Cloud Migration Factory on AWS: Implementation Guide
Copyright © 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 execution 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 .................................................................................................................................................. 9
IAM roles .............................................................................................................................................. 9
Amazon Cognito .................................................................................................................................. 9
Amazon CloudFront ............................................................................................................................. 9
Implementation considerations ............................................................................................................. 10
Choosing between CloudEndure Migration and AWS Application Migration Service (MGN) ............ 10
Regional Deployments .......................................................................................................................... 10
AWS CloudFormation templates ......................................................................................................... 11
Automated deployment .......................................................................................................................... 12
Prerequisites ......................................................................................................................................... 12
Domain permissions ............................................................................................................................... 12
AWS Application Migration Service (AWS MGN) ................................................................................. 12
Deployment overview ............................................................................................................................ 12
Step 1. Launch the stack ......................................................................................................................... 13
Step 2. (For deploying AWS MGN ONLY) Launch the target account stack in the target AWS account .... 16
Step 3. Create the first user .................................................................................................................... 17
Create the initial user and log in to the solution .................................................................................... 17
Add a user to the admin group ............................................................................................................. 17
Identify the CloudFront URL ............................................................................................................... 18
Step 4. Update the factory schema ....................................................................................................... 18
(CloudEndure Migration only) Update the clouednure_projectname for CloudEndure Migration ................. 18
(AWS MGN only) Update the aws_accountid for AWS MGN ................................................................ 19
Step 5. Build a migration execution server ............................................................................................ 20
Build a Windows Server 2016 or later server ......................................................................................... 20
Download the sample automation scripts and update the configuration file ........................................... 21
Configure AWS permissions for the migration execution server .......................................................... 22
Step 6. Test the solution using the automation scripts .......................................................................... 24
Access the domains ............................................................................................................................... 24
Conduct a test run of the migration automation ................................................................................... 24
Step 7. (Optional) Build a migration tracker dashboard ......................................................................... 24
Set the permission and connections ..................................................................................................... 25
Create a dashboard ............................................................................................................................... 31
Additional resources ............................................................................................................................. 45
List of automated migration activities for CloudEndure Migration ......................................................... 46
Import the application and server data .................................................................................................. 46
Check the prerequisites ......................................................................................................................... 50
Install the CloudEndure agents ............................................................................................................ 53
Push the post-launch scripts .................................................................................................................. 55
Push the post-launch scripts for Windows ............................................................................................. 55
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. 98): October 2021)

The Cloud Migration Factory on AWS solution is designed to coordinate and automate manual processes for larger scaled 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.
- Rehost 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 CloudEndure Migration Factory in the AWS Prescriptive Guidance CloudEndure 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 August 2021, 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 **$5.83 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.

<table>
<thead>
<tr>
<th>AWS service</th>
<th>Factors</th>
<th>Cost/month</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Core services</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Amazon Elastic Container Service</td>
<td>15 mins x 2 vCPU x $0.04/vCPU per hour</td>
<td>$0.02</td>
</tr>
<tr>
<td>Amazon API Gateway</td>
<td>10,000 requests/month x ($3.50/million)</td>
<td>$0.035</td>
</tr>
<tr>
<td>AWS Lambda</td>
<td>10,000 invocations/month (avg 3,000 ms duration and 128 MB memory)</td>
<td>$0.065</td>
</tr>
<tr>
<td>Amazon DynamoDB</td>
<td>20,000 write requests/month x ($1.25/million)</td>
<td>$0.035</td>
</tr>
<tr>
<td></td>
<td>40,000 read requests/month x ($0.25/million)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Data storage: 1 GB x $0.25</td>
<td></td>
</tr>
<tr>
<td>Amazon S3</td>
<td>Storage (10MB) &amp; 50,000 get requests/month</td>
<td>$0.25</td>
</tr>
<tr>
<td>Amazon CloudFront</td>
<td>Regional data transfer out to internet: first 10 TB</td>
<td>$0.92</td>
</tr>
<tr>
<td></td>
<td>Regional data transfer out to origin: all data transfer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>HTTPS requests:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50,000 requests/month x ($0.01/10,000 requests)</td>
<td></td>
</tr>
<tr>
<td>Amazon Cognito</td>
<td>20 user x ($0.0055/monthly active users (MAUs))</td>
<td>$0.11</td>
</tr>
<tr>
<td>Amazon Athena</td>
<td>10MB daily x $5.00 per TB of data scanned</td>
<td>$0.0015</td>
</tr>
<tr>
<td><strong>Optional services</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS Glue (optional migration tracker)</td>
<td>2 mins daily x Default 10 DPU x $0.44 per DPU-Hour</td>
<td>$4.40</td>
</tr>
</tbody>
</table>
(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.

<table>
<thead>
<tr>
<th>AWS service</th>
<th>Factors</th>
<th>Cost/month</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon EC2</td>
<td>176 hours a month x $0.1108/ per hour (t3.large)</td>
<td>$19.50</td>
</tr>
<tr>
<td>Amazon Elastic Block Store (Amazon EBS)</td>
<td>30 GB x $0.08/GB-month (gp3) x (176 hours/720 hours)</td>
<td>$0.59</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td><strong>~$20.09</strong></td>
</tr>
</tbody>
</table>

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

Deploying this solution builds the following serverless environment in the AWS Cloud.

![Cloud Migration Factory on AWS architecture diagram](image)

**Figure 1: Cloud Migration Factory on AWS architecture diagram**

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

1. The template deploys Amazon Elastic Container Service (Amazon ECS) to build a web interface and uploads the code to an Amazon Simple Storage Service (Amazon S3) bucket.
   
   **Note**
   
   The Cloud Migration Factory on AWS solution uses a migration execution server which is not a part of the AWS CloudFormation deployment. For more details on manually building the server, refer to Build a Migration Execution Server (p. 20).

2. Amazon API Gateway receives migration requests from the migration execution server via RestAPIs.

3. **AWS Lambda** functions provide the necessary services for you to log in to the web interface, perform the necessary administrative functions to manage the migration, and connect to third party APIs to automate the migration process.
   
   - The **user** Lambda function ingests the migration metadata into an Amazon DynamoDB table. Standard HTTP status codes are returned to you via the RestAPI from the API Gateway. An Amazon Cognito user pool is used for user authentication to the web interface and RestAPIs.
   
