InterSystems IRIS Data Platform

First Look: InterSystems Products in Docker Containers

Version 2019.4
2020-01-28

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

Table of Contents

First Look: InterSystems Products in Docker Containers

This First Look guide introduces you to the fundamentals of using Docker containers with InterSystems IRIS® data platform 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.

For the full documentation on Docker containers and InterSystems IRIS, see *Running InterSystems Products in Containers*, as well as the "ICM Overview" chapter of the *InterSystems Cloud Manager Guide*. The section Learn More About InterSystems IRIS and Docker Containers provides links to additional sources.

To browse all of the First Looks, including those that can be performed on a free evaluation instance of InterSystems IRIS, see InterSystems First Looks.

1 Why Containers?

Containers package applications into platform-independent, fully portable runtime solutions, with all dependencies satisfied and isolated. Docker containers, specifically, are ubiquitous. Because all major public cloud Infrastructure as a Service (IaaS) providers support Docker, organizations can reduce costs and complexity by using Docker containers and letting the cloud provider handle the infrastructure.

Containers bring all of the following benefits:

- Containers cleanly partition code and data, providing full separation of concerns and allowing applications to be easily deployed and upgraded.
- Containers are very efficient; 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, and the container runs as a single process that takes no more memory than any other executable.
- Containers support clean movement of an application between environments — for example, from development to test and then to production — thereby reducing the process and management conflicts typical of departments with different objectives. Developers can focus on the latest code and libraries, quality developers on testing and defect description, and operations engineers on the overall solution infrastructure including networking, high availability, data durability, and so on.
- Containers provide the agility, flexibility, 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, including a more agile delivery methodology (DevOps) and architecture (microservices).

These advantages make containers a natural building block for applications, promoting application delivery and deployment approaches that are simpler, faster, more repeatable, and more robust.

2 InterSystems IRIS in Docker 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.

- *InterSystems-provided 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. InterSystems provides Docker images containing a fully-installed instance of InterSystems IRIS, as well as other associated images. For more information on Docker images from InterSystems, see InterSystems IRIS Docker Images from InterSystems in *Running InterSystems Products in Containers*.

  In the hands-on experience in this First Look, you will create and start a container from an InterSystems IRIS image provided by InterSystems.

- *The iris-main program* — The *iris-main* program enables InterSystems IRIS and other products to satisfy the requirements of applications running in Docker containers. The *entrypoint application*, the main process started when a container is started, is required to block (that is, wait) until its work is complete, but the command starting InterSystems IRIS does not run as a blocking process. The **iris-main** program solves this by starting InterSystems IRIS and then continuing to run as the blocking entrypoint application. For more information about **iris-main**, see The iris-main Program in *Running InterSystems Products in Containers*.

  The program also offers a number of options to help tailor the behavior of InterSystems IRIS within a container; you will use some **iris-main** options in the hands-on experience in this First Look.

- *The durable %SYS feature* — 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.

  The durable %SYS features enables persistent storage of instance-specific data — such as user definitions, audit records, and the log, journal, and WIJ files — when InterSystems IRIS is run in a container, allowing an instance to span multiple containers. For example, if you run an InterSystems IRIS container using durable %SYS, you can upgrade the instance by stopping the original container and running a new one that uses the instance-specific data created by the old one.

  You will explore the durable %SYS feature in the hands-on experience in this First Look. For detailed information on durable %SYS, see Durable %SYS for Persistent Instance Data in *Running InterSystems Products in Containers*.

Important: Docker images from InterSystems comply with the OCI support specification, and are supported on Docker Enterprise Edition and Community Edition, version 18.03 and later. Docker EE only is supported for production environments.

Not all combinations of platform and Docker version are supported by Docker; for detailed information from Docker on compatibility, see the Compatibility Matrix and About Docker CE.

InterSystems Cloud Manager (ICM) provides automated deployment of InterSystems IRIS containers and others on cloud infrastructure it provisions, as well as existing virtual and physical infrastructure. For more information about using ICM to deploy containerized InterSystems IRIS instances, see *First Look: InterSystems Cloud Manager* and the *InterSystems Cloud Manager Guide*.

