First Look: InterSystems Cloud Manager

Version 2018.1
2018-01-31

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

First Look: InterSystems Cloud Manager
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: InterSystems Cloud Manager **1**
1 What Can ICM Do for You? 1
2 How Does ICM Work? 1
3 Try It! Deploy InterSystems IRIS in the Cloud with ICM 2
3.1 Install Docker CE 17.06 2
3.2 Identify the Docker Repository and Credentials 2
3.3 Launch ICM 3
3.4 Get an AWS Account and Credentials 3
3.5 Generate Security Keys 4
3.6 Customize the Sample Configuration Files 4
3.7 Provision the Infrastructure 7
3.8 Deploy InterSystems IRIS 7
3.9 Try ICM Management Commands 8
3.10 Unprovision the Infrastructure 10
4 ICM Can Do Much More Than That! 10
5 Learn More About ICM 11

List of Figures
Figure 1: ICM Makes It Easy 2
Figure 2: Interactive ICM Commands 9

First Look: InterSystems Cloud Manager

This First Look introduces you to InterSystems Cloud Manager (ICM), the end-to-end cloud provisioning and deployment solution for InterSystems IRIS-based applications.

As part of this first look you will use ICM to provision infrastructure in a public cloud and deploy InterSystems IRIS on that infrastructure.

1 What Can ICM Do for You?

Welcome to the Cloud Age! Are you eyeing its opportunities but wary of its challenges? Specifically,

- Are you eager to take advantage of the cloud, but hesitant to commit resources to a complex migration?
- Are you already in the cloud, but struggling to find a way to manageably deploy and version your application across a wide range of software environments?
- Do you want to bring continuous integration and delivery to your software factory and a DevOps approach to your deployment process? That is, would you like to free yourself from the limitations and risks of legacy practices, library dependencies, system drift, manual upgrade, and other overhead?

ICM can help! ICM gives you a simple, intuitive way to provision cloud infrastructure and deploy services on it, helping you get into the cloud *now* without major development or retooling. The benefits of infrastructure as code (IaC) and containerized deployment make it easy to deploy InterSystems IRIS-based applications on public cloud platforms such as Google, Amazon, and Azure, or on your private VMware vSphere cloud. Define what you want, issue a few commands, and ICM does the rest.

Even if you are already using cloud infrastructure, containers, or both, ICM dramatically reduces the time and effort required to provision and deploy your application by automating numerous otherwise manual steps.

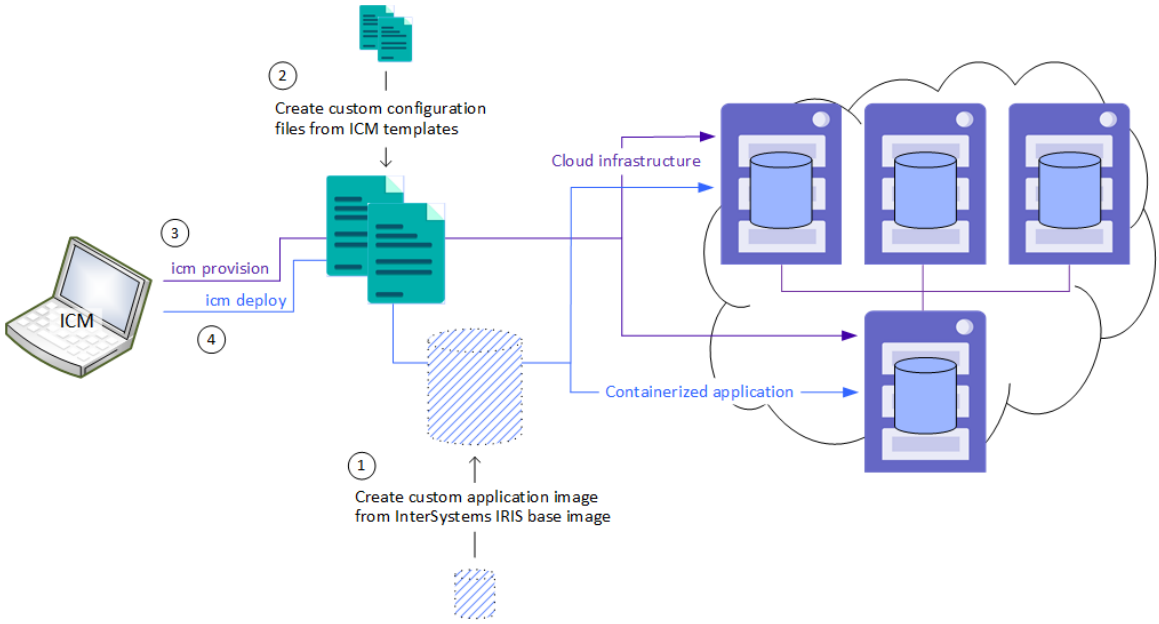
2 How Does ICM Work?

