InterSystems IRIS デモ：InterSystems Cloud Manager

Version 2024.1
2024-06-03

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

InterSystems IRIS デモ : InterSystems Cloud Manager
InterSystems IRIS Data Platform　Version 2024.1　2024-06-03

インターシステムズ社の製品に関するサポートやご質問は、以下にお問い合わせください：

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

目次

図一覧

InterSystems IRIS デモ：InterSystems Cloud Manager

重要　InterSystems IRIS リリース 2023.2 以降、ICM は非推奨になっています。ICM は今後のリリースで削除されます。

ここでは、InterSystems Cloud Manager (ICM) について説明します。ICM は、InterSystems IRIS® データ・プラットフォームを基盤とするアプリケーション向けのエンドツーエンドのクラウド・プロビジョニングおよび導入ソリューションです。また、ICM を使用してパブリック・クラウドでインフラストラクチャをプロビジョニングして、そのインフラストラクチャに InterSystems IRIS を導入するデモを紹介します。

1 ICM の利点

クラウドの時代へようこそ。クラウドがもたらす機会に注目しながらも、その課題に慎重になっていませんか。具体的には次のような状況にありませんか。

- クラウドを活用したいが、複雑な移行にリソースをつぎ込みたくないと考えている。
- 既にクラウドを活用しているが、さまざまなソフトウェア環境にわたって管理しやすい方法でアプリケーションを導入し、そのバージョンを管理するための方法を模索している。
- ソフトウェア・ファクトリにおける継続的な統合と配布、および導入プロセスにおける DevOps アプローチを実現したいと考えている。つまり、従来の手法、ライブラリへの依存、システム・ドリフト、手動アップグレード、およびその他のオーバーヘッドの制限やリスクから解放されたいと考えている。

このような皆様には ICM が役立ちます。ICM なら、シンプルかつ直感的にクラウド・インフラストラクチャをプロビジョニングし、そのインフラストラクチャにサービスを導入して、今すぐクラウドに移行できます。大規模な開発や設備更新の必要はありません。Infrastructure as Code (IaC) およびコンテナ化された導入の利点を生かして、InterSystems IRIS を基盤とするアプリケーションを Google、Amazon、Azure などのパブリック・クラウド・プラットフォームや VMware vSphere のプライベート・クラウドに簡単に導入することができます。必要なものを定義し、いくつかのコマンドを発行すればよいだけです。その他の処理は ICM が行います。

クラウド・インフラストラクチャまたはコンテナ、あるいはその両方を既に使用している場合でも、ICM は、本来なら手動で行う必要がある多数のステップを自動化するので、アプリケーションのプロビジョニングと導入に必要な時間と労力が大幅に節約されます。

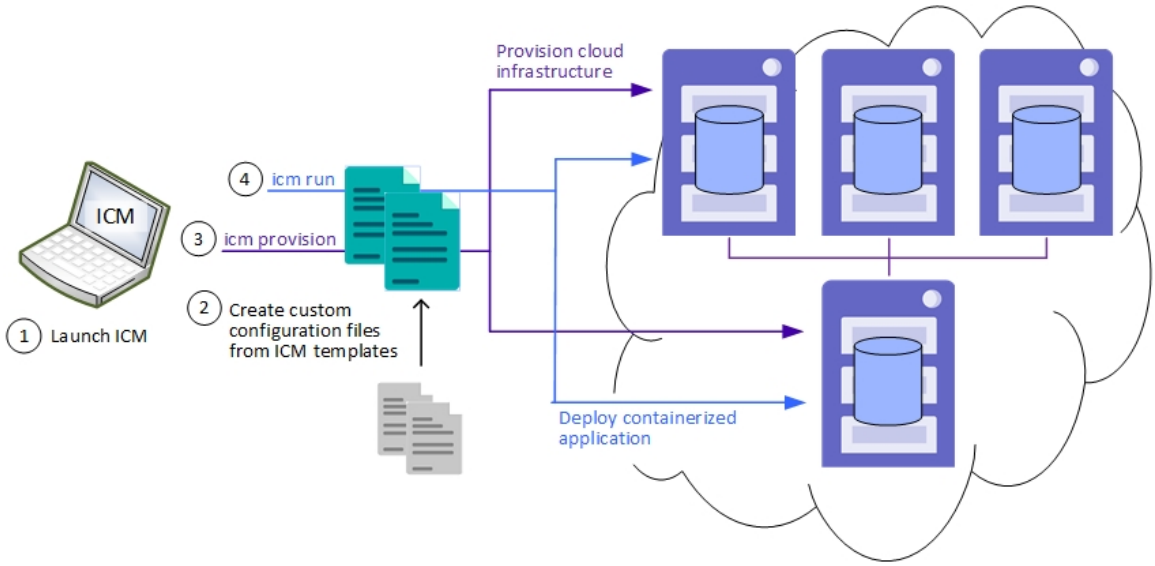
2 ICM の仕組み

ICM は、プレーン・テキストの構成ファイルに入力された情報に基づいて、広く使用されている Hashicorp の Terraform IaC ツールを使用してインフラストラクチャをプロビジョニングし、プロビジョニングされたホスト・ノードを必要に応じて構成します。次のフェーズでは、InterSystems IRIS とアプリケーション、および必要に応じて他のサービスを Docker コンテナに導入します。目的の導入に必要な InterSystems IRIS 構成のすべてが自動的に実行されます。ICM は、コンテナ化されたサービスを既存の仮想クラスタと物理クラスタに導入することもできます。