3 Try It! Create Your Own InterSystems IRIS-based Container

Now that you have had an introduction to containers, this section will walk you through a simple, hands-on experience with InterSystems IRIS containers, in which you will do the following:

- Prepare by installing Docker, locating the InterSystems-provided InterSystems IRIS image, and readying a storage location with your InterSystems IRIS license key.
- Create and start an InterSystems IRIS container using the **docker run** command and the InterSystems IRIS image.
- Change the InterSystems IRIS instance running in the container, then use the **docker commit** command to commit the container as a second InterSystems IRIS-based image that includes the change you made.
- Create a Dockerfile based on the committed image that installs an application along with InterSystems IRIS, then use the **docker build** command to build a third InterSystems IRIS-based image from the Dockerfile.
- Run a container from the image that you built, using the durable %SYS feature to store instance-specific data, and:
  - Confirm that the change you made to the InterSystems IRIS instance in the first container before committing it as a new image is reflected in the new container.
  - Change a setting in the InterSystems IRIS instance running inside the new container and see it reflected in the durable %SYS directory outside the container.
  - Confirm that the installed application exists and can be run inside or outside the new container.

These instructions assume you have an InterSystems IRIS license and access to InterSystems software downloads.

Because this example is intended to be brief, it does not go into detail about matters such as 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.

Note: Some of the commands included here may require root privileges.

3.1 Select a Platform

The instructions in this hands-on were created for, and are most easily used in, a Linux environment. If you do not have a Linux system available, InterSystems recommends that you create an account on a public cloud provider such as Google Cloud Platform (GCP), Amazon Web Services (AWS), or Microsoft Azure and provision a CentOS VM with Docker installed.

3.2 Install Docker

InterSystems IRIS is provided as a Docker image that includes everything you need. Therefore the only requirements for the system on which you launch the InterSystems IRIS container are that Docker EE or CE 18.03+ is installed, with the Docker daemon running, and that the system is connected to the Internet.

You may have to change Docker's default storage driver. For more information, see Docker Storage Driver in *Running InterSystems Products in Containers*.

Note: The instructions in this hands-on are written for a Linux platform, but can be adapted to Windows or MacOS. On Windows, you can execute the instructions in a Command Prompt or Windows PowerShell window, opened with **Run as administrator**; do not use PowerShell ISE. For more information about Docker for Windows, see Using InterSystems IRIS Containers with Docker for Windows on InterSystems Developer Community and Getting started with Docker for Windows in the Docker documentation.

3.3 Download the InterSystems IRIS Docker Image

To make the InterSystems IRIS Docker image from InterSystems available for use in this hands-on, you can do one of the following:

- Download an InterSystems IRIS Community Edition image from the Docker Store's InterSystems IRIS data platform page.

  InterSystems IRIS Community Edition comes with a free built-in 13-month license (and some functionality restrictions). For more information, see Deploying InterSystems IRIS Community Edition on Your Own System in *Getting Started with InterSystems IRIS Community Edition*.

  Note: You can also provision a cloud node hosting a running InterSystems IRIS Community Edition container on GCP, AWS, or Azure; for more information, see Getting Started with InterSystems IRIS Community Edition.

- Download the archive from InterSystems and load the image. To use this image, you must have an InterSystems IRIS license.

  InterSystems IRIS images are distributed as Docker tar archive files, available in the InterSystems Worldwide Response Center (WRC) download area. Once you have downloaded the tar file, you can make it available on your system using the **docker load** command, as follows:

```
docker load -i iris-2018.1.1.888.0-docker.tar.gz
c6dba2103a94: Loading layer [==================================================>]  1.217GB/1.217GB
1b229d298c03: Loading layer [==================================================>]  1.479MB/1.479MB
Loaded image: docker.intersystems.com/intersystems/iris:2018.1.1.888.0

$ docker images
REPOSITORY                                  TAG              IMAGE ID        CREATED        SIZE
docker.intersystems.com/intersystems/iris   2018.1.1.888.0   e3cd4844771e    13 days ago    1.39GB
acme/iris                                   stable           e3cd4844771e    13 days ago    1.39GB
centos                                      7.3.1611         262f7381844c    2 weeks ago    192MB
hello-world                                 latest           05a3bd381fc2    7 months ago   1.84kB
```

