

---

# Cloud Migration Factory on AWS

## Implementation Guide



## **Cloud Migration Factory on AWS: Implementation Guide**

Copyright © 2022 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

## Table of Contents

Welcome .....	1
Cost .....	2
(Optional/Recommended) Deploy an Amazon Elastic Compute Cloud instance to help run automation scripts .....	3
Architecture overview .....	4
Optional migration tracker .....	5
Solution components .....	6
Migration automation server .....	6
Migration services RestAPIs .....	6
Log in services .....	6
Admin services .....	7
User services .....	7
Tools services .....	7
Migration Factory web interface .....	7
Security .....	8
IAM roles .....	8
Amazon Cognito .....	8
Amazon CloudFront .....	8
Implementation considerations .....	9
Regional Deployments .....	9
AWS CloudFormation templates .....	10
Automated deployment .....	11
Prerequisites .....	11
Domain permissions .....	11
AWS Application Migration Service (AWS MGN) .....	11
Deployment overview .....	11
Step 1. Launch the stack .....	12
Step 2. Launch the target account stack in the target AWS account .....	14
Step 3. Create the first user .....	15
Create the initial user and log in to the solution .....	15
Add a user to the admin group .....	16
Identify the CloudFront URL .....	16
Step 4. Update the factory schema .....	17
Update the aws_accountid for AWS MGN .....	17
Step 5. Build a migration automation server .....	17
Build a Windows Server 2016 or later server .....	17
Configure AWS permissions for the migration automation server .....	18
Step 6. Test the solution using the automation scripts .....	22
Step 7. (Optional) Build a migration tracker dashboard .....	25
Additional resources .....	45
User guide .....	46
Metadata management .....	46
Viewing data .....	46
Adding or editing a record .....	46
Deleting a record .....	47
Exporting data .....	47
Importing data .....	48
Credentials management .....	49
Add a secret .....	49
Edit a secret .....	50
Delete a secret .....	50
Run automation from console .....	50
Run automations from command prompt .....	51
Manually running an automation package .....	52

Creation of the FactoryEndpoints.json .....	52
Launch AWS MGN jobs from Cloud Migration Factory .....	53
Prerequisite activities .....	53
Initial definition .....	53
Initiating a job .....	54
Replatform to EC2 .....	55
Prerequisites .....	56
Initial configuration .....	56
Deployment actions .....	58
Scripts management .....	59
Upload new script package .....	59
Download script packages .....	59
Add new version of a script package .....	59
Deleting script packages and versions .....	60
Composing a new script package .....	60
Schema management .....	62
Adding/editing an attribute .....	63
Permission management .....	68
Policies .....	69
Roles .....	70
List of automated migration activities using factory web console .....	71
Check prerequisites .....	71
Install the replication agents .....	72
Push the post-launch scripts .....	73
Verify the replication status .....	73
Validate launch template .....	74
Launch instances for testing .....	74
Verify the target instance status .....	75
Mark as ready for cutover .....	76
Shut down the in-scope source servers .....	77
Launch instances for Cutover .....	77
List of automated migration activities using command prompt .....	79
Check prerequisites .....	79
Install the replication agents .....	80
Push the post-launch scripts .....	82
Verify the replication status .....	83
Verify the target instance status .....	83
Shut down the in-scope source servers .....	84
Retrieve the target instance IP .....	84
Verify the target server connections .....	85
Update the stack .....	86
Uninstall the solution .....	87
Empty the Amazon S3 buckets .....	87
Using the AWS Management Console to delete the stack .....	87
Using AWS Command Line Interface to delete the stack .....	87
Collection of operational metrics .....	88
Source code .....	89
Revisions .....	90
Contributors .....	91
Notices .....	92
AWS glossary .....	93

# Coordinate and automate large scale migrations to the AWS Cloud using the Cloud Migration Factory on AWS solution

Publication date: *June 2020 (last update (p. 90): June 2022)*

The Cloud Migration Factory on AWS solution is designed to coordinate and automate manual processes for large-scale migrations involving a substantial number of servers. This solution helps enterprises improve performance and prevents long cutover windows by providing an orchestration platform for rehosting servers to AWS at scale. [AWS Professional Services](#), [AWS Partners](#), and other enterprises have already used this solution to help customers migrate thousands of servers to the AWS Cloud.

This solution helps you to:

- Integrate the many different types of tools that support migration, such as discovery tools, migration tools, and configuration management database (CMDB) tools.
- Automate migrations that involve many small, manual tasks, which take time to run and are slow and hard to scale.

For a complete end-to-end deployment guide using this solution, refer to [Automating large-scale server migrations with Cloud Migration Factory](#) in the *AWS Prescriptive Guidance Cloud Migration Factory Guide*.

This implementation guide discusses architectural considerations and configuration steps for deploying the Cloud Migration Factory on AWS solution in the Amazon Web Services (AWS) Cloud. It includes links to [AWS CloudFormation](#) templates that launch and configure the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, administrators, and DevOps professionals who have practical experience architecting in the AWS Cloud.

# Cost

You are responsible for the cost of the AWS services used while running this solution. As of June 2022, the estimated cost for running this solution with default settings in the US East (N. Virginia) Region and assuming that you are migrating 200 servers a month with this solution is approximately **\$7.81 per month**. The cost for running this solution depends on the amount of data being loaded, requested, stored, processed, and presented as shown in the following table.

AWS service	Factors	Cost/month
<b>Core services</b>		
Amazon API Gateway	10,000 requests/month x (\$3.50/million)	\$0.035
AWS Lambda	10,000 invocations/month (avg 3,000 ms duration and 128 MB memory)	\$0.065
Amazon DynamoDB	20,000 write requests/month x (\$1.25/million)  40,000 read requests/month x (\$0.25/million)  Data storage: 1 GB x \$0.25	\$0.035
Amazon S3	Storage (10MB) & 50,000 get requests/month	\$0.25
Amazon CloudFront	Regional data transfer out to internet: first 10 TB  Regional data transfer out to origin: all data transfer  HTTPS requests:  50,000 requests/month X (\$0.01/10,000 requests)	\$0.92
AWS Systems Manager	10,000 steps/month	\$0.00
AWS Secrets Manager	5 secrets x 30 days duration	\$2.00
Amazon Cognito	20 user x (\$0.0055/monthly active users (MAUs))	\$0.11
Amazon Athena	10MB daily x \$5.00 per TB of data scanned	\$0.0015
<b>Optional services</b>		
AWS Glue (optional migration tracker)	2 mins daily x Default 10 DPU x \$0.44 per DPU-Hour	\$4.40

AWS service	Factors	Cost/month
Total:		~\$7.81/month

## (Optional/Recommended) Deploy an Amazon Elastic Compute Cloud instance to help run automation scripts

We highly recommend deploying an Amazon Elastic Compute Cloud (Amazon EC2) instance to automate the connection to the solution's APIs and AWS boto3 APIs with IAM roles. The following cost estimate assumes that the Amazon EC2 instance is located in the us-east-1 Region and runs eight hours a day, five days a week.

AWS service	Factors	Cost/month
Amazon EC2	176 hours a month x \$0.1108/ per hour (t3.large)	\$19.50
Amazon Elastic Block Store (Amazon EBS)	30 GB x \$0.08/GB-month (gp3) x (176 hours/720 hours)	\$0.59
Total:		~\$20.09

Prices are subject to change. For full details, refer to the pricing webpage for each AWS service you will be using in this solution.

The diagram illustrates the AWS Migration Automation Framework architecture. A central **Migration Automation Server** (MaaS) is connected to various AWS services. The framework is organized into three main components:

- 1. Automation:** This component involves **Automation Scripts** and **AWS Systems Manager** to manage the migration process. It triggers the **Automation** process, which is managed by **AWS Systems Manager** and **Automation Scripts**.
- 2. Admin Functions:** These functions are used for managing the migration process. They interact with **Amazon CloudFront** (Frontend code), **Amazon API Gateway**, **Amazon S3** (Frontend code), **Amazon Cognito**, **AWS Secrets Manager**, and **Amazon EC2**.
- 3. User Functions:** These functions are used for managing the migration process. They interact with **Amazon S3** (Frontend code), **Amazon Cognito**, **AWS Secrets Manager**, and **Amazon EC2**.

The framework also includes an **Optional Migration Tracker Component** (Amazon QuickSight dashboard) and **AWS Managed Services** for rehosting and replatforming.

The solution's AWS CloudFormation template launches the AWS services necessary to help enterprises migrate their servers.

The Cloud Migration Factory on AWS solution uses a migration automation server which is not a part of the AWS CloudFormation deployment. For more details on manually building the server, refer to [Build a migration automation server \(p. 10\)](#).

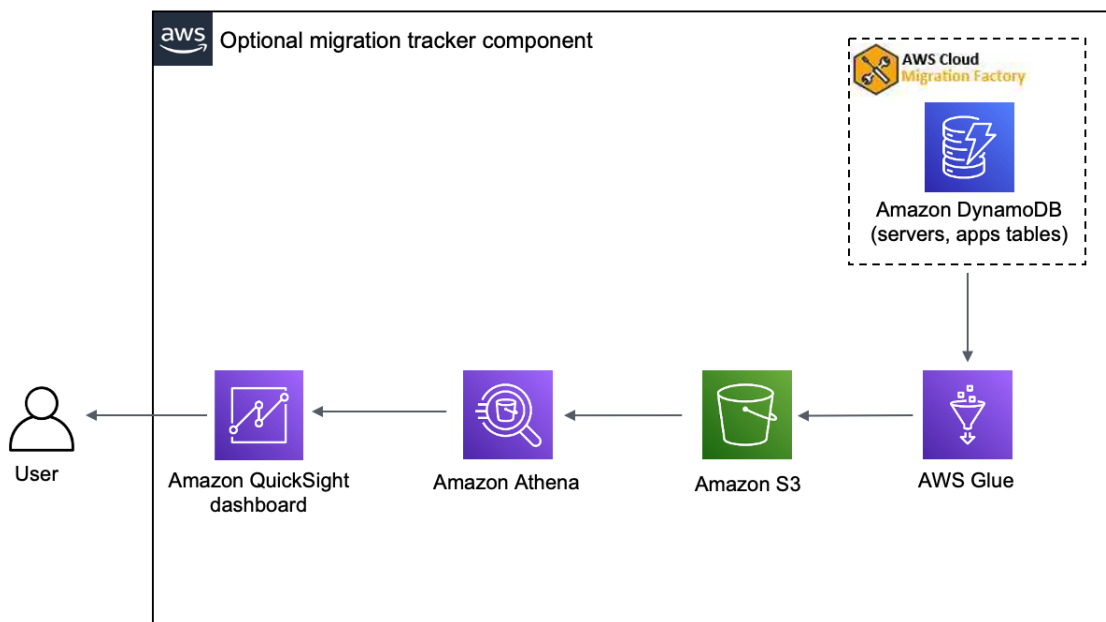
- 
- 4



environment. If your migration pattern is Replatform to Amazon EC2, the `tools` Lambda functions launch CloudFormation templates in the target AWS account to launch Amazon EC2 instances.

## Optional migration tracker

This solution also deploys an optional migration tracker component that tracks the progress of your migration.



**Figure 2: Optional migration tracker component**

The CloudFormation template deploys [AWS Glue](#) to get the migration metadata from the Cloud Migration Factory DynamoDB table and exports the metadata to Amazon S3 twice a day (at 5:00 AM and 1:00 PM UTC). After the AWS Glue job completes, an Amazon Athena save query is initiated, and you can set up Amazon QuickSight to pull the data from the Athena query results. You can then create the visualizations and build a dashboard that meets your business needs. For guidance on creating visuals and building a dashboard, refer to [Build a migration tracker dashboard \(p. 25\)](#).

This optional component is managed by the **Tracker** parameter in the CloudFormation template. By default, this option is activated, but you can deactivate this option by changing the **Tracker** parameter to `false`.

# Solution components

## Migration automation server

This solution leverages a migration automation server to run migrations using Rest APIs. This server isn't automatically deployed with the solution and must be built manually. For more information, refer to [Build a Migration Automation Server \(p. 17\)](#). We highly recommend that you build the server in your AWS environment, but you can also build on-premises in your network environment. The server must meet the following requirements:

- Windows Server 2016 or later versions
- Minimum 4 CPUs with 8 GB RAM
- Deployed as a new virtual machine with no additional applications installed
- (If building in AWS) In the same AWS account and Region as Cloud Migration Factory

Once installed, the server requires internet access and non-restrictive internal network connectivity to the in-scope source servers (source servers)—the on-premises servers to be migrated to AWS.

If port restriction is required from the migration automation server to the source servers, the following ports must be open from the migration automation server to the source servers:

- SMB port (TCP 445)
- SSH port (TCP 22)
- WinRM port (TCP 5985, 5986)

We recommend that the migration automation server be in the same domain as the source servers. If the source servers reside in multiple domains, the security configuration for domain trust in each domain determines whether you need more than one migration automation server.

- If domain trust exists in all the domains with source servers, a single migration automation server will be able to connect to and run automation scripts for all domains.
- If domain trust doesn't exist in all the domains, you must create an additional migration automation server for each untrusted domain, or for each action to be performed on the automation server alternative credentials will need to be provided with appropriate permissions on the source servers.

## Migration services RestAPIs

The Cloud Migration Factory on AWS solution automates the migration process using Rest APIs that are processed through AWS Lambda functions, an Amazon API Gateway, AWS Managed Services, AWS Application Migration Service (AWS MGN) and CloudEndure Migration. When you make a request or initiate a transaction, such as adding a server or viewing a list of servers or applications, Rest API calls are made to Amazon API Gateway which initiates an AWS Lambda function to run the request. The following services detail the components for the automated migration process.

### Log in services

Log in services include the `login` Lambda functions and Amazon Cognito. Once you log in to the solution using the `login` API via the API Gateway. The `login` Lambda functions then validates the

credentials, retrieves an authentication token from Amazon Cognito, and returns the token details back to you. You can use this authentication token to connect to the other services in this solution.

## Admin services

Admin services include the Amazon API Gateway, `admin` Lambda functions, and Amazon DynamoDB. Administrators for the solution can use the `admin` Lambda function to define the migration metadata schema, which are the application and server attributes. The admin services API provides the schema definition for the DynamoDB table. User data including application and server attributes must adhere to this schema definition. Typical attributes include the `app_name`, `wave_id`, `server_name`, and other fields as identified in [Import migration metadata into the factory \(p. 22\)](#). By default, the AWS CloudFormation template deploys a common schema automatically, but this can be customized after deployment.

Administrators can also use admin services to define migration roles for the members of their migration team. The administrator has granular control to map specific user roles to specific attributes and migration stages. A migration stage is a period of time to run certain migration tasks, for example, a build stage, a testing stage, and a cutover stage.

## User services

User services include the Amazon API Gateway, `user` Lambda functions, and Amazon DynamoDB. Users can manage the migration metadata, allowing them to read, create, update, and delete the wave, application, and server data in the migration metadata pipeline.

### Note

A migration wave is a concept of application grouping with a start and an end or cutover date. Wave data includes the migration candidate applications and application groupings scheduled for a particular migration wave.

User services offer an API for the migration team to manipulate the data in the solution: create, update, and delete the data using the Python script and source CSV files. For detailed steps, refer to [Automated migration activities using Migration Factory web console \(p. 71\)](#) and [Automated migration activities using command prompt \(p. 79\)](#).

## Tools services

Tool services include the Amazon API Gateway, `tools` Lambda functions, Amazon DynamoDB, AWS Managed Services, AWS Application Migration Service, and CloudEndure Migration. You can use these services to connect to third-party APIs and automate the migration process. Using CloudEndure Migration and AWS Application Migration Service, the migration team can orchestrate the server launch process with a single button press to launch all servers in the same wave consisting of a group of applications and servers that have the same cutover date. Using AWS Managed Services, the Cloud Migration Factory on AWS solution automates the workload ingestion RFC process, and reduces the manual effort needed during the migration process.

# Migration Factory web interface

The solution includes a Migration Factory web interface hosted in an Amazon Simple Storage Service (Amazon S3) bucket which allows you to complete the following tasks using a web browser:

- Update wave, application, and server metadata from your web browser
- Manage application and server schema definitions
- Connect to third-party services such as AWS Application Migration Service, CloudEndure Migration and AWS Managed Services to automate the migration process

# Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This [shared model](#) can reduce your operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. For more information about security on AWS, visit [AWS Cloud Security](#).

## IAM roles

AWS Identity and Access Management (IAM) roles allow you to assign granular access policies and permissions to services and users in the AWS Cloud. This solution creates IAM roles that grants the AWS Lambda function access to the other AWS services used in this solution.

## Amazon Cognito

The Amazon Cognito user created by this solution is a local user with permissions to access only the RestAPIs for this solution. This user does not have permissions to access any other services in your AWS account. For more information, refer to [Amazon Cognito User Pools](#) in the *Amazon Cognito Developer Guide*.

## Amazon CloudFront

This solution deploys a web console [hosted](#) in an Amazon Simple Storage Service (Amazon S3) bucket. To help reduce latency and improve security, this solution includes an Amazon CloudFront distribution with an origin access identity, which is a special CloudFront user that helps provide public access to the solution's website bucket contents. For more information, refer to [Restricting Access to Amazon S3 Content by Using an Origin Access Identity](#) in the *Amazon CloudFront Developer Guide*.

# Implementation considerations

## Regional deployments

This solution uses Amazon Cognito and Amazon QuickSight, which are currently available in specific AWS Regions only. Therefore, you must launch this solution in a Region where these services are available. For the most current service availability by Region, refer to the [AWS Regional Services List](#).

**Note**

Data transfer during the migration process is not affected by Regional deployments.

# AWS CloudFormation templates

This solution uses AWS CloudFormation to automate the deployment of the Cloud Migration Factory on AWS solution in the AWS Cloud. It includes the following AWS CloudFormation template, which you can download before deployment.



View template

**aws-cloud-migration-factory-solution.template** -

Use this template to launch the Cloud Migration Factory on AWS solution and all associated components. The default configuration deploys AWS Lambda functions, Amazon DynamoDB tables, an Amazon API Gateway, [Amazon CloudFront](#), Amazon Simple Storage Service (Amazon S3) buckets, an Amazon Cognito user pool, and an Amazon Elastic Container Service (Amazon ECS) cluster, but you can also customize the template based on your specific needs.



View template

**aws-cloud-migration-factory-solution-target-**

**account.template** - Use this template to launch the Cloud Migration Factory on AWS solution target account(s). The default configuration deploys IAM roles and an IAM user, but you can also customize the template based on your specific needs.

# Automated deployment

Before you launch the automated deployment, review the architecture, components, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the Cloud Migration Factory on AWS solution into your account.

**Time to deploy:** Approximately 20 minutes

## Prerequisites

### Domain permissions

A domain user with local admin permissions to the in-scope source servers targeted for migration is required for Windows and Linux (sudo permissions) servers. If Linux is not in the domain, other users may be used including an LDAP user with sudo permissions or a local sudo user. Before launching this solution, verify that you have the necessary domain permissions or have coordinated with the appropriate person in your organization with domain permissions.

### AWS Application Migration Service (AWS MGN)

If you use AWS MGN for this solution, you must first initialize the AWS MGN service in every target account and region before launching the target account stack, refer to [Initializing Application Migration Service](#) in the *Application Migration Service User Guide* for more details.

## Deployment overview

The procedure for deploying this architecture on AWS consists of the following steps. For detailed instructions, follow the links for each step.

#### [Step 1. Launch the stack \(p. 12\).](#)

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters.
- Review the template parameters, and adjust if necessary.

#### [Step 2. Launch the target account stack in the target account \(p. 14\).](#)

- Repeat this step for each target AWS account.

#### [Step 3. Create the first user \(p. 15\).](#)

- Create the initial user and log in to the solution.

#### [Step 4. Update the schema for CloudEndure Migration or AWS Application Migration Service \(p. 17\).](#)

- Update the factory schema.

[Step 5. Build a migration automation server \(p. 17\).](#)

- Build a Windows Server 2016 or later server.
- Install the required packages.
- Download the sample automation scripts and update the configuration file.

[Step 6. Test the solution using the automation scripts \(p. 22\).](#)

- Import migration metadata into the factory.
- Get the necessary domain permissions.
- Conduct a test run of the migration automation.

[Step 7. \(Optional\) Build a migration tracker dashboard \(p. 25\).](#)

**Note**

For questions or to get support for this solution, email [<migration-factory-support@amazon.com>](mailto:migration-factory-support@amazon.com). Report technical issues with the solution on the [Issues page](#) of the GitHub repository.

## Step 1. Launch the stack

**Important**

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. AWS owns the data gathered through this survey. Data collection is subject to the [AWS Privacy Policy](#).

To opt out of this feature, download the template, modify the AWS CloudFormation mapping section, and then use the AWS CloudFormation console to upload your template and deploy the solution. For more information, refer to the [Collection of operational metrics \(p. 88\)](#) section of this guide.

This automated AWS CloudFormation template deploys the Cloud Migration Factory on AWS solution in the AWS Cloud.

**Note**

You are responsible for the cost of the AWS services used while running this solution. Refer to the [Cost \(p. 2\)](#) section for more details. For full details, refer to the pricing webpage for each AWS service you will be using in this solution.

1. Sign in to the AWS Management Console and select the button to launch the `cloud-migration-factory-solution` AWS CloudFormation template.



You can also [download the template](#) as a starting point for your own implementation.

2. The template launches in the US East (N. Virginia) Region by default. To launch this solution in a different AWS Region, use the Region selector in the console navigation bar.

**Note**

This solution uses Amazon Cognito and Amazon QuickSight, which are currently available in specific AWS Regions only. Therefore, you must launch this solution in an AWS Region



where these services are available. For the most current availability by Region, refer to the [AWS Regional Services List](#).

3. On the **Create stack** page, verify that the correct template URL shows in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack.
5. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
<b>Application name</b>	migration-factory	<p>Enter a prefix to the AWS CloudFormation <b>Physical ID</b> that identifies the AWS services deployed by this solution.</p> <p><b>Note</b> The <b>Application name</b> is used as a prefix to identify the AWS resources that are deployed: <code>&lt;application-name&gt;-&lt;environment-name&gt;-&lt;aws-resource&gt;</code>. If changing the default name, we recommend that you keep the combined prefix labels to 40 characters or less to ensure you do not exceed character limitations.</p>
<b>Environment name</b>	test	<p>Enter a name to identify the network environment where the solution is deployed. We recommend a descriptive name such as test, dev, or prod.</p> <p><b>Note</b> The <b>Environment name</b> is used as a prefix to identify the AWS resources that are deployed: <code>&lt;application-name&gt;-&lt;environment-name&gt;-&lt;aws-resource&gt;</code>. If changing the default name, we recommend that you keep the combined prefix labels to 40 characters or</p>

Parameter	Default	Description
		less to ensure you do not exceed character limitations.
<b>Migration Tracker</b>	true	By default, the optional migration tracker dashboard is activated, but you can deactivate it by changing this parameter to false.
<b>Replatform EC2</b>	true	By default, the Replatform EC2 feature is activated, but you can deactivate it by changing this parameter to false.
<b>ServiceAccountEmail</b>	serviceaccount@yourdomain	Default service account email address, the migration factory automation scripts uses this account to connect to the factory API.

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Check the boxes acknowledging that the template will create [AWS Identity and Access Management](#) (IAM) resources and that it might require the capability **CAPABILITY\_AUTO\_EXPAND**.
9. Choose **Create stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a **CREATE\_COMPLETE** status in approximately 20 minutes.

### Important

If you are using AWS MGN, you must complete the prerequisite for AWS MGN before continuing to Step 2. Refer to [Application Migration Service initialization and permissions](#).

## Step 2. Launch the target account stack in the target AWS account

This automated AWS CloudFormation template deploys IAM roles in the target AWS account to allow the factory account to assume roles and perform MGN actions in the target account. Repeat this step for each target account.

### Note

The target account must be initialized for AWS Application Migration Service before launching this stack, refer to [Initializing Application Migration Service](#) in the *Application Migration Service User Guide* for more details.

The target account stack must be launched in the same Region as the factory stack in the previous step regardless of which Region will be used as the migration target Region. This stack is for cross account permissions only.

1. Sign in to the [AWS CloudFormation console](#). Choose **Create stack** then select **With new resources**, to start the deployment of the template. You can also [download the template](#) as a starting point for your own implementation.

2. On the **Specify stack details** page, assign a name to your solution stack.
3. Under **Parameters**, review the parameters for the template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
<b>FactoryAWSAccountId</b>	111122223333	Enter an account ID where the Migration Factory was deployed.  <b>Note</b> Launch this stack in the same AWS Region as the Migration Factory stack.
<b>Replatform</b>	Yes	Turn on this option if you plan to use the Replatform EC2 module of this solution
<b>RehostMGN</b>	Yes	Turn on this option if you plan to use the Rehost MGN module of this solution

4. Choose **Next**.
5. On the **Configure stack options** page, choose **Next**.
6. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create [AWS Identity and Access Management](#) (IAM) resources.
7. Choose **Create stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a **CREATE\_COMPLETE** status in approximately 5 minutes.

## Step 3. Create the first user

### Create the initial user and log in to the solution

Use the following procedure to create the initial user.

1. Navigate to the [Amazon Cognito console](#).
2. Choose **Manage User Pools**.
3. On the **Your User Pools** page, choose the user pool that starts with the **migration-factory** prefix.
4. In the left navigation pane, choose **Users and groups**.
5. Select the **Users** tab and choose **Create user**.
6. In the **Create user** dialog box, **Username** field, do the following:
  - Enter an email address.
  - Select the checkbox next to **Email**.
  - Verify that the **Send an invitation** option is selected.

**Important**

This email address must be different from the one you used in the **ServiceAccountEmail** parameter, which is used during the deployment of the primary CloudFormation template.

7. In the **Temporary password** field, enter a password.

**Note**

The password must be at least eight characters in length, including upper and lower case letters, numbers, and special characters.