   - The **tools** Lambda functions processes external RestAPI and calls the CloudEndure Migration and AWS Application Migration Service (AWS MGN) for AWS migration, and initiates workload ingestion (WIG) request for change (RFC) calls to AWS Managed Services (AMS) for AMS migration.

4. The migration metadata stored in Amazon DynamoDB is routed to CloudEndure or AWS MGN using a CloudEndure Migration or AWS MGN API to initiate a migration job and launch servers housed...
in CloudEndure. If your migration target is AMS, the tools Lambda function will automatically initiate an AMS RFC to automate the workload ingestion process to ingest your servers into the AMS production environment.

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 CloudEndure 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. 24).

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 execution server

This solution leverages a migration execution server to run migrations using RestAPIs. This server isn't automatically deployed with the solution and must be built manually. For more information, refer to Build a Migration Execution Server (p. 20). 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

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 execution server to the source servers, the following ports must be open from the migration execution server to the source servers:

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

We recommend that the migration execution 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 execution server.

- If domain trust exists in all the domains with source servers, a single migration execution 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 will need to create an additional migration execution server for each untrusted domain, or for each action to be performed on the execution 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 RestAPIs that are processed through AWS Lambda functions, an Amazon API Gateway, AWS Managed Services, 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, RestAPI 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
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 automated migration activities for CloudEndure Migration (p. 46) and automated migration activities for AWS Application Migration Service (p. 68). 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 migrate 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 for CloudEndure Migration (p. 46) and automated migration activities for AWS Application Migration Service (p. 68).

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

Choosing between CloudEndure Migration and AWS Application Migration Service

AWS Application Migration Service (AWS MGN) is the primary migration service recommended for lift-and-shift migrations to AWS. For many use cases, AWS Application Migration Service (AWS MGN) can be the fastest route to the cloud. For more information about the uses cases, refer to When to Choose AWS Application Migration Service.

If your preferred AWS Region is not currently supported by AWS MGN, consider using CloudEndure Migration. Additionally, if the operating system on which your applications run is not currently supported by AWS MGN, consider using CloudEndure Migration.

If you choose to use AWS Application Migration Service, then you must complete the Prerequisites (p. 12) prior to deploying the CloudFormation templates.

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 AWS Cloud Migration Factory Solution in the AWS Cloud. It includes the following AWS CloudFormation template, which you can download before deployment.

**cloud-migration-factory-solution.template**: As of August 2021, the latest template is version 2.0.0. 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.

**cloud-migration-factory-solution-mgn-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 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. 13).

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters: Stack name, Application name, Environment name, Version, Security Group Id, and Subnet Id.
- Review the template parameters, and adjust if necessary.

Step 2. Launch the target account stack in the target account (p. 16).

- Repeat this step for each target AWS account.

Step 3. Create the first user (p. 17).

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

Step 4. Update the schema for CloudEndure Migration or AWS Application Migration Service (p. 18).

- Update the factory schema.
Step 5. Build a migration execution server (p. 20).
- 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. 24).
- Get the necessary domain permissions.
- Conduct a test run of the migration automation.

Step 7. (Optional) Build a migration tracker dashboard (p. 24).

Note
For questions or get support for this solution, email <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. 96) 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 is launched 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application name</td>
<td>migration-factory</td>
<td>Enter a prefix to the AWS CloudFormation Physical ID that identifies the AWS services deployed by this solution.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Application name is used as a prefix to identify the AWS resources that are deployed: &lt;application-name&gt;-&lt;environment-name&gt;-&lt;aws-resource&gt;. 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.</td>
</tr>
<tr>
<td>Environment name</td>
<td>test</td>
<td>Enter a name to identify the network environment where the solution is deployed. We recommend a descriptive name such as test, dev, or prod.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The Environment name is used as a prefix to identify the AWS resources that are deployed: &lt;application-name&gt;-&lt;environment-name&gt;-&lt;aws-resource&gt;. 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.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>less to ensure you do not exceed character limitations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tracker</td>
<td>true</td>
<td>By default, the optional migration tracker dashboard is activated, but you can deactivate it by changing this parameter to false.</td>
</tr>
<tr>
<td>ServiceAccountEmail</td>
<td>serviceaccount@yourdomain</td>
<td>Default service account email address, the migration factory automation scripts uses this account to connect to the factory API.</td>
</tr>
<tr>
<td>Security Group Id</td>
<td>default</td>
<td>Enter the ID that the Amazon ECS cluster will use during the initial deployment of the solution to connect to the endpoint. <strong>Note</strong> The default security group ID in your AWS account is used, but you can provide a different ID. The security group must allow outbound access to the internet.</td>
</tr>
<tr>
<td>Subnet Id</td>
<td>default</td>
<td>Enter the ID that the Amazon ECS cluster will use during the initial deployment of the solution to connect to the endpoint. <strong>Note</strong> The default subnet ID in your AWS account is used, but you can provide a different ID. The subnet must allow outbound access to the internet.</td>
</tr>
</tbody>
</table>

**Note**
The solution uses the Security Group Id and Subnet Id from the default Amazon Virtual Private Cloud (Amazon VPC) in your AWS account. If your AWS account does not contain a default VPC, you will need to provide these identifiers from the primary Amazon VPC in your AWS account. For more information, refer to Default VPC and default subnets and Security groups for your VPC in the Amazon Virtual Private Cloud User Guide.

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.

---

**Step 2. (For deploying AWS MGN ONLY) Launch the target account stack in the target AWS account**

**Important**
If you are not using AWS MGN, then skip this step and proceed directly to Step 3 (p. 17).

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 Management Console and go to the CloudFormation Service. From here press Create stack selecting 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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FactoryAWSAccountld</td>
<td>111122223333</td>
<td>Enter an account ID where the Migration Factory was deployed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note: Launch this stack in the same AWS Region as the Migration Factory stack.</td>
</tr>
</tbody>
</table>

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 will allow 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 might take up to 2 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**

**Note**
Depends on your use cases, you may choose to update schema for CloudEndure Migration, AWS Application Migration Service (AWS MGN), or both.

