InterSystems™
IRIS Data Platform

First Look: InterSystems Cloud Manager

Version 2020.3
2021-02-04

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

First Look: InterSystems Cloud Manager
InterSystems IRIS Data Platform Version 2020.3 2021-02-04
Copyright © 2021 InterSystems Corporation
All rights reserved.

Table of Contents

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 applications based on InterSystems IRIS® data platform.

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

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

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?

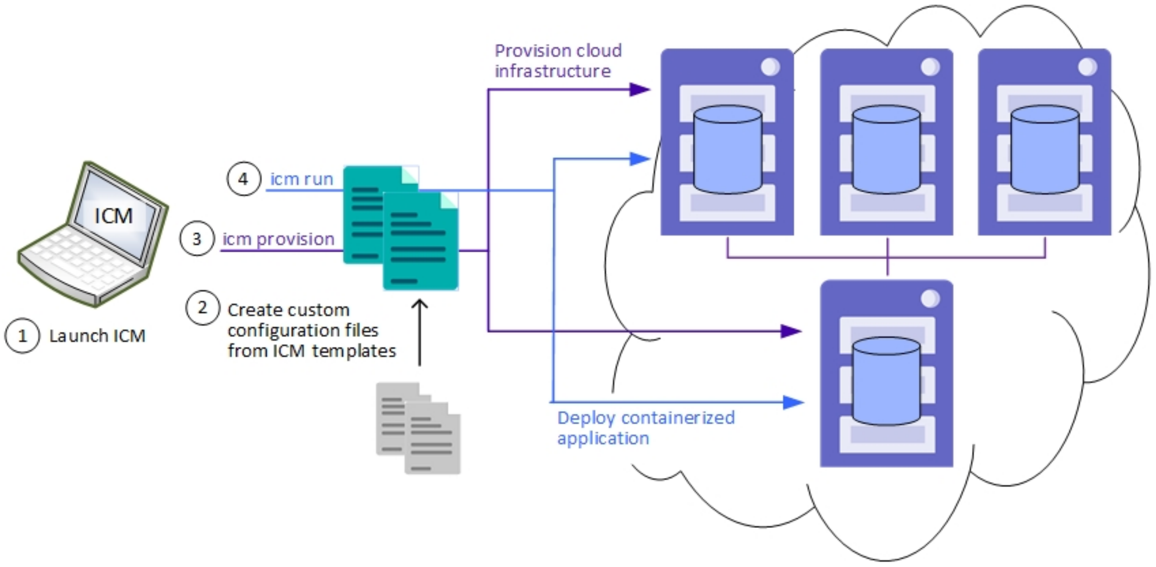
Guided by your input in plain-text configuration files, ICM provisions your infrastructure using the popular Terraform IaC tool from Hashicorp and configures the provisioned host nodes as needed. In the next phase, ICM deploys InterSystems IRIS and your application in Docker containers, as well as other 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 container image that includes everything you need. Download and run a container from 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 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


Provision cloud infrastructure
4 icm run
3 icm provision
ICM
2 Create custom configuration files from ICM templates
1 Launch ICM
Deploy containerized application

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, and links are provided where appropriate.

These instructions assume you have the following:

- Container-specific InterSystems IRIS sharding license and access to InterSystems software downloads.
- An account on Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure (Azure), or Tencent Cloud (Tencent).

Many of the properties to be specified in your configuration files are common across these cloud platforms, but others are platform-specific. Detailed information about these differences can be found in the Provider-Specific Parameters section of the "ICM Reference" chapter of the *ICM Guide*.

3.1 Install Docker

ICM is provided as a container 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. For information about installing Docker on your platform, see Install Docker in the Docker documentation.

Important: ICM is supported on Docker Enterprise Edition and Community Edition version 18.09 and later; Enterprise Edition only is supported for production environments.

