

記事

[Tomohiro Iwamoto](#) · 2020年8月25日 6m read

CloudFormationを使ってAWSにICMをデプロイする

本稿について

ICM(InterSystems Cloud Manager)のセットアップは難しいものではありませんが、様々な理由でそもそもDockerが使いづらいという状況があり得ます。
また、セキュリティ的に堅固な環境を得るために、既存VPC内のプライベートサブネット上にIRISクラスタをデプロイする方法のひとつに、[同VPC内でICM実行する方法](#)があります。
本稿では、ICMをAWSにデプロイする作業を、CloudFormationで自動化する方法をご紹介します。ICMに関しては、[こちらの記事](#)をご覧ください。

更新: 2020年11月24日 デフォルトVPC以外でも動作するように変更しました。

ICMは、マルチクラウドに対してIRISクラスタをプロビジョン、アンプロビジョンするためのユーティリティですので、CloudFormationとは機能的に重なる部分があります。必要であれば、[CloudFormationでIRISのミラー構成をプロビジョン](#)することも可能です。ここでは割り切って、CloudFormationはICMをデプロイすることだけに利用しています。

また、ICMキット入手方法が今までのWRCサイトからの手動ダウンロードに加えて、[InterSystems Container Registry](#)から直接Pullする方法が利用可能となりました。本稿はこれを利用しています。

前提条件

EC2インスタンスをプロビジョンする先としてVPC、サブネット、セキュリティグループ(SSHがインバウンド可)が存在していること。

事前準備

もしお持ちでなければSSH接続のためのEC2キーペアを作成します。
ICMで使用するAWSリソースの操作に必要なポリシーを持つロールを用意します。本稿ではAmazonEC2FullAccessポリシーを持つロール(iwamoto-ICM-role)を使用しています。

このロールはICMをデプロイするEC2インスタンスに付与されますので、使用するEC2キーペアの管理にはご注意ください(わざわざ言うまでもありませんね)。

事前準備はこれだけです。

本稿では扱いませんが、同ロールに、AmazonS3ReadOnlyAccessを加えておくと、コンテナレスのデプロイを行う際に使用するIRISのインストーラキット(tar形式)やライセンスキーを自前のS3バケットからダウンロード出来るので、ICMのスクリプト機能を利用して、アプリケーション配布まで含めた完全な自動化を実現できます。

アクセス権限

信頼関係

タグ

アクセスアドバイザー

セッションの無効化

▼ Permissions policies (2 適用済みポリシー)

ポリシーをアタッチします

インラインポリシーの追加

ポリシー名	ポリシータイプ	
AmazonEC2FullAccess	AWS 管理ポリシー	×
AmazonS3ReadOnlyAccess	AWS 管理ポリシー	×

Permissions boundary (not set)

手順

1. CloudFormationテンプレートの入手

[こちら](#)から入手できます。

2. スタックの作成

AWS CloudFormationコンソールで「スタックの作成」「新しいリソースを使用」を選択します。「テンプレートファイルのアップロード」で先ほど入手したymlをアップロードします。

CloudFormation

スタック

スタックの作成

ステップ 1
テンプレートの指定

ステップ 2
スタックの詳細を指定

ステップ 3
スタックオプションの設定

ステップ 4
レビュー

スタックの作成

前提条件 - テンプレートの準備

テンプレートの準備

各スタックはテンプレートに基づきます。テンプレートとは、スタックに含む AWS リソースに関する設定情報を含む JSON または YAML ファイルです。

テンプレートの準備完了

サンプルテンプレートを使用

デザイナーでテンプレートを作成

テンプレートの指定

テンプレートは、スタックのリソースおよびプロパティを表す JSON または YAML ファイルです。

テンプレートソース

テンプレートを選択すると、保存先となる Amazon S3 URL が生成されます。

Amazon S3 URL

テンプレートファイルのアップロード

テンプレートファイルのアップロード

ファイルの選択

icm.yml

JSON または YAML 形式のファイル

S3 URL: https://s3-ap-northeast-1.amazonaws.com/ icm.yml

デザイナーで表示

キャンセル

次へ

ウィザードに沿って「次へ」進みます。

[スタックの詳細を指定]画面では、各種パラメータを指定します。

スタックの名前	任意の文字列
ICMKitVersionParameter	ICMのキットのバージョン
IamInstanceProfileParameter	事前準備したロール名
InstanceSecurityGroupParameter	EC2インスタンスに適用するセキュリティグループと同じVPCに属してSSHをインバウンド可能にしておく