(CloudEndure Migration only) Update the `cloudendure_projectname` for CloudEndure Migration

1. On the **Migration Factory** web interface, upper-right corner of the page, select the **Resource List** drop-down menu and select **Admin**.
2. Select the **Attribute Configuration** tab.
3. From the **Select Attribute Type** drop-down menu, select the **Application Attributes**.
4. From the left navigation pane, select `cloudendure_projectname`.
5. From the right side of the page, locate the **app Attribute Details** tab.
6. In the **List Value** field, replace the placeholder values with your specific CloudEndure Project name, then choose **Save Attribute**.
(AWS MGN only) Update the aws_accountid for AWS MGN

1. On the Migration Factory web interface, upper-right corner of the page, select the Resource List drop-down menu and select Admin.
2. Select the Attribute Configuration tab.
3. From the Select Attribute Type drop-down menu, select the Application Attributes.
4. From the left navigation pane, select aws_accountid.
5. From the right side of the page, locate the app Attribute Details tab.
6. In the List Value field, replace the placeholder values with a specific Id for your target AWS account, then choose Save Attribute.

Figure 3: Migration Factory web interface Attribute Details tab

Note
The project name is case sensitive. If you have more than one CloudEndure project, separate the project names with commas.

(AWS MGN only) Update the aws_accountid for AWS MGN
Step 5. Build a migration execution server

Build a Windows Server 2016 or later server

The migration execution 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. To review the server requirements, refer to Migration execution server (p. 6).

Note
The migration execution server must be in the source AD domain for both options.

On your migration execution 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 and install Python v3.9.2.
2. Set the Python installation folder as PATH environment variables. For example: C:\Users \<username>\AppData\Local\Programs\Python\<Python Version>\n
3. 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

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

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

4. **Install AWS CLI (Command Line Interface).**

**Download the sample automation scripts and update the configuration file**

Use the following procedure to download the sample automation scripts and update the configuration file.

1. Sign in to the migration execution server.
2. Take one of the following steps, depending on whether you have CloudEndure Migration or AWS Application Migration Service:
   - For CloudEndure Migration, download the `automation-scripts-ce.zip` sample automation scripts from the S3 bucket.
   - For AWS Application Migration Service, download the `automation-scripts-mgn.zip` sample automation scripts from the S3 bucket.
3. Open `cmd.exe` and take the following actions:
   1. From the `c:\` prompt, run the following command to create a folder named `migrations`:
      ```
      mkdir c:\migrations
      ```
   2. From the `c:\migrations` prompt, run the following command to create a folder named `scripts`:
      ```
      mkdir c:\migrations\scripts
      ```
4. Unzip the sample automation scripts file and copy all the files to the `scripts` folder.
5. Navigate to the [AWS CloudFormation console](https://aws.amazon.com/cloudformation/) and select the solution’s stack.
6. Select the Outputs tab and copy the `LoginAPI`, `UserAPI`, `Region` and `UserPoolId` values.
7. Navigate back to the command window, verify that you are in the `c:\migrations\scripts` folder, and open the `FactoryEndpoints.json` file.

```json
{
  "FactoryEndpoints.json
  1 {
  2   "LoginApiUrl": "<LoginApi-value>",
  3   "UserApiUrl": "<UserApi-value>",
  4   "region": "<Region-value>",
  5   "UserPoolId": "<UserPoolId-value>"
  6 }
}
```

Replace `<LoginApi-value>, <UserApi-value>, <Region-value>`, and `<UserPoolId-value>` with the corresponding values you retrieved from the AWS CloudFormation console. Do not add a slash (/) at the end of the URL.
Configure AWS permissions for the migration execution server

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

(Highly Recommended) Option 1: Use the following procedure to configure the permissions for the migration execution server to AWS in Amazon EC2.

1. Navigate to the AWS CloudFormation console and select the solution's stack.
2. Select the Outputs tab, under the Key column, locate ExecutionServerIAMRole and record the Value. You will need to provide this value later in the deployment.

   ![Figure 5: AWS CloudFormation console, Outputs tab](image)

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
8. Select Modify IAM role.
9. From the list of IAM roles, locate and select the IAM role containing the value for ExecutionServerIAMRole that you recorded in Step 2, and choose Save.

Option 2: Use the following procedure to configure the permissions for the migration execution 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. You will need to provide this value later in the deployment.

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

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 for CloudEndure Migration (p. 46) and automated migration activities for AWS Application Migration Service (p. 68).

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 for CloudEndure Migration (p. 46) and automated migration activities for AWS Application Migration Service (p. 68).

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. 31).

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.

![QuickSight select Amazon S3 buckets dialog box](image)

*Figure 8: QuickSight select Amazon S3 buckets 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](image)

**Figure 9:** QuickSight Amazon Athena resources dialog box

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.

![New Athena data source dialog box](image)

**Figure 10: 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**.
9. In the next dialog box, Finish data set creation, select Visualize.
Set the permission and connections

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.

4. In the `<migration-factory>-general-view` dialogue box, choose Schedule refresh.
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 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.
Create a dashboard

Figure 16: 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.
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 17: 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**.
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 18: Field wells page**

The data type of a field used in this view will automatically update your analysis, replacing the current field with a different one.
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.
Create a dashboard

Figure 20: 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 21: 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 22: Migration factory dataset dialog box](image)

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

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

7. Choose Save.

8. In the Data page, choose Save & visualize.
Your newly added query will be listed in the **Data set Fields list**.

**Figure 26: 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.

![Dashboard visualization]

**Figure 27: 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:

```
Total Server Breakdown for all Waves
Quick summary of the total count of servers across all waves:
- **InProgress** with 15,979
- **Completed** with 4,205
```

**Figure 28: Example dashboard insights**

You can also customize the data by breaking down the metadata into waves. For example:
Figure 29: 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**.

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.
Cloud Migration Factory on AWS Implementation Guide

Create a dashboard

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

   ![Figure 31: Add an insight to a visual](image1)

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

   ![Figure 32: Add an Insight to your dashboard](image2)

   ![Figure 33: Customize narrative option](image3)
Create a dashboard

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 34: 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 35: 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 36: 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
- Amazon Elastic Container Service

AWS resources

- CloudEndure Migration Factory guide
List of automated migration activities for CloudEndure Migration

Note
For a list of automated migration activities for AWS Application Migration Service (AWS MGN), refer to the List of automated migration activities for AWS Application Migration Service (p. 68) section in this guide.