ICM 自体が、必要なものすべてを含むコンテナ・イメージとして提供されます。インターシステムズから ICM イメージをダウンロードしてコンテナを実行し、コマンド行を開けば、準備完了です。ICM なら、以下の要素を備えているので簡単です。

- テンプレートとして使用することで目的の導入をすばやく定義できるサンプル構成ファイル
- アプリケーションを追加できる InterSystems IRIS イメージ
- 各タスクに応じて簡単に使えるコマンド
- プロビジョニングされたノードやそれらに導入されたサービスを管理および操作するためのさまざまな方法

図 1: ICM で実現が容易に



3 体験 : ICM によるクラウドへの InterSystems IRIS の導入

ICM は、ユーザに代わって多くのタスクを実行し、必要なものを正確に導入するためのさまざまなオプションを提供するため、プロダクション環境で使用するには、ある程度の計画と準備が必要です (ただし、手動による方法と比べると、はるかに少なくて済みます)。しかし、プロビジョニングおよび導入プロセスはシンプルで、ICM がユーザに代わって多くの判断を下すことができます。この演習は、ICM の仕組み、および ICM を使用することで簡単に InterSystems IRIS 構成を Amazon Web Services (AWS) に導入できることを自分自身で確かめられるように設計されています。すぐに終わる作業ではありませんが、この演習にはそれほど時間がかからないので、機会が生じたときに段階的に実行できます。

詳細にとらわれずに ICM の基本機能を体験していただくため、以下の手順はシンプルなものにしてあります。例えば、できるだけ多く既定の設定を使用するように指定されています。ただし、ICM をプロダクション・システムで使用する際には、特にセキュリティに関して (それに限定されない)、多くの異なる処理が必要になります。したがって、ここに示す ICM の操作例と実際の操作を混同しないようにしてください。このドキュメントの最後に示すソースでは、ICM をプロダクション環境で使用するために必要な良策が提供されています。"InterSystems Cloud Manager ガイド" には、ICM を使用するための包括的な情報および手順が提示されており、必要に応じてリンクも提供されています。

これらの手順では、以下が前提となっています。

- コンテナ固有の InterSystems IRIS シャード・ライセンスを有し、インターシステムズ・ソフトウェアのダウンロードにアクセスできる。
- Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure (Azure)、または Tencent Cloud (Tencent) のアカウントを有している。

構成ファイルに指定されるプロパティの多くは、これらのクラウド・プラットフォーム間で共通していますが、プラットフォーム固有のものもあります。これらの相違に関する詳細は、"InterSystems Cloud Manager ガイド" の "ICM リファレンス" の章にある "プロバイダ固有のパラメータ" を参照してください。

3.1 Docker のインストール

ICM は、必要なものをすべて含むコンテナ・イメージとして提供されます。そのため、ICM を起動する Linux、macOS、または Microsoft Windows システムの要件は、Docker がインストールされ、Docker デーモンが実行されており、システムがインターネットに接続されていることだけです。Docker をプラットフォームにインストールする手順の詳細は、Docker のドキュメントの "Docker のインストール" を参照してください。

重要　ICM は、Docker Enterprise Edition および Community Edition バージョン 18.09 以降でサポートされています。プロダクション環境についてサポートされているのは Enterprise Edition のみです。

3.2 ICM イメージのダウンロード

ICM を使用するには、作業中のシステムに ICM イメージをダウンロードする必要があります。このためには、ダウンロード元のレジストリと、アクセスに必要な認証情報を特定することが必要になります。同様に、ICM で InterSystems IRIS およびその他のインターシステムズのコンポーネントを導入するには、関係するイメージのこの情報が必要です。ICM がイメージをダウンロードするレジストリは、使用するクラウド・プロバイダからアクセス可能である (つまり、ファイアウォールの背後にない) ことが必要です。また、セキュリティを確保するために、ICM はユーザから提供された認証情報を使用して認証する必要があります。関係するレジストリの識別と、ICM イメージのダウンロードに関する詳細は、"InterSystems Cloud Manager ガイド" の "ICM イメージのダウンロード" を参照してください。

注釈　ICM の起動元のイメージと導入するインターシステムズのイメージのメジャー・バージョンが一致している必要があります。例えば、2019.3 バージョンの ICM を使用して 2019.4 バージョンの InterSystems IRIS を導入することはできません。

イメージのダウンロードを含め、コマンド行で InterSystems IRIS コンテナをすばやく実行する方法は、"InterSystems IRIS の基礎 : InterSystems IRIS コンテナの実行" を参照してください。ICM 以外の方法を使用してコンテナに InterSystems IRIS および InterSystems IRIS ベースのアプリケーションを導入する方法は、"コンテナ内でのインターシステムズ製品の実行" を参照してください。

3.3 ICM の起動

コマンド行で ICM を起動するには、以下の docker run コマンドを使用してレジストリから ICM イメージをプル (ダウンロード) し、そのイメージからコンテナを作成して、そのコンテナを起動します。

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

注釈　このドキュメントで示しているイメージ・タグは、例として紹介しているだけです。InterSystems Container Registry (ICR) に移動して、最新のリポジトリとタグを参照してください。

ICM コンテナ内の /Samples ディレクトリは、プロバイダ (/AWS、/GCP など) ごとに提供されており、ICM をそのまますぐに使用してプロビジョニングと導入を簡単に実行できるようになっています。これらのディレクトリのいずれかにあるサンプル構成ファイルを使用して、そのディレクトリからプロビジョニングおよび導入を実行することもできます。ただし、これらのディレクトリは ICM コンテナ内にあるため、コンテナが削除されるとその中のデータも失われます。こういったデータには、構成ファイルおよび ICM で作成されプロビジョニング済みインフラストラクチャの管理に使用される状態ディレクトリも含まれます。プロダクションでは、ベスト・プラクティスとして、ICM の起動時に外部ボリュームをマウントして、このデータの格納にコンテナ外部の場所を使用することをお勧めします (Docker ドキュメントの "Manage data in Docker" を参照してください)。