Page 2 of 9

InstanceTypeParameter	EC2インスタンスタイプ
KeyNameParameter	事前準備したEC2キーペアの名前
LatestAmildParameter	使用するAMI(本稿ではAmazon Li
SSHLocationParameter	SSH接続を許可するホストのCIDRブロック (0.0.0.0/0は全て許可。自分のグローバルIPを指定すると、そのIPからのみ許可)
SubnetIdParameter	EC2インスタンスを起動するサブネットID (セキュリティグループと同じVPCに属する)
WRCTokenParameter	https://containers.intersystems.com
WRCUserParameter	同サイトへのログインに使用したユーザー名

スタックの詳細を指定

スタックの名前

スタックの名前

スタック名では、大文字および小文字 (A-Z~a-z)、数字 (0-9)、ダッシュ (-) を使用することができます。

パラメータ

パラメータは、テンプレートで定義されます。また、パラメータを使用すると、スタックを作成または更新する際にカスタム値を入力できます。

ICMKitVersionParameter

ICM Kit version

IamInstanceProfileParameter

IAM instance role for ICM + s3:read

InstanceSecurityGroupParameter

Security group for the instance

InstanceTypeParameter

WebServer EC2 instance type

KeyNameParameter

Name of an existing EC2 KeyPair to enable SSH access to the instance

LatestAmildParameter

SSHLocationParameter

The IP address range that can be used to SSH to the EC2 instances

SubnetIdParameter

SubnetId

WRCTokenParameter

Leave it empty. Don't use this yet. Token for container registry server.

WRCUserParameter

キャンセル

戻る

次へ

アップロードしたテンプレートファイルは、自動でS3に保存されます。わずかとはいえ課金対象ですので、デプロイ後、不要であれば削除します。

ウィザードに沿って「次へ」進みます。

[スタックオプションの設定]はデフォルトのままに構いません。
ウィザードに沿って「次へ」進みます。

[レビュー]画面で値の最終確認を行います。

レビュー test-stack

ステップ 1: テンプレートの指定

編集

テンプレート

テンプレート URL

https://s3-ap-northeast-1.amazonaws.com/cf-templates-[redacted]-icm.yml

スタックの説明

Based on AWS CloudFormation Sample Template EC2InstanceWithSecurityGroupSample.

予想コスト [🔗](#)

ステップ 2: スタックの詳細を指定

編集

パラメータ (10)

🔍 検索パラメータ



キー	値
ICMKitVersionParameter	2020.1.0.215.0
IamInstanceProfileParameter	iwamoto-ICM-role
InstanceSecurityGroupParameter	sg-[redacted]
InstanceTypeParameter	t2.micro
KeyNameParameter	iwamoto20200714
LatestAmildParameter	/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2
SSHLocationParameter	0.0.0.0/0
SubnetIdParameter	subnet-[redacted]
WRCTokenParameter	*****
WRCUserParameter	iwamoto

通知オプション

通知オプションはありません
定義されている通知オプションはありません

スタックの作成オプション

失敗時のロールバック
有効

タイムアウト
-

削除保護
無効

▶ クイック作成リンク

キャンセル 戻る 変更セットの作成 **スタックの作成**

先ほど指定したパラメータに誤りがなければ「スタックの作成」を実行します。

スタック名と実行日時に下にステータスがCREATE_IN_PROGRESSと表示されていると思います。リフレッシュ矢印ボタンを押してステータスがCREATE_COMPLETEになれば完了です。私の場合、1分ほどで完了しました。

test-stack

削除更新するスタックアクション▼スタックの作成▼

スタックの情報

イベント

リソース

出力

パラメータ

テンプレート

変更セット

イベント (5)

🔄

🔍 検索イベント

⚙️

タイムスタンプ	論理 ID	ステータス	状況の理由
2020-11-24 16:30:26 UTC+0900	test-stack	✔ CREATE_COMPLETE	-
2020-11-24 16:30:24 UTC+0900	EC2Instance	✔ CREATE_COMPLETE	-
2020-11-24 16:29:52 UTC+0900	EC2Instance	ⓘ CREATE_IN_PROGRESS	Resource creation Initiated
2020-11-24 16:29:50 UTC+0900	EC2Instance	ⓘ CREATE_IN_PROGRESS	-
2020-11-24 16:29:45 UTC+0900	test-stack	ⓘ CREATE_IN_PROGRESS	User Initiated

もし、EC2Instanceのステータスが以下のような「状況の理由」でCREATE_FAILEDした場合は、セキュリティグループとサブネットが同じVPCに属しているか確認してください。

Security group sg-XXXXXXX and subnet subnet-YYYYYYY belong to different networks.