Guided by your input in plain-text configuration files, ICM provisions your infrastructure using the popular Terraform IaC tool from Hashicorp and configures the provisioned compute nodes as needed. In the next phase, ICM deploys InterSystems IRIS and your application in Docker containers, as well as other services if desired. All of the InterSystems IRIS configuration needed for the deployment you want is carried out automatically. ICM can also deploy containerized services on existing virtual and physical clusters.

ICM itself comes in a Docker container that includes everything you need. Download and run the ICM image from InterSystems to open the command line, and you are ready to go. ICM makes it easy for you by combining these elements:

- Sample configuration files you can use as templates to quickly define the deployment you want.
- InterSystems IRIS Docker images to which you can add your application.
- User-friendly commands for each task.
- Multiple ways to manage and interact with the provisioned nodes and the services deployed on them.

Figure 1: ICM Makes It Easy



3 Try It! Deploy InterSystems IRIS in the Cloud with ICM

ICM carries out many tasks for you and gives you numerous options to help you deploy exactly what you need, so its use in production involves a certain amount of planning and preparation (although much less than manual methods!). But the provisioning and deployment process is simple, and ICM can make many decisions for you. This exploration is designed to let you see for yourself how ICM works, and how easy it is to deploy an InterSystems IRIS configuration on Amazon Web Services (AWS) using ICM. While it is not the work of a moment, this exploration should not take up too much of your time, and you can do it in stages as opportunity arises.

To give you a taste of ICM without bogging you down in details, we've kept this simple; for example, we've had you use as many default settings as possible. When you bring ICM to your production systems, though, there are many things you will need to do differently, especially in regard to (but not limited to) security. So be sure not to confuse this exploration of ICM with the real thing! The sources provided at the end of this document will give you a good idea of what's involved in using ICM in production. The *ICM Guide* provides complete information and procedures for using ICM.

If you prefer, you can deploy InterSystems IRIS on Google Cloud Platform (GCP) or Microsoft Azure instead of AWS; links to the needed information in the *ICM Guide* are provided where appropriate.

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

3.1 Install Docker CE 17.06

ICM 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 ICM are that Docker is installed, with the Docker daemon running, and that the system is connected to the Internet.

3.2 Identify the Docker Repository and Credentials

To download and run the ICM image, and to deploy the InterSystems IRIS image in the cloud using ICM, you need to identify the repository in which these images are located and the credentials you need to log into that repository. The

repository must be accessible from the internet (that is, not behind a firewall) in order for the cloud provider to download the InterSystems IRIS image.

InterSystems images are distributed as Docker tar archive files, available in the InterSystems Worldwide Response Center (WRC) download area. Your enterprise may 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 ICM and InterSystems IRIS images from the WRC, and add each of them to the repository using the following steps on the command line:

1. **docker load -i** *archive_file*
2. **docker tag docker.intersystems.com/intersystems/***image_name***:***version your_repository/image_name***:***version*

   For example:

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

3. **docker login** *your_repository*

   For example:

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

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

   For example:

   ```
   docker push acme/icm:2018.1.0.583
   ```

If your organization does not have an internet accessible Docker repository, you can use the free (or extremely low cost) Docker Hub for your testing.

3.3 Launch ICM

To launch ICM on the command line, use the following **docker run** command to download the ICM image from the repository, create a container from it, and start the container:

```
docker run --name icm -it --cap-add SYS_TIME acme/icm:stable
```

3.4 Get an AWS Account and Credentials

ICM can provision and deploy on three public cloud platforms: Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure. Here we will use AWS. You can create a free trial account at aws.amazon.com/free.

Once you have created the account, download administrative credentials by following the instructions on the AWS Managing Access Keys page. Next, copy the credentials from the .csv file in which they are downloaded to another file, in the following format

```
[default]
aws_access_key_id = <value in "Access key ID" column>
aws_secret_access_key = <value in "Secret access key" column"
```

It is this new file that you will identify to ICM as your AWS credentials file.