3.4 クラウド・プロバイダのアカウントおよび資格情報の入手

ICM は、Amazon Web Services (AWS)、Google Cloud Platform (GCP)、Microsoft Azure、および Tencent Cloud (Tencent) の 4 つのパブリック・クラウド・プラットフォームでプロビジョニングおよび導入を行うことができます。

AWS、GCP、Azure、または Tencent のアカウントにログインします。まだアカウントを持っておらず、勤務先にもアカウントがない場合は、AWS、GCP、Azure、または Tencent のポータル・ページに移動して、すばやく無料アカウントを作成できます。

各クラウド・プロバイダに対する認証で ICM が必要とするアカウント資格情報の取得の詳細は、"InterSystems Cloud Manager ガイド" の "ICM リファレンス" の章にある "セキュリティ関連のパラメータ" を参照してください。

3.5 セキュリティ・キーの生成

ICM は、インフラストラクチャをプロビジョニングする対象のクラウド・プロバイダ、プロビジョニングされたノード上の Docker、およびコンテナの導入後はいくつかの InterSystems IRIS サービスとの間で、セキュリティ保護された通信を行います。既にクラウドの認証情報をダウンロードまたは確認しましたが、セキュリティ保護された SSH および TLS 通信を有効にするには、他のファイルも必要です。

ICM に付属している 2 つのスクリプトを使用して、必要なファイルを作成できます。**keygenSSH.sh** スクリプトは必要な SSH ファイルを作成し、ICM コンテナ内のディレクトリ **/Samples/ssh** に配置します。**keygenTLS.sh** スクリプトは必要な TLS ファイルを作成し、**/Samples/tls** に配置します。

ICM コマンド行でこれらのスクリプトを実行するには、以下のように入力します。

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

これらのスクリプトによって生成されるファイルの詳細は、"InterSystems Cloud Manager ガイド" の "セキュリティ関連ファイルの入手"、"ICM セキュリティ"、および "セキュリティ関連のパラメータ" を参照してください。

重要　これらの鍵は、このデモやその他のテストで便利に使用できるように生成されます。プロダクションでは、企業のセキュリティ・ポリシーに沿って必要な鍵を生成または入手する必要があります。これらのスクリプトによって生成される鍵、およびクラウド・プロバイダの資格情報により、これらが使用される ICM 導入環境へのフルアクセスが可能になるので、これらの情報は完全にセキュリティ保護する必要があります。

3.6 サンプル構成ファイルのカスタマイズ

ICM を使用して導入する構成を定義するには、必要な設定を JSON 形式の 2 つの構成ファイル (既定値ファイルと定義ファイル) に追加します。前者 (**defaults.json**) には、選択したクラウド・プロバイダなど、導入環境全体に適用される設定が含まれます。後者 (**definitions.json**) では、必要なノードのタイプおよびそれぞれのノードの数を定義し、シャード・クラスタ、単一のスタンドアロン InterSystems IRIS インスタンス、その他の構成のいずれを導入するのかを指定します。

ICM コンテナ内の **/Samples** ディレクトリには、4 つのクラウド・プロバイダすべてのサンプル構成ファイルが用意されています。例えば、AWS 用のサンプル構成ファイルは、**/Samples/AWS** ディレクトリにあります。これらのファイルをカスタマイズするには、使用するプロバイダを選択し、以下のテーブルで説明しているように、そのプロバイダ用のサンプルの既定値ファイルおよび定義ファイルを変更します。これを行うには、vi などのエディタを使用してコンテナ内で直接編集

するか、以下のように、ローカルのコマンド行で docker cp コマンドを使用してコンテナからローカル・ファイル・システムにそれらをコピーし、編集した後で再びコンテナに戻します。

```
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
```

重要　フィールド名と値の両方で大文字と小文字が区別されます。例えば、クラウド・プロバイダとして AWS を選択するには、既定値ファイルで、“provider”:”AWS” や “Provider”:”aws” などではなく “Provider”:”AWS” と指定する必要があります。

3.6.1 defaults.json のカスタマイズ

/Samples で提供されている既定値ファイルに必要な最小限のカスタマイズと、オプションの変更に関する推奨事項を以下のテーブルに示します。ここで取り上げられていない設定は、サンプル・ファイルでそのままにしておくことができます。

各設定が、″InterSystems Cloud Manager ガイド″ の “ICM リファレンス” の章にある ″ICM の構成パラメータ″ のセクションの関連するテーブルにリンクされており、詳細を確認できます。このセクションでは目的のパラメータを検索してください。一番右の列には、各パラメータまたはパラメータ・セットがサンプルの既定値ファイルに出現するとおりに示されています。

すべてのプロバイダに共通の設定をすべて確認するには、そのセクションの ″一般パラメータ″ を参照してください。特定のプロバイダに固有の設定をすべて確認するには、″プロバイダ固有のパラメータ″ を参照してください。