  You can tag the image if you want a simpler name:

```
$ docker tag docker.intersystems.com/intersystems/iris:2018.1.1.888.0 acme/iris:stable
```

- Download the image from your organizations's Docker repository, if it has already been placed there

  If your organization has a Docker repository and the InterSystems IRIS image from InterSystems is already loaded, obtain the location of the repository and the needed credentials, then log into the repository and list the images, as follows:

```
$ docker login docker.acme.com
Username: pmartinez@acme.com
Pasword: **********
$ $ docker images
REPOSITORY          TAG            IMAGE ID          CREATED         SIZE
acme/iris           stable         15627fb5cb76      3 days ago      1.39GB
centos              7.3.1611       262f7381844c      2 weeks ago     192MB
hello-world         latest         05a3bd381fc2      7 months ag     1.84kB
```

For simplicity, these instructions assume you are working with the image acme/iris:stable

3.4 Add the License Key to the External Storage Location

Like any InterSystems IRIS instance, an instance running in a container requires a license key (typically called iris.key).

The InterSystems IRIS Community Edition image available from the Docker Store (described in the previous section) comes with a free built-in temporary license. Generally, however, license keys are not and cannot be included in an InterSystems IRIS container image, but instead must be copied into a container after it is started to be activated for the InterSystems IRIS instance running there. The **iris-main** program provides an option for this, but it requires you to place the license key in a storage location to be mounted as an external volume; instructions for using it are provided in the next section. To learn more about license keys for InterSystems IRIS containers, see License Keys for InterSystems IRIS Containers in *Running InterSystems Products in Containers*.

Copy your InterSystems IRIS license key file, iris.key, to the external storage location.

3.5 Run a Container from the InterSystems IRIS Image

Once you have made the InterSystems IRIS image available on your local machine and have identified the external storage location and placed your license key on it, you are ready to use the **docker run** command to create and start a container. The **docker run** command actually combines three separate Docker commands, as follows:

- **docker pull** — Downloads an image if it is not already present locally.
- **docker create** — Creates a container from the image.
- **docker start** — Starts the container.

Each of these commands is useful separately, for various purposes in different contexts. For more information, see Docker run reference in the Docker documentation.

A sample **docker run** command follows; all of its options are explained in the accompanying text. Note that options to the **docker run** command appear on the command line before the image specification, while options to the InterSystems **iris-main** program (see InterSystems IRIS in Docker Containers) come after.

```
docker run --name iris
  --init
  --detach
  --publish 52773:52773
  --volume /nethome/pmartinez/iris_external:/external
  acme/iris:stable
  --key /external/iris.key
```

- **--name** *container_name*

  Specifies the name of the container, which you can use to refer to the container in other Docker commands, for example **docker stop** *container_name* when you want to stop the container.

- **--init**

  Indicate that an init process should be used as PID 1 within the container, ensuring that the usual responsibilities of an init system are performed inside the container.

- **--detach**

  Runs the container in the background (and displays the container's unique ID).

- **--publish** *host_port:container_port*

  Publishes a port within the container to a port on the host so that entities outside the container (on the host or on other machines) can contact a program in the container. For example, an InterSystems IRIS instance's Management Portal is reached through the instance's web server port (52773 by default). If this port inside the container is published to a port on the host, the instance's Management Portal can be loaded into a browser using the host's port.

--volume *external_storage_path***:***internal_volume*

Mounts an external storage location accessible by the container as a storage volume inside the container. For information about which storage locations can be mounted in this way and Docker configuration that may be required, see Use Volumes in the Docker documentation.

Important: InterSystems does not support mounting NFS locations as external volumes in InterSystems IRIS containers.

- *repository/image***:***tag*

  Specifies the image to be pulled and used to create a container (see Download the InterSystems IRIS Docker Image). Use the **docker images** command to list available images and make sure you are specifying the right one. The instructions here assume the image is iris:stable.