8. Optionally, enter a **Phone Number**. If you do not, verify that the **Mark phone number as verified** option is not checked.
9. In the **Email** field, enter the same email address used for the **Username**. Verify that the **Mark email as verified** option is selected.
10. Choose **Create user**.

**Note**

You will receive an email with the temporary password. Until you change the temporary password, the **Account status** for this user will display as **FORCE\_CHANGE\_PASSWORD**. You can update the password later in the deployment.

## Add a user to the admin group

In the Amazon Cognito console, use the following procedure to add a user to the default Admin group.

1. Ensure that you have selected **Users and groups** option from left nav menu. From the **Users** page, select the created **Username**.
2. On the **Users** page, select **Add to group**.
3. In the dialog box, select the drop-down arrow and choose **Admin**. Then choose **Add to group**. This default Admin group authorizes the user to manage migrations in the solution.

## Identify the CloudFront URL

Use the following procedure to identify the solution's Amazon CloudFront URL. This allows you to log in and change the password.

1. Navigate to the [AWS CloudFormation console](#) and select the solution's stack.
2. On the **Stacks** page, select the **Outputs** tab and select the **Value** for the **MigrationFactoryURL**.

**Note**

If you launched the solution in an AWS Region other than US East (N. Virginia), CloudFront may take longer to deploy and the **MigrationFactoryURL** may not be accessible immediately (you will receive an access denied error). It can take up to four hours before the URL becomes available. The URL includes `cloudfront.net` as part of the string.

3. Sign in with your username and temporary password, then create a new password and choose **Change Password**.

**Note**

The password must be at least eight characters in length, including upper and lower case letters, numbers, and special characters.

## Step 4. Update the factory schema

### Update the aws\_accountid for AWS MGN

1. On the **Migration Factory** web interface, select **Administration**, then select **Attributes**.
2. On the **Attribute Configuration** page, select **Application**, then select **Attributes**.
3. Select **AWS Account Id**, then choose **Edit**.

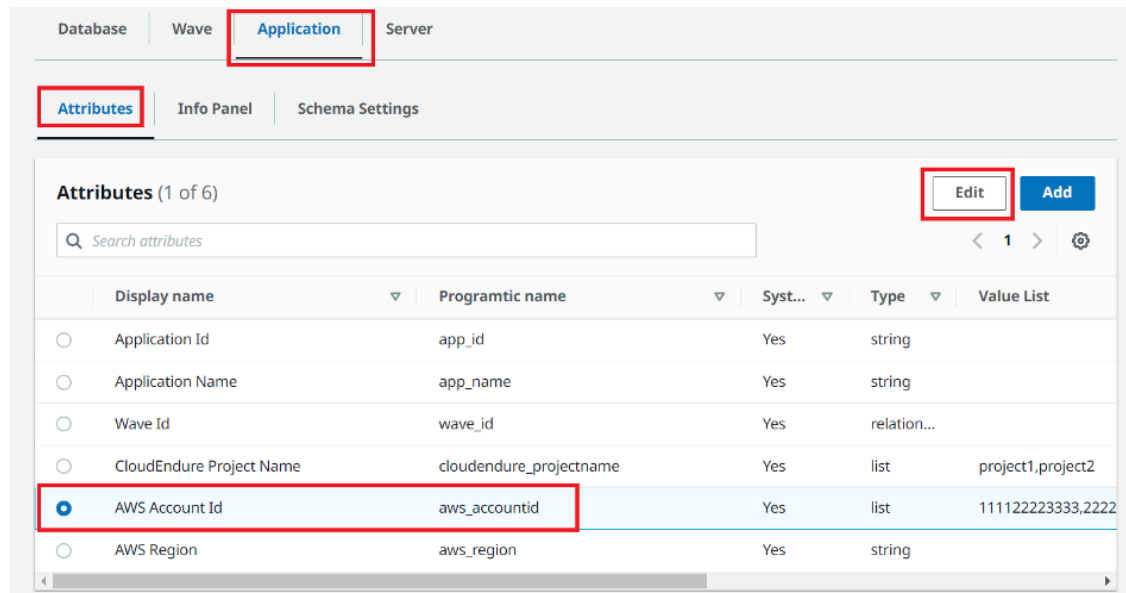


Figure 4: Migration Factory web interface Attribute Details tab

4. On the **Amend attribute** page, update **Value list** with your target AWS account ID and choose **Save**.

#### Note

If you have more than one AWS account ID, separate the ID with commas.

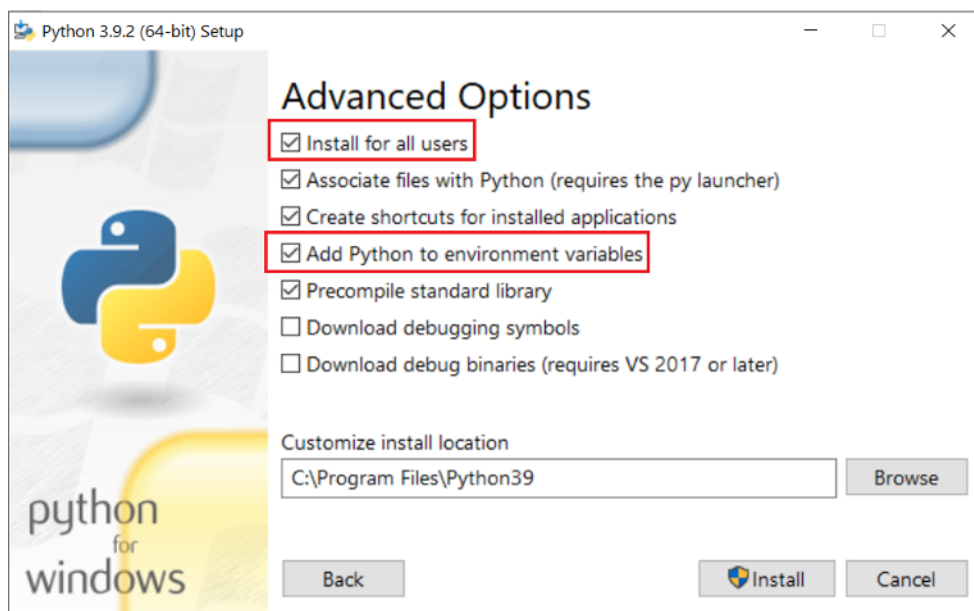
## Step 5. Build a migration automation server

### Build a Windows Server 2016 or later server

The migration automation server is used to run migration automation. We recommend creating the server in your AWS account, but it can also be created in your on-premises environment. If built in your AWS account, it must be in the same AWS account and Region as Cloud Migration Factory. To review the server requirements, refer to [Migration automation server \(p. 6\)](#).

On your migration automation server, use the following procedure to install the packages needed to build the server, download the sample automation scripts, and update the configuration file.

1. Download [Python v3.9.2](#).
2. Log in as administrator and install Python v3.9.2, and choose **Customize installation**.
3. Choose **Next**, and select **Install for all users** and **Add Python to environment variables**. Choose **Install**.



**Figure 5: Python installation - Advanced options tab**

4. Verify that you have administrator privileges, open `cmd.exe`, and run the following commands to install the Python packages one at a time:

```
python -m pip install requests
python -m pip install paramiko
python -m pip install boto3
```

If any of these commands fail, upgrade pip by running the following command:

```
python -m pip install --upgrade pip
```

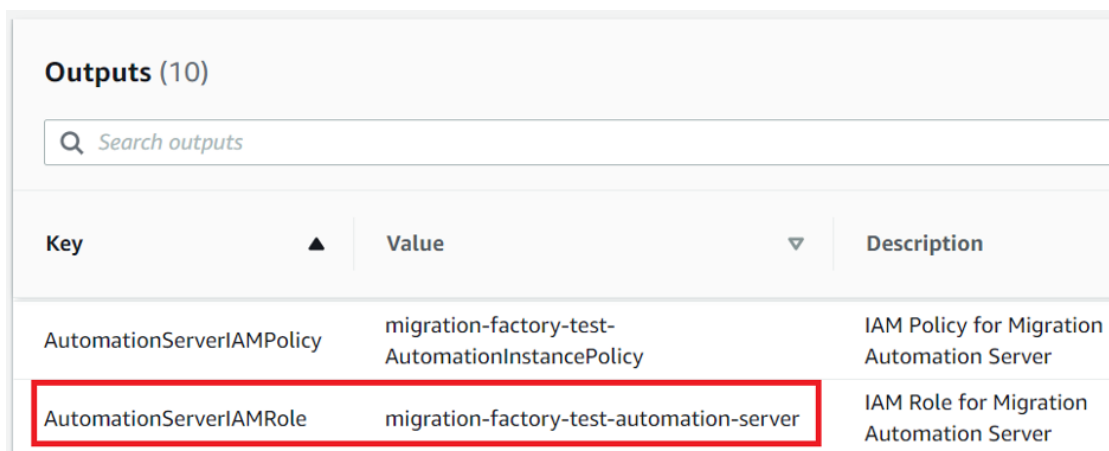
5. Install [AWS CLI \(Command Line Interface\)](#).

## Configure AWS permissions for the migration automation server

Depending on where you deploy the migration execution server, choose one of the options below to configure AWS permissions for the migration automation server. The IAM role or policy provides the permission to the automation server and the access to AWS Secrets Manager to get agent installation keys and factory service account credentials. You can deploy the migration automation server either to AWS as an EC2 instance or on-premises.

**(Recommended)** Option 1: Use the following procedure to configure the permissions for the migration automation server in Amazon EC2 and in the same AWS account and Region as factory.

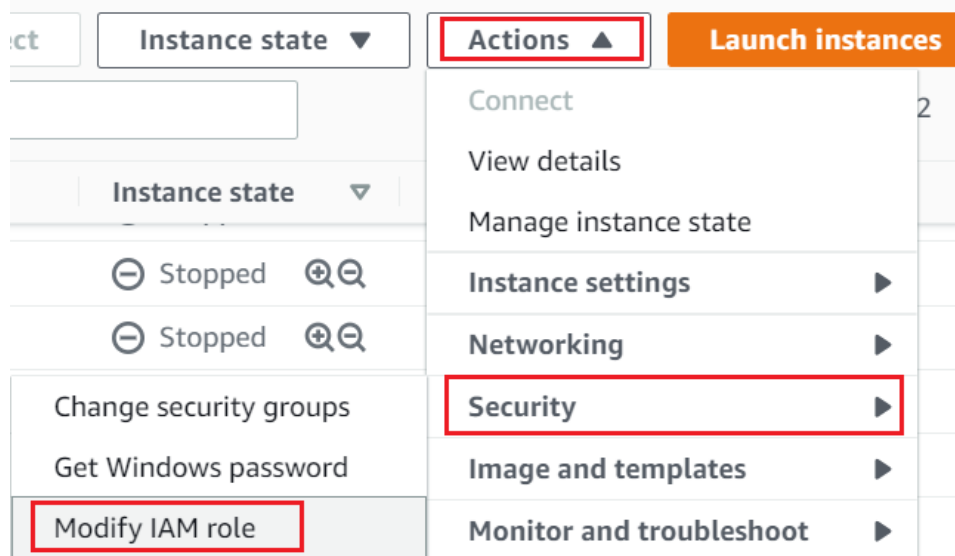
1. Navigate to the [AWS CloudFormation console](#) and select the solution's stack.
2. Select the **Outputs** tab, under the **Key** column, locate `AutomationServerIAMRole` and record the **Value** to use later in the deployment.



Key	Value	Description
AutomationServerIAMPolicy	migration-factory-test-AutomationInstancePolicy	IAM Policy for Migration Automation Server
AutomationServerIAMRole	migration-factory-test-automation-server	IAM Role for Migration Automation Server

**Figure 6: AWS CloudFormation console, Outputs tab**

3. Navigate to the [Amazon Elastic Compute Cloud](#) console.
4. From the left navigation pane, select **Instances**.
5. On the **Instances** page, use the Filter Instances field and enter the name of the migration execution server to find the instance.
6. Select the instance and select **Actions** on the menu.
7. Select **Security** from the drop down list then select **Modify IAM role**.



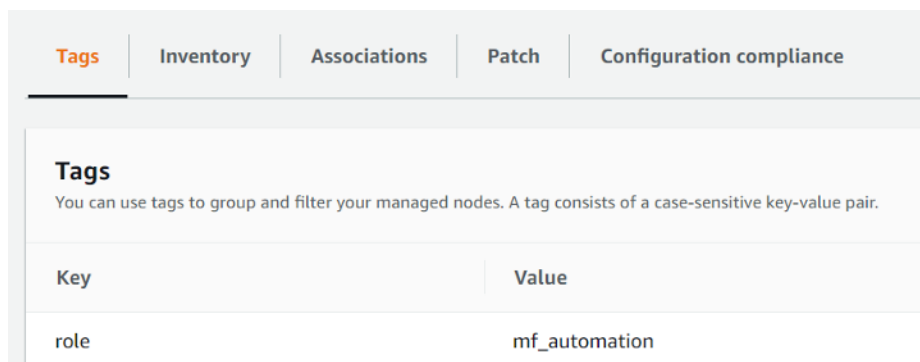
**Figure 7: Amazon EC2 console**

8. From the list of IAM roles, locate and select the IAM role containing the value for AutomationServerIAMRole that you recorded in Step 2, and choose **Save**.
9. Use your remote desktop protocol (RDP) to log in to the migration automation server.
10. Download and install [SSM Agent](#) on the migration automation server.

**Note**

By default, AWS Systems Manager agent is preinstalled on Windows server 2016 Amazon Machine Images. Perform this step only if the SSM Agent is not installed.

11. Add the following tag to the migration automation server EC2 instance: **Key**= role and **Value** = mf\_automation.



The screenshot shows the 'Tags' tab in the AWS Systems Manager console. Below the tab name, there is a description: 'You can use tags to group and filter your managed nodes. A tag consists of a case-sensitive key-value pair.' Below this is a table with two columns: 'Key' and 'Value'. A single row is visible with the key 'role' and the value 'mf\_automation'.

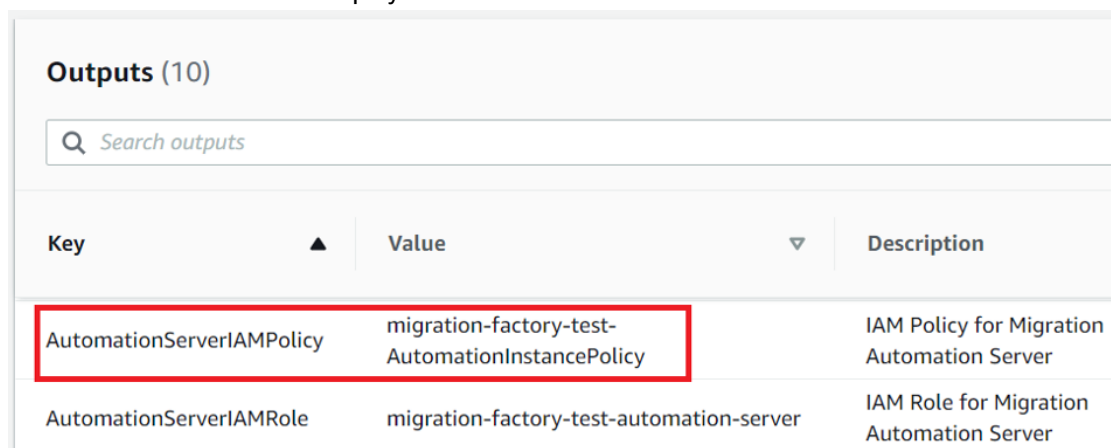
Key	Value
role	mf_automation

**Figure 8: Add tag to migration automation server**

12. Open the AWS Systems Manager console and choose **Fleet Manager**. Check the automation server status, and make sure the SSM Agent ping status is **online**.

Option 2: Use the following procedure to configure the permissions for the migration automation server on-premises.

1. Navigate to the [AWS CloudFormation console](#) and select the solution's stack.
2. Select the **Outputs** tab, under the **Key** column, locate `ExecutionServerIAMPolicy` and record the **Value** to use later in the deployment.



The screenshot shows the 'Outputs (10)' tab in the AWS CloudFormation console. There is a search bar labeled 'Search outputs'. Below is a table with three columns: 'Key', 'Value', and 'Description'. The first row is highlighted with a red box. The 'Key' is 'AutomationServerIAMPolicy', the 'Value' is 'migration-factory-test-AutomationInstancePolicy', and the 'Description' is 'IAM Policy for Migration Automation Server'.

Key	Value	Description
AutomationServerIAMPolicy	migration-factory-test-AutomationInstancePolicy	IAM Policy for Migration Automation Server
AutomationServerIAMRole	migration-factory-test-automation-server	IAM Role for Migration Automation Server

**Figure 9: AWS CloudFormation console, Outputs tab**

3. Navigate to the [Identity and Access Management](#) console.
4. From the left navigation pane, select **Users**, then select **Add User**.
5. In the **User name** field, create a new user.
6. Under Select AWS access type, tick the checkbox for **Programmatic access**.
7. Select **Next: Permissions**.
8. On the **Add user** page, under **Set permissions**, select **Attach existing policy directly**. A list of policies displays.
9. From the list of policies, locate and select the policy containing the value for `AutomationServerIAMPolicy` that you recorded in [Step 2 \(p. 14\)](#).
10. Select **Next: Tags**, then select **Next: Review** and verify that the correct policy is selected.
11. Select **Create User**.
12. On the **Add user** confirmation page, record the **Access Key Id** and **Secret Access Key**, and select **Close**.



13. Use your remote desktop protocol (RDP) to log in to the migration execution server.
14. Signed in as an administrator, open a command prompt (CMD . exe).
15. Run the following command to configure the AWS credentials on the server. Replace `<your_access_key_id>`, `<your_secret_access key>`, and `<your_region>` with your values.

```
SETX /m AWS_ACCESS_KEY_ID <your_access_key_id>
SETX /m AWS_SECRET_ACCESS_KEY <your_secret_access key>
SETX /m AWS_DEFAULT_REGION <your_region>
```

16. Install the agent using Hybrid mode (on-prem servers).
  - a. Navigate to the [AWS Systems Manager console](#).
  - b. In the navigation pane, choose **Hybrid Activations**.
  - c. Choose **Create activation**.
  - d. In the **Instance limit** field, specify the total number of automation servers that you want to register. The default value is 1 instance.
  - e. In the **IAM role name** section, choose **Select an existing custom IAM role that has the required permissions**, and select an IAM role with **automation-server** suffix.
  - f. Choose **Create activation**.
  - g. Copy the **Activation Code** and **Activation ID** to the notepad to use later.
  - h. Copy the command below to a notepad and replace `<activation-code>` and `<activation-id>` with the value from previous step, and replace `<region>` with your factory region value such as `us-east-1`.

```
$code = <activation-code>
$id = <activation-id>
$region = <region>
$dir = $env:TEMP + "\ssm"
New-Item -ItemType directory -Path $dir -Force
(New-Object
System.Net.WebClient).DownloadFile("https://amazon-ssm-$region.s3.
$region.amazonaws.com/latest/windows_amd64/AmazonSSMAgentSetup.exe", $dir +
"\AmazonSSMAgentSetup.exe")
t-Process .\AmazonSSMAgentSetup.exe -ArgumentList
@("/q", "/log", "install.log",
"CODE=$code", "ID=$id",
"REGION=$region") -Wait
Get-Content ($env:ProgramData +
"\Amazon\SSM\InstanceData\registration")
Get-Service -Name "AmazonSSMAgent"
```

- i. Log in to Migration Automation Server as administrator.
  - j. Open Microsoft PowerShell and run the command above.
  - k. After that, you will see the SSM Agent up and running.
  - l. In the AWS Systems Manager console, choose **Fleet Manager**. Identify the node ID with **mi-** prefix with **Online** status.

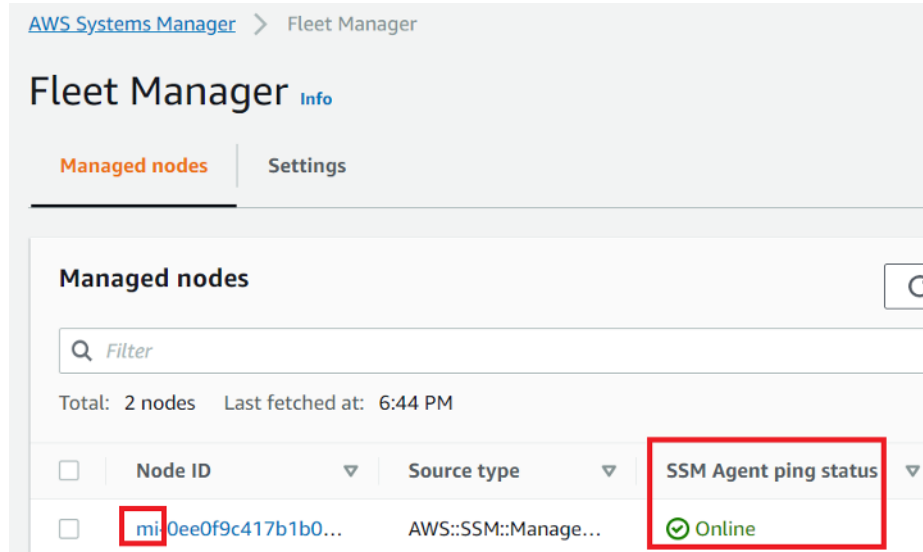


Figure 10: Fleet manager - managed node ID

- m. Select the **Node ID** and make sure the IAM role is the one you selected with **automation-server** suffix.
- n. Add the following tag for this Hybrid node: **Key** = `role` and **Value** = `mf_automation`. All lower case.

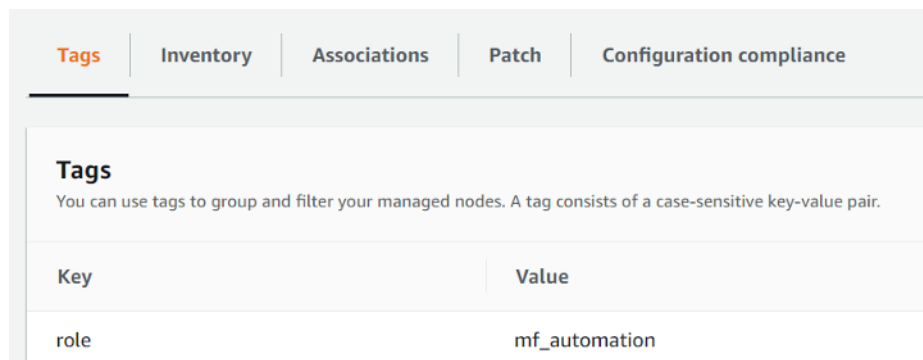


Figure 11: Add tag - hybrid node

## Step 6. Test the solution using the automation scripts

### Import migration metadata into the factory

To start the migration process, download the [server-list.csv](#) file from the GitHub repository. The `server-list.csv` file is an example AWS MGN Service migration intake form to import the attributes for the in-scope source servers.

#### Note

The `.csv` file and the sample automation scripts were part of the package from the same GitHub repository.

You can customize the form for your migration by replacing the sample data with your specific server and application data. The following table details the data to replace to customize this solution for your migration needs.

Field name	Required?	Description
wave_name	Yes	The wave name is based on priority and application server dependencies. Obtain this identifier from your migration plan.
app_name	Yes	The names of the applications that are in-scope for migration. Confirm that your application grouping includes all the applications sharing the same servers.
aws_accountid	Yes	A 12-digit identifier for your AWS account located in your account profile. To access, select your account profile from the upper-right corner of the AWS Management Console and select <b>My Account</b> from the drop-down menu.
aws_region	Yes	AWS Region code. For example, us-east-1. Refer to the <a href="#">full Region code list</a> .
server_name	Yes	The name of the on-premises servers that are in-scope for migration.
server_os_family	Yes	The operating system (OS) that is running on the in-scope source servers. Use either <b>windows</b> or <b>linux</b> since this solution supports only these operating systems.
server_os_version	Yes	<p>The version of the OS running on the in-scope source servers.</p> <p><b>Note</b> Use the OS version, not the Kernel version, for example, use RHEL 7.1, Window Server 2012 R2, or CentOS 7.5, 7.6. Do not use Linux 3.xx, 4.xx, or Windows 8.1.x.</p>
server_fqdn	Yes	The source server's fully qualified domain name, which is the server name followed by the domain name. For example, server123.company.com.
server_tier	Yes	A label to identify whether the source server is a <b>web</b> , <b>app</b> , or a

Field name	Required?	Description
		<b>database</b> server. We recommend designating the source server as <b>app</b> if the server functions as more than one tier, for example, if the server runs web, app, and database tiers together.
server_environment	Yes	A label to identify the server's environment. For example, <b>dev</b> , <b>test</b> , <b>prod</b> , <b>QA</b> , or <b>pre-prod</b> .
r_type	Yes	A label to identify the migration strategy. For example, <b>Retire</b> , <b>Retain</b> , <b>Relocate</b> , <b>Rehost</b> , <b>Repurchase</b> , <b>Replatform</b> , <b>Rearchitected</b> , <b>TBC</b> .
subnet_IDs	Yes	The subnet ID for the target Amazon EC2 instance for the migration post-cutover.
securitygroup_IDs	Yes	The security group ID for the target Amazon EC2 instance for the migration post-cutover.
subnet_IDs_test	Yes	The target subnet ID for the source server that will be tested.
securitygroup_IDs_test	Yes	The target security group ID for the source server that will be tested.
instanceType	Yes	The Amazon EC2 instance type identified in the discovery and planning effort. For information about EC2 instance types, refer to <a href="#">Amazon EC2 Instance Types</a> .
tenancy	Yes	The tenancy type, which is identified during the discovery and planning efforts. Use one of the following values to identify the tenancy: <b>Shared</b> , <b>Dedicated</b> , or <b>Dedicated Host</b> . You can use <b>Shared</b> as the default value unless an application's license requires a specified type.
Tags	No	The tags for the server resources, such as CostCenter=123;BU=IT;Location=US
private_ip	No	The private IP for the target instance. If not included, instance will get an IP from DHCP.