3.2 Download the ICM Image

To use ICM, you need to download the ICM image to the system you are working on; this requires you to identify the registry from which you will download it and the credentials you need for access. Similarly, for ICM to deploy InterSystems IRIS and other InterSystems components, it requires this information for the images involved. The registry from which ICM downloads images must be accessible to the cloud provider you use (that is, not behind a firewall), and for security must require ICM to authenticate using the credentials you provide to it. For details on identifying the registries involved and downloading the ICM image, see Downloading the ICM Image in the *InterSystems Cloud Manager Guide*.

Note: The major versions of the image from which you launch ICM and the InterSystems images you deploy must match. For example, you cannot deploy a 2019.4 version of InterSystems IRIS using a 2019.3 version of ICM.

For a brief introduction to the use of InterSystems IRIS in containers, including a hands-on experience, see *First Look: InterSystems Products in Containers*; for detailed information about deploying InterSystems IRIS and InterSystems IRIS-based applications in containers using methods other than ICM, see *Running InterSystems IRIS in Containers*.

3.3 Launch ICM

To launch ICM on the command line, use the following **docker run** command to pull (download) the ICM image from the registry, create a container from it, and start the container, for example:

```
docker run --name icm -it --cap-add SYS_TIME
  containers.intersystems.com/intersystems/icm:2020.3.0.221.0
```

The /Samples directories in the ICM container, one for each provider (/AWS, /GCP, and so on) intended to make it easy for you to provision and deploy using ICM out of the box; you can use the sample configuration files in one of these directories and provision and deploy from that directory as well. Because these directories are inside the ICM container, however, data in them is lost when the container is removed, including your configuration files and the state directory ICM creates and uses to manage the provisioned infrastructure. In production, the best practice is to use locations outside the container to store this data by mounting external volumes when you launch ICM (see Manage data in Docker in the Docker documentation).

3.4 Get a Cloud Provider Account and Credentials

ICM can provision and deploy on four public cloud platforms: Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure (Azure), and Tencent Cloud (Tencent).

Log into your AWS, GCP, Azure, or Tencent account. If neither you nor your employer have one yet, you can go to the AWS, GCP, Azure, or Tencent portal page to quickly create a free account.

For information about obtaining the account credentials ICM needs to authenticate to each cloud provider, see Security-Related Parameters in the “ICM Reference” chapter of 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 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 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 Obtain Security-Related Files, ICM Security, and 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.