- **--key** *license_key_path*

  An **iris-main** option that identifies the InterSystems IRIS license key to be installed in the instance in the container; this location must be on a mounted volume.

Use the preceding sample and explanations to construct your own **docker run** command and execute it on the command line. When the command has completed, use the **docker ps** command to see your container in the list with a status of **Up**.

```
$ docker run --name iris --init --detach --publish 52773:52773
    --volume /nethome/pmartinez/iris_external:/external
    acme/iris:2018.1.0.583 --key /externaliris.key
426d4a511d6746d89ec2a24cf93b29aa546ea696b479a52210d37da4c6d04883
$ docker ps
CONTAINER ID  IMAGE              COMMAND                 CREATED        STATUS
426d4a511d67  acme/iris:stable  "/iris-main --key ..."  5 seconds ago  Up 3 seconds
    PORTS                       NAMES
    0.0.0.0:52773->52773/tcp  iris
```

Note: The **--key** option is not needed with the InterSystems IRIS Community Edition image (see Download the InterSystems IRIS Docker Image), which comes with a free built-in license.

If the image is not yet present locally but is in your organization's repository (see Download the InterSystems IRIS Docker Image), Docker pulls (downloads) the image before creating and starting the container.

As shown in the example, after creating the container, Docker outputs the *UUID long identifier*; the first 12 characters make up the *UUID short identifier*, which is used to identify the container in other output, for example from the **docker ps** command.

3.6 Change the Instance and Commit the Container as a New Image

When you alter the program running inside a container, you can use the **docker commit** command to create a new image from the container. This new image is the same as the original image from which you created the container, but includes the alterations you made to the container. To see how this works, follow these steps:

1. Open the Management Portal for the InterSystems IRIS instance in the container. The URL for an instance's Management Portal incorporates both the host identifier and web server port number of the instance.

   - The host identifier is the hostname or IP address of the system the container is running on; if your browser is running on the same system as the container, you can use localhost.
   - The web server port number is the host port number to which you published the instance's web server port number, which is 52773, when you started the container with **docker run**. Assuming you included **--publish 52773:52773** as provided in the sample command at the end of the previous section, the web server port number is 52773.

   For example, on the container host, with web server port 52773, the Management Portal URL would be http://localhost:52773/csp/sys/%25CSP.Portal.Home.zen.

Log in using one of the predefined user accounts, for example **_SYSTEM**, which have the default password **SYS** (see Predefined Users Accounts in the "Users" chapter of the *Security Administration Guide*). For security reasons, you are immediately prompted to change the password on first login (using the Management Portal or the **iris terminal** command) to any predefined account on a containerized InterSystems IRIS instance.

Note: The **iris-main --password-file** option can be used to change the default password, including in scripts and other automation; for more information, see The iris-main Program in *Running InterSystems Products in Docker Containers*.

2. From the home page, select **System Administration** > **Configuration** > **System Configuration** > **Namespaces** to display the **Namespaces** page, then click the **Create New Namespace** button to display the **New Namespace** page.
3. Create a namespace named **USER2** by entering **USER2** in the **Name of the namespace box**, selecting **USER** from the **Copy from** dropdown, clearing the **Enable namespace for interoperability productions** check-box and confirming that, clicking the **Save** button, and finally confirming that you want to copy all properties and mappings. Then click **Close** on the **Copy Namespace Mappings** page to return to the **Namespaces** page, on which the **USER2** namespace is now listed. You have now altered the instance in the container.
4. Next, stop the container and commit it as a new image called **iris2**, then list the available images.

```
$ docker stop iris
$ docker commit iris acme/iris2:test
sha256:7b4adb9f7abf1490a39086665ccd3d255c05163c25cb9a3de8e9421f6ca16b40
$ docker images
REPOSITORY          TAG            IMAGE ID         CREATED        SIZE
acme/iris2          test           421f6ca16b40     8 seconds ago  1.40GB
acme/iris           stable         15627fb5cb76     3 days ago     1.39GB
centos              7.3.1611       262f7381844c     2 weeks ago    192MB
hello-world         latest         05a3bd381fc2     7 months ag    1.84kB
```