Field name	Required?	Description
iamRole	No	IAM role for the target instance. If not included, no IAM role will be attached to the target instance.

1. Log in to the Cloud Migration Factory web console.
2. Under **Migration Management**, select **Import** and choose **Select file**. Select the intake form you completed previously and choose **Next**.
3. Review the changes and make sure you do not see any errors (Information message is normal), and choose **Next**.
4. Choose **Upload** to upload servers.

## Access the domains

The sample automation scripts included with this solution connect to the in-scope source servers to automate migration tasks, such as the installation of the replication agent, and shutting down the source servers. In order to conduct a test run of the solution, a domain user with local admin permissions to the source servers is required, for Windows and Linux (sudo permissions) servers. If Linux is not in the domain, other users such as LDAP user with sudo permissions or a local sudo user may be used. For more information about automated migration tasks, refer to [Automated migration activities using Migration Factory web console \(p. 71\)](#) and [Automated migration activities using command prompt \(p. 79\)](#).

## Conduct a test run of the migration automation

This solution lets you conduct a test run of the migration automation. Using automation scripts, the migration process imports the data from the migration CSV file into the solution. Prerequisite checks are conducted for the source servers, the replication agent is pushed to the source servers, replication status is verified, and the target server is launched from the Migration Factory web interface. For step-by-step instructions on running a test, refer to [Automated migration activities using Migration Factory web console \(p. 71\)](#) and [Automated migration activities using command prompt \(p. 79\)](#).

## Step 7. (Optional) Build a migration tracker dashboard

If you deployed the optional migration tracker component, you can set up a QuickSight dashboard that will visualize the migration metadata stored in the Amazon DynamoDB table. Use the following procedures to (1) set up the QuickSight permissions and connections and (2) [create a dashboard \(p. 32\)](#).

### Note

If the Migration Factory is empty and there is no wave, application, and server data, then there will not be any data to build a QuickSight dashboard.

## Set the permission and connections

If you have not set up Amazon QuickSight in your AWS account, refer to [Setting Up for Amazon QuickSight](#) in the *Amazon QuickSight User Guide*. After you have set up a QuickSight subscription, use the following procedure to set the permissions and connections between QuickSight and this solution.

### Note

This solution uses Amazon QuickSight enterprise license. However if you don't want the email reporting, insights, and the hourly data refresh, you can opt for a standard license, which can also be used with migration tracker.

First, connect QuickSight with the Amazon S3 bucket:

1. Navigate to the [QuickSight console](#).
2. On the QuickSight page, in the top right corner, choose the icon displaying your username to access the drop-down menu and select **Manage QuickSight**.
3. On the **Account name** page, from the left menu pane, select **Security & permissions**.
4. On the **Security & permissions** page, under the **QuickSight access to AWS services** section, select **Add or remove**.
5. From the **QuickSight access to AWS services** page, tick the checkbox for **Amazon S3**.
6. On the **Select Amazon S3 buckets** dialog box, verify that you are in the **S3 Buckets Linked to QuickSight Account** tab and tick both the right and left checkboxes for the **athena-results** and **migration-tracker** S3 buckets, as shown in Figure 6.

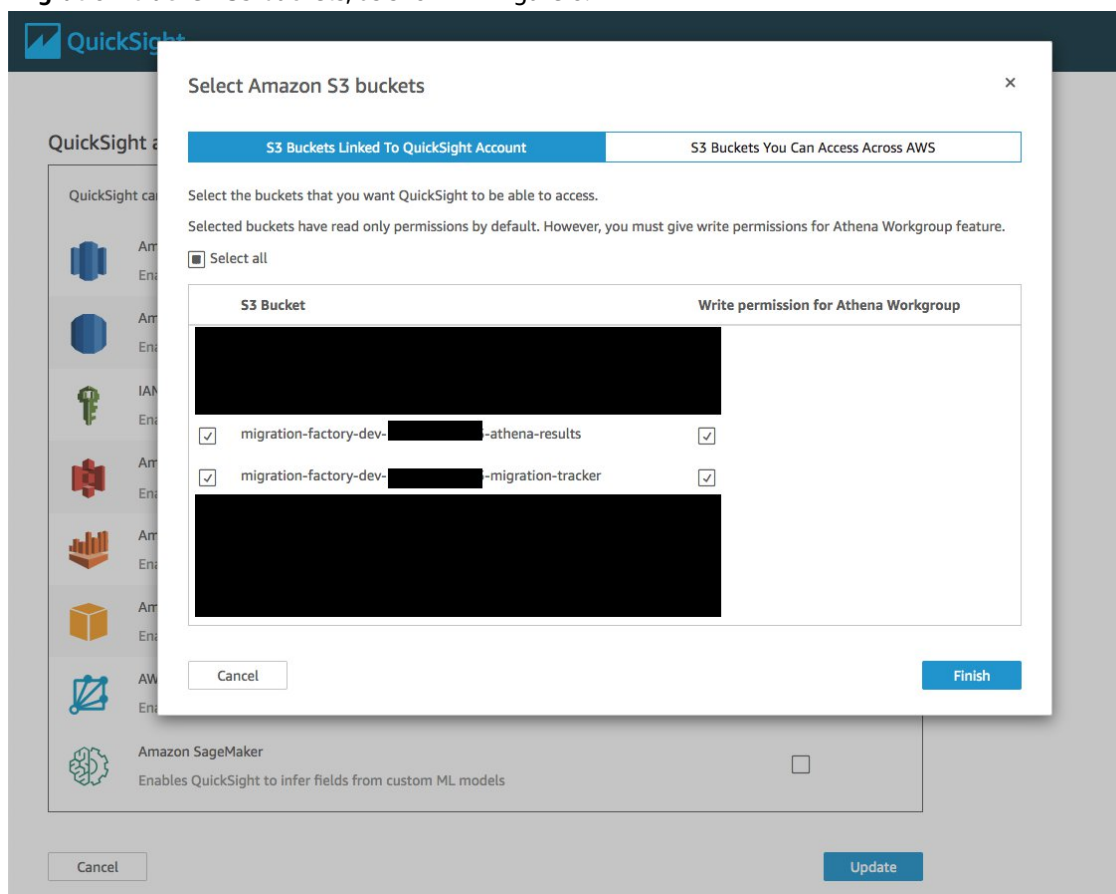


Figure 12: QuickSight select Amazon S3 buckets dialog box

### Note

If you are already using QuickSight for other S3 data analysis, untick and retick the Amazon S3 option to display the bucket selection dialog box.

7. Choose **Finish**.

Next, set up permissions for Amazon Athena:

1. From the **QuickSight access to AWS services** page, tick the checkbox for **Amazon Athena**.
2. On the **Amazon Athena permissions** dialog box, choose **Next**.
3. On the **Amazon Athena resources** dialog box, verify that you are in the **S3 Buckets Linked to QuickSight Account** tab and verify that the same S3 buckets are checked - **athena-results** and **migration-tracker**, as shown in Figure 9.

Select Amazon S3 buckets

S3 Buckets Linked To QuickSight Account S3 Buckets You Can Access Across AWS

Select the buckets that you want QuickSight to be able to access.  
Selected buckets have read only permissions by default. However, you must give write permissions for Athena Workgroup feature.

☐ Select all

S3 Bucket	Write permission for Athena Workgroup
[redacted]	<input type="checkbox"/>
<input checked="" type="checkbox"/> migration-factory [redacted]-athena-results	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> migration-factory [redacted]-migration-tracker	<input checked="" type="checkbox"/>
[redacted]	<input type="checkbox"/>
[redacted]	<input type="checkbox"/>
[redacted]	<input type="checkbox"/>

Cancel Finish

Figure 13: QuickSight Amazon Athena resources dialog box

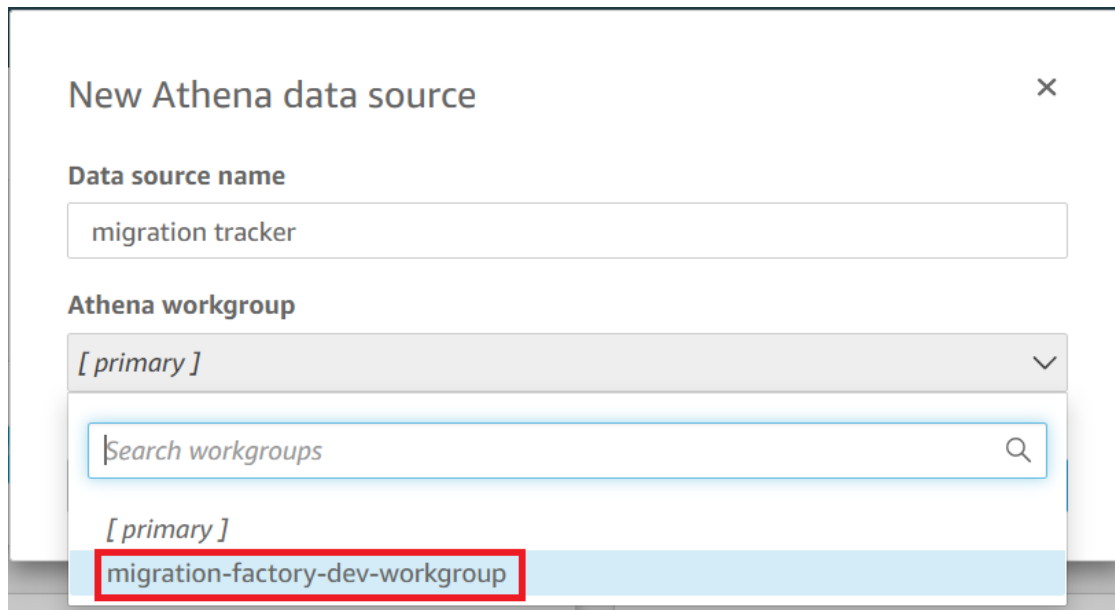
4. Choose **Finish**.
5. From the **QuickSight access to AWS services** page, choose **Update**.

Next, set up a new analysis:

1. Select the QuickSight logo to return to the QuickSight homepage.
2. On the **Analysis** page, choose **New Analysis**.
3. Choose **New dataset**.
4. On the **Create a Data Set** page, choose **Athena**.
5. In the **New Athena data source** dialog box, take the following actions:
  - a. In the **Data source name** field, enter a name for the data source
  - b. In the **Athena workgroup** field, select the appropriate **<migration-factory>-workgroup**.

**Note**

If you have deployed this solution multiple times, there will be more than one workgroup. Select the one that was created for your current deployment.



The screenshot shows a 'New Athena data source' dialog box. It has a title bar with a close button (X). Below the title, there is a 'Data source name' label and a text input field containing 'migration tracker'. Below that is an 'Athena workgroup' label and a dropdown menu. The dropdown menu is open, showing a search bar with the placeholder text 'Search workgroups' and a magnifying glass icon. Below the search bar, there is a list of workgroups. The first workgroup is '[ primary ]'. The second workgroup is 'migration-factory-dev-workgroup', which is highlighted with a red rectangular box.

**Figure 14: New Athena data source dialog box**

6. Choose **Validate connection** to ensure that QuickSight can communicate with Athena.
7. After the connection is validated, choose **Create data source**.
8. In the next dialog box, **Choose your table**, take the following actions:
  - a. From the **Catalog** list, choose **AwsDataCatalog**.
  - b. From the **Database** list, choose **<Athena-table>-tracker**.
  - c. From the **Tables** list, choose **<tracker-name>-general-view**.
  - d. Choose **Select**.



Choose your table ×

migration tracker

**Catalog: contain sets of databases.**

AwsDataCatalog ▼

**Database: contain sets of tables.**

migration-factory-dev-tracker ▼

**Tables: contain the data you can visualize.**

☐ migration\_factory\_dev\_apps

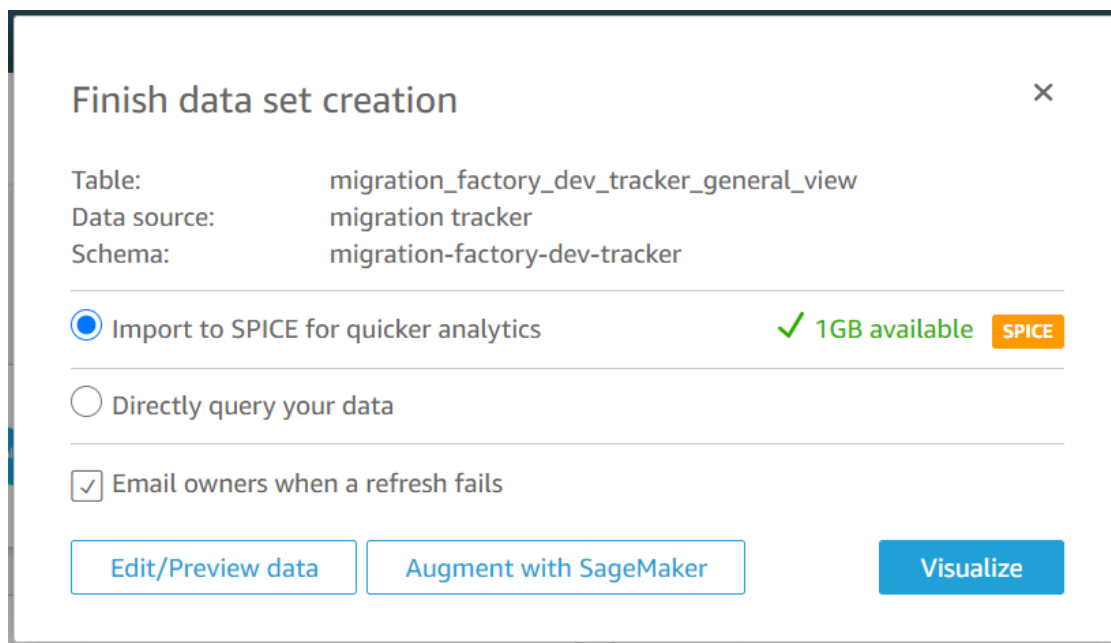
☐ migration\_factory\_dev\_servers

☒ migration\_factory\_dev\_tracker\_general\_view

[Edit/Preview data](#) [Use custom SQL](#) [Select](#)

**Figure 15: Choose your table dialog box**

9. In the next dialog box, **Finish data set creation**, select **Visualize**.



The dialog box titled "Finish data set creation" contains the following information:

- Table: migration\_factory\_dev\_tracker\_general\_view
- Data source: migration tracker
- Schema: migration-factory-dev-tracker

Below this information are two radio button options:

- ☒ Import to SPICE for quicker analytics (with a green checkmark and "1GB available" text, and a "SPICE" button)
- ☐ Directly query your data

There is also a checkbox option:

- ☒ Email owners when a refresh fails

At the bottom are three buttons: "Edit/Preview data", "Augment with SageMaker", and "Visualize".

Figure 16: Finish data set creation dialog box

After the data is imported, you will be redirected to the Analysis page. However, before creating your visuals, set up a schedule to refresh your dataset.

1. Select the QuickSight logo to return to the QuickSight homepage.
2. From the left menu pane, choose **Datasets**.
3. On the Datasets page, select the *<migration-factory>-general-view* dataset.

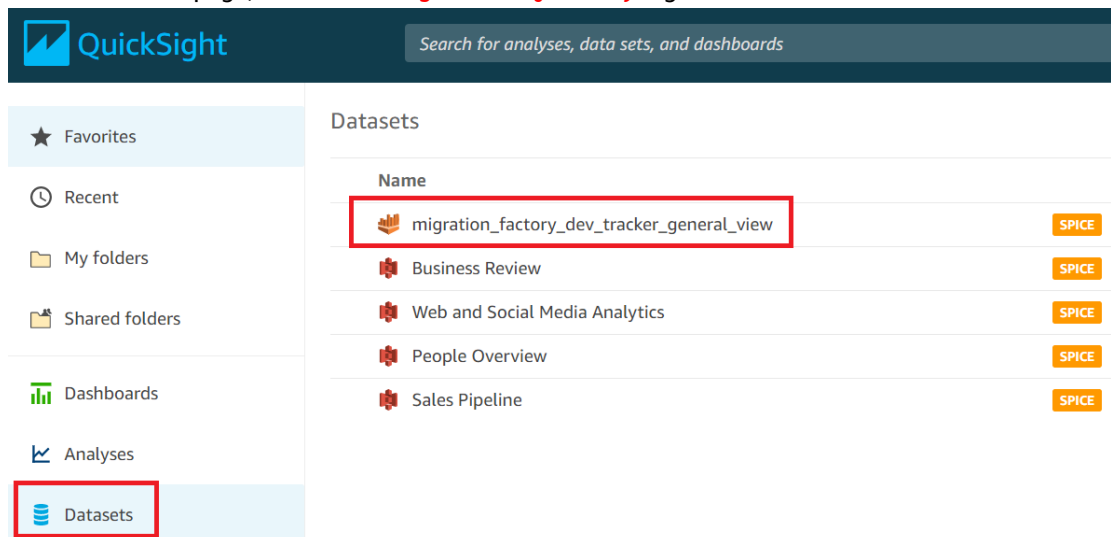



Figure 17: QuickSight Datasets page

4. In the *<migration-factory>-general-view* dialog box, choose **Schedule refresh**.

 migration\_factory\_dev\_tracker\_general\_view ×

SPICE

Data Set 0 bytes

Import complete:  
success  
0 rows were imported to SPICE  
0 rows were skipped

Last refreshed: a minute ago  
[View history](#)

Refresh Now

Schedule refresh

☒ Email owners when a refresh fails

Data source name: migration tracker  
Database name: ATHENA

Delete data set

Share

Edit data set

Duplicate data set

Row-level security

Column-level security

Create analysis

**Figure 18: Migration tracker general view dialog box**

5. Choose **Create**.
6. In the **Create a schedule** dialog box, select the appropriate time zone, the refresh frequency, and enter a starting time.
7. Choose **Create**.

Create a schedule

After you create an hourly refresh schedule, you can't create any other refresh schedules for this data set.

Time zone

Europe/London

Repeats

Hourly

Starting

2020-09-24 14:00

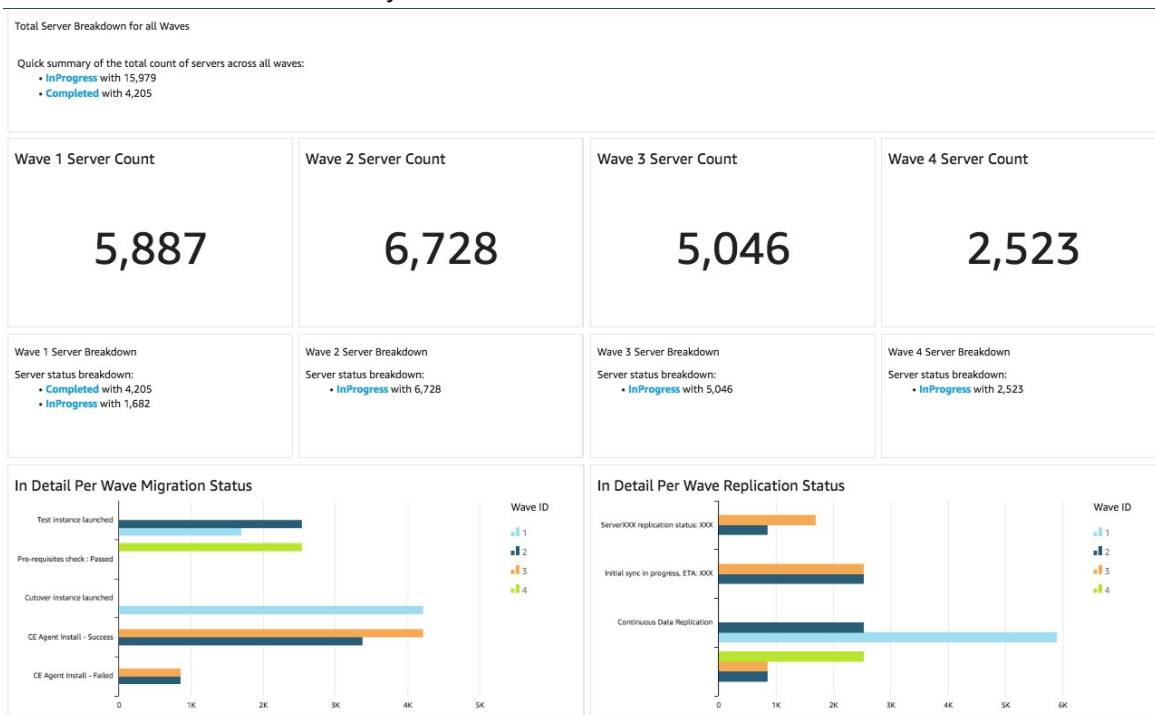
Cancel

Create

Figure 19: Create a schedule dialog box

## Create a dashboard

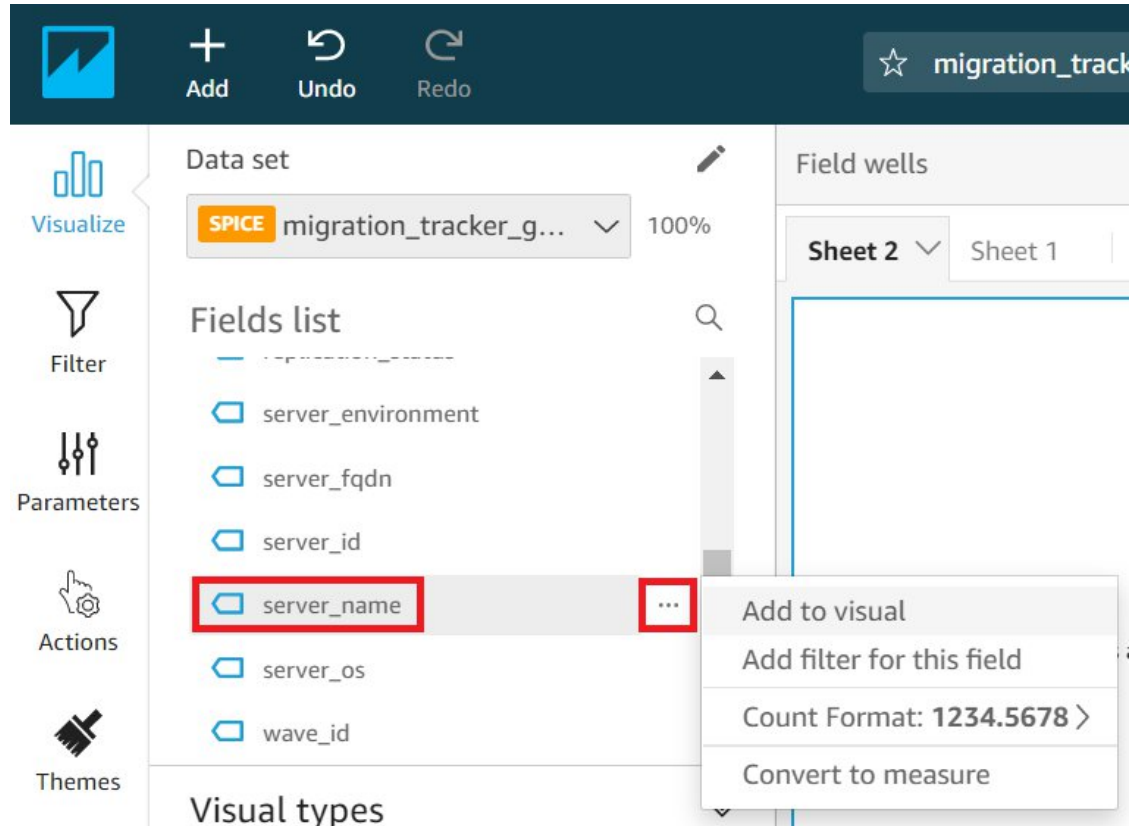
Amazon QuickSight offers the flexibility of building a custom dashboard that helps you to visualize your migration metadata. The following tutorial creates a dashboard containing a count visual that shows the server count by waves and bar charts showing the migration status, as shown in Figure 16. You can customize this dashboard to meet your business needs.



**Figure 20: Example QuickSight dashboard**

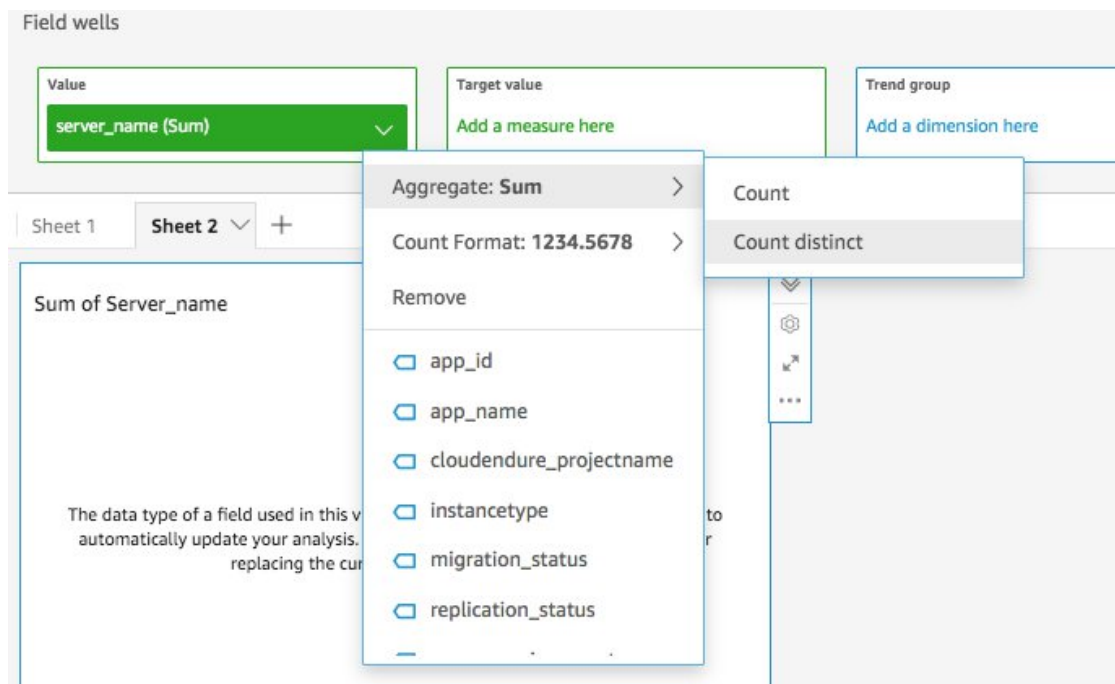
Use the following steps to create a count overview by migration waves. This view counts all the servers in the dataset that are grouped per wave, providing a granular view of the total number of servers in a wave. To create this view, you will convert the **server\_name** into a measure, which allows you to count distinct servers names. Then you will create a wave by wave filter.