設定	説明	サンプルの defaults.json ファイルでのエントリ
Provider	クラウド・インフラストラクチャ・プロバイダを特定します。サンプルの defaults.json の値のままにします。	“Provider”:[”AWS”|"GCP"|"Azure"|"Tencent"],
Label	プロビジョニングされるノードの名前付け方式 Label-Role-Tag-NNNN のフィールド (以下の Role を参照)。導入の所有者および目的を指定するように更新します。例えば、会社名と “TEST” を使用して Acme-DATA-TEST-0001 のようなノード名を作成します。	"Label": "Sample",
Tag	(上記の Label 参照)	"Tag": "TEST",
DataVolumeSize	各ノードと共にプロビジョニングする永続データ・ボリュームのサイズ。これは、definitions.json ファイル内の個々のノード定義でオーバーライドできます。Tencent でのプロビジョニングを除き、サンプルの defaults.json の値を受け入れます (Tencent を使用する場合は 60 に変更します)。	"DataVolumeSize": "10",

設定	説明	サンプルの defaults.json ファイルでのエントリ
SSHUser	プロビジョニングされたノードに対する sudo アクセス権を持つ非 root アカウント。ICM でアクセスのために使用されます。サンプルの defaults.json の既定値のままにできますが、AWS または Tencent でマシン・イメージ (以下を参照) を変更する場合は、このエントリの更新が必要なことがあります。	`"SSHUser": "ubuntu",` (AWS & Tencent) `"SSHUser": "sample",` (GCP & Azure)
SSHPublicKey	SSH 公開鍵の場所。"セキュリティ・キーの生成" で説明したキー生成スクリプトを使用した場合、キーはサンプル・ファイルに指定されているディレクトリにあります。このため、変更は不要です。独自のキーを指定している場合は、docker cp を使用して、そのキーをローカル・ファイル・システムからこれらの場所にコピーします。	`"SSHPublicKey": "/Samples/ssh/insecure-ssh2.pub",`
SSHPrivateKey	SSH 秘密鍵の場所。上記の "SSHPublicKey" を参照してください。	`"SSHPrivateKey": "/Samples/ssh/insecure",`
TLSKeyDir	TILS ファイルの場所。上記の "SSHPublicKey" を参照してください。	`"TLSKeyDir": "/Samples/tls/",`
DockerVersion	プロビジョニングされるノードにインストールされる Docker バージョン。既定値のままにしてください。	`"DockerVersion": "5:19.03.8~3-0~ubuntu-bionic",`
DockerImage	プロビジョニングされるノードに導入されるイメージ。"Docker のリポジトリと資格情報の識別" で識別したリポジトリとイメージの情報を反映するよう更新します。	`"DockerImage": "containers.intersystems.com/intersystems/iris:latest-em",`
DockerUsername DockerPassword	プライベート・リポジトリにある場合は DockerImage によって指定されるイメージをダウンロードするのに必要な資格情報。"Docker のリポジトリと資格情報の識別" で識別したリポジトリの情報と資格情報を反映するよう更新します。	`"DockerUsername": "xxxxxxxxxxxx",` `"DockerPassword": "xxxxxxxxxxxx",`
LicenseDir	InterSystems IRIS ライセンスのステージング・ディレクトリ。このディレクトリに、コンテナ固有の InterSystems IRIS シャード・ライセンスを配置します。	`"LicenseDir": "/Samples/Licenses",`

設定	説明	サンプルの defaults.json ファイルでのエントリ
Region, Location (Azure)	インフラストラクチャのプロビジョニングが行われる、プロバイダの計算リソースの地理的地域。サンプルの defaults.json の既定値を受け入れるか、プロバイダ提供のリージョンとゾーン (以下を参照) の別の組み合わせを選択します。	`"Region": "us-west-1",` (AWS) `"Region": "us-east1",` (GCP) `"Location": "Central US",` (Azure) `"Region": "na-siliconvalley",` (Tencent)
Zone	選択したリージョン (上記を参照) 内のアベイラビリティ・ゾーン。サンプルの defaults.json の既定値を受け入れるか、プロバイダ提供のリージョンとゾーンの別の組み合わせを選択します。	`"Zone": "us-west-1c",` (AWS) `"Zone": "us-east1-b",` (GCP) `"Zone": "1",` (Azure) `"Zone": "na-siliconvalley-1",` (Tencent)

マシン・イメージ (プロバイダ固有)	プロビジョニングされたノードのプラットフォームおよび OS のテンプレート。サンプルの defaults.json の既定値を受け入れるか、プロバイダ提供のマシン・イメージとインスタンス・タイプ (以下を参照) の別の組み合わせを選択します。	`"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)

インスタンス・タイプ (プロバイダ固有)	プロビジョニングされたノードの計算リソースのテンプレート。サンプルの defaults.json の値を受け入れるか、プロバイダ提供のマシン・イメージ (上記を参照) とインスタンス・タイプの別の組み合わせを選択します。	`"InstanceType": "m4.large",` (AWS) `"MachineType": "n1-standard-1",` (GCP) `"Size": "Standard_DS2_v2",` (Azure) `"InstanceType": "S2.MEDIUM4",` (Tencent)

設定	説明	サンプルの defaults.json ファイルでのエントリ
資格情報とアカウント設定（プロバイダ固有）	プロバイダに対する認証で ICM が必要とするファイルまたは ID。プロバイダによって異なります。アカウントの必要なファイルの場所または ID を指定するよう更新します（手順については、プロバイダのリンクをクリックしてください）。	“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/xxxxxxxxxxxxxx=",（Azure） "TenantId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",（Azure） "SecretID": "xxxxxxxxxxxx",（Tencent） "SecretKey": "xxxxxxxxxxxx",（Tencent）
ISCPassword	導入される InterSystems IRIS イメージの事前定義アカウントのパスワード。導入フェーズで入力をマスクしてインタラクティブにパスワードを指定するには（セキュリティのために推奨）、このフィールドを削除します。そうでなければ目的のパスワードに変更します。	"ISCPassword": "",
Mirror	DATA、DM、および DS ノードに導入した InterSystems IRIS インスタンスをミラーとして構成するかどうかを指定します。既定値のままにします。	“Mirror”: “false”
UserCPF	導入したインスタンスの初期 CPU 設定のオーバーライドに使用する構成マージ・ファイルを指定します（構成マージ機能または CPF に精通していない場合は、このエントリを削除します。構成マージの詳細は、″ICM・ガイド″ の “ICM リファレンス” の章にある ″カスタマイズされた InterSystems IRIS 構成を使用した導入″ を参照してください）。	"UserCPF": "/Samples/cpf/iris.cpf"