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 execution server as a domain user with local administrator permission on the in-scope source servers.

Important
All the activities listed in this topic require you to be logged in as an administrator.

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

Import the application and server data

To start the migration process, navigate to the 0-Migration-intake-form.csv file located in the c:\migrations\scripts folder. This file is used as the 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 you unzipped in Step 5 (p. 20) of the Automated deployment section.

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wave_id</td>
<td>Yes</td>
<td>The wave number which is based on priority and application server dependencies. Obtain this identifier from your migration plan.</td>
</tr>
<tr>
<td>app_name</td>
<td>Yes</td>
<td>The names of the applications that are in-scope for migration. Confirm that your application grouping includes all the applications sharing the same servers.</td>
</tr>
<tr>
<td>cloudendure_projectname</td>
<td>Yes</td>
<td>The project name(s) you created in CloudEndure containing the AWS account destination for the migration.</td>
</tr>
</tbody>
</table>
## Import the application and server data

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws_accountid</td>
<td>Yes</td>
<td>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 <strong>My Account</strong> from the drop-down menu.</td>
</tr>
<tr>
<td>server_name</td>
<td>Yes</td>
<td>The name of the on-premises servers that are in-scope for migration.</td>
</tr>
<tr>
<td>server_os_family</td>
<td>Yes</td>
<td>The operating system (OS) that is running on the in-scope source servers. Use either <strong>windows</strong> or <strong>linux</strong> since this solution supports only these operating systems.</td>
</tr>
<tr>
<td>server_os_version</td>
<td>Yes</td>
<td>The version of the OS running on the in-scope source servers. Note: Use the OS version, not the Kernel version, for example, use RHEL 7.1, Windows Server 2012 R2, or CentOS 7.5, 7.6. Do not use Linux 3.xx, 4.xx, or Windows 8.1.x.</td>
</tr>
<tr>
<td>server_fqdn</td>
<td>Yes</td>
<td>The source server's fully qualified domain name, which is the server name followed by the domain name. For example, server123.company.com.</td>
</tr>
<tr>
<td>server_tier</td>
<td>Yes</td>
<td>A label to identify whether the source server is a <strong>web</strong>, <strong>app</strong>, or a <strong>database</strong> server. We recommend designating the source server as <strong>app</strong> if the server functions as more than one tier, for example, if the server runs web, app, and database tiers together.</td>
</tr>
<tr>
<td>server_environment</td>
<td>Yes</td>
<td>A label to identify the server's environment. For example, <strong>dev</strong>, <strong>test</strong>, <strong>prod</strong>, <strong>QA</strong>, or <strong>pre-prod</strong>.</td>
</tr>
<tr>
<td>subnet_IDs</td>
<td>Yes</td>
<td>The subnet ID for the target Amazon EC2 instance for the migration post-cutover.</td>
</tr>
</tbody>
</table>
Import the application and server data

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>securitygroup_IDs</td>
<td>Yes</td>
<td>The security group ID for the target Amazon EC2 instance for the migration post-cutover.</td>
</tr>
<tr>
<td>subnet_IDs_test</td>
<td>Yes</td>
<td>The target subnet ID for the source server that will be tested.</td>
</tr>
<tr>
<td>securitygroup_IDs_test</td>
<td>Yes</td>
<td>The target security group ID for the source server that will be tested.</td>
</tr>
<tr>
<td>instanceType</td>
<td>Yes</td>
<td>The Amazon EC2 instance type identified in the discovery and planning effort. For information about EC2 instance types, refer to Amazon EC2 Instance Types.</td>
</tr>
<tr>
<td>tenancy</td>
<td>Yes</td>
<td>The tenancy type, which is identified during the discovery and planning efforts. Use one of the following values to identify the tenancy: Shared, Dedicated, or Dedicated Host. You can use Shared as the default value unless an application's license requires a specified type.</td>
</tr>
</tbody>
</table>

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

```
python 0-Import-intake-form.py --Intakeform "<file-path>"
```

3. Replace `<file-path>` with the correct intake form location, for example, c:\migrations\scripts \0-Migration-intake-form.csv.
4. Sign in to the solution with your username and password.

The script loads the CSV data and then creates the waves in the solution.
After the waves are created, the script automatically creates the applications and servers based on the information from the CSV file.

Figure 37: Migration factory creating the waves

Figure 38: Migration factory creating the applications and servers
Check the 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 CloudEndure replication.

Before you can conduct the check, the CloudEndure agent must be installed manually on one source server. After installation, CloudEndure creates the CloudEndure replication server in Amazon Elastic Compute Cloud (Amazon EC2). You will need to verify the TCP port 1500 from the source server to the CloudEndure replication server in this activity. For information about installing the CloudEndure agent in your source servers, refer to Install the CloudEndure Agents manually.

Figure 39: Search for CloudEndure replication server in Amazon EC2

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

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

   python 0-Prerequisites-checks.py --Waveid <wave-id> --CEServerIP <ces-server-ip>

   Replace <wave-id> and <ces-server-ip> with the appropriate values:
   • The Waveid is a unique integer value to identify your migration waves.
   • The CEServerIP value identifies the CloudEndure 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 CloudEndure 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. Sign in to the solution with your username and password.

   The script automatically retrieves a server list for the specified wave and CloudEndure projects.

   Figure 40: Retrieve 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.

```
#Checking Pre-requisites for Windows servers#

--- Windows Server result for Server1 ---

TCP 443 to CE Console : Pass
TCP 1500 to Rep Server : Pass
.Net 3.5 version : Pass
3GB C:\ Free Space : Pass

--- Windows Server result for Server2 ---

TCP 443 to CE Console : Pass
TCP 1500 to Rep Server : Pass
.Net 3.5 version : Pass
3GB C:\ Free Space : Pass
```

Figure 41: Windows servers prerequisite checks

Next, the script checks the Linux servers.

```
#Checking Pre-requisites for Linux servers#

Linux Username: stackadmin
If you use a private key to login, press Y or if use password press N: N
Linux Password:
Re-enter Password:

--- Linux Server result for MF-LX ---

SSH 22 to source server : Fail

--- Linux Server result for MF-RHEL ---

SSH 22 to source server : Pass
SUDO privileges : Pass
TCP 443 to CE Console : Pass
TCP 1500 to Rep Server : Pass
3.0 GB / FreeSpace : Pass
0.5 GB / tmp FreeSpace : Pass
DAGCLIENT Package : Pass
```

Figure 42: Linux servers prerequisite checks

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

```
***** Finally results for all servers *****

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

servers1.mymdomain.local
servers2.mymdomain.local

--- Windows server failed one or more Pre-requisites checks ---

servers3.mymdomain.local : Unexpected error, please check error details
```

Figure 43: Windows servers prerequisite check results
Check the prerequisites

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. Use the following procedure to check the status details.