1. From the QuickSight Visualize page, choose **+ Add** and select **Add Visual**.
2. From the left Dataset pane, open the drop-down list and select **server\_name**. Choosing this field ensures you get the unique entries.
3. Hover over the **server\_name** and choose the ellipsis to the right.



**Figure 21: QuickSight Visualize a data set page**

4. Select **Convert to measure** to convert the dataset from a dimension to a measure. The **server\_name** text turns green to indicate that the dataset has been converted to a measure.
5. Select **server\_name** to visualize the image. The visual will contain an error message indicating that the field data types must be updated.
6. On the **Field wells** page, Value box, choose the drop-down arrow for **server\_name (Sum)**, select **Aggregate: Sum**, then select **Count distinct**.



**Figure 22: Field wells page**

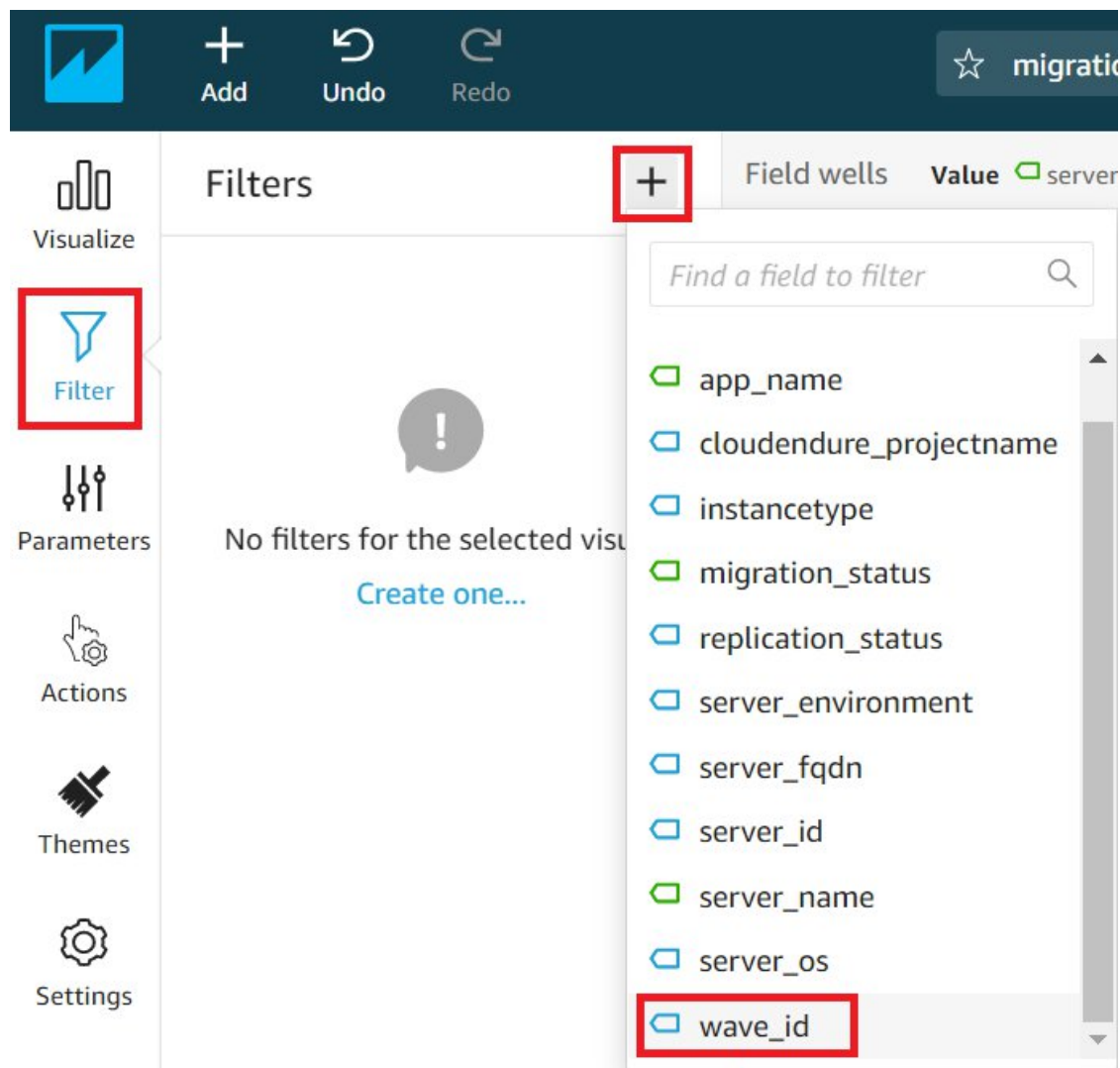
The visual should now show a count of the number of unique server names you have in your dataset. You can resize the visualization as needed to ensure it displays the information clearly on your monitor.

**Note**

You may need to convert your dataset back to dimension when you create another visual.

Next, add filters to the visualization to identify the server count for each migration wave. The following steps will apply a **wave\_id** filter to your visualization.

1. Verify that the visualization is selected and in the left menu pane, select **Filter**.
2. From the left Filters pane, choose the **+** button and then, in the drop-down list, select **wave\_id**.



**Figure 23: Filters pane drop-down list**

3. In the **Edit filter** pane, choose **wave\_id** and then under **Filter type**, select the checkbox next to the value **1**.
4. In the visualization, change the title to **Wave 1 Server Count**.

Repeat these steps for the other waves that are visualized in your dashboard.

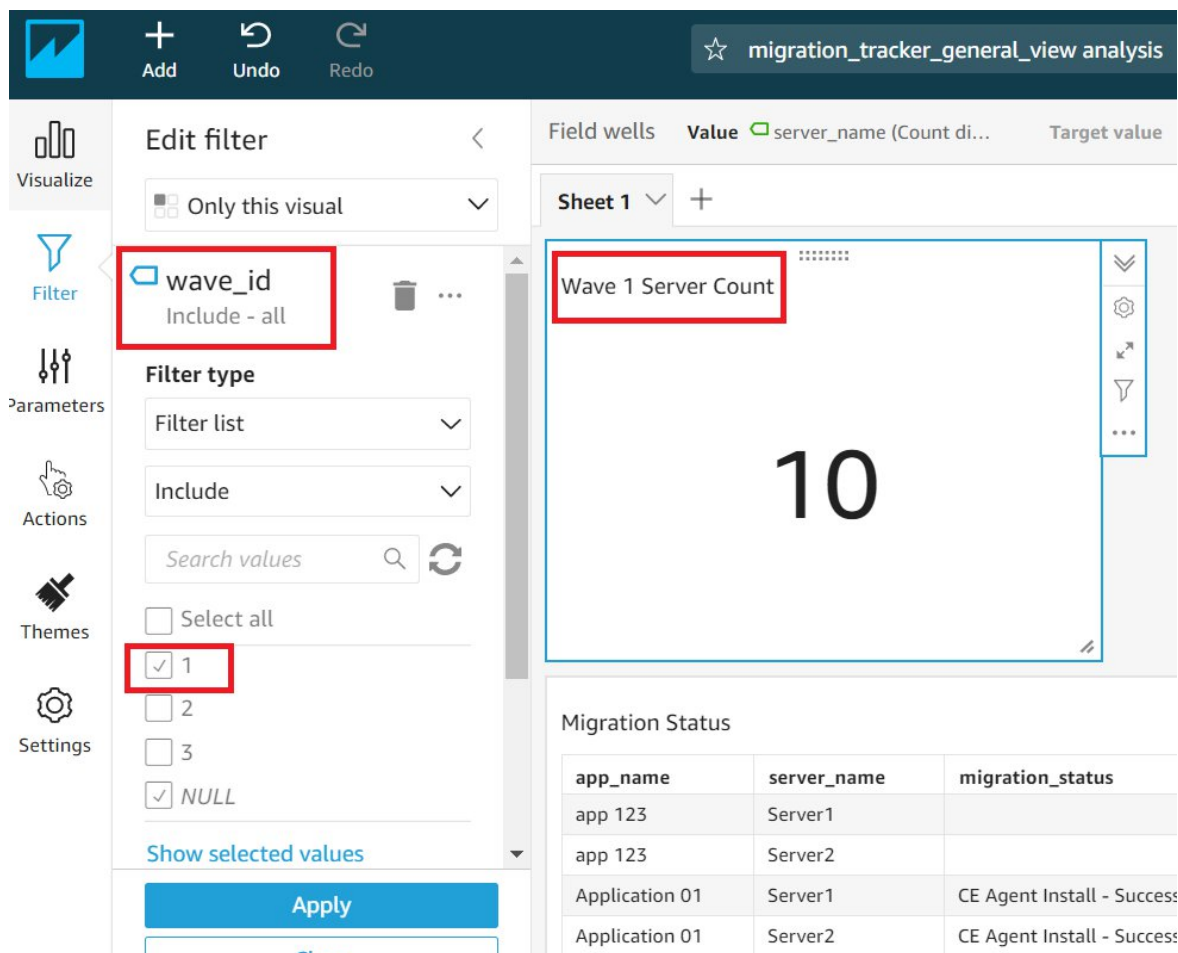


Figure 24: Edit filter pane

The next visualization we will add in the dashboard is a doughnut graph showing servers that are in progress of being migrated versus ones that have completed the migration. This chart uses Super-fast, Parallel, In-memory Calculation Engine (SPICE) Queries by creating a new column in the dataset that determines that an incomplete status will be identified as *in progress*. All values in the dataset that are not completed are combined and categorized as *in progress*.

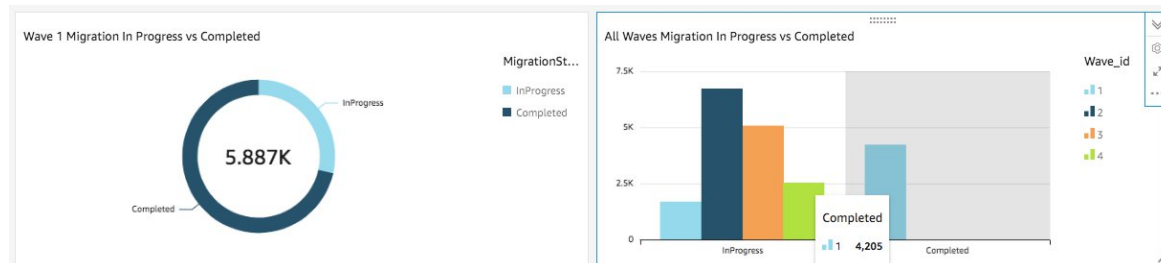


Figure 25: Doughnut graph and bar chart visualizing migration progress

#### Note

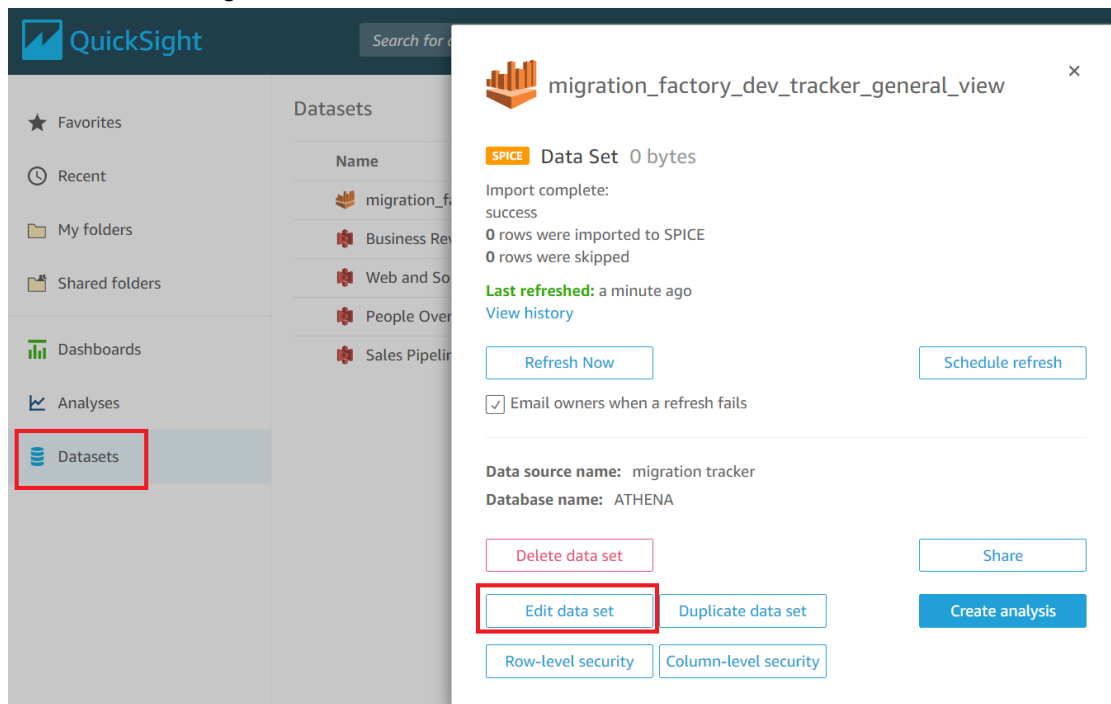
By default, when there is no custom query applied to the dataset, up to five migration/replication statuses can be shown. For this solution, a **MigrationStatusSummary** query is created in a new column: `ifelse(migration_status = 'Cutover instance launched', 'Completed', 'InProgress')`



This query combines the values of the statuses to create one column that is used for the visualization. For information about creating a query, refer to [Using the Query Editor](#) in the *Amazon QuickSight User Guide*.

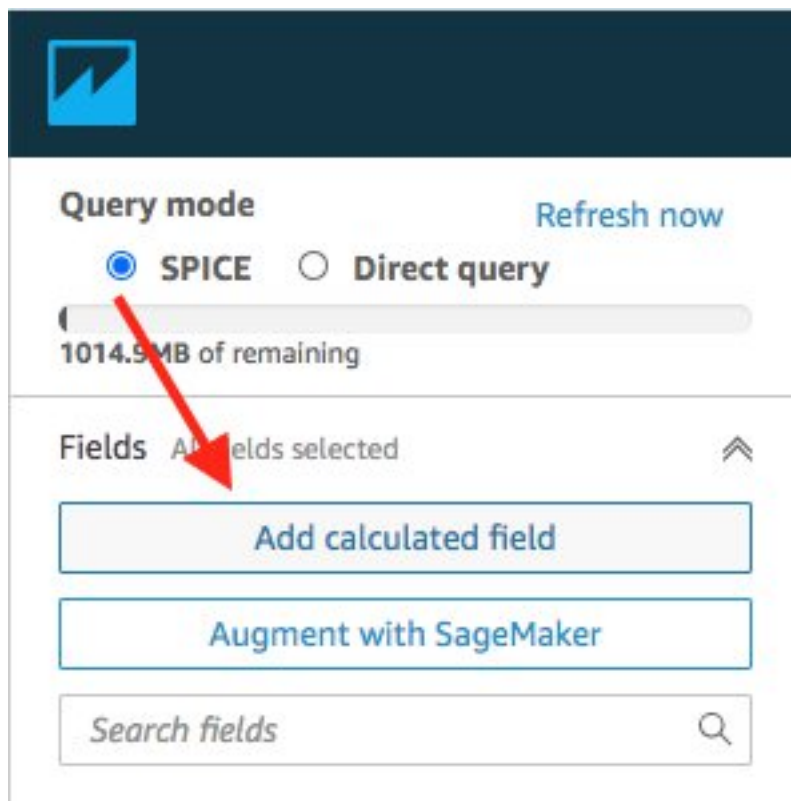
Use the following steps to create the **MigrationStatusSummary** column:

1. In the left menu pane, select **Datasets**.
2. On the **Datasets** page, select the `<migration-factory>-general-view` dataset.
3. In the dataset dialog box, select **Edit data set**.



**Figure 26: Migration factory dataset dialog box**

4. In the next dialog box, in the **Query mode** pane, choose **Add calculated field**.



**Figure 27: Query mode dialog box**

5. In the Add calculated field dialog box, enter a name for your SQL query, for example, **MigrationStatusSummary**.
6. Enter the following SQL query into the SQL editor:

```
ifelse(migration_status = 'Cutover instance launched', 'Completed', 'InProgress')
```

7. Choose **Save**.



**Figure 28: Add calculated field dialog box**

8. In the Data page, choose **Save & visualize**.

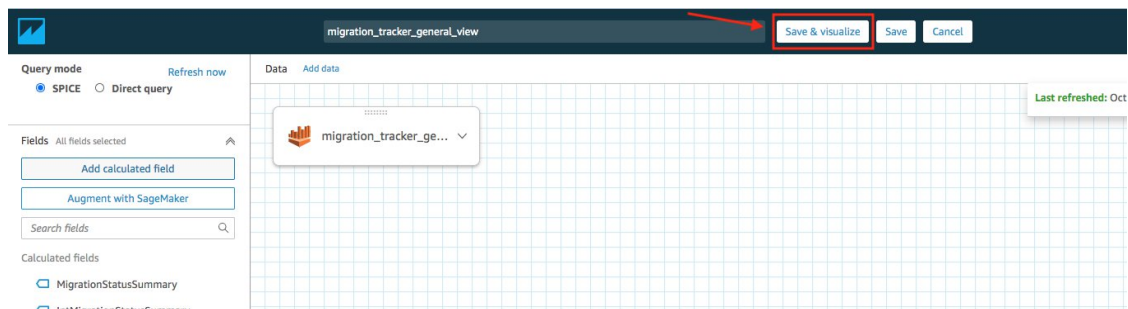


Figure 29: Data page

Your newly added query will be listed in the **Data set Fields list**.

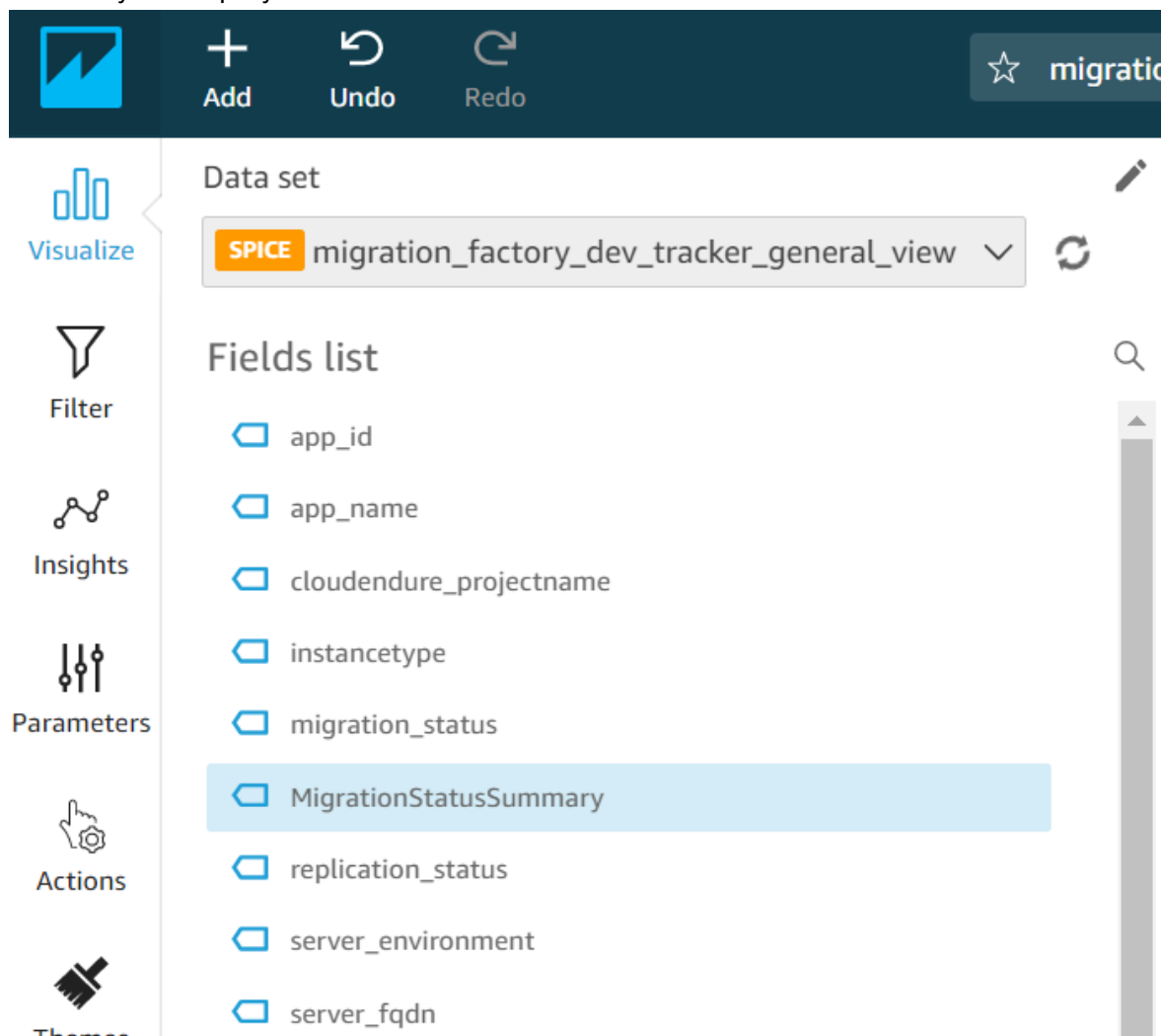


Figure 30: Data set Fields list

Next, you will build the dashboard.

1. On the Visualize page, choose **+ Add**.

2. In the Dataset pane, access the drop-down list and select **MigrationStatusSummary** as the main value (x-axis) and **wave\_id** as the grouping.

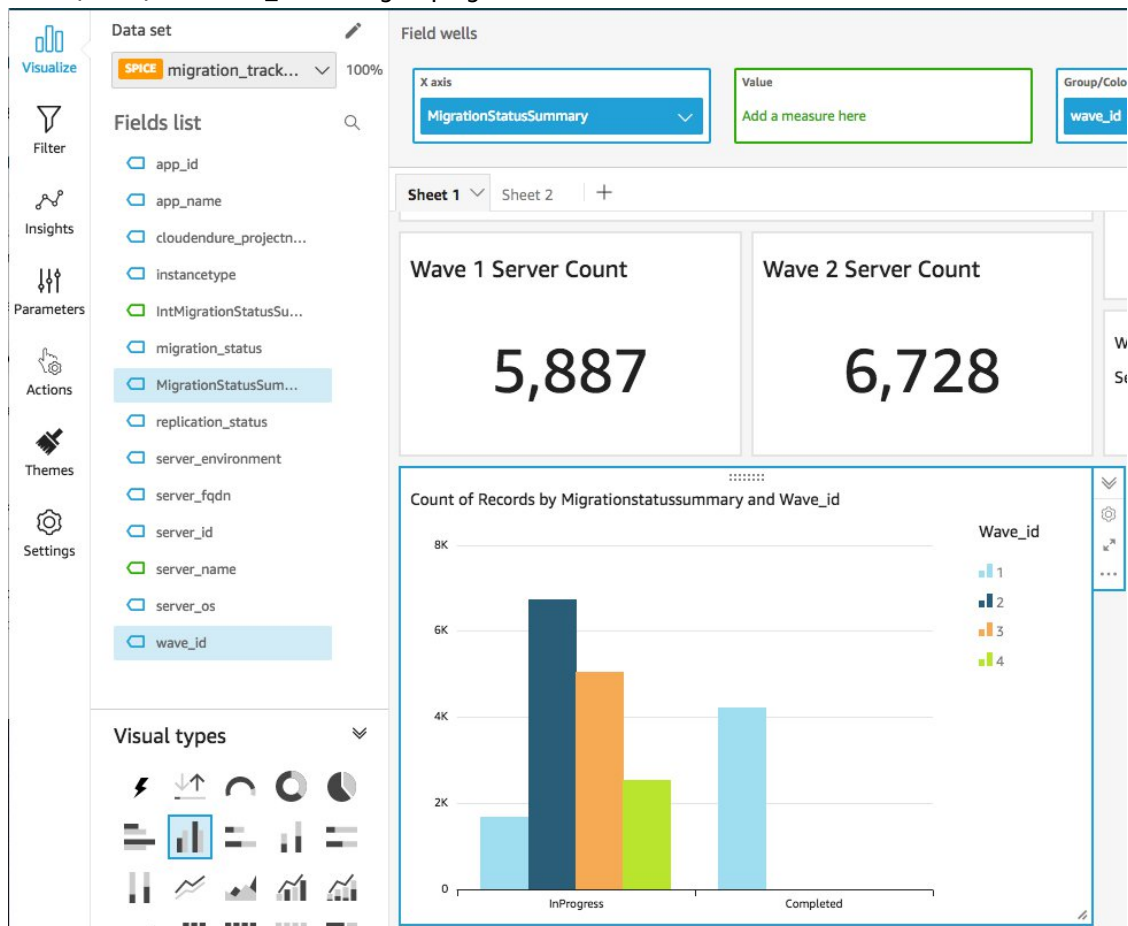


Figure 31: Dataset page

If you have an enterprise license for Amazon QuickSight, insights will be generated after the custom columns are created. You can customize your narratives for each insight. For example:

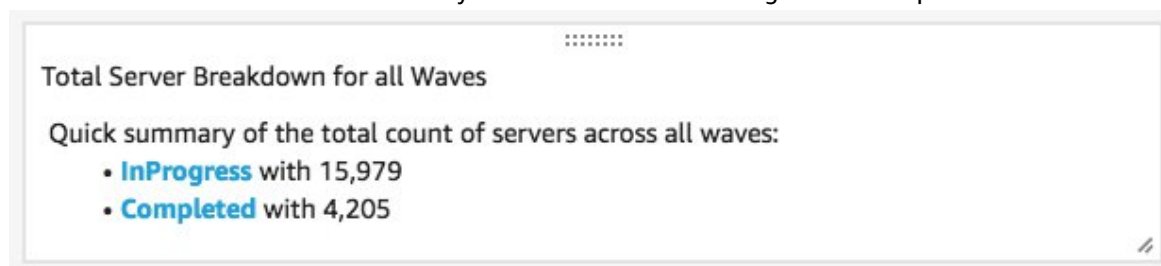


Figure 32: Example dashboard insights

You can also customize the data by breaking down the metadata into waves. For example:



Figure 33: Example wave 1 server breakdown

### (Optional) View Insights on the QuickSight dashboard

**Note**

You can use the following procedure if you have an enterprise license for Amazon QuickSight.