The /Samples directory in the ICM container provides sample configuration files for all four cloud providers. For example, sample configuration files for AWS are located in the /Samples/AWS directory. To customize these files, select the provider you want to use, then modify the sample defaults and definitions files for that provider as described in the tables that follow. To do this, you can 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
```

Important: Both field names and values are case-sensitive; for example, to select AWS as the cloud provider you must include “Provider”:”AWS” in the defaults file, not “provider”:”AWS”, “Provider”:”aws”, and so on.

3.6.1 Customize defaults.json

The following table details the minimum customization needed for the defaults files provided in the /Samples files, plus suggestions for optional changes. Any setting not covered here can be left as is in the sample file.

Each setting is linked to the relevant table in the ICM Configuration Parameters section of the “ICM Reference” chapter of the *ICM Guide*, where you will find more detailed descriptions; once there, search for the parameter you want. The rightmost column shows each parameter or set of parameters as they appear in the sample defaults files.

To review all of the settings that are common to all providers, see General Parameters in that section; to review all that are specific to a particular provider, see Provider-Specific Parameters.

Setting	Description	Entry in sample defaults.json file
Provider	Identifies the cloud infrastructure provider; keep the value in the sample defaults.json.	`“Provider”:[”AWS”|"GCP"|"Azure"|"Tencent"],`
Label	Field in naming scheme for provisioned nodes, *Label-Role-Tag-NNNN* (see Role below); update to identify the owner and purpose of the deployment, for example use your company name and “TEST” to create node names like **Acme-DATA-TEST-0001**.	`"Label": "Sample",`
Tag	(see Label above)	`"Tag": "TEST",`
DataVolumeSize	Size of the persistent data volume to be provisioned with each node, which can be overridden in individual node definitions in the definitions.json file; accept the value in the sample defaults.json unless you are provisioning on Tencent, in which case change it to 60.	`"DataVolumeSize": "10",`
SSHUser	Nonroot account with **sudo** access on provisioned nodes, used by ICM for access; you can keep the default in the sample defaults.json, but if you change the machine image (below) on AWS or Tencent, you may need to update this entry.	`"SSHUser": "ubuntu",` (AWS & Tencent) `"SSHUser": "sample",` (GCP & Azure)

SSHPublicKey	Location of the SSH public key. If you used the key generation scripts discussed in Generate Security Keys, the keys are located in the directories specified in the sample files, so make no changes; if you are providing your own keys, use **docker cp** to copy them from the local file system to these locations.	`"SSHPublicKey":`
		`"/Samples/ssh/insecure-ssh2.pub",`

SSHPrivateKey	Location of the SSH private key; see SSHPublicKey above.	`"SSHPrivateKey": "/Samples/ssh/insecure",`
TLSKeyDir	Location of TILS files; see SSHPublicKey above.	`"TLSKeyDir": "/Samples/tls/",`

Setting	Description	Entry in sample defaults.json file
DockerVersion	The Docker version to be installed on provisioned nodes; keep the default value.	`"DockerVersion": "5:19.03.8~3-0~ubuntu-bionic",`
DockerImage	The image to be deployed on provisioned nodes; update to reflect the repository and image information you identified in Identify Docker Repository and Credentials.	`"DockerImage": "containers.intersystems.com/intersystems/iris:2020.3.0.221.0",`

DockerUser-name
 	
Docker-Password