5. Finally, remove the container created from the original iris image.

```
$ docker rm iris
iris
```

3.7 Build an InterSystems IRIS-based Image Using a Dockerfile

Committing a running container is a convenient way to save changes for testing purposes or as a minor delta. But the preferred method for creating a production image is to define the image in a Dockerfile and then build it using the **docker build** command, which:

- ensures that the source for each image is directly traceable.
- avoids accidentally committing unwanted changes.
- maintains separation of code and data, a best practice when using containers.

In this section you will create a Dockerfile based on your modified InterSystems IRIS image and use it to build a new image that installs an application with InterSystems IRIS. In the next section, Run and Investigate the InterSystems IRIS-based Container, you will run and investigate a container from this image that uses durable %SYS, a feature of InterSystems IRIS containers that lets you save instance-specific data outside the container.

Each Dockerfile specifies an existing base image which provides (at a minimum) the runtime environment for whatever is to be executed in the container; if the base image contains installed software, that software is also included in the image built from the Dockerfile. For example, if you use the InterSystems IRIS image from InterSystems as a base, the resulting image inherits the Ubuntu 18.04 LTS runtime environment on which the InterSystems image is based and includes an installed InterSystems IRIS instance. To this you can add specifications for everything needed to add your application — for example, copying or downloading files, setting environment variables, installing the application, and launching the application.

Before creating your Dockerfile, choose or create a very simple application to install in the container and identify or create an installer to be copied into the Dockerfile. To make it easy, you can use the example that follows — a command to create a file. In the Dockerfile for a production image, you might install your InterSystems IRIS-based application instead.

To build your InterSystems-IRIS image, follow these steps:

1. Using your favorite text editor, create a file named Dockerfile containing the following contents. The image you specify in the **FROM** statement should be iris2, the one you created by committing the container you modified in Change the Instance and Commit the Container as a New Image. If you have a simple application to install, replace the **RUN** command shown with one installing that application.

   ```
   # Build from the modified IRIS image
   FROM acme/iris2:test

   # create a demo file
   RUN echo "This is the file added to the image" > /demo.txt
   ```

2. Issue the **docker build** command, specifying the build context (the location of the Dockerfile) and naming and tagging the resulting image (you may need to run this command with root privileges):

   ```
   $ docker build /nethome/pmartinez --tag acme/iris3:test
   Sending build context to Docker daemon  478.1MB
   Step 1/2 : FROM acme/iris2:test
    ---> 15627fb5cb76
   Step 2/2 : RUN echo "This is the file added to the image" > /demo.txt
    ---> Running in 000b94705814
    ---> 64350c828716
   Removing intermediate container 000b94705814
   Successfully built 64350c828716
   Successfully tagged iris3:test
   ```

3. When the image has built successfully, list the available images

   ```
   $ docker images
   REPOSITORY           TAG            IMAGE ID         CREATED         SIZE
   acme/iris3           test           64350c828716     9 seconds ago   1.44
   acme/iris2           test           421f6ca16b40     1 hour ago      1.40GB
   acme/iris            stable         15627fb5cb76     3 days ago      1.39GB
   centos               7.3.1611       262f7381844c     2 weeks ago     192MB
   hello-world          latest         05a3bd381fc2     7 months ag     1.84kB
   ```

Note: InterSystems provides a sample Dockerfile and code file that serve as an example of creating an InterSystems IRIS-based image incorporating your application. For more information, see Creating InterSystems IRIS Docker Images in *Running InterSystems Products in Containers*. InterSystems also provides tools to help you build InterSystems IRIS-based images; see Containerization Tools Provided with InterSystems IRIS in the same document.