Use the following steps to add an insight to your dashboard which shows a breakdown of completed and in progress migrations.

1. On the left menu pane, choose **Insights**.

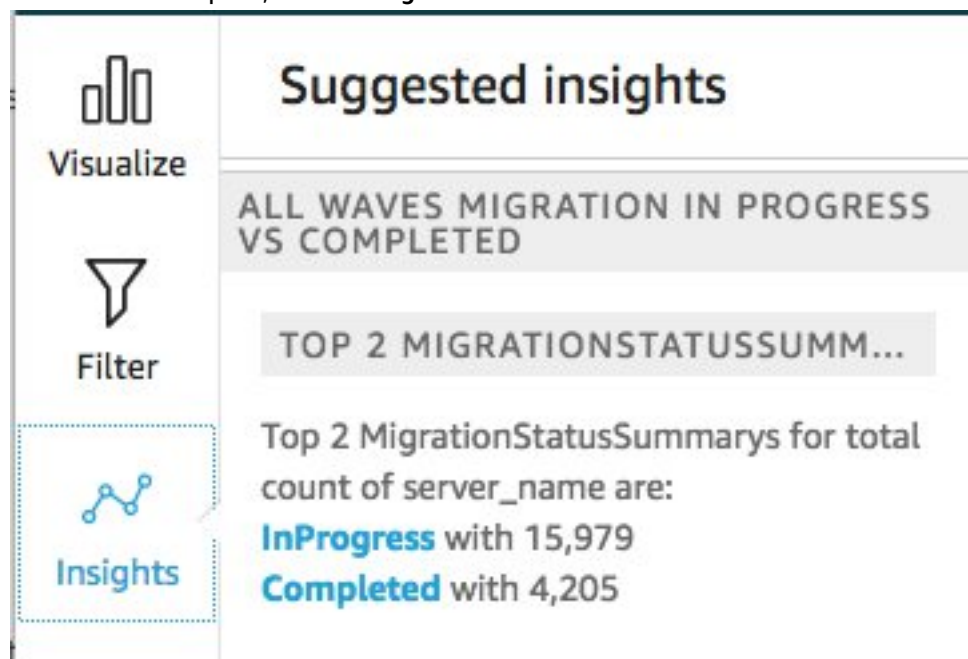
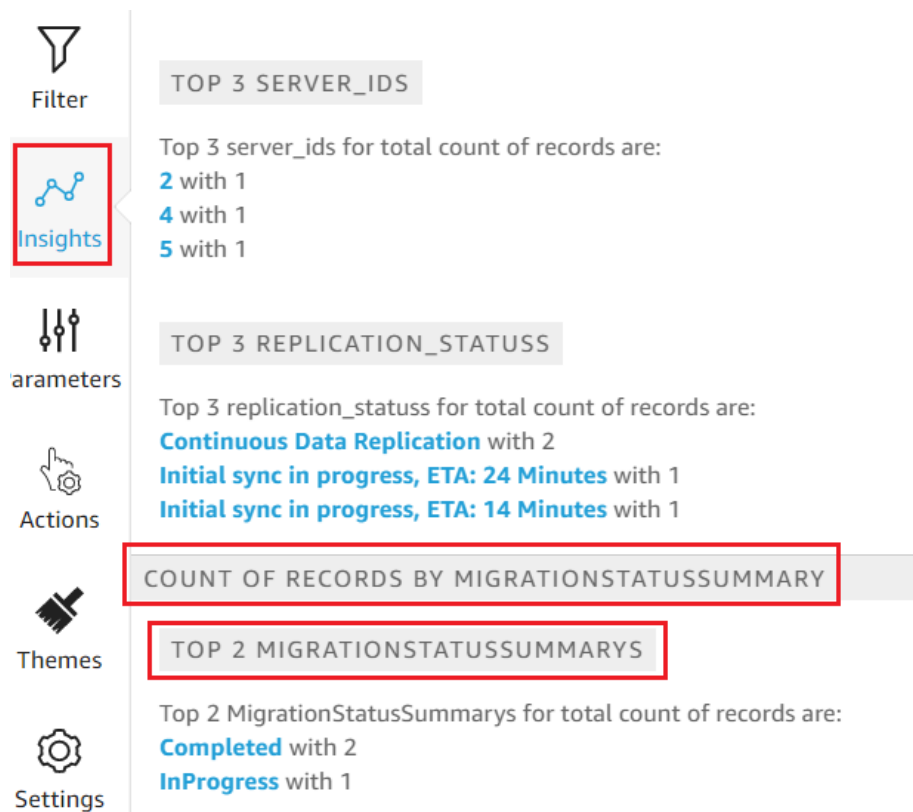


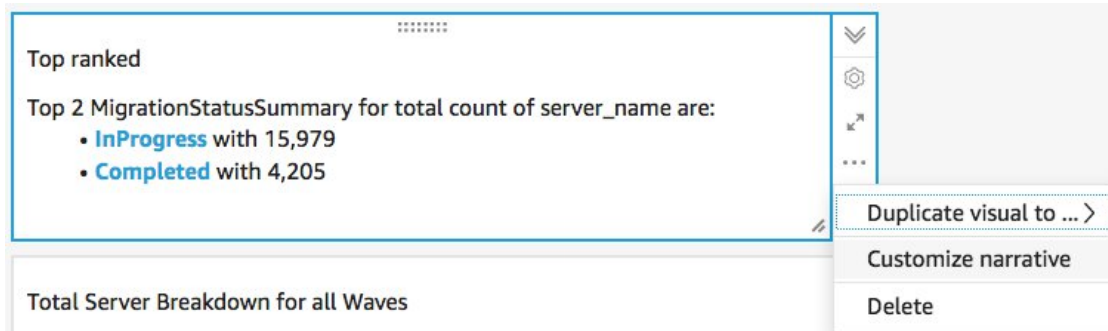
Figure 34: Insights available with an enterprise license

2. On the **Insights** page, in the **Count of Records BY MIGRATIONSTATUSSUMMARY** section, hover over **Top 2 MigrationSummarys** item and choose the **+** button to add an insight to the visual.

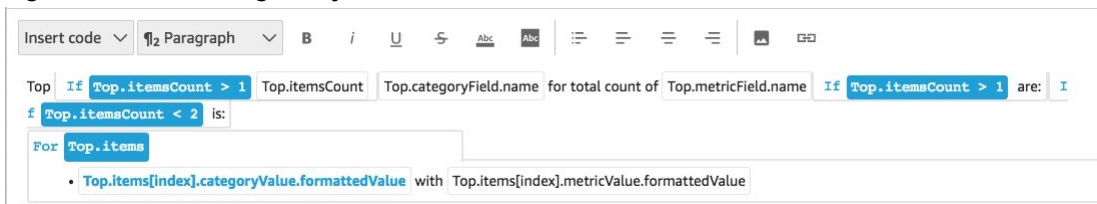


**Figure 35: Add an insight to a visual**

3. Customize the insight for your analysis by choosing **Customize Narrative** on the visual.

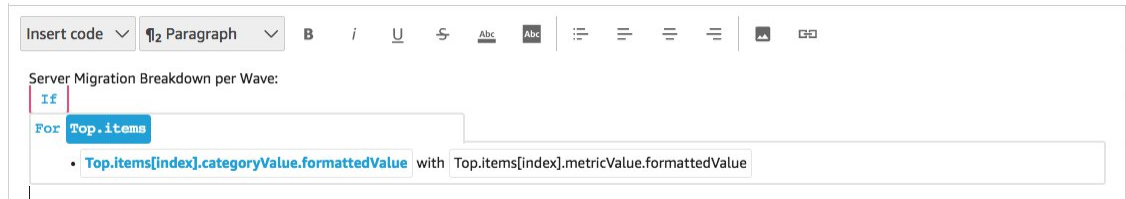


**Figure 36: Add an Insight to your dashboard**



**Figure 37: Customize narrative option**

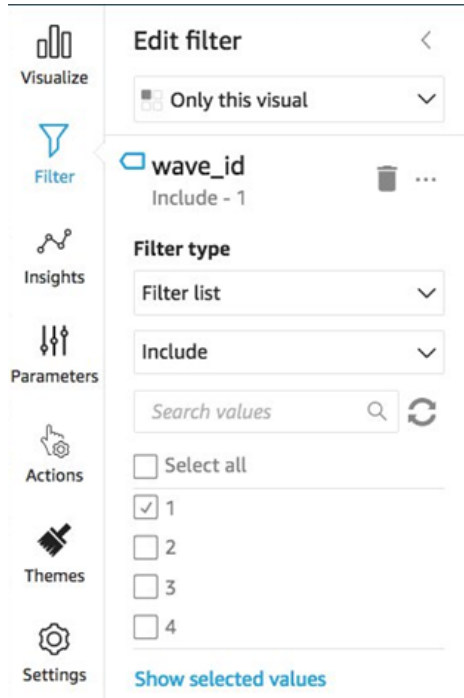
4. Edit the narrative to fit your use case and choose **Save**. For example:



**Figure 38: Edit your narrative**

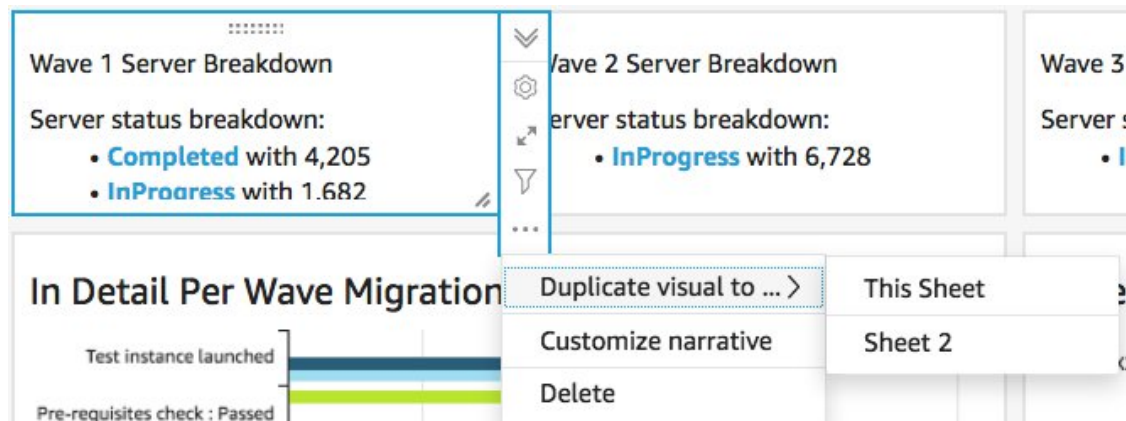
Return to the dashboard and filter it to show each wave:

5. On the left menu pane, choose **Filter**.
6. Choose the **+** button and select **wave\_id**.
7. Select a wave to visualize and choose **Apply**.



**Figure 39: Edit filter pane**

8. To visualize all the migration waves, duplicate the visuals by choosing the ellipsis to the left side of the visual and selecting **Duplicate visual to**.



**Figure 40: Visualize the migration waves**

9. Modify the filter for each visual to show a breakdown for each migration wave.

This insight is customized summarize the total count of servers across all waves. For more information and guide on how to customize insights, refer to [Working with Insights](#) in the *QuickSight User Guide*. You can access this QuickSight dashboard from any device and seamlessly embed it into your applications, portals, and websites. For more information about QuickSight dashboards, refer to [Working with Dashboards](#) in the *Amazon QuickSight User Guide*.



# Additional resources

## **AWS services**

- [AWS CloudFormation](#)
- [AWS Lambda](#)
- [Amazon API Gateway](#)
- [Amazon CloudFront](#)
- [Amazon Cognito](#)
- [Amazon DynamoDB](#)
- [Amazon Simple Storage Service](#)
- [AWS Systems Manager](#)

## **AWS resources**

- [CloudEndure Migration Factory guide](#)

# User guide

The following sections provide guidance on how to use the various features available in a deployed Cloud Migration Factory on AWS instance with a large-scale migration to AWS.

## Metadata management

The Cloud Migration Factory on AWS solution provides an extensible datastore that allows records to be added, edited, and deleted from within the user interface. All updates to the data stored in the datastore are audited with record level audit stamps, which provide creation and update timestamps along with user details. All update access to records is controlled by the groups and associated policies that the logged in user is assigned. For more details about granting user permissions, refer to [Permission management \(p. 68\)](#).

### Viewing data

Through the **Migration Management** navigation pane, you can select the record types (application, wave, database, server) held in the datastore. After you select a view, a table of the existing records for the chosen record type is shown. Each record type's table shows a default set of columns which can be changed by the user. Changes are persistent between sessions, and are stored in the browser and computer used for making the changes.

### Changing the default columns displayed in tables

To change the default columns, select the settings icon located on the upper-right corner of any data table, and then select the columns to display. From this screen, you can also change the default number of rows to be displayed and activate line wrapping for columns with large amounts of data.

### Viewing a record

To view a specific record in a table, you can click anywhere on the row, or select the check box next to the row. Selecting multiple rows will result in no record being displayed. This will then display the record in read-only mode under the data table at the bottom of the screen. The displayed record will have the following default tables available.

**Details** – This is a summary view of the required attributes and values for the record type.

**All Attributes** – This displays a complete list of all the attributes and their values.

Other tabs may be present depending on the record type selected that provide related data and information. For example, **Application** records will have a **Servers** tab showing a table of the servers related to the **Application** selected.

### Adding or editing a record

Operations are controlled by record type through user permissions. If a user does not have the required permission to add or edit a specific type of record then the **Add** and/or **Edit** buttons are greyed out and deactivated.

To add a new record:

1. Choose **Add** from the upper-right corner of the table for the record type you want to create.

By default, the **Add application** screen displays the **Details** and **Audit** sections, but depending on the type and any customizations to the schema, other sections might display too.

2. Once you have completed the form and resolved all errors, choose **Save**.

To edit an existing record:

1. Select a record from the table you want to edit, then choose **Edit**.
2. Edit the record and ensure no validation errors exist, then choose **Save**.

## Deleting a record

If a user does not have the permission to delete a specific type of record then the **Delete** button is grayed out and deactivated.

### Important

Records deleted from the datastore are not recoverable. We recommend making regular backups of the DynamoDB table, or exporting the data to ensure there is a recovery point in the event of an issue.

To delete one or more records:

1. Select one or more records from the table.
2. Choose **Delete** and confirm the action.

## Exporting data

The majority of data stored within the Cloud Migration Factory on AWS solution can be exported into Excel (.xlsx) files. You can export data at the record type level or a complete output of all data and types.

To export a specific record type:

1. Go to the table to export.
2. *Optional:* Select the record(s) to export to an excel sheet. If none are selected, then all records will be exported.
3. Choose the **Export** icon in the upper-right corner of the data table screen.

An excel file with the name of the record type (for example, servers.xlsx) will be downloaded to the browser's default download location.

To export all data:

1. Go to **Migration Management**, and select **Export**.
2. Check **Download All Data**.

An excel file with the name all-data.xlsx will be downloaded to the browser's default download location. This excel file contains a tab per record type, and all records for each type will be exported.

## Importing data

The Cloud Migration Factory on AWS solution provides a data import capability that can import simple record structures into the data store, for instance a list of servers. It can also import more complex relational data, for instance it could create a new application record and multiple servers contained in the same file and relate them to one another in a single import task. This allows a single import process to be used for any data type needed to be imported. The import process validates the data using the same validation rules used when the user edits data in the user interface.

## Downloading a template

To download template intake forms from the import screen, select the required template from the **Actions list**. The following two default templates are available.

**Template with only required attributes** – This contains only the attributes marked as required. It provides the minimum set of attributes required to import data for all record types.

**Template with all attributes** – This contains all the attributes in the schema. This template contains additional schema helper information for each attribute to identify the schema that it was found in. These helper prefixes to the column headers can be removed if required. If left in place during an import, the values within the column will only be loaded into the specific record type and not used for relational values. Refer to **Import header schema helpers** for more details.

## Importing a file

Import files can be created in either .xlsx or .csv format. For CSV, it must be saved using UTF8 encoding, otherwise the file will appear to be empty when viewing the pre-upload validation table.

To import a file:

1. Go to **Migration Management**, and select **Import**.
2. Choose **Select file**. By default, you can only select files with either the .csv or .xlsx extensions. If the file is successfully read, then the file name and size of the file will be displayed.
3. Choose **Next**.
4. The **Pre-upload validation** screen shows the outcome of the mapping of the headers within the file to attributes within the schema, and validation of the values provided.
  - a. The mappings of the file column headers are shown on the on-screen table column names. To check which file column header was mapped, select the expandable name in the header for more information about the mapping, including the original file header and the schema name it has been mapped to. You will see a warning in the **Validation** column for any unmapped file headers, or where there are duplicate names in multiple schemas.
  - b. All headers validate the values for each row of the file against the requirements for the mapped attribute. Any warnings or errors in the file content are displayed in the **Validation column**. Refer to Figure 39.
5. Once no validation errors are present, choose **Next**.
6. The **Upload data** step shows an overview of the changes that will be made once this file is uploaded. For any item where a change will be performed on upload, you can select **Details** under the specific update type to view the changes that will be performed.
7. Once the review has been completed, choose **Upload** to commit these changes to the live data.

A message appears at the top of the form if the upload is successful. Any errors that occur during the upload are displayed under **Upload Overview**.

## Import header schema helpers

By default, the column headers in the intake file should be set to the name of an attribute from any schema. The import process searches all schemas and attempts to match the header name with an attribute. If an attribute is found in multiple schemas you will see a warning, especially for relationship attributes which can be ignored in most cases. However, if the intention is to map a specific column to a specific schema attribute then you can override this behavior by prefixing the column header with a schema helper prefix. This prefix is in the format `[{schema name}]{attribute name}`, where `{schema name}` is the name of the schema based on its system name (wave, application, server, database) and the `{attribute name}` is the system name of the attribute in the schema. If this prefix is present then all values will only be populated into records for this specific schema, even if the attribute name is present in other schemas.

As shown in Figure 41, the header in column C has been prefixed with `[ database ]`, forcing the attribute to map to the `database_type` attribute in the database schema.

	B	C	D	E	F	G	H	I
1	database_name	[database]database_type	wave_name	aws_accountid	server_name	server_os_family	server_os_version	server_fqdn
2	importdb1	mssql	importwave1	123456789012	importserver1	linux	RH	importserver1

Figure 41 – Import header schema helper

# Credentials management

The Cloud Migration Factory on AWS solution features a Credentials Manager which integrates with AWS Secrets Manager within the account in which the instance is deployed. The feature allows administrators to save system credentials to AWS Secrets Manager for use in automation scripts without providing users access to retrieve the credentials directly, or needing to provide users' access to AWS Secrets Manager. Users can select stored credentials based on their name and description when providing them to an automation job. The automation job will then only retrieve the credentials requested when running on the automation server, and at this point the IAM Role allocated to the EC2 instance will be used to access the required secrets.

The Credentials Manager administration area is only visible to users that are members of the **admin** group within Amazon Cognito. Non-admin users will only be able to view credential names and descriptions when referenced through an automation, or other records relationship.

The following three secret types can be stored in AWS Secrets Manager via Credentials Manager.

**OS Credentials** – In the form of a username and password.

**Secret key/value** – In the form of a key and value.

**Plaintext** – In the form of a single plain text string.

## Add a secret

1. Choose **Add** from the **Credential Manager Secrets** list.
2. Select the **Secret Type** to add.
3. Enter a **Secret Name**. This will be the same name that will be displayed inside AWS Secrets Manager for the secret name.

4. Enter a **Secret Description**. This will be the same description that will be displayed inside AWS Secrets Manager for the secret description.
5. Enter the credential information for the secret type.

**Note**

For **OS Credentials** secret type, there is an option to select the **OS Type** which can be referenced in custom scripts.

## Edit a secret

Except the secret name and type, you can edit all properties of the secret using the Credentials Manager user interface.

## Delete a secret

From the Credentials Manager view, select the secret you want to delete and choose **Delete**. The secret will be scheduled for deletion within AWS Secrets Manager which may take a few minutes to complete. Any attempt to add a new secret with the same name during this time will fail.

# Run automation from console

The Cloud Migration Factory on AWS solution provides an automation engine allowing users to run jobs in the form of scripts against the inventory within the datastore. With this feature, you can manage, customize, and deploy all automations required to complete the end-to-end migration activities.

Jobs initiated from AWS CMF are run on automation servers that can be hosted in the AWS Cloud or on-premise. These servers need to run Windows with the AWS SSM agent installed, along with Python and Microsoft PowerShell. You can also install other frameworks as required for custom automations. Refer to [Step 5. Build a migration automation server \(p. 17\)](#) for details of the automation server build. At least one automation server is required to run jobs from the AWS CMF console.

On deployment, you can use scripts for the most common tasks needed to rehost workloads using AWS MGN. Download the scripts from the web interface and use it as a starting point for custom scripts. For details on creating custom automation script, refer to [Scripts management \(p. 59\)](#).

To initiate a job from the console, select a wave to run the automation against, then select **Actions**, and choose **Run Automation**. Or, you can select a job to run the automation against, then select **Actions**, and choose **Run Automation**.

From the **Run Automation**:

1. Enter the **Job Name**. This will be used to identify the job in the log.

**Note**

Job names do not have to be unique, as all jobs are also allocated a unique ID and timestamps to further identify them.

2. Select the **Script Name** from the list. This is a list of all scripts that have been loaded into the AWS CMF instance. When the job is submitted, the default version of the script selected will be run. To check the details of the script, including the current default version, choose **Related details** under the script name. Refer to **Change default version of script package** for details about updating the default version of scripts. When you select the script to run, the required parameters are shown under **Script Arguments**.

3. From the **Instance ID**, select the automation server for the job from the list.

**Note**

The list will only show instances that have the SSM agent installed and where either the EC2 instance, or for non-EC2 hosted automation servers, the Managed Instance tag of role is set to `mf_automation`.

4. In **Script Arguments**, enter the required input arguments for the script.
5. Once you have entered all the required parameters and verified them, choose **Submit Automation Job**.

When you submit the automation job, the following process is initiated:

1. A job record will be created with the AWS Cloud Migration Factory **Jobs** view containing the details of the job and current status.
2. An AWS Systems Manager automation job will be created, and will start to run the AWS Cloud Migration Factory SSM automation document against the automation server provided via the **Instance ID**. The automation document:
  - a. Downloads the current default version of the script package from the AWS Cloud Migration Factory S3 bucket to the automation server into the `C:\migration\scripts` directory.
  - b. Unzips and verifies the package.
  - c. Launches the master file python script specified in the `package-structure.yml` included in the zip.
3. Once the master file python script has been launched, any output from the script is captured by the SSM agent and fed into CloudWatch. It is then captured regularly and stored in the AWS Cloud Migration Factory datastore with the original job record, providing a complete audit of the job run.
  - a. If the script requires credentials to AWS Cloud Migration Factory then the script will contact AWS Secrets Manager to obtain the service account credentials. If the credentials are incorrect or not present then the script will return a failure.
  - b. If the script has a requirement to access other secrets stored using the AWS Cloud Migration Factory Credentials Manager feature, then it will contact AWS Secrets Manager to access those credentials. If this is not possible then the script will return a failure.
4. Once the master file python script exits, the outcome of this script will determine the status provided to the AWS Cloud Migration Factory job record. A non-zero return will set Job Status to Failed.

**Note**

Currently, if a failure occurs in the initial run of the AWS SSM document, it is not shown in the web interface. Failures are only logged once the master file python is launched. All jobs initiated from the console will timeout after 12 hours if they have not returned a success or failure status.

## Run automations from command prompt

Although we recommend running automation jobs through the web interface, you can run automation scripts manually from a command line on the automation server. This provides additional options where organizations cannot or do not want to use the combination of AWS CMF Credentials Manager, AWS Secrets Manager and AWS Systems Manager in the environment, or if Cloud Migration Factory on AWS users need to provide MFA one-time access codes to logon to Cloud Migration Factory on AWS.

When scripts are run from the command line, job history and logs are not available from within the **Jobs** view in the web interface. Log output will only be directed to the command-line output only. The scripts

can still access the Cloud Migration Factory on AWS APIs to read and update records, and other functions available through the APIs.

We recommend storing scripts in the scripts library or another central location to ensure you are accessing and using the latest version of the script, or the version that is currently approved for use.

## Manually running an automation package

This section describes the steps to download a package from Cloud Migration Factory on AWS and manually run it on the automation server. You can also follow the process for other script source locations by replacing steps 1 and 2 with the source specific download steps.

1. If scripts are stored in Cloud Migration Factory on AWS, follow the steps covered in [Download script packages \(p. 59\)](#) to obtain the automation package zip file.
2. Copy the zip file to a location on the automation server, such as `c:\migrations\scripts`, and unzip the contents.
3. Copy the `FactoryEndpoints.json` file to each of the unzipped script folder. Configure the file with the specific API endpoints for the Cloud Migration Factory instance that contains the servers, or other records this automation job will reference. Refer to [Creation of the FactoryEndpoints.json \(p. 52\)](#) for more information on how to create this file.
4. From the command line, ensure that you are within the root directory of the unzipped package, and run the following command:

```
python [package master script file] [script arguments]
```

*package master script file* – this can be obtained from the `Package-Structure.yml` under the `MasterFileName` key.