1. Log in to the Migration Factory web interface and select Resource List from the drop-down menu on the upper-right corner of the page.
2. Choose the Server List tab and enter the wave name in the search field. In the following screenshot of an example project, we searched for Wave 1.

   ![Figure 45: Search for a server in the Server List tab](image)

   A list of source servers targeted in Wave 1 displays on the page.
3. Choose the settings icon on the upper-right corner of the page.

   ![Figure 46: Settings icon](image)

4. In the Show/Hide Columns dialog box, scroll down the Server Attributes options, and select migration_status.
5. Choose Update View. The migration_status column will display.

When the check is successful, the migration_status will show Pre-requisites check: Passed as shown in the following screenshot of an example project.
Install the CloudEndure agents

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

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

   ```python
def CEAgentInstall.py --Waveid <wave-id>
```

   Replace `<wave-id>` with the appropriate Wave ID value to install the CloudEndure 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.

3. Log in to the solution with your username and password.
4. Enter the CloudEndure API token.

   **Note**
   To find your CloudEndure API token, navigate to the CloudEndure console, choose Setup & info, choose Other Settings, then choose API Token. If the token does not exist, select Generate token.

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

   **Figure 48: Migration factory server list**

   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 machines, then proceeds to the Linux machines for each project.

Figure 49: Install CloudEndure agents

Results are displayed after the script finishes installing the CloudEndure agents. Review the results for error messages to identify servers that failed to install the agents. You will need to manually install CloudEndure agents on the failed servers. If manual installation does not succeed, go to the AWS support center and log a CloudEndure case.

Figure 50: Check CloudEndure agent install results

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

Figure 51: Migration factory web interface showing migration_status
Cloud Migration Factory on AWS Implementation Guide

Push the post-launch scripts

CloudEndure 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.

Push the post-launch scripts for Windows

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

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

```
python 1-FileCopy-Windows.py --Waveid <wave-id> --Source <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. This command copies all files from the source folder to the destination folder.
3. Log in to the solution with your username and password.

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. In the following screenshot of an example project, the solution created a directory labeled post_launch.

```
+Coping files to post_launch folder+
Copying files to server: server1.nydomain.local
Copying files to server: server2.nydomain.local

Directory: \server1.nydomain.local\c:\Program Files (x86)\CloudEndure

<table>
<thead>
<tr>
<th>Mode</th>
<th>LastWriteTime</th>
<th>Length</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8/31/2019 8:20 PM</td>
<td></td>
<td>post_launch</td>
</tr>
</tbody>
</table>

Copying files to server: server2.nydomain.local
Copying files to server: server3.nydomain.local
```

Figure 52: Copy files to post_launch folder

Push the post-launch scripts for Linux

Use the following procedure from the migration execution server to push the post-launch scripts to Linux machines.

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

```
python 1-FileCopy-Linux.py --Waveid <wave-id> --Source <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. This command copies all files from the source folder to the destination folder.
3. Log in to the solution with your username and password.
4. Enter your Linux sudo username and password to copy the files to the in-scope source servers.

```
Getting Server List
Successfully retrieved server list
MF-RHEL.mydomain.local
MF-Ubuntu.mydomain.local

*Provide Linux credentials to copy files*
Linux Username: stackadmin
If you use a private key to login, press [Y] or if use password press [N]: N
Linux Password: 
Re-enter Password: 
```

Figure 53: Migration factory Linux login

You will receive the following message if the files were successfully copied to the source servers.

```
*Copying files to post_launch folder*
Successfully copied files to MF-RHEL.mydomain.local
Successfully copied files to MF-Ubuntu.mydomain.local
```

Figure 54: Successful copy files to post_launch folder message

## Verify the CloudEndure 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 *continuous data replication* status.

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

1. Signed in as an administrator, open a command prompt (`CMD.exe`).
2. Navigate to the `c:\migrations\scripts` 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 CloudEndure projects and updates the `replication_status` attribute for the source server identified in the solution.

3. Log in to the solution with your username and password.
4. Enter the CloudEndure API token.

   **Note**
   To find your CloudEndure API token, navigate to the CloudEndure console, choose Setup & info, choose Other Settings, then choose API Token. If the token does not exist, select Generate token.

   The script generates a list identifying the servers included for the specified wave.
Verify the CloudEndure replication status

The expected status for the in-scope source servers that is ready to launch is **Continuous Data Replication**. If you find a different status for a server, then it is not ready for launch yet. Other server status include:

- **Initial sync in progress**: the server has not finish initial replication
- **Booting_replication_server**: the initial sync has not started

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

<table>
<thead>
<tr>
<th>* Verify replication status *</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Server List *</td>
</tr>
<tr>
<td>* Server1 * replication status: Initial sync in progress</td>
</tr>
<tr>
<td>* Server2 * replication status: Initial sync in progress</td>
</tr>
<tr>
<td>* Server3 * replication status: Initial sync in progress</td>
</tr>
<tr>
<td>* Server5 * replication status: BOOTING_REPLICATION_SERVER</td>
</tr>
</tbody>
</table>

**Figure 56: Replication status for servers**

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

1. Log in to the **Migration Factory** web interface and select **Resource List** from the drop-down menu on the upper-right corner of the page.
2. Choose the **Server List** tab and enter the wave name in the search field.
3. Choose the settings icon on the upper-right corner of the page.
4. In the **Show/Hide Columns** dialog box, scroll down the **Server Attributes** options and select **replication_status**.