3.8 Run and Investigate the InterSystems IRIS-based Container

To wrap up this experience, you will use the **docker run** command to create and start a container from the InterSystems-IRIS based image you just built, including the durable %SYS feature for persisting instance-specific data. Durable %SYS is a much more useful way of saving instance-specific data and any changes you have made to it. Because this data is saved outside the container, it can become the data for a new InterSystems IRIS container, allowing you to upgrade an IRIS instance by running a container from a later image while retaining the data from the previous container; this is not possible with internal container changes committed to a new image. (For detailed information on durable %SYS, see Durable %SYS for Persistent Instance Data in *Running InterSystems Products in Containers*.)

When you have started the new container, you will do the following:

- Confirm that the namespace you created in the container you committed as a new image (see Change the Instance and Commit the Container as a New Image) exists in the InterSystems IRIS instance in the new container.

- Change a setting in the InterSystems IRIS instance in the container and see it reflected in the durable %SYS data outside the container.
- Confirm that the demo file you added exists inside the container.

To do this, follow these steps:

1. Identify an external storage location for this container. You can use the one you chose for the previous container in Add the License Key to the External Storage Location or select a new one. The license key should still be in place in the previous location. (If you use a new location, ensure that the license key is in place.)
2. Create a **docker run** command like the one you executed in Run a Container from the InterSystems IRIS Image, based on the instructions there but with two changes.
   - Add the option **--env ISC_DATA_DIRECTORY=***pathname*

     Identifies the durable %SYS directory, that is, the location in which the InterSystem IRIS instance's persistent data is written. The durable %SYS directory must be on a mounted volume (see the **--volume** option and Add the License Key to the External Storage Location).
   - Specify the new image with *image***:***tag*

     Previously, you created the container from acme/iris:stable, the InterSystems-provided image; this time, you are using acme/iris3:test, the image you built from your Dockerfile. Remember that acme/iris3:test is based on acme/iris2:test, the image you created by committing the altered iris container.

   Call the container iris3. When the **docker run** command has completed, use the **docker ps** command to list the container and see its status. For example:

```
$ docker run --name iris3 --init --detach --publish 52773:52773
    --volume /nethome/pmartinez/iris_external:/external
    --env ISC_DATA_DIRECTORY=/external/durable
    iris3:test
    --key /external/iris.key
bdfe214ef76a34290a8308cddce92162aae14df1ba1bc244e692af3c8d911a3e
$ docker ps
CONTAINER ID  IMAGE            COMMAND                  CREATED        STATUS
af3c8d911a3e  acme/iris3:test  "/iris-main --key ..."  5 seconds ago  Up 3 seconds
  PORTS                     NAMES
  0.0.0.0:52773->52773/tcp  iris3
```

   Note: The **--key** option is not needed with the InterSystems IRIS Community Edition image (see Download the InterSystems IRIS Docker Image), which comes with a free built-in license.

   Docker Compose, a tool for defining and running multicontainer Docker applications, offers an alternative to command-line interaction with Docker. To use Compose, you create a docker-compose.yml containing specifications for the containers you want to create, start, and manage, then use the **docker-compose** command. For more information, see Running an InterSystems IRIS Container: Docker Compose Example in *Running InterSystems Products in Containers* and Overview of Docker Compose in the Docker documentation.

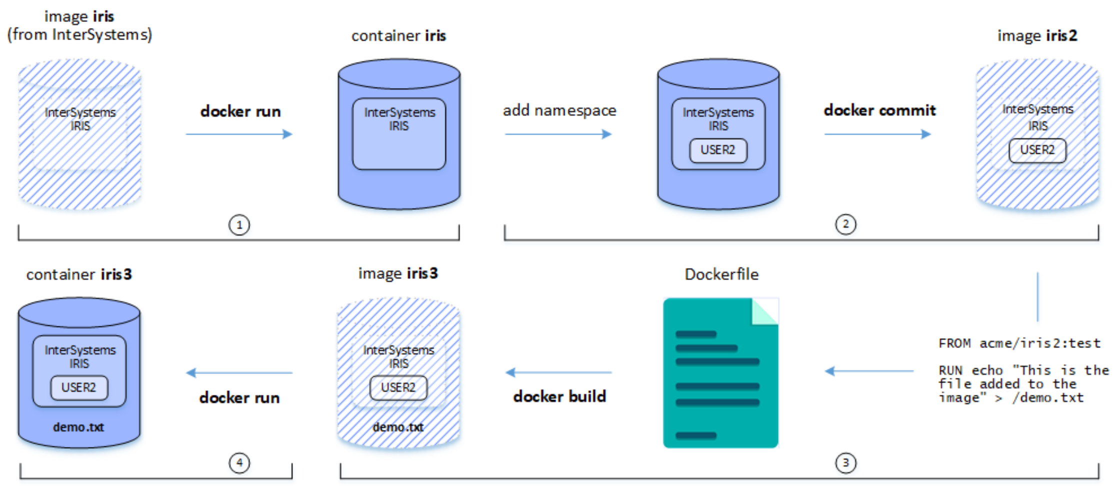
3.8.1 Confirm the Change You Committed from the Altered InterSystems Container