[リソース]で、どのようなリソースを作成したのかを確認できます。

test-stack

削除 更新する スタックアクション ▼ スタックの作成 ▼

スタックの情報 イベント リソース 出力 パラメータ テンプレート 変更セット

リソース (1)

Q リソースの検索

論理 ID	物理 ID	タイプ	ステータス	状況の理由
EC2Instance	i-0aaa33da1842f875a	AWS::EC2::Instance	CREATE_COMPLETE	-

[出力]を表示して、PublicDNS値を記録してください。

test-stack

削除 更新する スタックアクション ▼ スタックの作成 ▼

スタックの情報 イベント リソース 出力 パラメータ テンプレート 変更セット

出力 (3)

Q 検索結果の出力

キー	値	説明	エクスポート名
AZ	ap-northeast-1d	Availability Zone of the newly created EC2 instance	-
InstanceId	i-0aaa33da1842f875a	InstanceId of the newly created EC2 instance	-
PublicDNS	ec2-18-179-52-235.ap-northeast-1.compute.amazonaws.com	Public DNSName of the newly created EC2 instance	-

3. 作成されたEC2インスタンスへのログイン

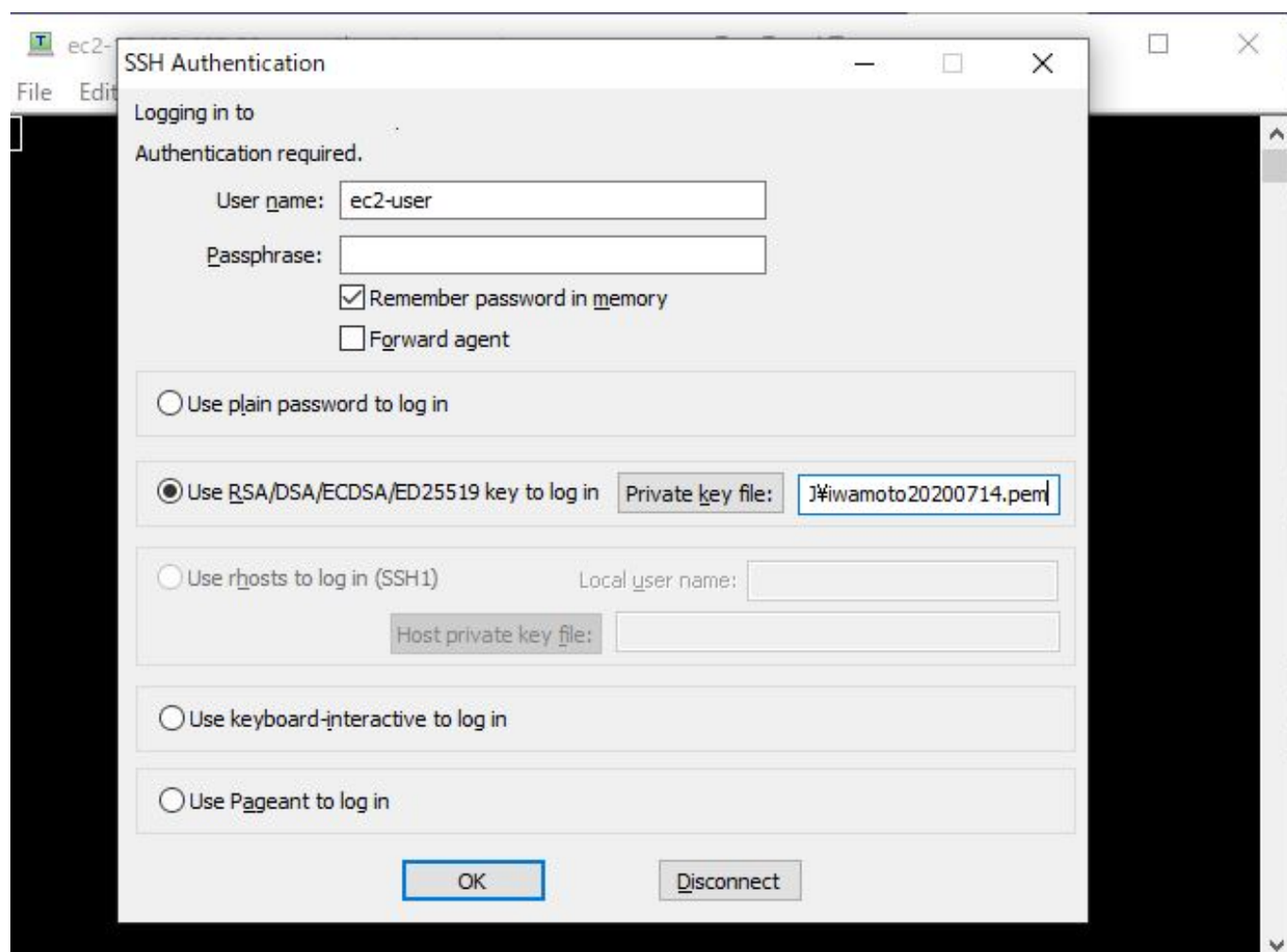
先ほど指定したEC2キーペアを使用して、PublicDNSのホストにsshします。

```
C:\Users\iwamoto>ssh -i iwamoto20200714.pem ec2-user@ec2-18-179-52-235.ap-northeast-1.compute.amazonaws.com
```

```
__|  __|_ )
_| ( /   Amazon Linux 2 AMI
___|\___|___|
```

```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-22-166 ~]$
```