Note: For information about how to download and use credentials for a GCP or Azure account, see Provider-Specific Parameters in the *ICM Guide*.

3.5 Generate Security Keys

ICM communicates securely with the cloud provider on which it provisions the infrastructure, with Docker on the provisioned nodes, and with several InterSystems IRIS services following container deployment. You have already downloaded or identified your cloud credentials; you also need other files to enable secure SSH and SSL/TLS communication.

You can create the files you need using two scripts provided with ICM, The keygenSSH.sh script creates the needed SSH files and places them in the directory /Samples/ssh in the ICM container. The keygenTLS.sh script creates the needed SSL/TLS files and places them in /Samples/tls.

To run these scripts on the ICM command line, enter the following:

```
# keygenSSH.sh
Generating keys for SSH authentication.
...
# keygenTLS.sh
Generating keys for TLS authentication.
...
```

For more information about the files generated by these scripts, see Security-Related Parameters in the *ICM Guide*.

Important: These generated keys are intended as a convenience for your use in this First Look experience and other tests; in production, you should generate or obtain the needed keys in keeping with your company's security policies. The keys generated by these scripts, as well as your cloud provider credentials, must be fully secured, as they provide full access to any ICM deployments in which they are used.

3.6 Customize the Sample Configuration Files

To define the configuration you want ICM to deploy, you include the needed settings in two JSON-formatted configuration files: the defaults file and the definitions file. The former (defaults.json) contains settings that apply to the entire deployment — for example, your choice of cloud provider — while the latter (definitions.json) defines which types of nodes you want and how many of each, thereby determining whether you deploy a sharded cluster, a single stand-alone IRIS instance, or some other configuration.

Sample configuration files for AWS are located in the /Samples/AWS directory in the ICM container (there are also directories for the other platforms). To customize these files, use an editor such as **vi** to edit them directly within the container, or use the **docker cp** command on the local command line to copy them from the container to the local file system, and then back again after you have edited them, for example:

```
docker cp icm:/Samples/AWS/defaults.json .
docker cp icm:/Samples/AWS/definitions.json .
...
docker cp defaults.json icm:/Samples/AWS
docker cp definitions.json icm:/Samples/AWS
```

3.6.1 Customize defaults.json

The following tables indicate the minimum customization needed for /Samples/AWS/defaults.json file, plus suggestions for optional changes. (The settings in the table are ordered differently than in the sample defaults.json file.) The first setting in the file is "Provider": "AWS"; do not change this. See Amazon Web Services (AWS) Parameters in the *ICM Guide* and the AWS documentation for more information about these settings.

Setting in /Samples/AWS/defaults.json	Description	What to do
"Credentials": "/Samples/AWS/sample.credentials"	Location of downloaded AWS credentials file.	Use **docker cp** to copy your AWS credentials file (see Get an AWS Account and Credentials) to this location.
"Region": "us-west-1" "Zone": "us-west-1c"	Geographical location and zone of compute resources used to provision infrastructure. About AWS Regions and Availability Zones	OPTIONAL: Change to your preferred region and zone.
"AMI": "ami-d1315fb1" "InstanceType": "m4.large"	Amazon Machine Image (AMI) and instance type used to provision compute nodes. About AWS AMIs (to list public AMIs available, in the EC2 Console, select AMIs in the navigation pane and filter for **Public AMIs**) About AWS instance types **Note:** ICM currently supports provisioning of and deployment of containers on compute nodes running Red Hat Enterprise Linux, version 7.3 or later.	OPTIONAL: Change to your preferred AMI and instance type. **Note:** Some instance types may not be compatible with some AMIs. The settings provided in the sample defaults.json are compatible.
"SSHUser": “ec2–user”	Nonroot account with **sudo** access on provisioned nodes, used by ICM for access; determined by selected AMI.	*Do not change* if using Red Hat Enterprise Linux image.
"Label": "Sample", "Tag": "TEST",	Arbitrary fields in naming scheme for provisioned cloud nodes: *Label-Role-Tag-NNNN*, for example **Acme-DM-TEST-0001** or **Acme-DS-TEST-0005**.	Update with custom information indicating ownership and purpose, to avoid conflicting with others testing ICM. Multiple deployments should not share the same Label and Tag.