In Change the Instance and Commit the Container as a New Image, you added a namespace to the InterSystems IRIS instance in the container you created from the InterSystems-provided image intersystems/iris:stable and then committed that container as a new image, acme/iris2:test. In Build an InterSystems IRIS-based Image Using a Dockerfile, you created a Dockerfile based on acme/iris2:test and used it to build the image acme/iris3:test. The namespace you added should therefore exist in the InterSystems IRIS instance running inside the iris3 container, which was created from acme/iris3:test. These steps are illustrated in the following list and illustration:

1. Run a Container from the InterSystems IRIS Image
2. Change the Instance and Commit the Container as a New Image

3. Build an InterSystems IRIS-based Image Using a Dockerfile
4. Run and Investigate the InterSystems IRIS-based Container

Figure 1: Steps in this Hands-on



To confirm this, do the following:

1. Open the Management Portal and log in, as described in Change the Instance and Commit the Container as a New Image.
2. From the home page, select **System Administration** > **Configuration** > **System Configuration** > **Namespaces** to display the Namespaces page; the **USER2** namespace you created in the iris container is listed.

3.8.2 Explore and Alter the Durable %SYS Directory

To explore the durable %SYS feature of InterSystems IRIS containers, do the following:

1. To see the instance-specific data written outside the container by InterSystems IRIS because you included the durable %SYS environment variable in your **docker run** command, display the contents of the directory you specified in that variable in the storage location you specified using the **--volume** option as the external volume to be mounted. For example, if that directory was specified as /nethome/pmartinez/iris_external/durable, as shown in the sample **docker run**, you'd do the following

```
$ cd /nethome/pmartinez/iris_external
$ ls
durable iris.key
$ ls durable
csp  dist  httpd  iris.cpf  iris.cpf_20180417  _LastGood_.cpf  mgr
$ ls durable/mgr
alerts.log    irisaudit          iris.ids      irislocaldata  iristemp  IRIS.WIJ  journal.log
startup.last  SystemMonitor.log  user          ilock          IRIS.DAT  iris.lck  iris.shid
iris.use      journal            messages.log  stream         Temp
```

2. Return the Management Portal for the InterSystems IRIS instance in the container and select **System Administration** > **Configuration** > **System Configuration** > **Journal Settings** to display the Journal Settings page. Change the **Secondary journal directory** setting from /external/durable/mgr/journal/ to /external/durable/mgr/journal2/ and click **Save**.
3. Return to the command line and list the mgr subdirectory of the durable %SYS directory again:

```
$ ls /nethome/pmartinez/iris_external/durable/mgr
alerts.log         irisaudit     iris.ids  iris.lck  iris.shid        iris.use      journal
journal.log        messages.log  stream    Temp      ilock            IRIS.DAT      iris.key
irislocaldata      iristemp      IRIS.WIJ  journal2  licmanager.port  startup.last
SystemMonitor.log  user
```

The journal2 subdirectory has been added *outside* of the container because of the change you made to the InterSystems IRIS instance *inside* the container.