	The credentials needed to download image specified by DockerImage if in a private repository; update to reflect the repository information and credentials you identified in Identify Docker Repository and Credentials.	`"DockerUsername": "xxxxxxxxxxxx",`
		
		`"DockerPassword": "xxxxxxxxxxxx",`

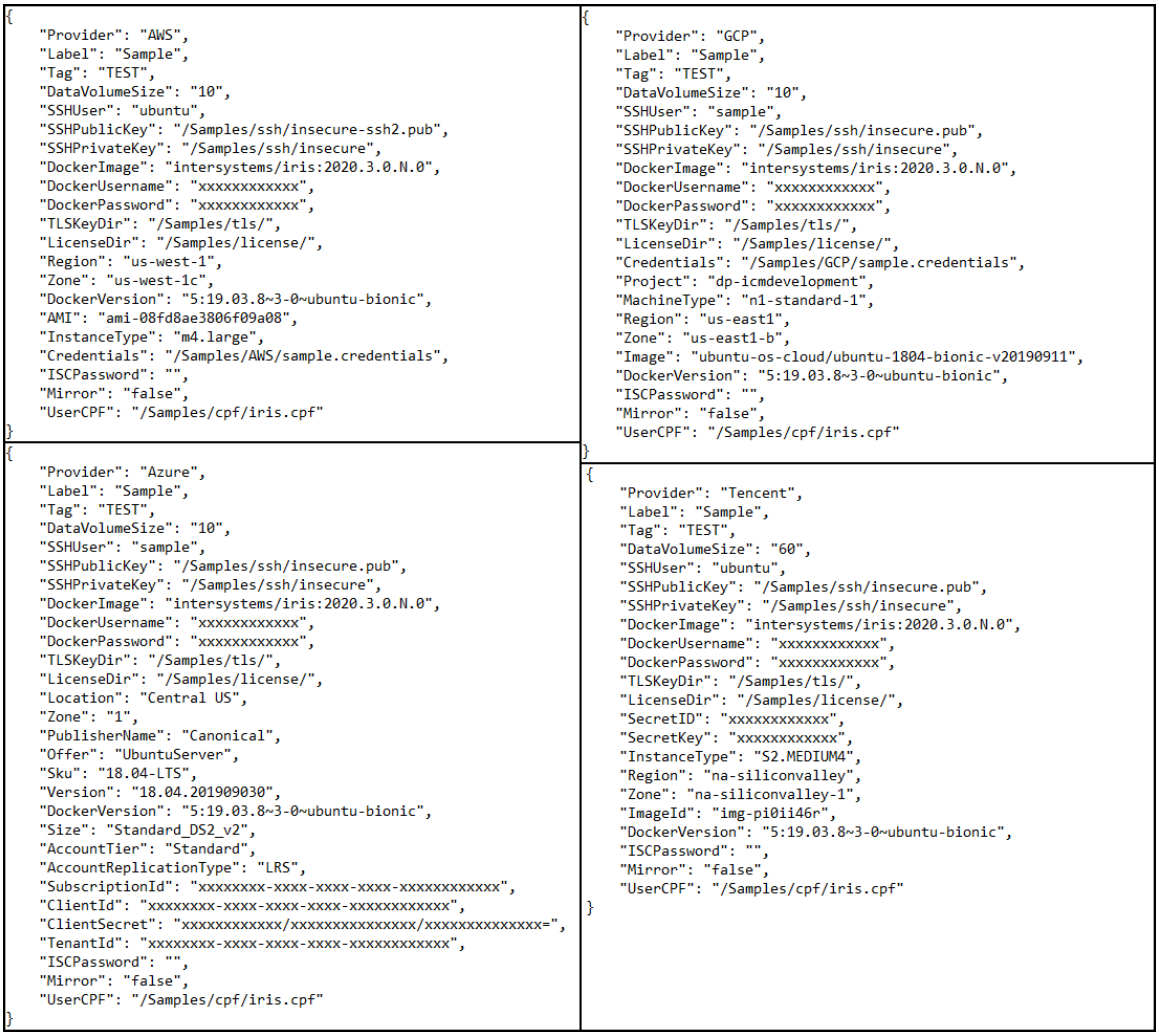
LicenseDir	Staging directory for InterSystems IRIS licenses; place your container-specific InterSystems IRIS sharding license in this directory.	`“LicenseDir”: “/Samples/Licenses”,`
Region, Location (Azure)	Geographical region of provider’s compute resources in which to provision infrastructure; accept the default in the sample defaults.json or select another combination of region and zone (below) from the provider.	`"Region": "us-west-1",` (AWS) `"Region": "us-east1",` (GCP) `"Location": "Central US",` (Azure) `"Region": "na-siliconvalley",` (Tencent)
Zone	Availability zone within the selected region (above); accept the default in the sample defaults.json or select another combination of region and zone from the provider.	`"Zone": "us-west-1c",` (AWS) `"Zone": "us-east1-b",` (GCP) `"Zone": "1",` (Azure) `"Zone": "na-siliconvalley-1",` (Tencent)

Machine image (provider-specific)	Template for platform and OS of provisioned nodes; accept the defaults in the sample defaults.json or select a different combination of machine image and instance type (below) from the provider.	`"AMI": "ami-c509eda6",` (AWS)
		
		`"Image": "ubuntu-os-cloud/ubuntu-1804-bionic-v20180617",` (GCP)
		
		`"PublisherName": "Canonical",` (Azure)
		
		`"Offer": "UbuntuServer",(Azure)`
		
		`"Sku": "18.04-LTS",(Azure)`
		
		`"Version": "18.04.201804262",(Azure)`
		
		`"ImageID": "img-pi0ii46r",` (Tencent)

Setting	Description	Entry in sample defaults.json file
Instance type (provider-specific)	Template for compute resources of provisioned nodes; accept the value in the sample defaults.json or select a different combination of machine image (above) and instance type from the provider.	`"InstanceType": "m4.large",` (AWS) `"MachineType": "n1-standard-1",` (GCP) `"Size": "Standard_DS2_v2",` (Azure) `"InstanceType": "S2.MEDIUM4",` (Tencent)
Credentials and account settings (provider-specific)	Files or IDs needed by ICM to authenticate to the provider, differing by provider; update to specify the needed file locations or IDs for your account (for instructions click the provider link).	`“Credentials”:”/Samples/AWS/credentials”,` (AWS) `"Credentials": "/Samples/GCP/sample.credentials",` (GCP) `"Project": "dp-icmdevelopment",` (GCP) `"SubscriptionId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",` (Azure) `"ClientId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",` (Azure) `"ClientSecret": "xxxxxxxxxxxx/xxxxxxxxxxxxxxx/xxxxxxxxxxxxxxx=",` (Azure) `"TenantId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",` (Azure) `"SecretID": "xxxxxxxxxxxx",` (Tencent) `"SecretKey": "xxxxxxxxxxxx",` (Tencent)
ISCPassword	Password for predefined accounts in deployed InterSystems IRIS images; to provide the password interactively with masked input during the deployment phase (as recommended for security), remove this field, otherwise change to your preferred password.	`"ISCPassword": "",`
Mirror	Determines whether InterSystems IRIS instances deployed on DATA, DM, and DS nodes are configured as mirrors; retain the default.	`“Mirror”: “false”`
UserCPF	Specifies the CPF merge file to be used to override initial CPF settings for deployed instances. (Remove this entry if you are not familiar with the CPF merge feature or the CPF; for information about CPF merge, see Deploying with Customized InterSystems IRIS Configurations in the “ICM Reference” chapter of the *ICM Guide*.	`"UserCPF": "/Samples/cpf/iris.cpf"`

The following illustration shows the contents of the sample defaults.json files for AWS, GCP, Azure, and Tencent:

Figure 2: Sample ICM defaults files for cloud providers