*script arguments* – information about the arguments is provided in the `Package-Structure.yml` under the `Arguments` key.

5. The scripts will request credentials required for Cloud Migration Factory on AWS APIs and the remote server. Any credentials manually entered are cached in memory for the duration of this process to avoid entering the same credentials again. If you enter script arguments to access secrets stored using the Credentials Manager feature, then access to AWS Secrets Manager and associated secrets is required. If secret retrieval fails for any reason, the script will prompt for user credentials.

## Creation of the FactoryEndpoints.json

It is recommended that this file is created once for a deployment of the Cloud Migration Factory on AWS solution, as the content do not change after initial deployment, and stored in a central location on the automation server. This file provides the automation scripts with the AWS Cloud Migration Factory API endpoints and other key parameters. An example of the default contents of the file is shown here:

```
{
  "UserApiUrl":
    "https://cmfuserapi.execute-api.us-east-1.amazonaws.com",
  "Region": "us-east-1",
  "UserPoolId": "us-east-1_AbCdEfG",
  "LoginApiUrl":
    "https://cmfloginapi.execute-api.us-east-1.amazonaws.com"
}
```

The information required to compose this file for a deployed AWS Cloud Migration Factory instance is available from the AWS CloudFormation **Outputs** tab of the deployed stack.



```
{  
  "UserApiUrl": <UserApi-value>,  
  "Region": <Region-value>,  
  "UserPoolId": <UserPoolId-value>,  
  "LoginApiUrl": <LoginApi-value>  
}
```

Replace `<LoginApi-value>`, `<UserApi-value>`, `<Region-value>`, and `<UserPoolId-value>` with the corresponding values you retrieved from the AWS CloudFormation Outputs console. Do not add a forward slash (/) to the end of the URLs.

The file has an optional `DefaultUser` key. You can set the value for this key to the default user ID to be used to access the Cloud Migration Factory on AWS instance to avoid having to enter it every time. When prompted for the Cloud Migration Factory user ID, you can either enter a user ID or use the default value by pressing the enter key. You can only do this when the scripts are run manually.

## Launch AWS MGN jobs from Cloud Migration Factory

The Cloud Migration Factory on AWS solution has built in automation to initiate and manage Rehost migration using AWS MGN. These automations allow migration teams to manage all aspects of their migration from a single user interface, combining the key actions available within the AWS MGN service console, with the AWS Cloud Migration Factory automation library that extends the functionality with prebuilt scripts for mass migrations, which helps to increase the speed of the migration activities. Refer to [List of automated migration activities for AWS Application Migration Service \(AWS MGN\) \(p. 71\)](#) for a full list of AWS MGN automation jobs available. Using AWS Cloud Migration Factory also provides seamless multi account migrations using AWS MGN as the Cloud Migration Factory has the ability to assume roles in different target accounts automatically based on the Cloud Migration Factory application and server definitions being migrated.

### Prerequisite activities

1. Target account AWS CMF CloudFormation deployed into each target account. For more information, review the [AWS CloudFormation templates \(p. 10\)](#) section in this document.
2. [AWS MGN is initialized in each target account.](#)

### Initial definition

Definition of the on-premises inventory is performed through the creation of wave, application and server items using either the user interface, or through the import of a CSV intake form. These definitions are used to provide the on-premise server identities and also the target EC2 parameters and well as other required data to manage the migration activity.

### User interface definition

In order to use the AWS MGN functionality, you need to create a wave record, with associated application records, and finally one or more server records associated to the applications. The wave record is used to group the applications, and does not provide parameters to the automation, whereas the application record defines the target AWS account ID and AWS Region that the application will be migrated to. The server records provide the automations actions and AWS MGN integration the target parameters for the EC2 instances, such as instance type, subnets, security groups, etc.

When defining a server in the AWS CMF datastore for use with the AWS MGN functionality the server needs to be configured with a **Migration Strategy of Rehost**. Once Rehost is selected then the additional attributes required for this functionality will be shown on the screen. The following attributes are required to be populated in order to successfully initiate an AWS MGN migration job:

## Required

**Server OS Family** – Set to either linux or windows depending on the OS family.

**Server OS Version** – Set to the detailed OS version running on the server.

**Instance Type** – EC2 instance type to be used.

**Tenancy** – Shared hosting, dedicated host.

**Security Group Ids** – List of security groups that will be assigned to the instance when the final cutover is initiated.

**Security Group Ids - Test** – List of security groups that will be assigned to the instance when the testing is initiated.

## Conditional

**Subnet Ids** – Subnet ID to assign this EC2 instance to when the final cutover is initiated. (not applicable when **Network Interface ID** specified)

**Subnet Ids - Test** – Subnet ID to assign this EC2 instance to when testing is initiated. (not applicable when **Network Interface ID -Test** specified)

**Network Interface ID** – ENI ID to be used when the final cutover is initiated.

**Network Interface ID - Test** - ENI ID to be used when testing is initiated.

**Dedicated Host ID** – Dedicated host ID that the instance will be launched on. (only applicable when **Tenancy** is set to **Dedicated host**).

## Optional

**Tags** – EC2 Instance tags to be applied to instance.

All other attributes not listed here do not have any bearing on the AWS MGN Jobs initiated from within the AWS CMF solution.

## Intake form definition

Intake forms can contain the details to create or update multiple types of record with the datastore in a single row of the csv file, this enables the import of related data. In the example below, the wave, application and server records will be created and related to each other automatically during the import.

To import the intake form, follow the same process as other data imports into the Cloud Migration Factory on AWS solution covered in [Importing data \(p. 48\)](#).

## Initiating a job

Initiating an AWS MGN job from AWS CMF is performed against a wave, from the wave list view select the wave, and then from the **Actions** select **Rehost>MGN**.

This screen requires the user to make the following choices before they can **Submit** the job.

1. Select the AWS MGN **Action** to perform against the applications and servers in the wave. These actions mostly replicate those available in the AWS MGN service console and API, with the exception of **Validate Launch Template** (see below for details on this action). For details of the effects of each action, refer to AWS MGN user guide.
2. Select the **Wave** to run the action against.
3. Select the **Applications** from the wave that the action will be run against. This list will only show applications that are associated to the Wave selected.
4. Once all options are correct, choose **Submit**.

The automation will now initiate the selected action against each selected application's target AWS account, as specified in the Application record. The results of the action will be displayed in the notification message, including any errors.

## Validate launch template

This action is used to validate the configuration data stored in CMF for each server is valid before attempting cutover activities. In order to run this action, you must have successfully deployed the AWS MGN agents onto the source server.

The validations performed for each server are:

- Verify instance type is valid.
- Verify IAM instance profile exists.
- Security groups exist for both test and live.
- Subnets exist for both test and live (if ENI not specified).
- Dedicated host exists (if specified).
  - If dedicated host is specified then the following checks are made:
    - Does the dedicated host support the instance type specified?
    - Does the dedicated host have free capacity for all the requirements of this wave, based on instance types required?
- ENI exists (if specified).

The results of the action will be displayed in the notification message, including any errors.

## Replatform to EC2

The Cloud Migration Factory on AWS solution allows groups of EC2 instances to be launched automatically from configurations defined in its datastore; deploying EC2 instances with EBS volumes attached. This provides the ability to provision new EC2 instances, allowing Replatform through AWS CloudFormation, and Rehost of on-premise servers with AWS MGN within a single CMF user interface. Before you can use this functionality the datastore must contain the definition of the servers. Once this has been addressed the servers should be linked to a wave. When the decision is taken to launch the EC2 instances, the user can initiate the following actions against the wave:

- EC2 Input Validation
- EC2 Generate CF Template

- EC2 Deployment

## Prerequisites

Permissions to add the Replatform attribute access.

## Initial configuration

Configuration of the new EC2 instances is performed through the creation of new server items using either the user interface or through the import of a CSV intake form containing the server items. These definitions are converted to AWS Cloud Formation templates stored in an S3 bucket within the same AWS account that the AWS CMF instance is deployed.

## User interface definition

When defining a server in the AWS Cloud Migration Factory datastore for use with the Replatform to EC2 functionality the server needs to be configured with a **Migration Strategy of Replatform**. Once **Replatform** is selected then the additional attributes required for this functionality will be shown on the screen. The following attributes are required to be populated for the functionality to work:

### Required attributes

**AMI Id** – ID of the Amazon Machine Image used to launch the EC2 instance.

**Availability Zone** – AZ that the EC2 instance will be deployed to.

**Root Volume Size** – Size in GB of the root volume for the instance.

**Instance Type** – EC2 instance type to be used.

**Security Group Ids** – List of security groups assigned to the instance.

**Subnet Ids** – Subnet ID to assign this EC2 instance to.

**Tenancy** – Currently the only supported option for the Replatform to EC2 integration is **Shared** any other option will be replaced with **Shared** when the template is generated.

### Optional Attributes

**Enable Detailed Monitoring** – Check to enable detailed monitoring.

**Additional Volume Names** – List of Additional EBS volume names. Each item in the list needs to map to the same line as the **Size** and **Type** lists.

**Additional Volume Sizes** – List of additional EBS volume sizes. Each item in the list needs to map to the same line as the **Names** and **Type** lists.

**Additional Volume Types** – List of additional EBS volume types. Each item in the list must map to the same line as the **Names** and **Size** lists, if not specified then defaults to **gp2** for all volumes.

**EBS KMS Key Id for Volume Encryption** – If EBS volumes will be encrypted then specify the **Key ID**, **Key ARN**, **Key Alias**, or **Alias ARN**.

**Enable EBS Optimized** – Select to turn on **EBS Optimized**.

**Root Volume Name** – Select from the options provided, if not specified then the ID will be used.

**Root Volume Type** – Provide the EBS type of the volume to be created, if not specified then defaults to gp2.

## Intake form definition

Intake forms can contain the details to create or update multiple types of record with the datastore in a single row of the csv file, this enables the import of related data. In the following example, the wave, application and server records will be created and related to each other automatically during the import.

Example: Replatform intake form

Column name	Example data	Required	Notes
wave_name	wave1	Yes	
app_name	app1	Yes	
aws_accountid	1234567890	Yes	
server_name	Server1	Yes	
server_fqdn	Server1	Yes	
server_os_family	linux	Yes	
server_os_version	Amazon	Yes	
server_tier	Web	No	
server_environment	Dev	No	
subnet_IDs	subnet-xxxxxxx	Yes	
securitygroup_IDs	sg-yyyyyyyyyy	Yes	
instanceType	m5.large	Yes	
iamRole	ec2customrole	Yes	
tenancy	Shared	Yes	
r_type	Replatform	Yes	
root_vol_size	50	Yes	
ami_id	ami-zzzzzzzzz	Yes	
availabilityzone	us-west-2a	Yes	
root_vol_type	gp2	No	
add_vols_size	40:100	No	
add_vols_type	gp2:gp3	No	
ebs_optimized	false	No	
ebs_kmskey_id	1111-1111-1111-1111	No	
detailed_monitoring	true	No	
root_vol_name	Server1_root_volume	No	

Column name	Example data	Required	Notes
add_vols_name	Server1_root_volumeA: Server1_root_volumeB	No	

To import the intake form, follow the same process as any other data imports into the Cloud Migration Factory on AWS solution.

## Deployment actions

### EC2 input validation

The first action that should be completed after definition of the instance parameters, is to run the wave action: **Replatform>EC2>EC2 Input Validation**. This action verifies that all the correct parameters have been provided for each server in order to create a valid CloudFormation template.

#### Note

This validation does not currently verify that the input parameters are valid, only that they are present in each server definition. Verification of the correct values should be performed before creation of the template otherwise the deployment of the template will fail.

### EC2 generate CloudFormation template

Once the definitions for all servers included in a wave have been verified the CloudFormation template can be generated. To do this, run the wave action: **Replatform>EC2>EC2 Generate CF Template**. This action creates a Cloud Formation template for each application in the wave, where the servers in the application have a **Migration Strategy of Replatform**; any servers with other migration strategies defined will not be included in the template.

Once run, the templates for each application will be stored in the S3 bucket: [instance specific prefix]-**gfbuild-cftemplates**, which was automatically created when the Cloud Migration Factory on AWS solution was deployed. The folder structure of this bucket is as follows:

- [Target AWS Account ID]
  - [Wave Name]
    - **CFN\_Template\_[Application ID]\_[Application Name].yaml**

Each time the generate action is run a new version of the template is stored in the S3 bucket. The S3 URIs for the templates will be provided in the notification, these templates can be reviewed or edited as required before deployment.

The CloudFormation templates generate the following Cloud Formation resource types currently:

- AWS::EC2::Instance
- AWS::EC2::Volume
- AWS::EC2::VolumeAttachment

### EC2 deployment

Once the time is right to deploy the new EC2 instances, the **EC2 Deployment** action can be initiated through the wave action **Replatform>EC2>EC2 Deployment**. This action will use the latest version of the Cloud Formation template for each application in the wave, and deploy these templates into the target accounts selected, through AWS CloudFormation. For more details on these actions, refer to [Deployment actions](#) (p. 58).

## Scripts management

The Cloud Migration Factory on AWS solution allows users to fully manage the library of automation scripts or packages within the user interface. You can upload new custom scripts as well as new versions of the script using the scripts management interface. When multiple versions are available, an administrator can switch between these versions allowing the ability to test updates before making them default. The script management interface also allows administrators to download script packages to update or review the content.

A supported script package is a compressed zip archive containing the following mandatory files in the root:

- **Package-Structure.yml** – Used to define the script's arguments and other metadata, such as description and default name. Refer to [Composing a new script package \(p. 60\)](#) for more details.
- **[custom python script].py** – This is the initial script that will be run when a job is submitted. This script can call other scripts and modules and if so these should be included in the archive. The name of this script must match the value specified in the `MasterFileName` key in the `Package-Structure.yml`.

### Upload new script package

#### Note

A script package must conform to the supported format. Refer to [Composing a new script package \(p. 60\)](#) for more details.

1. Choose **Add** on the **Automation Scripts** table.
2. Select the package archive file you want to upload.
3. Enter a unique name for the script. Users will reference the script by this name for initiating jobs.

### Download script packages

You can download script packages from the console to activate updates and content verification.

1. Select **Automation**, then **Scripts**.
2. Select the script you want to download from the table, then select **Actions** and choose **Download default version** or **Download latest version**.

You can download specific versions of a script. To do so, select the script, then **Actions** and choose **Change default version**. From the **Script Default Version** list, choose **Download selected version**.

### Add new version of a script package

Updates to AWS Cloud Migration Factory script packages can be uploaded in the **Automation > Scripts** section by following these steps:

1. Select **Automation**, then **Scripts**.
2. Select the existing script to add a new version, then select **Actions** and choose **Add new version**.
3. Select the updated package archive file you want to upload, and choose **Next**. The new script version will keep the existing name by default. Enter a unique script name. Any name change will only be applied to this version of the script.
4. You can make the new version of the script the default version by selecting **Make default version**.

5. Choose **Upload**.

## Deleting script packages and versions

You cannot delete scripts or versions of a script for auditing purposes. This allows reviewing the exact script that was run against a system at a point in time. Every script version has a unique signature and ID when uploaded which is recorded against the job history that the script and version was used in.

## Composing a new script package

Cloud Migration Factory on AWS script packages support Python as the primary scripting language. You can initiate other shell scripting languages as required from within a Python main program or wrapper. To create a new script package quickly, we recommend downloading a copy of one of the prepackaged scripts and updating it to perform the required task. You must first create a master Python script that will perform the core functionality of the script. Then, create a `Package-Structure.yml` file to define the arguments and other metadata the script requires. Refer to `Package-Structure.yml` options for more details.

### Main Python script

This is the initial main script that runs when a job is initiated. Once the script completes running, the task is finished and the final return code determines the status of the job. All output from this script is captured when run remotely and passed into the output audit log of the job for reference. This log is also stored in Amazon CloudWatch.

### Accessing Cloud Migration Factory on AWS data and APIs from a script

To provide access to the Cloud Migration Factory on AWS APIs and data, you can use the included python helper module. The module provides the following common functions:

#### Factorylogin

Returns an access token that can be used to call Cloud Migration Factory on AWS APIs. This function will attempt login to CMF using a number of attempts for credentials:

1. By attempting to access the default secret containing the service account userid and password if it exists and access is allowed. This secret name **MFServiceAccount-[userpool id]** will be checked.
2. If, 1 is unsuccessful, and the user is running the script from the command line, then the user will be prompted to supply an AWS Cloud Migration factory userid and password. If run from a remote automation job, the job will fail.

#### getServerCredentials

Returns the login credentials for a server stored in AWS Cloud Migration Factory in either Credentials Manager, or through user input. This function will check a number of different sources to determine the credentials for a specific server, the order of the sources is:

1. If `local_username` and `local_password` is set and valid then these will be returned.
2. If `secret_override` is set then this will be used to retrieve the secret specified from AWS Secret Manager, otherwise, checks if the server record contains the key **secret\_name** and this is not empty then this secret name will be used.
3. If there is a failure locating or accessing the secrets specified then the function will fall back to prompting the user for the credentials, but only if the **no\_user\_prompts** is set to **False**, otherwise it will return a failure.



### Parameters

local\_username – If passed then will be returned.

local\_password – If passed then will be returned.

server – Server name, as stored in the server\_name attribute in AWS Cloud Migration Factory.

Secret\_override – Is passed this will set the secret name to retrieve from Secrets Manager for this server.

No\_user\_prompts – Tells the function not to prompt a user for a userid and password if not stored, this should be True for any remote automation script.

getCredentials

Gets the credentials stored using AWS Cloud Migration Factory Credentials Manager from Secrets Manager.

### Parameters

secret\_name – name of the secret to retrieve.

get\_factory\_servers

Returns an array of servers from the AWS Cloud Migration Factory datastore based on the waveid provided.

### Parameters

Waveid – Wave record ID of the servers that will be returned.

Token – Authentication token obtained from the FactoryLogin Lambda function.

UserHOST – AWS Cloud Migration Factory User API Endpoint URL.

osSplit – If set to true, then two lists will be returned one for Linux and one windows servers, if False, then a single combined list will be returned.

## Final message summary

It is recommended to provide a summary message of the script's outcome as the final output to the screen or sysout. This will be shown on the console in the **Last Message** property (see Figure 46 - Example script output), which provides a quick status of the script outcome without the user having to read the full output log.

## Return code

The main python script should return a non-zero return code on exit if the function of the script was not completely successful. On receiving a non-zero return code the job status will be shown as **Failed** in the jobs log indicating to the user that they should review the output log for details of the failure.

## Package-Structure.yml options

### Example

```
Name: "0-Check MGN Prerequisites"
Description: "This script will verify the source servers meet the basic requirements for
  AWS MGN agent installation."
```

```
MasterFileName: "0-Prerequisites-checks.py"
UpdateUrl: ""
Arguments:
-
  name: "ReplicationServerIP"
  description: "Replication Server IP."
  long_desc: "IP Address of an AWS MGN Replication EC2 Instance."
  type: "standard"
  required: true
-
  name: "SecretWindows"
  long_desc: "Windows Secret to use for credentials."
  description: "Windows Secret"
  type: "relationship"
  rel_display_attribute: "Name"
  rel_entity: "secret"
  rel_key: "Name"
-
  name: "SecretLinux"
  long_desc: "Linux Secret to use for credentials."
  description: "Linux Secret"
  type: "relationship"
  rel_display_attribute: "Name"
  rel_entity: "secret"
  rel_key: "Name"
-
  name: "Waveid"
  description: "Wave Name"
  type: "relationship"
  rel_display_attribute: "wave_name"
  rel_entity: "wave"
  rel_key: "wave_id"
  validation_regex: "^(?!\\s*$).+"
  validation_regex_msg: "Wave must be provided."
  required: true
```

## Keys

Required

**Name** – Default name that the script will use on import.

**Description** – Description of the usage of the script.

**MasterFileName** – This is the starting point for the script to run, it has to be a python file name that is included within the script package archive.

**Arguments** – A list of arguments that the MasterFileName Python script accepts. Each argument needs specified is in the AWS Cloud Migration Factory Attribute definition format. Required properties for each argument are **Name** and **Type**, all other properties are optional.

Optional

**UpdateUrl** – Provide a URL where the script package's source is available for providing updates. Currently this is for reference only.

# Schema management

The Cloud Migration Factory on AWS solution provides a fully extensible metadata repository, allowing data for automation, auditing and status tracking to be stored in a single tool. The repository provides

a default set of entities (Waves, Applications, Servers, and Databases) and attributes at deployment time to get you started with capturing and using the most frequently used data, and from here you can customize the schema as required.

Only Cognito admin group users have permissions to manage the schema. To make a user a member of the admin or other groups, refer to [User management \(p. 16\)](#).

Go to **Administration**, and select **Attributes** for the default entity tabs. The following tabs are available to support management of the entity.

**Attributes** – Allows for adding, editing and deleting attributes.

**Info Panel** – Allows for editing of the Info panel help content, this is shown on the right of the entities' screen in the Migration Management section.

**Schema Settings** – Currently this tab only provides the ability to change the friendly name of the entity, this is the name that is shown on the user interface. If not defined, then the user interface uses the programmatic name of the entity.

## Adding/editing an attribute

Attributes can be amended dynamically through the **Attributes** administration section of the Migration Factory solution. When attributes are added, edited or deleted the updates will be applied in real-time for the administrator making the change. Any other user that is currently logged into the same instance will have their session updated automatically within a minute of the changes being saved by the administrator.

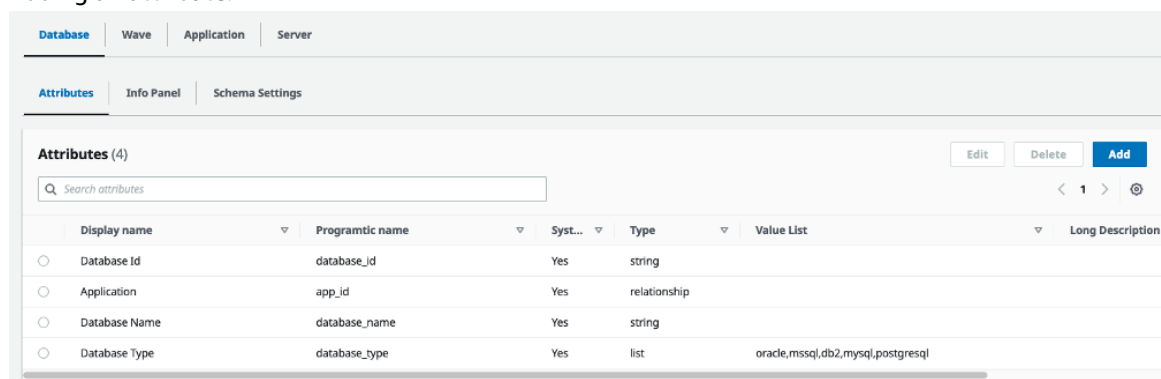
Some attributes are defined as system attributes, which means that the attribute is key to the core functionality of Cloud Migration Factory and so only some properties are available for administrators to amend. Any attribute that is a system attribute will be shown with a warning on the top of the **Amend attribute** screen.

For system defined attributes only the following can be edited:

- Info Panel
- Advanced Options
  - Attribute Grouping & Positioning
  - Input Validation

All other properties of the system defined attribute are read-only.

Adding an attribute:



Display name	Programtic name	Syst...	Type	Value List	Long Description
Database Id	database_id	Yes	string		
Application	app_id	Yes	relationship		
Database Name	database_name	Yes	string		
Database Type	database_type	Yes	list	oracle,mssql,db2,mysql,postgresql	

Figure 42: Attribute management

You can add new attributes by choosing the **Add** button on the attributes tab of the entity you wish to add the attribute to. In the example above, choosing **Add** will add a new attribute to the database entity.

On the **Amend attribute** dialog, you must to provide the following required properties:

**Programmatic name** – This is the key that will be used to store data for the attribute against items in the DynamoDB table. It is also referenced when using the Migration Factory APIs, and in automation scripts.

**Display name** – This is the label that will be displayed on the web interface against the data entry field.

**Type** – This drop-down selection defines the type of data that the user will be allowed to store against the attribute. The following options are available:

Type	Usage
String	Users can enter any single line of text carriage returns are not permitted.
Multivalue String	Similar to a <b>String</b> the only difference being that the user can enter multiple values on separate lines within the field, these are then stored as an array/list.
Password	Provides the user with a way to securely enter data that should not be displayed on the screen by default.  <b>NOTE: data is not stored encrypted when using this attribute type, and is shown in clear text when viewed in API payloads, thus should not be used to store sensitive data. Any passwords or secrets should be stored in the Migration Factory Credential Manager (covered in this document) which utilizes AWS Secrets Manager to securely store and provide access to credentials.</b>
Date	Provides a field with a date picker for the user to select a date, or they can manually enter the date required.
Checkbox	Provides a standard checkbox, when checked the key value will store 'true' if not checked then it will be 'false' or the key will not exist in the record.
TextArea	Unlike the <b>String</b> type <b>TextAreas</b> provide the ability to store multi line text, it supports basic text characters only.
Tag	Allows users to store a list of key/value pairs.
List	Provides the user a list of predefined options to select from, these options are defined in the schema attribute definition in the Value List property of the attribute.

Type	Usage
Relationship	<p>This attribute type provides the ability to store relationships between any two entities or records. When defining a relationship attribute you select the entity that the relationship will be to, and then the key value used to relate the items and select the attribute from the related item that you wish to display to the user.</p> <p>The user is presented with a drop-down list based on the entity and display values available for the relationship.</p> <p>Under each relationship field the user has a quick link to show the summary of the related item.</p>
JSON	<p>Provides a JSON editor field where JSON data can be stored and edited. This could be used for storing script input/output parameters or other data needed for automation of tasks, or any other use.</p>

When adding a new attribute, you will need to grant users access to the new attribute through a policy, please see the permissions section for details on how to grant attribute access.