以下のタブは、AWS、GCP、Azure、および Tencent のサンプルの **defaults.json** ファイルのコンテンツを示しています。

AWS

```
{
    "Provider": "AWS",
    "Label": "Sample",
```

```
    "Tag": "TEST",
    "DataVolumeSize": "10",
    "SSHUser": "ubuntu",
    "SSHPublicKey": "/Samples/ssh/insecure-ssh2.pub",
    "SSHPrivateKey": "/Samples/ssh/insecure",
    "DockerRegistry": "https://containers.intersystems.com",
    "DockerImage": "containers.intersytems.com/intersystems/iris:some-tag",
    "DockerUsername": "xxxxxxxxxxxx",
    "DockerPassword": "xxxxxxxxxxxx",
    "TLSKeyDir": "/Samples/tls/",
    "LicenseDir": "/Samples/license/",
    "Region": "us-east-1",
    "Zone": "us-east-1a",
    "AMI": "ami-07267eded9a267f32",
    "DockerVersion": "5:20.10.17~3-0~ubuntu-jammy",
    "InstanceType": "m5.large",
    "Credentials": "/Samples/AWS/sample.credentials",
    "ISCPassword": "",
    "Mirror": "false",
    "UserCPF": "/Samples/cpf/iris.cpf"
}
```

GCP

```
{
    "Provider": "GCP",
    "Label": "Sample",
    "Tag": "TEST",
    "DataVolumeSize": "10",
    "SSHUser": "sample",
    "SSHPublicKey": "/Samples/ssh/insecure.pub",
    "SSHPrivateKey": "/Samples/ssh/insecure",
    "DockerRegistry": "https://containers.intersystems.com",
    "DockerImage": "containers.intersytems.com/intersystems/iris:some-tag",
    "DockerUsername": "xxxxxxxxxxxx",
    "DockerPassword": "xxxxxxxxxxxx",
    "TLSKeyDir": "/Samples/tls/",
    "LicenseDir": "/Samples/license/",
    "Credentials": "/Samples/GCP/sample.credentials",
    "Project": "sample-project",
    "MachineType": "n1-standard-1",
    "Region": "us-east1",
    "Zone": "us-east1-b",
    "Image": "ubuntu-os-cloud/ubuntu-2204-jammy-v20220607",
    "DockerVersion": "5:20.10.17~3-0~ubuntu-jammy",
    "ISCPassword": "",
    "Mirror": "false",
    "UserCPF": "/Samples/cpf/iris.cpf"
}
```

Azure

```
{
    "Provider": "Azure",
    "Label": "Sample",
    "Tag": "TEST",
    "DataVolumeSize": "10",
    "SSHUser": "sample",
    "SSHPublicKey": "/Samples/ssh/insecure.pub",
    "SSHPrivateKey": "/Samples/ssh/insecure",
    "DockerRegistry": "https://containers.intersystems.com",
    "DockerImage": "containers.intersytems.com/intersystems/iris:some-tag",
    "DockerUsername": "xxxxxxxxxxxx",
    "DockerPassword": "xxxxxxxxxxxx",
    "TLSKeyDir": "/Samples/tls/",
    "LicenseDir": "/Samples/license/",
    "Location": "Central US",
    "Zone": "1",
    "PublisherName": "Canonical",
    "Offer": "0001-com-ubuntu-server-jammy",
    "Sku": "22_04-lts",
    "Version": "22.04.202206040",
    "DockerVersion": "5:20.10.17~3-0~ubuntu-jammy",
    "Size": "Standard_DS2_v2",
    "AccountTier": "Standard",
    "AccountReplicationType": "LRS",
    "SubscriptionId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "ClientId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "ClientSecret": "xxxxxxxxxxxx/xxxxxxxxxxxxxx/xxxxxxxxxxxxx=",
    "TenantId": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
    "ISCPassword": "",
```

```
    "Mirror": "false",
    "UserCPF": "/Samples/cpf/iris.cpf"
}
```

Tencent

```
{
    "Provider": "Tencent",
    "Label": "Sample",
    "Tag": "TEST",
    "DataVolumeSize": "60",
    "SSHUser": "ubuntu",
    "SSHPublicKey": "/Samples/ssh/insecure.pub",
    "SSHPrivateKey": "/Samples/ssh/insecure",
    "DockerRegistry": "https://containers.intersystems.com",
    "DockerImage": "containers.intersytems.com/intersystems/iris:some-tag",
    "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-22trbn9x",
    "DockerVersion": "5:20.10.10~3-0~ubuntu-focal",
    "ISCPassword": "",
    "Mirror": "false",
    "UserCPF": "/Samples/cpf/iris.cpf"
}
```

3.6.2 definitions.json のカスタマイズ

/Samples ディレクトリ内のサンプルの definitions.json ファイル（すべてのプロバイダで同一）では、以下に示すように、2 つのデータ・ノードと 2 つの計算ノードが含まれるシャード・クラスタを定義します。