```
{
    "Provider": "AWS",
    "Label": "Sample",
    "Tag": "TEST",
    "DataVolumeSize": "10",
    "SSHUser": "ubuntu",
    "SSHPublicKey": "/Samples/ssh/insecure-ssh2.pub",
    "SSHPrivateKey": "/Samples/ssh/insecure",
    "DockerImage": "intersystems/iris:2020.3.0.N.0",
    "DockerUsername": "xxxxxxxxxxxx",
    "DockerPassword": "xxxxxxxxxxxx",
    "TLSKeyDir": "/Samples/tls/",
    "LicenseDir": "/Samples/license/",
    "Region": "us-west-1",
    "Zone": "us-west-1c",
    "DockerVersion": "5:19.03.8~3-0~ubuntu-bionic",
    "AMI": "ami-08fd8ae3806f09a08",
    "InstanceType": "m4.large",
    "Credentials": "/Samples/AWS/sample.credentials",
    "ISCPassword": "",
    "Mirror": "false",
    "UserCPF": "/Samples/cpf/iris.cpf"
}
```

```
{
    "Provider": "GCP",
    "Label": "Sample",
    "Tag": "TEST",
    "DataVolumeSize": "10",
    "SSHUser": "sample",
    "SSHPublicKey": "/Samples/ssh/insecure.pub",
    "SSHPrivateKey": "/Samples/ssh/insecure",
    "DockerImage": "intersystems/iris:2020.3.0.N.0",
    "DockerUsername": "xxxxxxxxxxxx",
    "DockerPassword": "xxxxxxxxxxxx",
    "TLSKeyDir": "/Samples/tls/",
    "LicenseDir": "/Samples/license/",
    "Credentials": "/Samples/GCP/sample.credentials",
    "Project": "dp-icmdevelopment",
    "MachineType": "n1-standard-1",
    "Region": "us-east1",
    "Zone": "us-east1-b",
    "Image": "ubuntu-os-cloud/ubuntu-1804-bionic-v20190911",
    "DockerVersion": "5:19.03.8~3-0~ubuntu-bionic",
    "ISCPassword": "",
    "Mirror": "false",
    "UserCPF": "/Samples/cpf/iris.cpf"
}
```