あるいは、定番のターミナルエミュレータからですと



ICMがインストールされている事を確認します。イメージ名のレポジトリ名がcontainers.intersystems.com/intersystems/icmになっていることにご注意ください。

```
[ec2-user@ip-172-31-22-166 ~]$ docker images
REPOSITORY                                TAG                IMAGE ID
CREATED          SIZE
containers.intersystems.com/intersystems/icm 2020.1.0.215.0    00d66fae9030
4 months ago      889MB
[ec2-user@ip-172-31-22-166 ~]$ docker run --rm containers.intersystems.com/intersystems/icm:2020.1.0.215.0 icm version
Version: 2020.1.0.215.0
[ec2-user@ip-172-31-22-166 ~]$
```

AWSのCLIを実行して、指定したロールが備わっていることを確認します。

```
[ec2-user@ip-172-31-22-166 ~]$ aws ec2 describe-
instances --query "Reservations[].Instances[].InstanceId"
[
  "i-0aaa33da1842f875a"
]
```

先ほど確認した[出力]のInstanceIdと同じですね。

ロールにS3へのアクセス権を付与していれば、S3からキットやライセンスキーをダウンロード出来るので便利(なにより別途クレデンシャル情報を取り扱わない分、セキュア)です。


```
[ec2-user@ip-172-31-22-166 ~]$ aws s3 cp s3://mybucket/IRIS-2020.1.0.215.0-lnxrhx64.tar.gz .
download: s3://mybucket/IRIS-2020.1.0.215.0-lnxrhx64.tar.gz to ./IRIS-2020.1.0.215.0-lnxrhx64.tar.gz
```

もし、上記のような結果が得られなければ何か問題が発生しています。下記にログや実行されたシェルが保存されていますので、内容を確認してください。

```
[ec2-user@ip-172-31-22-166 ~]$ cat /var/log/cloud-init-output.log
[ec2-user@ip-172-31-22-166 ~]$ sudo cat /var/lib/cloud/instances/[EC2??????ID]/user-data.txt
```

以後、ICMの操作を行います。

4. スタックの削除

不要になったら忘れずにリソース(特にEC2インスタンス)を削除しましょう。その際、EC2インスタンス等を個別に削除するのではなく、AWS CloudFormationのコンソールの[削除]で全てのリソースを一括削除します。[スタックの情報]で、ステータスがDELETE_IN_PROGRESSからDELETE_COMPLETEに変われば削除終了です。

ICMのEC2インスタンスを削除すると、ICMでプロビジョンしたIRISクラスタへの制御情報も完全に失われますのでご注意ください。

[#AWS](#) [#クラウド](#) [#InterSystems IRIS](#) [#InterSystems IRIS for Health](#)

ソースURL:

<https://jp.community.intersystems.com/post/cloudformation%E3%82%92%E4%BD%BF%E3%81%A3%E3%81%A6aws%E3%81%ABicm%E3%82%92%E3%83%87%E3%83%97%E3%83%AD%E3%82%A4%E3%81%99%E3%82%8B>