---

**Figure 55: Migration factory server list**

<table>
<thead>
<tr>
<th>* Getting Server List *</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Servers for CE Project: Demo2 *</td>
</tr>
<tr>
<td>* Server1 *</td>
</tr>
<tr>
<td>* Server2 *</td>
</tr>
<tr>
<td>* Server3 *</td>
</tr>
</tbody>
</table>

| \* Servers for CE Project: Demo3 \* |
| \* Server5 \* |

<table>
<thead>
<tr>
<th>* Verify replication status *</th>
</tr>
</thead>
<tbody>
<tr>
<td>* Replication Status for CE Project: Demo2 *</td>
</tr>
<tr>
<td>* Server1 * replication status: Initial sync in progress</td>
</tr>
<tr>
<td>* Server2 * replication status: Initial sync in progress</td>
</tr>
<tr>
<td>* Server3 * replication status: Initial sync in progress</td>
</tr>
<tr>
<td>* Server5 * replication status: BOOTING_REPLICATION_SERVER</td>
</tr>
</tbody>
</table>

| \* Replication Status for CE Project: Demo3 \* |
| \* Server1 \* replication status: Continuous Data Replication |
| \* Server2 \* replication status: Continuous Data Replication |
| \* Server3 \* replication status: Continuous Data Replication |
| \* Server5 \* replication status: Continuous Data Replication |

**Figure 57: Settings icon**
Create the admin user

Create the local admin user for a Windows environment

This activity creates a local admin user on Windows machines. This user may be needed to troubleshoot issues that may occur after migration cutover from the in-scope source servers to AWS. The migration team will use this user to log in to the target server when the authentication server (for example, the DC or LDAP server) is not reachable.

**Note**

This activity is not required if you have a local admin user and log in credentials for all source servers.

Use the following procedure to create a local admin user.

1. Use a remote desktop protocol (RDP) client to log in to the migration execution server with domain credentials. This user needs admin permissions to all source Windows machines.
2. Signed in as an administrator, open a command prompt (CMD.exe).
3. Navigate to the c:\migrations\scripts folder and run the following Python command to create a local user and add the user to the local administrator group.

   ```
   python 2-UserMgmt-Windows.py --Waveid <wave-id>
   ```

   Replace `<wave-id>` with the appropriate Wave ID value to create a local admin user for all the Windows servers in that wave.

4. Select option 1 to create a user.
5. Log in to the solution with your username and password.
6. Enter a new local admin username and password. A local admin user for all source Windows servers in the specified wave is created.

![Figure 58: Local admin user for Windows servers](image)

---

5. Choose **Update View**. The replication_status column displays.
Create a local sudo user for a Linux environment

Note

The migration team will use this user to log in to the target server when the authentication server (for example, the DC or LDAP server) is not reachable, for troubleshooting purposes.

You will receive the following message when the user is successfully created.

```
- Creating a local user on: server1.mydomain.local -
The command completed successfully.

Adding user to local admin group on server: server1.mydomain.local
The command completed successfully.

- Creating a local user on: server2.mydomain.local -
The command completed successfully.

Adding user to local admin group on server: server2.mydomain.local
The command completed successfully.
```

Figure 59: Local admin user created confirmation message

Create a local sudo user for a Linux environment

This activity creates a local sudo user on Linux machines. This user role may be needed to troubleshoot issues that may occur after migration cutover from the in-scope source servers to AWS. The migration team will use this role to log in to the target server when the authentication server (for example, the DC or LDAP server) is not reachable.

Note

This activity is not required if you have a sudo user and log in credentials for all source servers.

Use the following procedure to create a local sudo user.

1. Use a remote desktop protocol (RDP) client to log in to the migration execution server with domain credentials.
2. Signed in as an administrator, open a command prompt (CMD.exe).
3. Navigate to the c:\migrations\scripts folder and run the following Python command to create a local sudo user and add the user to the local administrator group.

```
python 2-UserMgmt-Linux.py --Waveid <wave-id>
```

Replace `<wave-id>` with the appropriate Wave ID value to create a local sudo user for all the Linux servers in that wave.

4. Select option 1 to create a user.
5. Log in to the solution with your username and password.
6. Enter a new Linux sudo username and password. A local sudo user for all source Linux servers in the specified wave is created.
Launch instances for testing or cutover

This activity launches all target machines for a given wave in CloudEndure either in **test mode** or **cutover mode**.

Use the following procedure to launch instances for either testing or cutover.

1. Log in to the **Migration Factory** web interface.
2. From the upper-right corner of the page, select **Tools** from the drop-down menu.
3. Choose the **CloudEndure** tab.
4. On the **CloudEndure Launch configuration** page, take the following actions:
   - In the **CloudEndure API Token** field, enter your CloudEndure API token.
     
     **Note**
     To find your CloudEndure API token, navigate to the CloudEndure console, choose **Setup & info**, choose **Other Settings**, then choose **API Token**. If the token does not exist, select **Generate token**.
   - For **Dry run**, select the drop-down arrow and select **yes**.
   - For **Launch Type**, select the drop-down arrow and select **test**.
     
     **Note**
     To launch the servers in cutover mode, select **cutover**.
   - Select **Launch Servers** to validate the CloudEndure blueprint.
Cloud Migration Factory on AWS Implementation Guide

Launch instances for testing or cutover

Figure 62: CloudEndure launch configuration page

When validation is successful, you will receive the following message: **Dry run was successful for all machines.**

**Note**
If validation is not successful, you will receive the following error message:
ERROR: Updating blueprint failed for machine: `<your-server-name>`, invalid blueprint config.
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.

5. On the **CloudEndure Launch configuration** page, in the **Dry run** field, select the drop-down arrow and select **no**.
6. Choose **Launch Servers** to create a job in CloudEndure.

When the servers are successfully launched in CloudEndure, you will receive a confirmation message: Test Job created for `<machine-names>` or Cutover Job created for `<machine-names>`, depending on your launch type.

If there are more than one CloudEndure project in the specified wave, repeat the steps for each CloudEndure project. The CloudEndure job and boot up process takes approximately 30 minutes to an hour to finish.
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 CloudEndure project. 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 execution 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 folder and run the following Python command.

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

   Replace <wave-id> with the appropriate Wave ID value and <project-name> with the CloudEndure project name to verify instance status. This script verifies the instance boot up process for all source servers in a CloudEndure project. If there are more than one CloudEndure project in a wave, you can run this script for each project.