Setting in /Samples/AWS/defaults.json	Description	What to do
"KeyFile": "/Samples/ssh/insecure", "SSHKey": "/Samples/ssh/insecure-ssh2.pub", "SSHKey2": "/Samples/ssh/insecure.pub", "TLSKeyDir": "/Samples/tls",	Security keys for SSH and SSL/TLS (see Security-Related Parameters in the *ICM Guide*)	If you used the key generation scripts discussed in Generate Security Keys, the keys are located in these directories; make no changes. If you provided your own keys, use **docker cp** to copy them from the local file system to these locations.
"DockerImage": "intersystems/iris:stable" "DockerUsername": "xxxxxxxxxxxx", "DockerPassword": "xxxxxxxxxxxx",	The Docker image to be deployed on all provisioned nodes, and the credentials needed to download that image if in a private repository; see Identify Docker Repository and Credentials for important information.	Change these values to reflect the repository information and credentials you identified in Identify Docker Repository and Credentials.
"ISCPassword": "",	Password for predefined accounts in deployed InterSystems IRIS images.	To provide the password interactively with masked input during the deployment phase, remove this field. Otherwise, change to your preferred password.

Note: Settings specific to GCP and Azure differ from those for AWS. For information about the settings you will need to modify in the defaults.json files in the /Samples/GCP and /Samples/Azure directories, see Provider-Specific Parameters in the *ICM Guide*.

3.6.2 Customize definitions.json

The /Samples/AWS/definitions.json file (which is the same for all providers) defines a basic sharded cluster with a shard master data server and two shard data servers, as follows. The Role field identifies the node type being provisioned, which in this case is DM for the shard master data server and DS for the shard data servers. The Count field indicates how many of that type to provision; StartCount starts numbering at 0002 for the DS nodes.

```
[
   {
    "Role": "DM",
    "Count": "1",
    "ISCLicense": "/Samples/license/ubuntu/ShardMaster/license.key"
    },
    {
    "Role": "DS",
    "Count": "2",
    "StartCount": "2"
    }
]
```

Any DM node requires an InterSystems IRIS sharding license. If the DM is the shard master data server in a sharded cluster, this license is used to generate licenses for use by the DS nodes, but single shard master license is required even if it is a stand-alone instance. Use **docker cp** to copy a sharding license to a location within the container, such as /Samples/AWS, and update the ISCLicense setting to specify this location.

That is the only definitions.json change required to provision the basic sharded cluster. To provisions a stand-alone InterSystems IRIS instance, remove the section defining the DS nodes so that the file looks like this:

```
[
   {
    "Role": "DM",
    "Count": "1",
    "ISCLicense": "/Samples/AWS/sharding_license.key"
    },
]
```

3.7 Provision the Infrastructure

By default, ICM takes input from the configuration files in the current directory, so all you need to do to provision your infrastructure is change to the /Samples/AWS directory and issue this command:.

```
icm provision
```

The **icm provision** command allocates and configures compute nodes on the platform you have selected. During the provisioning operation, ICM creates or updates state and log files in the state subdirectory and when finished creates the instances.json file, which serves as input to subsequent deployment and management commands.

Because ICM runs multiple tasks at once, the steps (Terraform plan and apply, copying files, mounting volumes, and so on) may not start and complete on the nodes in the same sequence. At completion, ICM provides a summary of the compute nodes that have been provisioned, and outputs a command line which can be used to delete the infrastructure at a later date, as shown in the following:

```
Machine             IP Address       DNS Name
-------             ----------       -------
ACME-DM-TEST-0001   00.53.183.209    ec2-00-53-183-209.us-west-1.compute.amazonaws.com
ACME-DS-TEST-0002   00.53.183.185    ec2-00-53-183-185.us-west-1.compute.amazonaws.com
ACME-DS-TEST-0003   00.56.59.42      ec2-00-56-59-42.us-west-1.compute.amazonaws.com
To destroy: icm unprovision -stateDir /Samples/AWS/ICM-8620265620732464265 [-cleanUp] [-force]
```

Important: Copy the **icm unprovision** command provided in the output and save this information, so you can easily replicate it when unprovisioning. This output also appears in the icm.log file.

Detailed information about errors that occur during the provisioning and deployment phasis written to terraform.err files in subdirectories of the state directory; when an error occurs, ICM directs you to the appropriate file, which can help you identify the cause of the problem.

If **icm provision** does not complete successfully due to timeouts and internal errors on the provider side, you can issue the command again, as many times as needed, until ICM completes all the required tasks for all the specified nodes without error. When doing so you must use the **-stateDir** option to specify the state directory (such as /Samples/AWS/ICM-8620265620732464265) created the first time, to indicate that provisioning is already in process and provide the needed information about what has been done and what hasn't. However, if you determine that the problem was caused by an error in a configuration file, unprovision the infrastructure, fix the error, and run **icm provision** again.

