and intuitively provision and deploy containers in a variety of ways. ICM brings the benefits of Infrastructure as Code (IaC), immutable infrastructure, and containerized deployment to InterSystems IRIS without requiring major investments in new technology, training, configuration, and management. This guide contains documentation on both ICM and Docker containers.
- Docker Documentation
- Containers, DevOps, and Cloud Deployment Resource Guide — Videos from InterSystems covering an array of topics related to containers, including:
  - Containers for InterSystems Technologies

  - Docker Containers: Essential Knowledge
  - Persistence in a World of Containers

- What are Containers and Why Do You Need Them? from CIO.
- What is a Container? from VMware.
- Containers 101 from VMware.