This example shows how durable %SYS enables you to upgrade a containerized InterSystems IRIS instance by creating a container from a new image. All persistent changes you make to the instance are stored *outside* the container in the durable %SYS directory; if you create and start a new container from any InterSystems IRIS image using the needed options — that is, the **--volume** option mounting the external storage location for durable %SYS and the **--env ISC_DATA_DIRECTORY** option specifying the durable %SYS location on that mounted volume, which must exist and contain a /mgr subdirectory — those changes are inherited by the instance because it uses the same data as the instance in the previous container.

3.8.3 Confirm the File in the Container

In Build an InterSystems IRIS-based Image Using a Dockerfile, you created a Dockerfile that added a demo file to an InterSystems IRIS-based image (acme/iris2:test) and used it to build the image acme/iris3:test. To verify that the file was created, use the **docker exec** command to display the file in the iris3 container from outside the container, as follows:

```
$ docker exec iris3 cat /demo.txt
This is the file added to the image

$
```

You could also do this interactively by using the **docker exec** command to open a Bash shell inside the container, as follows:

```
$ docker exec -it iris3 bash
root@af3c8d911a3e:/# ls
bin       dev      external   iris-main.log  media  proc  sbin  tmp  waitISC.log
boot      durable  home       lib            mnt    root  srv   usr
demo.txt  etc      iris-main  lib64          opt    run   sys   var
root@af3c8d911a3e:/# cat /demo.txt
This is the file added to the image
root@af3c8d911a3e:/#
```

While in the Bash shell, list the durable %SYS directory, which you listed from outside the container in Explore and Alter the Durable %SYS Directory, from inside the container instead:

```
root@af3c8d911a3e:/# ls /external/durable/mgr
alerts.log         irisaudit     iris.ids   iris.lck    iris.shid        iris.use      journal
journal.log        messages.log  stream     Temp        ilock            IRIS.DAT      iris.key
irislocaldata      iristemp      IRIS.WIJ   journal2    licmanager.port  startup.last
SystemMonitor.log  user
```

Another way to confirm the demo.txt file is to add the **iris-main --create** option, which executes the specified shell command before any other **iris-main** arguments are processed, to the **docker run** command you executed in Run and Investigate the InterSystems IRIS-based Container and displaying the log after the container has started, as follows:

```
$ docker run --name iris3 --init --detach --publish 52773:52773
    --volume /nethome/pmartinez/iris_external:/external
    --env ISC_DATA_DIRECTORY=/external/durable
    --create "cat demo.txt" --key /external/iris.key
bdfe214ef76a34290a8308cddce92162aae14df1ba1bc244e692af3c8d911a3e
$ docker logs -f iris3
[INFO] Executing command cat demo.txt...
[INFO] This is the file added to the image

[INFO] ...executed command cat demo.txt
[INFO] .
Waited 0 seconds for InterSystems IRIS to start
This copy of InterSystems IRIS has been licensed for use exclusively by:
License missing or unreadable.
Copyright (c) 1986-2018 by InterSystems Corporation
Any other use is a violation of your license agreement

[INFO] Copying InterSystems IRIS license key from /external/iris.key to /usr/irissys/mgr...
[INFO] ...copied key
[INFO] Starting InterSystems IRIS instance IRIS...
[INFO] This copy of InterSystems IRIS has been licensed for use exclusively by:
Acme Corp.
```

```
Copyright (c) 1986-2018 by InterSystems Corporation
Any other use is a violation of your license agreement
Starting IRIS
```

4 Learn More About InterSystems IRIS and 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.

- Docker Containers and InterSystems IRIS (video)
- Running InterSystems Products in Containers
- Articles from the InterSystems Developer Community:
  - What is a Container?
  - What is a Container Image?
  - Using InterSystems IRIS Containers with Docker for Windows
- Docker Documentation
- InterSystems Cloud Manager Guide — Use InterSystems Cloud Manager (ICM) to easily and intuitively provision infrastructure and deploy containers on it in a variety of ways. ICM brings the benefits of Infrastructure as Code (IaC),nd containerized deployment to InterSystems IRIS without requiring major investments in new technology, training, configuration, and management. This guide contains documentation on both ICM and using InterSystems IRIS with Docker containers.
- First Look: InterSystems Cloud Manager