## Info panel

Provides the facility to specify contextual help and guidance for the attribute's usage. When specified, the attribute's label on the UI will have an **Info** link displayed to the right. Clicking this link provides the user with the **Help content** and **Help links** specified in this section to the right of the screen.

The Info panel section provides two views of the data, the **Edit** view where you can define the content, and the **Preview** view to provide a quick preview of what the user will see when updates to the attribute are saved.

The **Help title** supports only plain text values. The **Help content** supports a subset of html tags that allow text formatting. For example, adding `<b>` start and `</b>` end tags around the text will make the enclosed text bold (i.e. `<b>Network Interface ID</b>` would result in **Network Interface ID**). The supported tags are as follows:

Tag	Usage	UI Example
<code>&lt;p&gt;&lt;/p&gt;</code>	Defines a paragraph.	<code>&lt;p&gt;My first paragraph&lt;/p&gt;</code> <code>&lt;p&gt;My second paragraph&lt;/p&gt;</code>
<code>&lt;a&gt;</code>	Defines a hyperlink.	<code>&lt;a href="https://aws.amazon.com/"&gt;Visit AWS!&lt;/a&gt;</code>
<code>&lt;h3&gt;</code> , <code>&lt;h4&gt;</code> and <code>&lt;h5&gt;</code>	Defines headings h3 to h5	<code>&lt;h3&gt;My heading 3&lt;/h3&gt;</code>
<code>&lt;span&gt;</code>	Defines a section of text, allowing additional formatting to be applied, such as text color, size, font.	<code>&lt;span style="color:blue"&gt;blue&lt;/span&gt;</code>

Tag	Usage	UI Example
<div>	Defines a block of the document, allowing additional formatting to be applied, such as text color, size, font.	<pre>&lt;div style="color:blue"&gt; &lt;h3&gt;This is a blue heading &lt;/h3&gt;  &lt;p&gt;This is some blue text in a div.&lt;/p&gt; &lt;/div&gt;</pre>
<ul> + <li>	Defines an unordered bulleted list.	<pre>&lt;ul&gt; &lt;li&gt;Rehost&lt;/li&gt; &lt;li&gt;Replatform&lt;/li&gt; &lt;li&gt;Retire&lt;/li&gt; &lt;/ul&gt;</pre>
<ol>, <li>	Defines an ordered/numbered list.	<pre>&lt;ol&gt; &lt;li&gt;Rehost&lt;/li&gt; &lt;li&gt;Replatform&lt;/li&gt; &lt;li&gt;Retire&lt;/li&gt; &lt;/ol&gt;</pre>
<code>	Defines a block or section of text containing code.	<pre>&lt;code&gt;background-color&lt;/code&gt;</pre>
<pre>	Defines a block of preformatted text, all line breaks, tabs, and spaces are output.	<pre>&lt;pre&gt; My preformatted text.  This is displayed in a fixed width font and will display as typed &lt;&lt;these spaces will be shown. &lt;/pre&gt;</pre>
<dl>, <dt> and <dd>	Defines a description list.	<pre>&lt;dl&gt; &lt;dt&gt;Rehost&lt;/dt&gt;  &lt;dd&gt;Lift and shift migration&lt;/dd&gt;  &lt;dt&gt;Retire&lt;/dt&gt;  &lt;dd&gt;Decommission the instance or service &lt;/dd&gt; &lt;/dl&gt;</pre>

Tag	Usage	UI Example
<hr>	Defines a horizontal rule across the page to show a switch in topic or section.	<hr>
 	Defines a line break in text. These are supported but not required as any carriage returns in the editor will be replaced with   when saved.	 
<i> and <em>	Defined the enclosed text in <i>Italic</i> or alternative localized format.	<i>This is in Italic</i> or <em>This is also Italic</em>
<b> and <strong>	Defines the enclosed text in <b>bold</b> font.	<b>I'm in bold</b> or <strong>This is different</strong>

Another option available to provide help is links to external content and guidance. To add an external link to the attribute's contextual help click **Add new URL** and provide a **Label** and URL. You can add multiple links to the same attribute type as required.

## Advanced options

### Attribute grouping and positioning

This section provides the administrator with the ability to set where on the Add/Edit UI the attribute will be positioned and also allows for grouping of attributes providing the user with a simple way to locate related attributes.

**UI Group** is a text value that defines the name of the group that the attribute should be shown in, all attributes with the same UI Group value will be placed in the same group, any attribute with no UI Group specified will be placed in the default group at the top of the form entitled **Details**. When UI Group is specified the user interface will display the text shown here as the title of the group.

The second property in this section is **Order in group**, this can be set to any positive or negative number, and when specified the attributes will be listed based on a sort of lowest to highest based on this value. Any attributes that do not have an **Order in group** specified will be lower priority and sorted alphabetically.

### Input validation

This section allows the administrator to define validation criteria that ensures that the user has entered valid data before they are able to save an item. The validation uses a regular expression or regex string which is a series of characters that specify a search pattern for a text value. For example, the pattern **^(subnet-([a-z0-9]{17})\*)\$** will search for the text **subnet-** followed by any combination of the characters a to z (lowercase) and digits 0 to 9 with an exact number of characters of **17**, if it finds anything else it will return false indicating that the validation has failed. Within this guide we cannot cover all the possible combinations and patterns available, but there are many resources on the internet that can provide help on creating the perfect one for your use case. Here are some common examples to get you started:

Regex Pattern	Usage
^(?!\\s*\$).+	Ensures that the value is set.

Regex Pattern	Usage
<code>^(subnet-([a-z0-9]{17})*)\$</code>	Checks that the value is a valid subnet ID.  [Starts with the text <b>subnet-</b> followed by 17 characters made up of letters and numbers only]
<code>^(ami-(([a-z0-9]{8,17})+)\$)</code>	Check that the value is a valid AMI ID.  [Starts with the text <b>ami-</b> followed by between 8 and 17 characters made up of letters and numbers only]
<code>^(sg-([a-z0-9]{17})*)\$</code>	Checks that the value is in a valid security group ID format.  [Starts with the text <b>sg-</b> followed by 17 characters made up of letters and numbers only]
<code>^(([a-zA-Z0-9][a-zA-Z0-9][a-zA-Z0-9\-\-]*[a-zA-Z0-9])\.)*([A-Za-z0-9][A-Za-z0-9][A-Za-z0-9\-\-]*[A-Za-z0-9])\$</code>	Ensures that server names are valid and only contain alphanumeric characters, hyphens and periods.
<code>^([1-9] [1-9][0-9] [1-9][0-9][0-9] [1-9][0-9][0-9][0-9] [1][0-6][0-3][0-8][0-4])\$</code>	Ensures that a number is entered that is between 1 and 1634.
<code>^(standard io1 io2 gp2 gp3)\$</code>	Ensures that the string enter matches either standard, io1, io2, gp2 or gp3.

Once you have created your regex search pattern, you can specify the specific error message that will be shown to the user under the field, enter this into the **Validation help message** property.

Once these two properties are set then in the same screen you will see below a **Validation simulator**, here you can test that your search pattern is working as expected and that the error message is displayed correctly. Just type some test text into the **Test validation** field to verify the pattern is matched correctly.

## Example data

The example data section provides the administrator with the ability to show the user an example of data format required for an attribute, this can be specified for the format of data required when provided in an intake form upload, through the user interface and/or through the API directly.

Example data shown in the **Intake form example data** property will be output in any intake template created where the attribute is included, when using the **Download, a template intake form** function, under **Migration Management > Import**.

User interface example data and API example data are stored in the attribute, but not currently exposed in the web interface. These can be used in integrations and scripts.

# Permission management

The Cloud Migration Factory on AWS solution provides granular role-based access control to the data and automation functions available in the solution, underlying this is Amazon Cognito, providing the user directory and authentication engine.

The following table shows the various elements that make up the access control framework within the Cloud Migration Factory and where each element is managed from.



Access control element	Management interface	Description
User	Amazon Cognito	Users are created, deleted and updated in Amazon Cognito, where the users' profile can be established as well as MFA if required.
Group	Amazon Cognito	Users can be added to groups within Amazon Cognito, by default the AWS CMF solution creates admin and read-only groups, but additional groups can be added.
Role	Cloud Migration Factory on AWS	<p>A Role is mapped to one or many Groups, changing the groups a Role is assigned to is performed in the AWS CMF administration section. Any user that is a member of a group assigned to a role will be assigned all policies that are mapped to the role.</p> <p>One or many policies can be assigned to a role.</p>
Policy	Cloud Migration Factory on AWS	A policy contains the detailed rights that are assigned to any user to which the policy applies (via group membership). A single policy can include data access rights for multiple entities or a single entity, along with access rights to run automation jobs and other actions within the AWS CMF user interface. These policies also apply when a user is interacting with the AWS CMF APIs.

## Policies

A policy provides the most granular permissions possible in Cloud Migration Factory on AWS, it holds the tasks level definition of what rights are provided to a user. Within a policy there are two main permission types that can be granted to a user group, **Metadata Permissions** and **Automation Action Permissions**. Metadata permissions allow an administrator to control the level of access a group has to individual schemas and their attributes, specifying rights to create, read, update and/or delete as required. Automation Action permissions grant users access to run specific automation actions, such as the AWS MGN integration action.

### Metadata permissions

For each schema or entity within AWS CMF an administrator can define a policy that allows users access to specific attributes and also define the level of access they have to those attributes. On creation of a

new policy the default rights for all schemas is no access, the first thing that should be set is the level of access required for this policy at the item/record level. Below is a table describing the record level access permissions available.

Access level	Description
Create	When selected, a user where this policy applies will have the ability to Add new records/items of this type to the metadata store. When create is selected but no other rights are allowed the user will have the ability to create records and set only required attributes to a value regardless of the selected attributes.
Read	<b>Not yet implemented (as of v3.0.0)</b> When selected, a user will have read rights to all records/items for this entity type, when not selected they will not see the data items in the UI or the API.
Update	When selected, a user where this policy applies will have the ability to update records/items of this type to the metadata store, but only for the attributes specified in the Attribute level access list. When update is selected at least one Attribute has to be selected or an error will be show on saving.
Delete	When selected, a user where this policy applies will have the ability to delete records/items of this type from the metadata store.

## Roles

Roles allow one or more policies to be assigned to one or more groups. The combination of all policies assigned to a role provides access permissions. Roles can be created based on job roles or functions within the project or organization.

# List of automated migration activities using factory web console

The Cloud Migration Factory on AWS solution deploys automated migration activities that you can leverage for your migration projects. You can follow the migration activities listed below and customize them based on your business needs.

Before starting any of the activities, make sure you read [User Guide – Run Automation from console \(p. 50\)](#) to understand how this works. Also, you must [Build an Automation server \(p. 17\)](#) and [create Windows and Linux users \(p. 49\)](#) to run automation from the console.

Use the following procedures in the same order to conduct a complete test run of the solution using the sample automation script and activities.

## Check prerequisites

Connect with the in-scope source servers to verify the necessary prerequisites such as TCP 1500, TCP 443, root volume free space, .Net framework version, and other parameters. These prerequisites are required for replication.

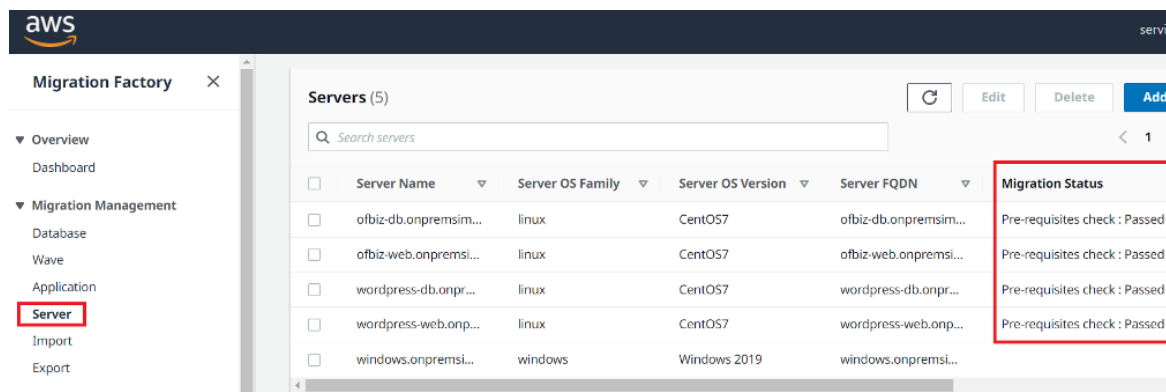
Before you can conduct the prerequisites check, you must install the first manually on one source server, so this will create a replication server in EC2. We will connect to this server for port 1500 testing. After installation, AWS Application Migration Service (AWS MGN) creates the replication server in Amazon Elastic Compute Cloud (Amazon EC2). You must verify the TCP port 1500 from the source server to the replication server in this activity. For information about installing the AWS MGN agent on your source servers, refer to [Installation instructions](#) in the *AWS Application Migration Service User Guide*.

Use the following procedure while signed in to the migration factory web console.

1. On the Migration Factory console, select **Jobs** on the left-hand side menu, and select **Actions**, and then **Run Automation** on the right-hand side.
2. Enter **Job Name**, select **0-Check MGN Prerequisites** script and your automation server to run the script. If automation server does not exist, make sure you complete [Build a migration automation server \(p. 17\)](#).
3. Select **Linux Secrets** and/or **Windows Secrets** depends on what OSs you have for this wave. Enter MGN replication server IP, choose the wave you want to run automation on, and choose **Submit Automation Job**.
4. You will be redirected to the **Jobs** list page, the job status should be running. Choose **Refresh** to see the status. It should change to Complete after a few minutes.

If the server failed one or more prerequisites checks, you can identify the faulty server by either reviewing the detailed error message provided at the completion of the check or by scrolling through the log details.

The script will also update the solution's **migration status** in the Migration Factory web interface as shown in the following screenshot of an example project.



Server Name	Server OS Family	Server OS Version	Server FQDN	Migration Status
ofbiz-db.onpremsim...	linux	CentOS7	ofbiz-db.onpremsim...	Pre-requisites check : Passed
ofbiz-web.onpremsi...	linux	CentOS7	ofbiz-web.onpremsi...	Pre-requisites check : Passed
wordpress-db.onpr...	linux	CentOS7	wordpress-db.onpr...	Pre-requisites check : Passed
wordpress-web.onp...	linux	CentOS7	wordpress-web.onp...	Pre-requisites check : Passed
windows.onpremsl...	windows	Windows 2019	windows.onpremsl...	

Figure 43: Migration status

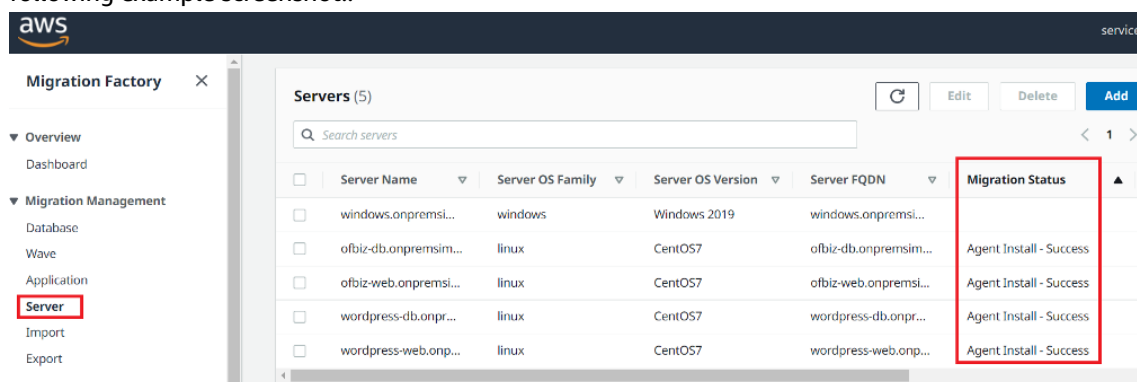
## Install the replication agents

### Note

Before you install the agent, make sure [AWS MGN is initialized in each target account](#) and region.

Use the following procedure to automatically install the Replication agents in the in-scope source servers.

1. On the Migration Factory console, select **Jobs** on the left-hand side menu, and select **Actions**, and then **Run Automation** on the right-hand side.
2. Enter **Job Name**, select **1-Install MGN Agents** script and your automation server to run the script. If automation server does not exist, make sure you complete [Build a migration automation server](#) (p. 17).
3. Select **Linux Secrets** and/or **Windows Secrets** depends on what OSs you have for this wave. Choose the wave you want to run automation, and choose **Submit Automation Job**.
4. You will be redirected to the **Jobs** list page, the job status should be running. Choose **Refresh** to see the status. It should change to **Complete** after a few minutes.
5. The script also provides the migration status in the Migration Factory web interface as shown in the following example screenshot..



Server Name	Server OS Family	Server OS Version	Server FQDN	Migration Status
windows.onpremsl...	windows	Windows 2019	windows.onpremsl...	Agent Install - Success
ofbiz-db.onpremsim...	linux	CentOS7	ofbiz-db.onpremsim...	Agent Install - Success
ofbiz-web.onpremsi...	linux	CentOS7	ofbiz-web.onpremsi...	Agent Install - Success
wordpress-db.onpr...	linux	CentOS7	wordpress-db.onpr...	Agent Install - Success
wordpress-web.onp...	linux	CentOS7	wordpress-web.onp...	Agent Install - Success

Figure 43: Migration status, Success

## Push the post-launch scripts

AWS Application Migration Service (MGN) supports post-launch scripts to help you automate OS-level activities, such as installing/uninstalling the software after launching target instances. This activity pushes the post-launch scripts to Windows and/or Linux machines, depending on the servers identified for migration.

### Note

Before you push the post-launch scripts, you must copy the files to a folder on the migration automation server.

Use the following procedure to push the post-launch scripts to Windows machines.

1. On the Migration Factory console, select **Jobs** on the left-hand side menu, and select **Actions**, and then **Run Automation** on the right-hand side.
2. Enter Job Name, select **1-Copy Post Launch Scripts** script and your automation server to run the script. If automation server does not exist, make sure you complete [Build a migration automation server](#) (p. 17).
3. Select **Linux Secrets** and/or **Windows Secrets** depends on what OSs you have for this wave. Provide a **Linux source location** and/or **Windows source location**.
4. Choose the wave you want to run automaton and choose **Submit Automation Job**.
5. You will be redirected to the **Jobs** list page, the job status should be running, and you can choose **Refresh** to see the status. It should change to **Complete** after a few minutes.

## Verify the replication status

This activity verifies the replication status for the in-scope source servers automatically. The script repeats every five minutes until the status of all source servers in the given wave changes to a *Healthy* status.

Use the following procedure to verify the replication status.

1. On the Migration Factory console, select **Jobs** on the left-hand side menu, and select **Actions**, and then **Run Automation** on the right-hand side.
2. Enter Job Name, select **2-Verify Replication Status** script and your automation server to run the script. If automation server does not exist, make sure you complete [Build a migration automation server](#) (p. 17).
3. Choose the wave you want to run automaton and choose Submit Automation Job.
4. You will be redirected to the **Jobs** list page, the job status should be running, and you can click **refresh** button to see the status. It should change to **Complete** after a few minutes.

### Note

Replication can take a while. You might not see the status update from the factory console for a few minutes. Optionally, you can also check the status in MGN service.

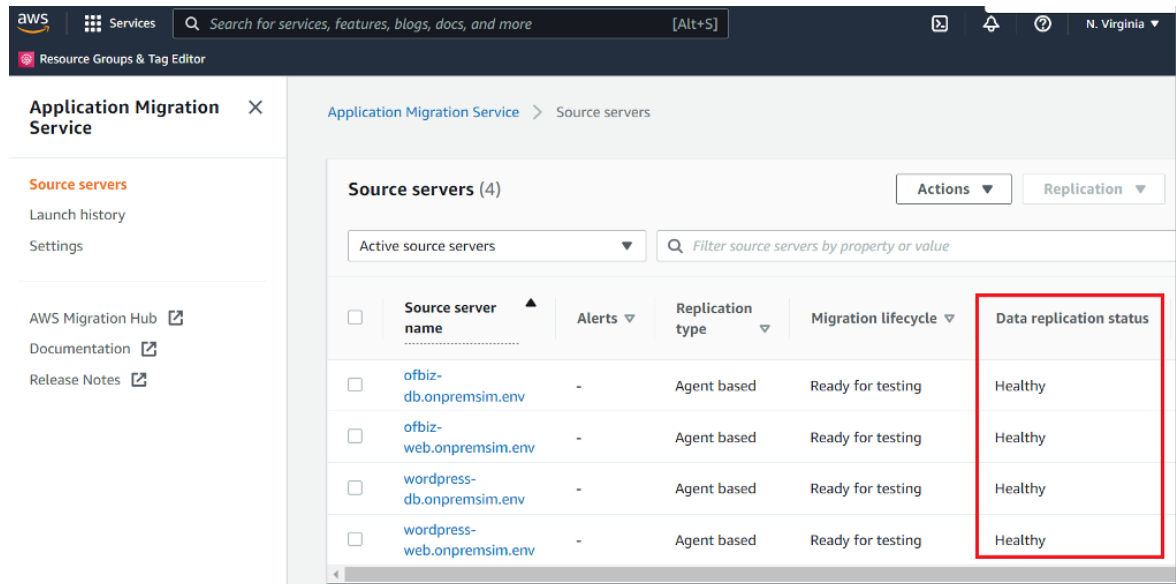


Figure 45: Data replication status for servers

## Validate launch template

This activity validates the server metadata in the migration factory and makes sure it works with EC2 template and no typos. It will validate both test and cutover metadata.

Use the following procedure to validate EC2 launch template.

1. Navigate to the Migration Factory console, and select **Wave** in the menu pane.
2. Select the target wave, and choose **Actions**. Select **Rehost**, and then select **MGN**.
3. Select **Validate Launch Template** for the **Action**, then select All **applications**.
4. Choose **Submit** to initiate the validation.

After some time, the validation will return a successful result.

**Note:** If validation is not successful, you will receive a specific error message:

The errors may be due to invalid data in the server attribute such as an invalid **subnet\_IDs**, **securitygroup\_IDs**, or **instanceType**.

You can switch to the Pipeline page from the Migration Factory web interface and select the problematic server to fix the errors.

## Launch instances for testing

This activity launches all target machines for a given wave in AWS Application Migration Service (MGN) in test mode.

Use the following procedure to launch test instances.

1. On the Migration Factory console, select **Wave** on the navigation menu.
2. Select target wave, and choose **Actions**. Select **Rehost**, and then select **MGN**.
3. Select **Launch Test Instances** Action, select **All applications**.
4. Choose **Submit** to launch test instances.
5. After some time, the validation will return a successful result.

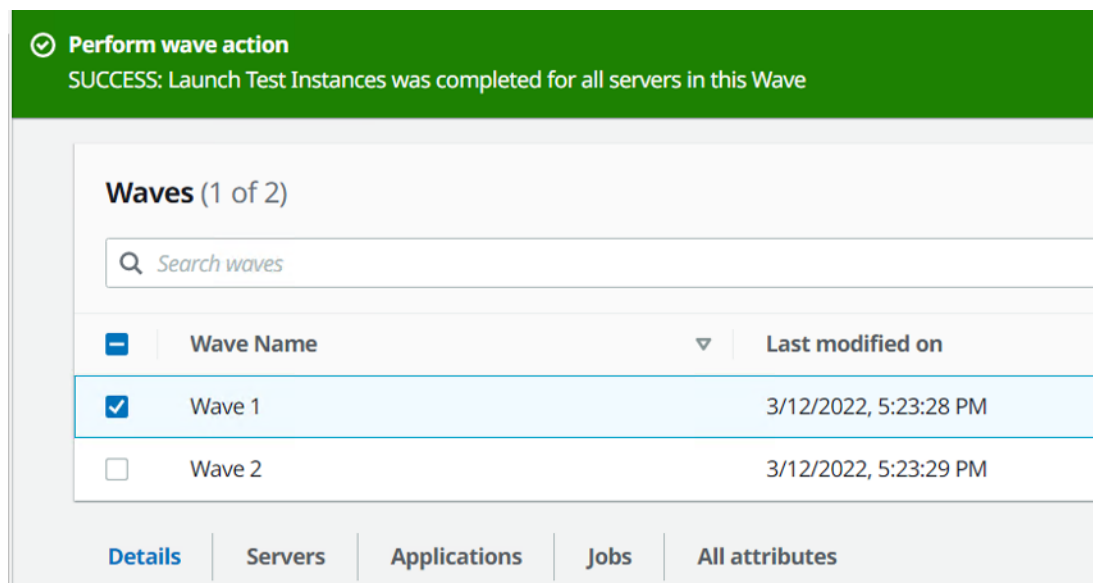


Figure 46: Wave action, success

**Note**

This action will also update the migration status for the server launched.

## Verify the target instance status

This activity verifies the status of the target instance by checking the boot up process for all in-scope source servers in the same wave. It may take up to 30 minutes for the target instances to boot up. You can check the status manually by logging into the Amazon EC2 console, searching for the source server name, and checking the status. You will receive a health check message stating *2/2 checks passed*, which indicates that the instance is healthy from an infrastructure perspective.