```
{
    "Provider": "Azure",
    "Label": "Sample",
    "Tag": "TEST",
    "DataVolumeSize": "10",
    "SSHUser": "sample",
    "SSHPublicKey": "/Samples/ssh/insecure.pub",
    "SSHPrivateKey": "/Samples/ssh/insecure",
    "DockerImage": "intersystems/iris:2020.3.0.N.0",
    "DockerUsername": "xxxxxxxxxxxx",
    "DockerPassword": "xxxxxxxxxxxx",
    "TLSKeyDir": "/Samples/tls/",
    "LicenseDir": "/Samples/license/",
    "Location": "Central US",
    "Zone": "1",
    "PublisherName": "Canonical",
    "Offer": "UbuntuServer",
    "Sku": "18.04-LTS",
    "Version": "18.04.201909030",
    "DockerVersion": "5:19.03.8~3-0~ubuntu-bionic",
    "Size": "Standard_DS2_v2",
    "AccountTier": "Standard",
    "AccountReplicationType": "LRS",
    "SubscriptionId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "ClientId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "ClientSecret": "xxxxxxxxxxxx/xxxxxxxxxxxxxxx/xxxxxxxxxxxxxx=",
    "TenantId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "ISCPassword": "",
    "Mirror": "false",
    "UserCPF": "/Samples/cpf/iris.cpf"
}
```

```
{
    "Provider": "Tencent",
    "Label": "Sample",
    "Tag": "TEST",
    "DataVolumeSize": "60",
    "SSHUser": "ubuntu",
    "SSHPublicKey": "/Samples/ssh/insecure.pub",
    "SSHPrivateKey": "/Samples/ssh/insecure",
    "DockerImage": "intersystems/iris:2020.3.0.N.0",
    "DockerUsername": "xxxxxxxxxxxx",
    "DockerPassword": "xxxxxxxxxxxx",
    "TLSKeyDir": "/Samples/tls/",
    "LicenseDir": "/Samples/license/",
    "SecretID": "xxxxxxxxxxxx",
    "SecretKey": "xxxxxxxxxxxx",
    "InstanceType": "S2.MEDIUM4",
    "Region": "na-siliconvalley",
    "Zone": "na-siliconvalley-1",
    "ImageId": "img-pi0ii46r",
    "DockerVersion": "5:19.03.8~3-0~ubuntu-bionic",
    "ISCPassword": "",
    "Mirror": "false",
    "UserCPF": "/Samples/cpf/iris.cpf"
}
```

3.6.2 Customize definitions.json

The sample definitions.json file in the /Samples directories, which is the same for all providers),defines a sharded cluster with two data nodes and two compute nodes, as shown in the following:.

```
[
   {
    "Role": "DATA",
    "Count": "2",
    "LicenseKey": "ubuntu-sharding-iris.key"
    },
    {
    "Role": "COMPUTE",
    "Count": "2",
    "StartCount": "3",
    "LicenseKey": "ubuntu-sharding-iris.key"
    }
]
```

The Role field identifies the node types being provisioned, which in this case are DATA and COMPUTE. The Count field indicates how many of that type to provision; StartCount starts numbering at 0003 for the COMPUTE nodes. The LicenseKey field indicates the name of an InterSystems IRIS license file located in the directory specified by the LicenseDir field in the defaults file.

For this exercise, remove the COMPUTE definition, leaving only the DATA definitions, as follows:

```
[
   {
    "Role": "DATA",
    "Count": "2",
    "LicenseKey": "ubuntu-sharding-iris.key"
   }
]
```

When a sharded cluster is deployed, all nodes must have a sharding license. Use **docker cp** to copy a sharding license to the location within the container specified by LicenseDir, such as /Samples/Licenses, and update the LicenseKey setting in the DATA node definition to specify license key to use.

Those are the only definitions.json changes required to provision the basic sharded cluster. To provision a stand-alone InterSystems IRIS instance instead, use this definition:

```
[
   {
    "Role": "DM",
    "Count": "1",
    "LicenseKey": "standard-iris.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 directory for the provider you have chosen, for example /Samples/AWS, and issue this command:

```
icm provision
```

The **icm provision** command allocates and configures host 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 host 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                                          Region    Zone
-------             ----------     --------                                          ------    ----
Acme-DATA-TEST-0001 00.53.183.209  ec2-00-53-183-209.us-west-1.compute.amazonaws.com  us-west-1 c
Acme-DATA-TEST-0002 00.53.183.185  ec2-00-53-183-185.us-west-1.compute.amazonaws.com  us-west-1 c
To destroy: icm unprovision [-cleanUp] [-force]
```

Important: Copy the **icm unprovision** command line 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.

The files and directories needed to both manage and unprovision your infrastructure are inside the ICM container, and thus are lost when you remove it. For this reason, you must not remove the ICM container until you are finished with the infrastructure and have successfully unprovisioned it. (In production, the best practice is to use locations outside the container to store this data by mounting external volumes when you launch ICM (see Manage data in Docker in the Docker documentation).

Detailed information about errors that occur during the provisioning and deployment phases is 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, or to an error in a configuration file, 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. For more information, see Reprovisioning the Infrastructure in the “Using ICM” chapter of the *ICM Guide*.

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

3.8 Deploy InterSystems IRIS

In addition to downloading images on the provisioned host 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, using different DockerImage fields in the node definitions in the definitions file (instead of once for all nodes in the defaults file, as here) lets you run different images on different node types, and beyond that you can execute the **icm run** command multiple times with certain options to deploy multiple containers with unique names on each provisioned node or only on specified nodes.

After InterSystems IRIS starts on each node, ICM does whatever configuration of the instance is needed — for example, resetting the password, 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:

```
$ icm run -definitions definitions_cluster.json
Executing command 'docker login' on ACME-DATA-TEST-0001...
...output in /Samples/AWS/state/ACME-DATA-TEST/ACME-DATA-TEST-0001/docker.out
...
Pulling image intersystems/iris:2020.3.0.221.0 on ACME-DATA-TEST-0001...
...pulled ACME-DATA-TEST-0001 image intersystems/iris:2020.3.0.221.0
...
Creating container iris on ACME-DATA-TEST-0002...
...
Management Portal available at:
http://ec2-00-53-183-209.us-west-1.compute.amazonaws.com:52773/csp/sys/UtilHome.csp
```

in this case, the provided link is for data node 1, 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.

As with **icm provision**, if **icm run** does not complete successfully due to timeouts and internal errors on the provider side, you can issue the command again; in most cases, deployment will succeed on repeated tries. If the error persists, however, and requires manual intervention — for example, if it is caused by an error in one of the configuration files — after fixing the problem you may need to delete the durable %SYS data on the node or nodes affected, as described in Redeploying Services in the “Using ICM” chapter of the *ICM Guide*, before reissuing the **icm run** command. 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.

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

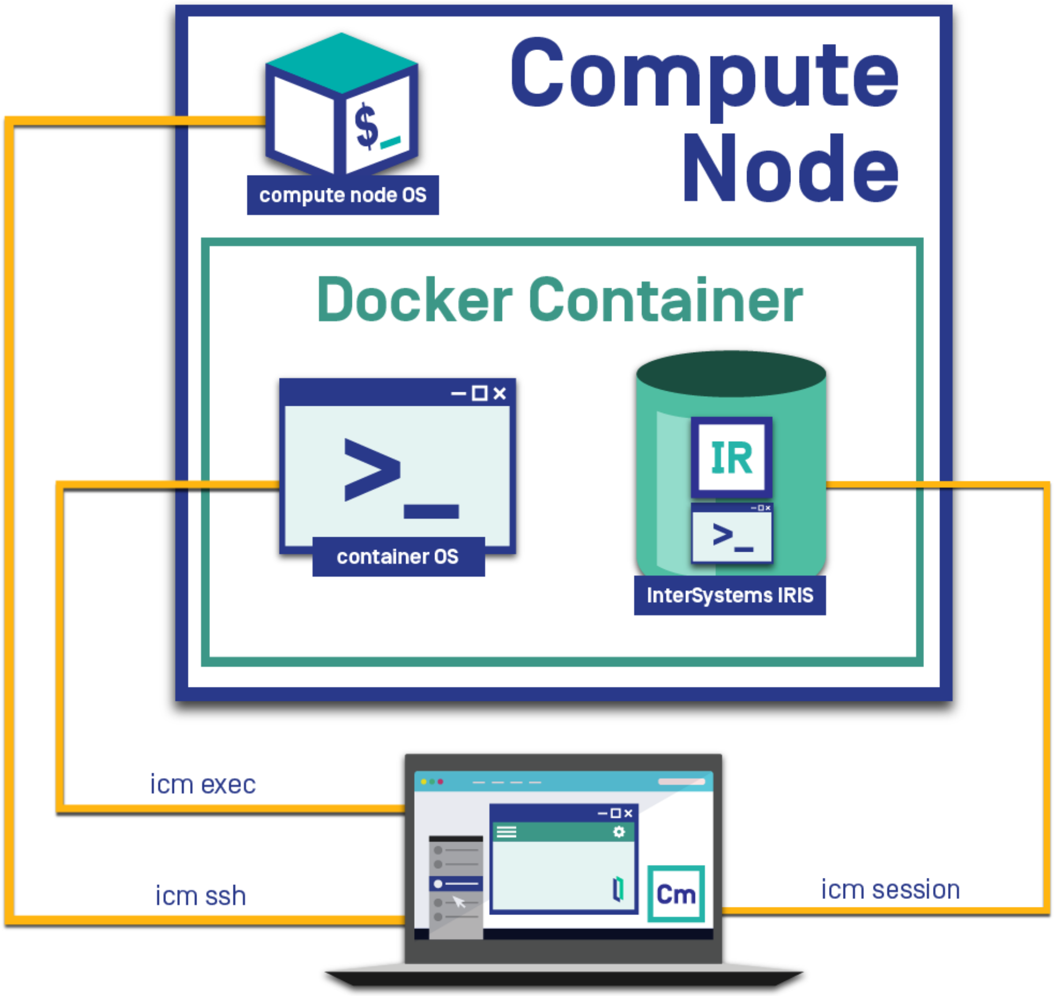
3.9 Try ICM Management Commands

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 session**) 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 3: Interactive ICM Commands



Try these commands on your newly deployed InterSystems IRIS configuration!

1. List the provisioned host nodes:

   ```
   icm inventory
   ```

2. Run a shell command on each of the host 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 host node:

   ```
   icm ssh -role DATA -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 Acme-DATA-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 /irissys/"
   ```

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

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

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

   ```
   icm session -machine Acme-DATA-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"
   icm sql -command "SELECT Name FROM Security.Users" -machine Acme-DATA-TEST-0002
   ```

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 -cleanUp
```

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

- Deploy a distributed cache cluster of application servers and a data server by defining AM nodes and a DM node, or a standalone InterSystems IRIS instance by defining just a DM node.
- Deploy mirrored DATA nodes or a mirrored DM node by simply including "Mirror": "True" in the defaults file and defining an even number of DATA nodes or two DM nodes and an AR (arbiter) node in the definitions file.
- Deploy a sharded cluster that includes COMPUTE nodes to separate the query and data ingestion workloads while maintaining the advantages of parallel processing and distributed caching, improving the performance of both.
- Add web servers (WS) and load balancers (LB or automatic) to the definitions 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 and about using InterSystems IRIS in containers, see

- InterSystems Cloud Manager Overview (video)
- The Benefits of InterSystems Cloud Manager (video)
- Experience InterSystems IRIS in the Cloud (online experience)
- InterSystems Cloud Manager Guide
- First Look: InterSystems Products in Containers
- Running InterSystems Products in Containers