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

Role フィールドは、プロビジョニングするノード・タイプ（ここでは DATA および COMPUTE）を指定します。Count フィールドは、プロビジョニングするそのノード・タイプの数を示します。StartCount は、COMPUTE ノードの番号付けを 0003 から開始します。LicenseKey フィールドは、既定値ファイルの LicenseDir フィールドで指定したディレクトリ内に配置されている InterSystems IRIS ライセンス・ファイルの名前を示します。

この演習では、以下のように、COMPUTE 定義を削除し、DATA 定義のみを残します。

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

シャード・クラスタを導入する場合、すべてのノードにシャード・ライセンスが必要です。docker cp を使用して、LicenseDir で指定したコンテナ内の場所（/Samples/Licenses など）にシャード・ライセンスをコピーし、使用するライセンス・キーを指定するように DATA ノード定義内の LicenseKey 設定を更新します。

基本的なシャード・クラスタをプロビジョニングするために必要な definitions.json の変更はこれだけです。代わりに、スタンドアロンの InterSystems IRIS インスタンスをプロビジョニングするには、以下の定義を使用します。

```
[
   {
    "Role": "DM",
    "Count": "1",
    "LicenseKey": "standard-iris.key""
    },
]
```

3.7 インフラストラクチャのプロビジョニング

既定では、ICM は現在のディレクトリ内の構成ファイルに入力された情報を使用します。したがって、インフラストラクチャをプロビジョニングするために必要なのは、選択したプロバイダの /Samples ディレクトリ (例えば、/Samples/AWS) に変更し、以下のコマンドを発行することだけです。

```
icm provision
```

icm provision コマンドは、選択したプラットフォームでホスト・ノードの割り当てと構成を行います。プロビジョニング操作時に、ICM は state サブディレクトリ内の状態ファイルとログ・ファイルを作成または更新し、完了時に instances.json ファイルを作成します。このファイルは、後続の導入コマンドと管理コマンドに対する入力となります。

ICM は一度に複数のタスクを実行するため、ステップ (Terraform の計画および適用、ファイルのコピー、ボリュームのマウントなど) が各ノードで同じ順序で開始され、完了するとは限りません。完了時に、ICM はプロビジョニングされたホスト・ノードの要約を表示し、後でインフラストラクチャを削除するために使用できるコマンド行を出力します。以下に例を示します。

```
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]
```

重要

プロビジョニング解除の際に簡単に複製できるように、出力に示された icm unprovision コマンド行をコピーし、この情報を保存してください。この出力は icm.log ファイルにも書き込まれます。

インフラストラクチャの管理とプロビジョニング解除の両方に必要なファイルおよびディレクトリは ICM コンテナ内にあるため、ICM コンテナを削除するとそれらも失われます。このため、インフラストラクチャに対する操作を完了し、これを正常にプロビジョニング解除するまで、ICM コンテナを削除しないでください (プロダクションでは、ベスト・プラクティスとして、ICM の起動時に外部ボリュームをマウントして、このデータの保存にコンテナ外部の場所を使用することをお勧めします (Docker ドキュメントの "Manage data in Docker" を参照してください))。

プロビジョニングおよび導入フェーズで発生したエラーに関する詳細情報は、状態ディレクトリのサブディレクトリ内の terraform.err ファイルに書き込まれます。エラーが発生した場合、該当するファイルが示され、問題の原因を特定するのに役立ちます。

プロバイダ側のタイムアウトや内部エラー、または構成ファイル内のエラーが原因で icm provision が正常に完了しなかった場合は、指定したすべてのノードについて必要なタスクがすべてエラーなしで完了するまで、必要に応じて何回でもコマンドを発行できます。詳細は、"InterSystems Cloud Manager ガイド" の “ICM の使用” の章にある "インフラストラクチャの再プロビジョニング" を参照してください。

icm provision の詳細は、"InterSystems Cloud Manager ガイド" の "icm provision コマンド" を参照してください。

3.8 InterSystems IRIS の導入

プロビジョニングされたホスト・ノードでイメージをダウンロードし、コンテナとして実行するだけでなく、ICM は InterSystems IRIS 固有の構成やその他のタスクも実行します。プロビジョニングされたノードに InterSystems IRIS を導入するには、構

成ファイルをカスタマイズし、インフラストラクチャをプロビジョニングした /Samples ディレクトリのまま、以下のコマンドを発行します。

```
icm run
```

既定では、icm run は、構成ファイルの DockerImage フィールドで指定されたイメージ (この場合は、インターシステムズのリポジトリにある InterSystems IRIS イメージ) をダウンロードして実行します。各コンテナの名前は iris です。実際には、(ここに示しているように既定値ファイルですべてのノードに対して 1 回ではなく) 定義ファイル内のノード定義のさまざまな DockerImage フィールドを使用することによって、ノード・タイプが異なれば異なるイメージを実行できます。また、それ以外にも、特定のオプションを指定して icm run コマンドを複数回実行することで、プロビジョニングされたノードごとに、または指定したノードのみに、固有の名前を持つ複数のコンテナを導入できます。

InterSystems IRIS が各ノードで起動すると、ICM は、パスワードのリセット、シャーディングの構成など、インスタンスで必要なすべての構成を行います。ICM は一度に複数のタスクを実行するため、導入プロセスのステップが各ノードで同じ順序で開始され、完了するとは限りません。

完了時に、ICM は該当する InterSystems IRIS インスタンスの管理ポータルへのリンクを出力します。

```
$ 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:latest-em on ACME-DATA-TEST-0001...
...pulled ACME-DATA-TEST-0001 image intersystems/iris:latest-em
...
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
```