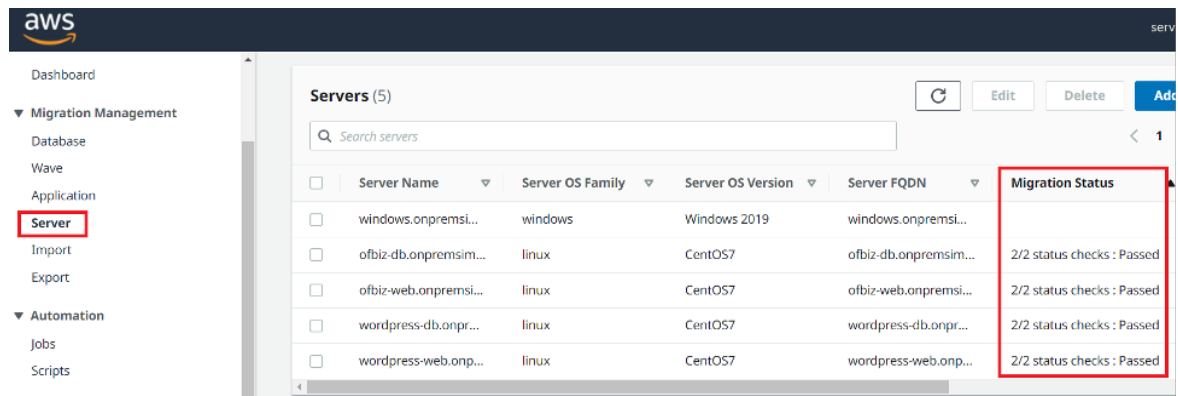
However, for a large-scale migration, it's time consuming to check the status of each instance, so you can run this automated script to verify the *2/2 checks passed* status for all source servers in a given wave.

Use the following procedure to verify the status of the target instance.

1. Navigate to the Migration Factory console, and select **Jobs** on the left-hand side menu.
2. Select **Actions**, then **Run Automation** on the right-hand side.
3. Enter **Job Name**, select **3-Verify Instance Status** script and your automation server to run the script. If automation server does not exist, ensure you complete [Build a migration automation server \(p. 17\)](#).
4. Choose the wave you want to run automaton, and choose **Submit Automation Job**.
5. You will be redirected to the **Jobs** list page, the job status should be running, and you can choose **Refresh** to see the status. It should change to **Complete** after a few minutes.

### Note

Instance boot up can take a while and you might not see the status update from the factory console for a few minutes. Migration factory also receives a status update from the script. Refresh the screen if necessary.



The screenshot shows the AWS Migration Factory console. On the left, the 'Server' option is highlighted in the 'Migration Management' sidebar. The main panel displays a table titled 'Servers (5)' with columns: Server Name, Server OS Family, Server OS Version, Server FQDN, and Migration Status. The table lists five servers, all of which have a 'Migration Status' of '2/2 status checks : Passed'. A red box highlights the 'Migration Status' column and the 'Server' sidebar item.

Server Name	Server OS Family	Server OS Version	Server FQDN	Migration Status
windows.onpremsi...	windows	Windows 2019	windows.onpremsi...	2/2 status checks : Passed
ofbiz-db.onpremsim...	linux	CentOS7	ofbiz-db.onpremsim...	2/2 status checks : Passed
ofbiz-web.onpremsi...	linux	CentOS7	ofbiz-web.onpremsi...	2/2 status checks : Passed
wordpress-db.onpr...	linux	CentOS7	wordpress-db.onpr...	2/2 status checks : Passed
wordpress-web.onp...	linux	CentOS7	wordpress-web.onp...	2/2 status checks : Passed

Figure 47: Migration status, 2/2 check

### Note

If your target instances fail the 2/2 health checks the first time, it may be due to the boot up process taking longer to complete. We recommend running the health checks a second time about an hour after the first health check. This ensures that the boot up process completes. If the health checks fail this second time, go to the [AWS support center](#) to log a support case.

## Mark as ready for cutover

Once testing is finished, this activity changes the status of the source server to mark as ready for cutover, so that user is able to launch a cutover instance.

Use the following procedure to validate EC2 launch template.

1. On the Migration Factory console, and select **Wave** on the left-hand side.
2. Select target wave, and click **Actions** button. Select **Rehost**, and then select **MGN**.
3. Select **Mark as Ready for Cutover** Action, select **All applications**.
4. Choose **Submit** to launch test instances.

After some time, the validation will return a successful result.



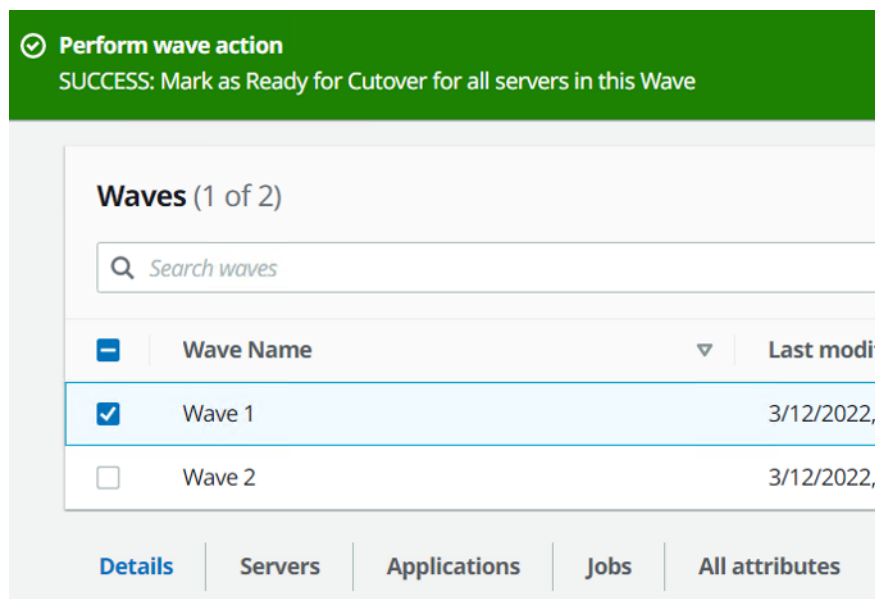


Figure 48: Wave action, ready for cutover

## Shut down the in-scope source servers

This activity shuts down the in-scope source servers involved with the migration. After you verify the source servers' replication status, you are ready to shut down the source servers to stop transactions from the client applications to the servers. You can shut down the source servers in the cutover window. Shutting down the source servers manually could take five minutes per server, and, for large waves, it could take a few hours in total. Instead, you can run this automation script to shut down all your servers in the given wave.

Use the following procedure to shut down all the source servers involved with the migration.

1. On the Migration Factory console, select **Jobs** on the left-hand side menu, and select **Actions**, and then **Run Automation** on the right-hand side.
2. Enter Job Name, select **3-Shutdown All Servers** script and your automation server to run the script. If automation server does not exist, make sure you complete [Build a migration automation server](#) (p. 17).
3. Select **Linux Secrets** and/or **Windows Secrets** depends on what OSs you have for this wave.
4. Choose the wave you want to run automaton and choose Submit Automation Job.
5. You will be redirected to the **Jobs** list page, the job status should be running, and you can click **refresh** button to see the status. It should change to **Complete** after a few minutes.

## Launch instances for Cutover

This activity launches all target machines for a given wave in AWS Application Migration Service (MGN) in Cutover mode.

Use the following procedure to launch test instances.

1. On the Migration Factory console, and select **Wave** on the left-hand side.

2. Select target wave, and choose **Actions**. Select **Rehost**, and then select **MGN**.
3. Select **Launch Cutover Instances** Action, select **All applications**.
4. Choose **Submit** to launch test instances.

After some time, the validation will return a successful result.

**Note**

This action will also update the migration status for the server launched.

# List of automated migration activities using command prompt

## Note

The recommended method to run automation is from the Cloud Migration Factory console. When running automation from the console is not a good option for you, you can use the following steps to run automation scripts. Make sure you download the automation scripts from the GitHub repo and configure the automation server with steps in [Run Automations from Command prompt \(p. 51\)](#), and following the instructions to configure permissions in [Configure AWS permissions for the migration automation server \(p. 18\)](#).

The Cloud Migration Factory on AWS solution deploys automated migration activities that you can leverage for your migration projects. You can follow the migration activities listed below and customize them based on your business needs.

Before starting any of the activities, verify that you are logged on to your migration automation server as a domain user with local administrator permission on the in-scope source servers.

## Important

You must log in as an administrator user to complete the activities listed in this section.

Use the following procedures in the same order to conduct a complete test run of the solution using the sample automation script and activities.

## Check prerequisites

Connect with the in-scope source servers to verify the necessary prerequisites such as TCP 1500, TCP 443, root volume free space, .Net framework version, and other parameters. These prerequisites are required for replication.

Before you can conduct the prerequisite check, you must install the first agent manually on one source server, so this will create a replication server in EC2, we will be connecting to this server for port 1500 testing. After installation, AWS Application Migration Service (AWS MGN) creates the replication server in Amazon Elastic Compute Cloud (Amazon EC2). You will need to verify the TCP port 1500 from the source server to the replication server in this activity. For information about installing the AWS MGN agent on your source servers, refer to [Installation instructions](#) in the *Application Migration Service User Guide*.

Use the following procedure while signed in to the migration automation server to check for the prerequisites.

1. Signed in as an administrator, open a command prompt (CMD.exe).
2. Navigate to the `c:\migrations\scripts\script_mgn_0-Prerequisites-checks` folder and run the following Python command.

```
python 0-Prerequisites-checks.py --Waveid <wave-id> --ReplicationServerIP <rep-server-ip>
```

Replace `<wave-id>` and `<rep-server-ip>` with the appropriate values:

- The Waveid is a unique integer value to identify your migration waves.

- The ReplicationServerIP value identifies the replication server IP address. Change this value to the Amazon EC2 IP address. To locate this address, sign in to the AWS Management Console, search for **Replication**, select one of the replication servers, and copy the private IP address. If the replication occurs over the public internet, use the public IP address instead.

3. The script automatically retrieves a server list for the specified wave.

The script then checks the prerequisites for Windows servers and returns a state of either pass or fail for each check.

**Note**

You may get a security warning like the following when the PowerShell script is not trusted. Run the following command in PowerShell to resolve the issue:

```
Unblock-File C:\migrations\scripts\script_mgn_0-Prerequisites-checks\0-Prerequisites-Windows.ps1
```

Next, the script checks the Linux servers.

Once the checks are completed, the script will return a final result for each server.

```
*****
**** Final results for all servers ****
*****

-----
-- Windows server passed all Pre-requisites checks --
-----

Server-T1.mydomain.local
server1.mydomain.local
Server-T15.mydomain.local
server2.mydomain.local

-----
-- Linux server passed all Pre-requisites checks --
-----

MF-RHEL.mydomain.local
MF-Ubuntu.mydomain.local
```

Figure 49: Script, final result

If the server failed one or more prerequisites checks, you can identify the faulty server by either reviewing the detailed error message provided at the completion of the check or by scrolling through the log details.

The script will also update the solution's **Migration Status** in the Migration Factory web interface as shown in the following screenshot of an example project.

## Install the replication agents

**Note**

Before you install the agent, make sure [AWS MGN is initialized in each target account](#).

Use the following procedure to automatically install the Replication agents in the in-scope source servers.

1. In the migration automation server, signed is as an administrator, open a command prompt (CMD.exe).
2. Navigate to the `c:\migrations\scripts\script_mgn_1-AgentInstall` folder and run the following Python command.

```
python 1-AgentInstall.py --Waveid <wave-id>
```

Replace `<wave-id>` with the appropriate Wave ID value to install the Replication agent on all servers in the identified wave. The script will install the agent on all source servers in the same wave one by one.

**Note**

To reinstall the agent, you can add `--force` argument

3. The script generates a list identifying the source servers included for the specified wave. In addition, servers that are identified in multiple accounts and for different OS versions may also be provided.

If there are Linux machines included in this wave, you must enter your Linux sudo username and password to log in to those source servers.

The installation starts on the Windows, then proceeds to the Linux for each AWS account.

```
*****
*** Installing Agents ***
*****

#####
#### In Account: 515800177220 , region: us-east-1 ####
#####

-----
- Installing Application Migration Service Agent for: Server-T1.mydomain.local -
-----

** Successfully downloaded Agent installer for: Server-T1.mydomain.local **
Verifying that the source server has enough free disk space to install the AWS Replication Agent.
(a minimum of 2 GB of free disk space is required)
Identifying volumes for replication.
Disk to replicate identified: c:\0 of size 30 GiB
All volumes for replication were successfully identified.
Downloading the AWS Replication Agent onto the source server... Finished.
Installing the AWS Replication Agent onto the source server... Finished.
Syncing the source server with the Application Migration Service Console... Finished.
The following is the source server ID: s-3fe3e5342c624e6a0.
The AWS Replication Agent was successfully installed.
The installation of the AWS Replication Agent has started.

** Installation finished for : Server-T1.mydomain.local **
```

Figure 50: Install replication agents

**Note**

You may get a security warning like the following when the PowerShell script is not trusted. Run the following command in PowerShell to resolve the issue:

```
Unblock-File C:\migrations\scripts\script_mgn_1-AgentInstall\1-Install-Windows.ps1
```

Results are displayed after the script finishes installing the replication agents. Review the results for error messages to identify servers that failed to install the agents. You will need to manually install the agents

on the failed servers. If manual installation does not succeed, go to the [AWS support center](#) and log a support case.

```
*****
*Checking Agent install results*
*****

-- SUCCESS: Agent installed on server: Server-T1.mydomain.local
-- SUCCESS: Agent installed on server: server1.mydomain.local
-- FAILED: Agent install failed on server: MF-RHEL.mydomain.local
-- SUCCESS: Agent installed on server: Server-T15.mydomain.local
-- SUCCESS: Agent installed on server: server2.mydomain.local
-- SUCCESS: Agent installed on server: MF-Ubuntu.mydomain.local
```

Figure 51: Check agent install result

The script also provides the migration status in the Migration Factory web interface as shown in the following screenshot of an example project.

## Push the post-launch scripts

AWS Application Migration Service supports post-launch scripts to help you automate OS-level activities such as the install/uninstall of software after launching target instances. This activity pushes the post-launch scripts to Windows and/or Linux machines, depending on the servers identified for migration.

Use the following procedure from the migration automation server to push the post-launch scripts to Windows machines.

1. Logged in as an administrator, open a command prompt (CMD.exe).
2. Navigate to the `c:\migrations\scripts\script_mgn_1-FileCopy` folder and run the following Python command.

```
python 1-FileCopy.py --Waveid <wave-id> --WindowsSource <file-path> --LinuxSource <file-path>
```

Replace `<wave-id>` with the appropriate Wave ID value and `<file-path>` with the full file path for Source, where the script is located. For example, `c:\migrations\scripts\script_mgn_1-FileCopy`. This command copies all files from the source folder to the destination folder.

### Note

At least one of these two arguments must be provided: `WindowsSource`, `LinuxSource`. If you provide `WindowsSource` path, this script will only push files to Windows servers in this wave, same as `LinuxSource`, which only pushes files to the Linux Servers in this wave. Provide both will push files to both Windows and Linux servers.

3. The script generates a list identifying the source servers included for the specified wave. In addition, servers that are identified in multiple accounts and for different OS versions may also be provided.

If there are Linux machines included in this wave, you must enter your Linux sudo username and password to log in to those source servers.

4. The script copies the files to the destination folder. If the destination folder does not exist, the solution creates a directory and notifies you of this action.

## Verify the replication status

This activity verifies the replication status for the in-scope source servers automatically. The script repeats every five minutes until the status of all source servers in the given wave changes to a *Healthy* status.

Use the following procedure from the migration automation server to verify the replication status.

1. Signed in as an administrator, open a command prompt (CMD.exe).
2. Navigate to the `\migrations\scripts\script_mgn_2-Verify-replication` folder and run the following Python command.

```
python 2-Verify-replication.py --Waveid <wave-id>
```

Replace `<wave-id>` with the appropriate Wave ID value to verify the replication status. The script verifies the replication details for the all servers in the specific wave and updates the **replication status** attribute for the source server identified in the solution.

3. The script generates a list identifying the servers included for the specified wave.

The expected status for the in-scope source servers that is ready to launch is **Healthy**. If you receive a different status for a server, then it is not ready for launch yet.

The following screenshot of an example wave shows that all servers in the current wave have finished replication and are ready for testing or cutover.

Optionally, you can verify status in the **Migration Factory** web interface.

## Verify the target instance status

This activity verifies the status of the target instance by checking the boot up process for all in-scope source servers in the same wave. It may take up to 30 minutes for the target instances to boot up. You can check the status manually by logging into the Amazon EC2 console, searching for the source server name, and checking the status. You will receive a health check message stating *2/2 checks passed*, which indicates that the instance is healthy from an infrastructure perspective.

However, for a large-scale migration, it's time-consuming to check the status of each instance, so you can run this automated script to verify the *2/2 checks passed* status for all source servers in a given wave.

Use the following procedure from the migration automation server to verify the status of the target instance.

1. Signed in as an administrator, open a command prompt (CMD.exe).
2. Navigate to the `c:\migrations\scripts\script_mgn_3-Verify-instance-status` folder and run the following Python command.

```
python 3-Verify-instance-status.py --Waveid <wave-id>
```

Replace `<wave-id>` with the appropriate Wave ID value to verify instance status. This script verifies the instance boot up process for all source servers in this wave.

3. The script returns a listing of the server list and Instance IDs for the specified wave.

4. The script will then return a list of target instance IDs.

**Note**

If you get an error message that the target instance Id does not exist, the launch job might still be running. Wait a few minutes before continuing.

5. You will receive instance status checks that indicates whether your target instances passed the 2/2 health checks.

**Note**

If your target instances fail the 2/2 health checks the first time, it may be due to the boot up process taking longer to complete. We recommend running the health checks a second time about an hour after the first health check. This ensures that the boot up process completes. If the health checks fail this second time, go to the [AWS support center](#) to log a support case.

## Shut down the in-scope source servers

This activity shuts down the in-scope source servers involved with the migration. After you verify the source servers' replication status, you are ready to shut down the source servers to stop transactions from the client applications to the servers. You can shut down the source servers in the cutover window. Shutting down the source servers manually could take five minutes per server, and, for large waves, it could take a few hours in total. Instead, you can run this automation script to shut down all your servers in the given wave.

Use the following procedure from the migration automation server to shut down all the source servers involved with the migration.

1. Signed in as an administrator, open a command prompt (CMD.exe).
2. Navigate to the `c:\migrations\scripts\script_mgn_3-Shutdown-all-servers` folder and run the following Python command.

```
Python 3-Shutdown-all-servers.py -Waveid <wave-id>
```

Replace `<wave-id>` with the appropriate Wave ID value to shut down the source servers.

3. The script returns a listing of the server list and Instance IDs for the specified wave.
4. The script first shuts down Windows servers in the specified wave. After the Windows servers are shut down, the script proceeds to the Linux environment and prompts for the login credentials. After successful login, the script shuts down the Linux servers.

## Retrieve the target instance IP

This activity retrieves the target instance IP. If the DNS update is a manual process in your environment, you would need to get the new IP addresses for all the target instances. However, you can use the automation script to export the new IP addresses for all the instances in the given wave to a CSV file.

Use the following procedure from the migration automation server to retrieve the target instance IPs.

1. Signed in as an administrator, open a command prompt (CMD.exe).



2. Navigate to the `c:\migrations\scripts\script_mgn_4-Get-instance-IP` folder and run the following Python command.

```
Python 4-Get-instance-IP.py --Waveid <wave-id>
```

Replace `<wave-id>` with the appropriate Wave ID value to get the new IP addresses for the target instances.

3. The script returns a server list and the target instance ID information.

4. The script will then return the target server IP.

The script exports the server name and IP addresses information to a CSV file (`<wave-id>-<project-name>-Ips.csv`) and places it in the same directory as your migration script (`c:\migrations\scripts\script_mgn_4-Get-instance-IP`).

The CSV file provides **instance\_name** and **instance\_ips** details. If the instance contains more than one NIC or IP, they will all be listed and separated by commas.

## Verify the target server connections

This activity verifies the connections for the target server. After you update the DNS records, you can connect to the target instances with the host name. In this activity, you check to determine if you can log in to the operating system by using Remote Desktop Protocol (RDP) or through Secure Shell (SSH) access. You can manually log in to each server individually, but it is more efficient to test the server connection by using the automation script.

Use the following procedure from the migration automation server to verify the connections for the target server.

1. Logged in as an administrator, open a command prompt (CMD.exe).
2. Navigate to the `c:\migrations\scripts\script_mgn_4-Verify-server-connection` folder and run the following Python command.

```
Python 4-Verify-server-connection.py --Waveid <wave-id>
```

Replace `<wave-id>` with the appropriate Wave ID value to get the new IP addresses for the target instances.

### Note

This script uses the default RDP port 3389 and SSH port 22. If needed, you can add the following arguments to reset to the default ports: `--RDPPort <rdp-port> --SSHPort <ssh-port>`.

3. The script returns a server list.
4. The script returns the test results for both RDP and SSH access.

# Update the stack

If you have previously deployed the solution, follow this procedure to update the Cloud Migration Factory on AWS solution CloudFormation stack to get the latest version of the solution's framework.

1. Sign in to the [AWS CloudFormation console](#), select your existing Cloud Migration Factory on AWS solution CloudFormation stack, and select **Update**.
2. Select **Replace current template**.
3. Under **Specify template**:
  - a. Select **Amazon S3 URL**.
  - b. Copy the link for the [latest template](#).
  - c. Paste the link in the **Amazon S3 URL** box.
  - d. Verify that the correct template URL shows in the **Amazon S3 URL** text box, and choose **Next**. Choose **Next** again.
4. Under **Parameters**, review the parameters for the template and modify them as necessary. Refer to [Step 1. Launch the Stack \(p. 12\)](#) for details about the parameters.
5. Choose **Next**.
6. On the **Configure stack options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template might create AWS Identity and Access Management (IAM) resources.
8. Choose **View change set** and verify the changes.
9. Choose **Update stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a **UPDATE\_COMPLETE** status in approximately 10 minutes.

# Uninstall the solution

You can uninstall the Cloud Migration Factory on AWS solution from the AWS Management Console or by using the AWS Command Line Interface. You must manually empty all the Amazon Simple Storage Service (Amazon S3) buckets created by this solution. AWS Solutions Implementations do not automatically delete S3 buckets in case you have stored data to retain.

## Empty the Amazon S3 buckets

If you decide to delete the AWS CloudFormation stack, this solution is configured to retain the created Amazon S3 bucket (for deploying in an opt-in Region) to prevent accidental data loss. You must manually empty all the S3 buckets before deleting the stack completely. Follow these steps to empty the Amazon S3 bucket.

1. Sign in to the [Amazon S3 console](#).
2. Choose **Buckets** from the left navigation pane.
3. Locate the `<stack-name>` S3 buckets.
4. Select the S3 bucket and choose **Empty**.

To delete the S3 bucket using AWS CLI, run the following command:

```
aws s3 rm s3://<bucket-name> --recursive
```

## Using the AWS Management Console to delete the stack

1. Sign in to the [AWS CloudFormation console](#).
2. On the **Stacks** page, select this solution's installation stack.
3. Choose **Delete**.

## Using AWS Command Line Interface to delete the stack

Determine whether the AWS Command Line Interface (AWS CLI) is available in your environment. For installation instructions, refer to [What Is the AWS Command Line Interface](#) in the *AWS CLI User Guide*. After confirming that the AWS CLI is available, run the following command.

```
aws cloudformation delete-stack --stack-name <installation-stack-name>
```

# Collection of operational metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When activated, the following information is collected and sent to AWS:

- **Solution ID:** The AWS solution identifier
- **Unique ID (UUID):** Randomly generated, unique identifier for each Cloud Migration Factory on AWS solution deployment
- **Timestamp:** Data-collection timestamp
- **Status:** Status is migrated once a server is launched in AWS MGN with this solution
- **Region:** The AWS Region where the solution is deployed

## Note

AWS will own the data gathered via this survey. Data collection will be subject to the [AWS Privacy Policy](#). To opt out of this feature, complete the following steps before launching the AWS CloudFormation template.

1. Download the [AWS CloudFormation template](#) to your local hard drive.
2. Open the AWS CloudFormation template with a text editor.
3. Modify the AWS CloudFormation template mapping section from:

```
Send:
  AnonymousUsage:
    Data: 'Yes'
```

to:

```
Send:
  AnonymousUsage:
    Data: 'No'
```

4. Sign in to the [AWS CloudFormation console](#).
5. Select **Create stack**.
6. On the **Create stack** page, **Specify template** section, select **Upload a template file**.
7. Under **Upload a template file**, choose **Choose file** and select the edited template from your local drive.
8. Choose **Next** and follow the steps in [Launch the stack \(p. 12\)](#) in the Automated deployment section of this guide.

# Source code

You can visit our [GitHub repository](#) to download the templates and scripts for this solution, and to share your customizations with others. If you require an earlier version of the CloudFormation template or have a technical issue to report, you can do so from the [GitHub issues](#) page.

# Revisions

Date	Change	
June 2020	Initial release	
February 2021	Release v1.1.0: Added optional migration tracker component; for more information on new features, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.	
May 2021	Release v1.1.1: Updated AWS Lambda functions to support Python v3.7; for more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.	
August 2021	Release v2.0.0: New feature to integrate with AWS Application Migration Service (AWS MGN); for more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.	
September 2021	Release v2.0.1: Bug fixes; for more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.	
October 2021	Release v2.0.2: Bug fixes; for more information, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.	
June 2022	Release v3.0.0: New web user interface and functionality to run all automation tasks directly from the UI using the new Remote Automation feature. This version removes the requirements to switch between automation server command lines and the web UI, providing a single migration automation view. Introduced Replatform to EC2, which allows the migration to perform deployments of new EC2 instances based on the configurations loaded into the Cloud Migration Factory datastore, providing migration waves the ability to have a mixture of migration strategies, with Replatform to EC2 and Rehost using MGN managed through a single Cloud Migration console. For more information about new features, refer to the <a href="#">CHANGELOG.md</a> file in the GitHub repository.	

# Contributors

The following individuals contributed to this document:

- Wally Lu
- Dev Kar
- Lyka Segura
- Chris Baker
- Dilshad Hussain
- Gnanasekaran Kailasam
- Jijo James
- Lakshmi Sudhakar Nekkanti
- Phi Nguyen
- Shyam Kumar
- Vijesh Vijayakumaran Nair

# Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

The Cloud Migration Factory on AWS solution is licensed under the terms of the [MIT No Attribution](#).



# AWS glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS General Reference*.