3. Log in to the solution with your username and password.

   ![Migration factory login](image)

   **Figure 80: Migration factory login**

4. Enter the CloudEndure API token.

   **Note**

   To find your CloudEndure API token, navigate to the CloudEndure console, choose Setup & info, choose Other Settings, then choose API Token. If the token does not exist, select Generate token.

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

   ```
   * Getting Server List *
   test1
   test2
   test3
   * Getting Target Instance Id *
   test3 : i-0707c3a64072e7209
   test2 : i-07ece94b9f9e2743b7
   test1 : i-0d2fa2015395643d7
   ```

   **Figure 63: Migration factory server list**

5. Enter the AWS access key and secret access key.

62
**Cloud Migration Factory on AWS Implementation Guide**

**Terminate the test instances**

Your AWS access key will need to have Amazon EC2 read only permission in the target AWS account.

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 CloudEndure case.

---

**Terminate the test instances**

This activity terminates all the test instances in a given wave. After testing, you can manually terminate the instances from the Amazon EC2 console to avoid charges. However, it is more efficient to terminate all instances by using the single automation script.

Use the following procedure from the migration execution server to terminate the test instances.

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

   ```python
   python 3-Terminate-test-instance.py --Waveid <wave-id>
   ```

---
Replace `<wave-id>` with the appropriate Wave ID value to terminate the test instances. The script terminates the test instances for all CloudEndure projects.

3. Log in to the solution with your username and password.

4. Enter the CloudEndure API token.

   **Note**
   To find your CloudEndure API token, navigate to the CloudEndure console, choose **Setup & info**, choose **Other Settings**, then choose **API Token**. If the token does not exist, select **Generate token**.

The script returns a server list and a cleanup job for each CloudEndure project.

```
***************
Getting Server List and Replica Id=
***************
***** Servers for CE Project: Demo2 *****
test1
test2
test3
***** Servers for CE Project: Demo3 *****
Server1

****************************************
Terminating Test instances, ...
cleanup job created for the following machines in Project: Demo2
***** test1 *****
***** test2 *****
***** test3 *****
cleanup job created for the following machines in Project: Demo3
***** Server1 *****
```

**Figure 66: Server list and cleanup jobs**

# 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 execution 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` folder and run the following Python command.

   ```python
   python 4-Shutdown-all-servers.py --WaveId <wave-id>
   ```

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

3. Log in to the solution with your username and password.

   After you are logged in, the script first shuts down Windows servers in the specified wave.
Retrieve the target instance IPs

This activity retrieves the target instance IPs. 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 execution 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` folder and run the following Python command.

   ```python
   python 4-Get-instance-IP.py --WaveId <wave-id> --CloudEndureProjectName <project-name>
   ```

   Replace `<wave-id>` with the appropriate Wave ID value and `<project-name>` with the CloudEndure project name to get the new IP addresses for the target instances.
3. Log in to the solution with your username and password.
4. Enter the CloudEndure API token.

   **Note**
   To find your CloudEndure API token, navigate to the CloudEndure console, choose Setup & info, choose Other Settings, then choose API Token. If the token does not exist, select Generate token.

   The script returns a server list and the target instance ID information for the specified project.
5. Enter the AWS access key and secret access key for the target account.

![Figure 70: AWS access key and secret key prompt](image)

**Note**
Your AWS access key will need to have Amazon EC2 read-only permission in the target AWS account.

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).

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. If you have more than one CloudEndure project in the specified wave, you can repeat this activity for each project.

![Figure 71: CSV file showing instance_name and instance_IPs](image)

### 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...
Verify the target server connections

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 execution server to verify the connections for the target server.

1. Signed in as an administrator, open a command prompt (CMD.exe).
2. Navigate to the c:\migrations\scripts 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. Log in to the solution with your username and password.

The script retrieves the server list.

![Server list](image)

**Figure 72: Server list**

4. Enter the Linux sudo username and password for source servers.

The script returns the test results for both RDP and SSH access.
List of automated migration activities for AWS Application Migration Service (AWS MGN)

**Note**
For a list of automated migration activities for CloudEndure Migration, refer to the List of automated migration activities for CloudEndure Migration (p. 46) section in this guide.

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 execution server as a domain user with local administrator permission on the in-scope source servers.

**Important**
All the activities listed in this topic require you to be logged in as an administrator.

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

### Import the application and server data

To start the migration process, navigate to the 0-Migration-intake-form.csv file located in the c:\migrations\scripts folder. This file is used as the 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 you unzipped in Step 5 (p. 20) of the Automated deployment section.

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

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wave_id</td>
<td>Yes</td>
<td>The wave number which is based on priority and application server dependencies. Obtain this identifier from your migration plan.</td>
</tr>
<tr>
<td>app_name</td>
<td>Yes</td>
<td>The names of the applications that are in-scope for migration. Confirm that your application grouping includes all the applications sharing the same servers.</td>
</tr>
</tbody>
</table>
## Cloud Migration Factory on AWS Implementation Guide

### Import the application and server data

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aws_accountid</td>
<td>Yes</td>
<td>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 <strong>My Account</strong> from the drop-down menu.</td>
</tr>
<tr>
<td>aws_region</td>
<td>Yes</td>
<td>AWS Region code. For example, us-east-1. Refer to the full Region code list.</td>
</tr>
<tr>
<td>server_name</td>
<td>Yes</td>
<td>The name of the on-premises servers that are in-scope for migration.</td>
</tr>
<tr>
<td>server_os_family</td>
<td>Yes</td>
<td>The operating system (OS) that is running on the in-scope source servers. Use either <strong>windows</strong> or <strong>linux</strong> since this solution supports only these operating systems.</td>
</tr>
<tr>
<td>server_os_version</td>
<td>Yes</td>
<td>The version of the OS running on the in-scope source servers. <strong>Note:</strong> Use the OS version, not the Kernel version, for example, use RHEL 7.1, Windows Server 2012 R2, CentOS 7.5, 7.6. Do not use Linux 3.xx, 4.xx, or Windows 8.1.x.</td>
</tr>
<tr>
<td>server_fqdn</td>
<td>Yes</td>
<td>The source server's fully qualified domain name, which is the server name followed by the domain name. For example, server123.company.com.</td>
</tr>
<tr>
<td>server_tier</td>
<td>Yes</td>
<td>A label to identify whether the source server is a <strong>web</strong>, <strong>app</strong>, or a <strong>database</strong> server. We recommend designating the source server as <strong>app</strong> if the server functions as more than one tier, for example, if the server runs web, app, and database tiers together.</td>
</tr>
<tr>
<td>server_environment</td>
<td>Yes</td>
<td>A label to identify the server's environment. For example, <strong>dev</strong>, <strong>test</strong>, <strong>prod</strong>, <strong>QA</strong>, or <strong>pre-prod</strong>.</td>
</tr>
<tr>
<td>subnet_IDs</td>
<td>Yes</td>
<td>The subnet ID for the target Amazon EC2 instance for the migration post-cutover.</td>
</tr>
</tbody>
</table>
Cloud Migration Factory on AWS Implementation Guide