ここでは、指定されたリンクは、データ・ノード 1 のリンクか、スタンドアロン・インスタンスのリンク (その構成を選択した場合) です。ブラウザでリンクを開き、管理ポータルを使用して InterSystems IRIS を確認します。

icm provision の場合と同様に、プロバイダ側のタイムアウトや内部エラーが原因で icm run が正常に完了しない場合は、このコマンドを再度発行できます。ほとんどの場合、繰り返し試行することで導入に成功します。どうしてもエラーが解決されず、手動による操作が必要な場合は (いずれかの構成ファイル内のエラーが原因である場合など)、問題を修正した後、icm run コマンドを再発行する前に、″InterSystems Cloud Manager ガイド″ の “ICM の使用” の章にある ″サービスの再導入″ の説明に従って、影響を受けるノード上の永続的な %SYS データを削除する必要が生じることがあります。エラーが発生した場合は、ICM から示されたログ・ファイルの情報を参照すると、問題の原因を特定するのに役立ちます。

icm run コマンドとそのオプションの詳細は、″InterSystems Cloud Manager ガイド″ の ″icm run コマンド″ を参照してください。

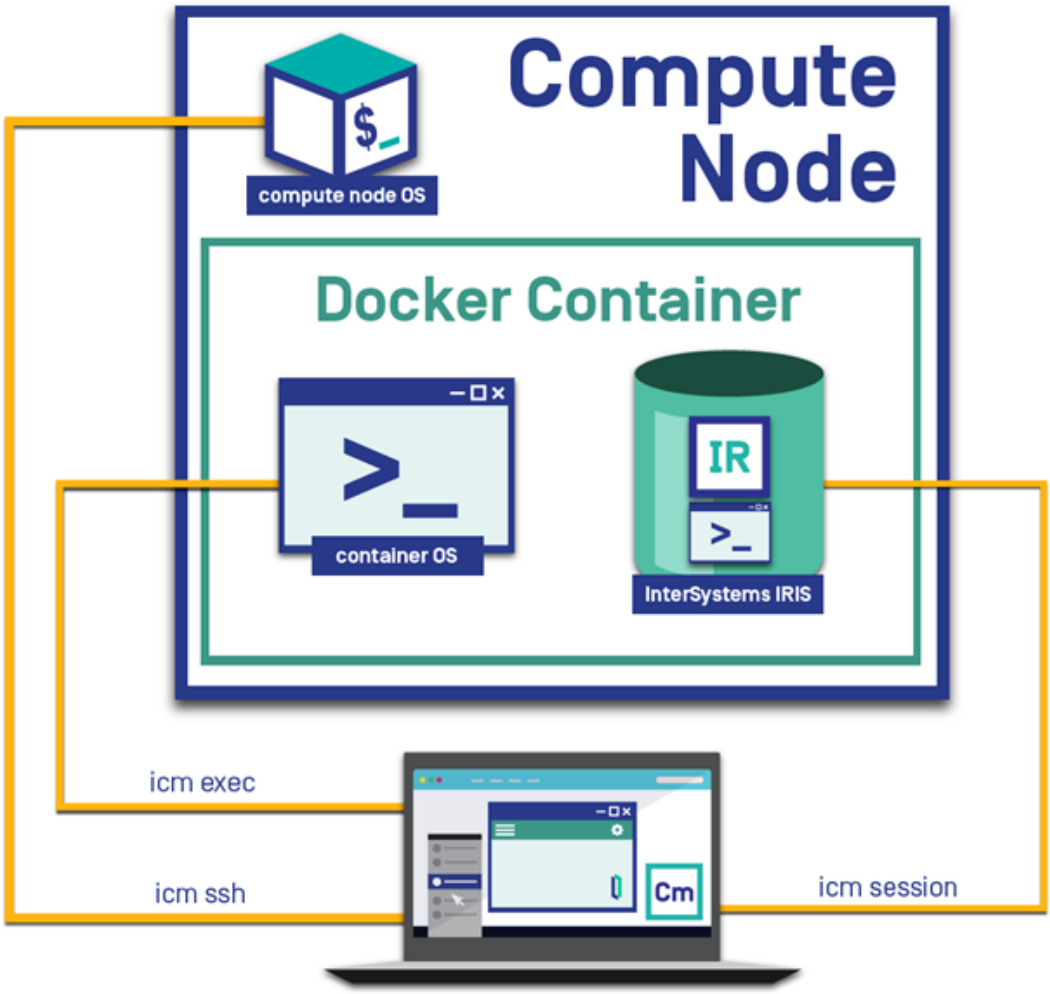
3.9 ICM 管理コマンドを体験

ICM には、以下を目的としたさまざまなコマンドが用意されています。

- プロビジョニングされたインフラストラクチャを管理する。
- 導入されたコンテナを管理する。
- InterSystems IRIS を含め、導入されたコンテナ内で実行されているサービスを操作する。

これらのコマンドの詳細は、″InterSystems Cloud Manager ガイド″ の ″インフラストラクチャ管理コマンド″、″コンテナ管理コマンド″、および ″サービス管理コマンド″ の各セクションを参照してください。″ICM コマンドとオプション″ には、すべてのコマンドのリストを掲載しています。導入環境で試せるように、ここではいくつかの例を示します。特に、3 つのコマンド (icm ssh、icm exec、および icm session) を使用すると、以下に示すように、ノード自体、ノードに導入されたコンテナ、コンテナ内で実行されている InterSystems IRIS インスタンスなど、導入環境のノードをさまざまなレベルで操作できます。