For more information about **icm provision**, see The icm provision Command in the *ICM Guide*.

3.8 Deploy InterSystems IRIS

In addition to downloading Docker images on the provisioned compute nodes and running them as containers, ICM carries out InterSystems IRIS-specific configuration and other tasks. To deploy InterSystems IRIS on your provisioned nodes, remain in the /Samples directory in which you customized the configurations files and provisioned the infrastructure and issue this command:

```
icm run
```

By default, **icm run** downloads and runs the image specified by the DockerImage field in the configuration files, in this case the InterSystems IRIS image from the InterSystems repository; each container is named **iris**. In practice, options allow you to execute the **icm run** command multiple times to deploy multiple containers with unique names on each provisioned node, only on specified nodes, or on nodes of specific types

After InterSystems IRIS starts on each node, ICM does whatever configuration of the instance is needed — for example, resetting the password, creating the namespace you specified, configuring sharding, and so on. Because ICM runs multiple tasks at once, the steps in the deployment process may not start and complete on the nodes in the same sequence.

At completion, ICM outputs a link to the Management Portal of the appropriate InterSystems IRIS instance; in this case, the provided link is for the shard master data server (DM) node, or for the stand-alone instance if that is the configuration you chose. Open the link in your browser and explore InterSystems IRIS using the Management Portal.

In the event of an error, See the log file ICM directs you to for information that can help you identify the cause of the problem.

Important: Unlike the **icm provision** command, the **icm run** command cannot simply be repeated if it fails on one or more nodes due to factors such as network latency and disconnects or interruptions in cloud provider service. Instead, you must manually roll back deployment on all nodes before executing **icm run** again. The procedure is as follows:

1. Enter **icm stop** on the ICM command line to stop any InterSystems IRIS containers that were successfully deployed.
2. Enter **icm rm** to remove those containers.
3. Execute the following command to remove InterSystems IRIS data from the storage volumes

   ```
   icm ssh -command "sudo rm -rf /intersys/*/*"
   ```

4. Enter **icm run** again.

Note that if the problem was caused by an error in a configuration file — for example, if you made a mistake typing the value for the DockerImage field — the instances.json file created during provisioning must also be updated. To make this simple and avoid further errors, after correcting the mistake you can unprovision the infrastructure and start again with the **icm provision** command, as described in the previous section.

For more information about the **icm run** command and its options, see The icm run Command in the *ICM Guide*.

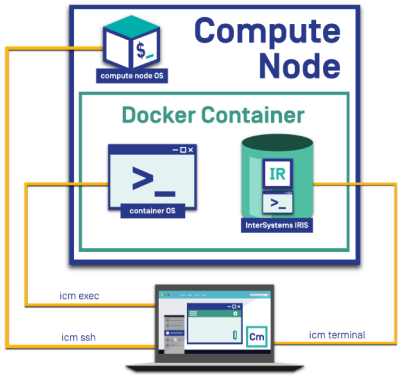
3.9 Try ICM Management Commands

ICM provides a number of commands for

- Managing provisioned infrastructure.
- Managing deployed containers.
- Interacting with services running in deployed containers, including InterSystems IRIS.

These commands are discussed in detail in the *ICM Guide* sections Infrastructure Management Commands, Container Management Commands, and Service Management Commands, and comprehensively listed in ICM Commands and Options. Some examples are provided here for you to try on your deployment. Note in particular that three commands (**icm ssh**, **icm exec**, and **icm terminal**) let you interact with the nodes of your deployment on several levels — with the node itself, with the container deployed on it, and with the running InterSystems IRIS instance inside the container, as shown in the following:

Figure 2: Interactive ICM Commands



Try these commands on your newly deployed InterSystems IRIS configuration!

1. List the provisioned compute nodes:

   ```
   icm inventory
   ```

2. Run a shell command on each of the compute nodes:

   ```
   icm ssh -command "df -k"
   ```

   When a command is executed on more than one node, the output is written to files and a list of output files provided. For example, in this case, if you deployed the sharded cluster, there is one output file for each of the three nodes.