図 2: インタラクティブ ICM コマンド



新しく導入された InterSystems IRIS 構成で以下のコマンドを試してみましょう。

1. プロビジョニングされたホスト・ノードをリストします。

```
icm inventory
```

2. それぞれのホスト・ノード上でシェル・コマンドを実行します。

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

複数のノード上でコマンドが実行される場合、出力はファイルに書き込まれ、出力ファイルのリストが提供されます。例えば、ここでは、シャード・クラスタを導入した場合、3 つのノードそれぞれに出力ファイルが 1 つずつあります。

3. 特定のホスト・ノード上でインタラクティブ・シェルを開きます。

```
icm ssh -role DATA -interactive
```

インタラクティブにするには、コマンドで -interactive オプションを使用し、1 つのノードを指定する必要があります。例えば、icm ssh は、前の例のように、複数のノード上でコマンドを実行できますが、以下の例に示すように、-interactive オプションを指定すると、1 つのノード上でのみインタラクティブ・シェルを開くことができます。-role DM オプションは、コマンドをそのタイプのノード (導入環境に 1 つだけある) に限定します。このオプションの代わりに -machine オプションを使用して、特定のノードの名前を指定することもできます。以下に例を示します。

```
icm ssh -machine Acme-DATA-TEST-0001 -interactive
```

4. 導入された InterSystems IRIS コンテナのステータスを表示します。

```
icm ps
```

複数のコンテナがノードに導入されている場合、このコマンドはそれらをすべてリストします。

5. InterSystems IRIS コンテナ、またはそれらのコンテナのいずれかにローカル・ファイルをコピーします。

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

6. それぞれの InterSystems IRIS コンテナ内でシェル・コマンドを実行します。

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

7. 特定の InterSystems IRIS コンテナ内でインタラクティブ・シェルを開きます (またはシェル・コマンドを実行します)。

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

8. InterSystems IRIS インスタンスに対してターミナル・ウィンドウを開きます (常にインタラクティブで、1 つのインスタンスのみを対象とします)。

```
icm session -machine Acme-DATA-TEST-0001
```

9. それぞれの InterSystems IRIS インスタンスに対して、または特定のインスタンスに対して SQL コマンドを実行します。

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

3.10 インフラストラクチャのプロビジョニング解除

AWS およびその他のパブリック・クラウド・プラットフォームのインスタンスは継続的に課金されるので、この体験が完了したらすぐに、インフラストラクチャのプロビジョニングを解除することが重要です。

そのためには、icm provision の出力から保存した icm unprovision コマンドを ICM コマンド行にコピーします。以下に例を示します。

```
$ icm unprovision -cleanUp
```

プロビジョニングの出力からコマンドを保存しなかった場合は、作業ディレクトリ内の **icm.log** ファイル (**/Samples/AWS/icm.log** など) で確認できます。-cleanUp オプションは、プロビジョニング解除の後に状態ディレクトリを削除します。このオプションを指定しなかった場合、状態ディレクトリは保持されます。

4 ICM で実行できるその他の操作

これまでに行った ICM の演習は意図的に簡素化されたものでしたが、それでも、実際のインフラストラクチャのプロビジョニングやサービスの導入が含まれていました。これまでに実行した操作のほかには、構成ファイル内の導入定義の拡張とさまざまなコマンド行オプションの利用に関するものがあるだけです。例えば、以下の操作を実行できます。

- 複数の AM ノードと 1 つの DM ノードを定義して、複数のアプリケーション・サーバと 1 つのデータ・サーバの分散キャッシュ・クラスタを導入したり、1 つの DM ノードだけを定義して、スタンドアロンの InterSystems IRIS インスタンスを導入することができます。

- 既定値ファイルに “Mirror”: “True” を追加し、定義ファイルで偶数個の DATA ノードまたは 2 つの DM ノードと 1 つの AR (アービター) ノードを定義するだけで、ミラーリングされた DATA ノードまたはミラーリングされた DM ノードを導入できます。
- COMPUTE ノードが含まれるシャード・クラスタを導入することで、クエリとデータ取り込みの作業負荷を分離する一方で、並列処理や分散キャッシュの利点を維持し、これら両方のパフォーマンスを向上させます。
- Web サーバ (WS) およびロード・バランサ (LB または自動) を定義ファイルに追加します。
- 定義ファイルでノードごとに異なる DockerImage 値を指定し、icm run コマンドを複数回発行することにより、個々のノードに異なるサービスを導入します。
- コマンド行で -container オプションと -image オプションを使用して異なるコンテナ名およびイメージ (カスタム・イメージとサードパーティ・イメージを含む) を指定しながら、icm run を複数回実行することにより、一部またはすべてのノードに複数のサービスを導入します。
- サードパーティ・ツールを使用したり、社内でスクリプトを作成することにより ICM の機能を拡張し、さらに自動化を強化すると共に労力を軽減します。
- 既存の仮想クラスタおよび物理クラスタにサービスを導入します。
- 単に ICM provision コマンドで操作を終えることにより、サービスを導入せずにインフラストラクチャをプロビジョニングします。

ICM には、これらの機能以外にもさまざまな機能が用意されています。詳細は、″InterSystems Cloud Manager ガイド″ を参照してください。

5 ICM の詳細情報

ICM の詳細およびコンテナでの InterSystems IRIS の使用法については、以下を参照してください。

- InterSystems Cloud Manager Introduction (ビデオ)
- The Benefits of InterSystems Cloud Manager (ビデオ)
- Deploying InterSystems IRIS in Containers and the Cloud (学習パス)
- InterSystems Cloud Manager ガイド
- InterSystems IRIS の基礎 : InterSystems IRIS コンテナの実行
- コンテナ内でのインターシステムズ製品の実行