3. Open an interactive shell on a specific compute node:

   ```
   icm ssh -role DM -interactive
   ```

   To be interactive, a command must use the **-interactive** option and specify a single node. For example, while **icm ssh** can execute a command on multiple nodes, as shown in the previous example, it can open an interactive shell on one node only with the **-interactive** option, as shown in this example. In place of the **-role DM** option, which restricts the command to that type of node (of which there is just one in your deployment), you could also use the **-machine** option to specify the name of a particular node, for example:

   ```
   icm ssh -machine LAURA-DM-TEST-0001 -interactive
   ```

4. Display the status of the deployed InterSystems IRIS containers:

   ```
   icm ps
   ```

   When multiple containers are deployed on the nodes, this command lists them all.

5. Copy a local file to the InterSystems IRIS containers, or to one of the containers:

   ```
   icm cp -localPath /Samples/ssh/ssh_notes -remotePath /home/sshuser
   icm cp -localPath /Samples/ssh/ssh_notes -remotePath /home/sshuser -role DM
   ```

6. Run a shell command within each of the InterSystems IRIS containers:

   ```
   icm exec -command "ls /intersys/"
   ```

7. Open an interactive shell (or run any shell command) within a particular InterSystems IRIS container:

   ```
   icm exec -command "bash" -role DM -interactive
   ```

8. Open a Terminal window for an InterSystems IRIS instance (always interactive, for a single instance):

```
icm terminal -machine LAURA-DM-TEST-0001
```

9. Run a SQL command against each of the InterSystems IRIS instances, or against a particular instance:

```
icm sql -command "SELECT Name FROM Security.Users" -namespace %SYS
icm sql -command "SELECT Name FROM Security.Users" -namespace %SYS -role DM
```

   The **-namespace** option determines the namespace in which the SQL executes.

3.10 Unprovision the Infrastructure

Because AWS and other public cloud platform instances continually generate charges, it is important that you unprovision your infrastructure immediately after completing this experience.

To do this, copy the **icm unprovision** command you saved from the output of **icm provision** to the ICM command line, for example:

```
$ icm unprovision -stateDir /Samples/AWS/ICM-2416821167214483124 -cleanUp
```

This command deallocates the provisioned infrastructure based on the state files created during provisioning, so the**-stateDir** option is required; you cannot simply enter **icm unprovision**. If you did not save the command from the provisioning output, you can find it in the icm.log file in your working directory (for example, /Samples/AWS/icm.log). The **-cleanUp** option deletes the state directory after unprovisioning; without it, the state directory is preserved.

4 ICM Can Do Much More Than That!

Although the ICM exploration you just completed was deliberately streamlined, it nonetheless included real-world infrastructure provisioning and service deployment. Adding to what you just accomplished is only a matter of extending your deployment definition in the configuration files and taking advantage of a host of command-line options. For example, you can

- Add shard master application servers and shard query servers to the sharded cluster by adding AM nodes and QS nodes to the definitions.json file.
- Deploy a tier of Enterprise Cache Protocol (ECP) application servers and a data server by defining AM nodes and a DM node but omitting a DS node definition.
- Deploy a mirrored DM node in any of its possible roles — the shard master data server of a sharded cluster, the data server of an ECP cluster, or a stand-alone InterSystems IRIS instance — by simply including “Mirror”: “True” in the defaults file and defining two DM nodes and an AR (arbiter) node in the definitions file.
- Add web servers (WS) and load balancers (LB or automatic) to the definitions file.
- Add Weave or Rancher monitoring to your provisioned nodes by including the Monitor field in the defaults file.
- Deploy different services on different nodes by specifying different DockerImage values for different nodes in the definitions file and issuing the **icm run** command multiple times.
- Deploy multiple services on some or all nodes by executing **icm run** multiple times while specifying different container names and images, including custom and third-party images, on the command line with the **-container** and **-image** options.
- Extend ICM’s functionality through the use of third-party tools and in-house scripting, further increasing automation and reducing effort.
- Deploy services on existing virtual and physical clusters.

- Provision infrastructure without deploying services, simply by stopping with the **ICM provision** command.

All of these possibilities and more are covered in the ICM Guide.

5 Learn More About ICM

To learn more about ICM, see

- What is InterSystems Cloud Manager? (video)
- Pat and Tracy Try ICM (video)
- Getting Started with ICM (online course)
- InterSystems IRIS Experience: Cloud
- InterSystems Cloud Manager Guide
- First Look: Running InterSystems IRIS in Docker Containers
- First Look: Deploying an InterSystems IRIS Sharded Cluster