Import the application and server data

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Required?</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>securitygroup_IDs</td>
<td>Yes</td>
<td>The security group ID for the target Amazon EC2 instance for the migration post-cutover.</td>
</tr>
<tr>
<td>subnet_IDs_test</td>
<td>Yes</td>
<td>The target subnet ID for the source server that will be tested.</td>
</tr>
<tr>
<td>securitygroup_IDs_test</td>
<td>Yes</td>
<td>The target security group ID for the source server that will be tested.</td>
</tr>
<tr>
<td>instanceType</td>
<td>Yes</td>
<td>The Amazon EC2 instance type identified in the discovery and planning effort. For information about EC2 instance types, refer to Amazon EC2 Instance Types.</td>
</tr>
<tr>
<td>tenancy</td>
<td>Yes</td>
<td>The tenancy type, which is identified during the discovery and planning efforts. Use one of the following values to identify the tenancy: Shared, Dedicated, or Dedicated Host. You can use Shared as the default value unless an application’s license requires a specified type.</td>
</tr>
<tr>
<td>private_ip</td>
<td>No</td>
<td>The private IP for the target instance. If not included, instance will get an IP from DHCP.</td>
</tr>
<tr>
<td>iamRole</td>
<td>No</td>
<td>IAM role for the target instance. If not included, no IAM role will be attached to the target instance.</td>
</tr>
</tbody>
</table>

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

   ```python
   python 0-Import-intake-form.py --Intakeform "<file-path>"
   ```

3. Replace `<file-path>` with the correct intake form location, for example, C:\migrations\scripts \0-Migration-intake-form.csv.
4. The script will first connects to the Migration Factory API to get server data.

   The script loads the CSV data and then creates the waves in the solution.
After the waves are created, the script automatically creates the applications and servers based on the information from the CSV file.
Import the application and server data

Figure 74: Migration factory creating the applications and servers
Check the 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 check, the first agent must be installed 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 execution 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 folder and run the following Python command.

```python
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.

![Figure 75: Search for replication server in Amazon EC2](image)

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

Note

To use an alternative credential for Windows server other than the current login user, use the `--WindowsUser` argument. For example, `--WindowsUser mydomain\userA`.
Check the prerequisites

Figure 76: Retrieve 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.

Figure 77: Windows servers prerequisite checks

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\0-Prerequisites-Windows.ps1
```

Figure 78: Security warning
Next, the script checks the Linux servers.

```
*Checking Pre-requisites for Linux servers*

-- Linux Server result for MF-RHEL --

SSH 22 to source server : Pass
SUDO permission : Pass
TCP 443 to MGN Endpoint : Pass
TCP 1500 to Rep Server : Pass
2.0 GB / FreeSpace : Pass
0.5 GB /tmp FreeSpace : Pass
DHCLIENT Package : Pass

-- Linux Server result for MF-Ubuntu --

SSH 22 to source server : Pass
SUDO permission : Pass
TCP 443 to MGN Endpoint : Pass
TCP 1500 to Rep Server : Pass
2.0 GB / FreeSpace : Pass
0.5 GB /tmp FreeSpace : Pass
DHCLIENT Package : Pass
```

Figure 79: Linux servers prerequisite checks

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

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. Use the following procedure to check the status details.

1. Log in to the Migration Factory web interface and select Resource List from the drop-down menu on the upper-right corner of the page.
2. Choose the Server List tab and enter the wave name in the search field. In the following screenshot of an example project, we searched for Wave 1. When the check is successful, the migration_status will show Pre-requisites check: Passed as shown in the following screenshot of an example project.

![Figure 80: Prerequisite check results](image)

![Figure 81: Prerequisites check results](image)
Install the replication agents

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

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

```python
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, to use an alternative credential for Windows server other than the current login user, use the `--WindowsUser` argument. For example, `--WindowsUser mydomain\userA`.

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.

```
*** Getting Server List ****

### Servers in Target Account: 51500000030, region: us-east-1 ###
MF-RHEL.mydomain.local
Server-T1.mydomain.local
server1.mydomain.local

### Servers in Target Account: 11450000000, region: us-east-2 ###
MF-Ubuntu.mydomain.local
Server-T15.mydomain.local
server2.mydomain.local
```

*Figure 82: Migration factory server list*

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.

```
* Enter Linux Sudo username and password *

Username: stackadmin
Password: stack123
```

*Figure 83: Migration factory Linux login*

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

*** Installing Agents ****

---------------------------
### In Account: 5158-..., 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: e-3fe3e53422624e6a0.
The AWS Replication Agent was successfully installed.
The installation of the AWS Replication Agent has started.
** Installation finished for : Server-T1.mydomain.local **

Figure 84: 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\1-Install-Windows.ps1

Security warning
Run only scripts that you trust. While scripts from the internet can be useful, this script can potentially harm your
computer. If you trust this script, use the Unblock-File cmdlet to allow the script to run without this warning
message. Do you want to run C:\migrations\scripts\1-Install-Windows.ps1?
[O] Do not run  [R] Run once  [S] Suspend  [?] Help (default is "D"): Traceback (most recent call last):

Figure 85: Security warning

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 86: Check agent install results

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

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 execution server to push the post-launch scripts to Windows machines.

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

   ```python
   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`. 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.
   
   To use an alternative credential for Windows server other than the current login user, use the `--WindowsUser` argument. For example, `--WindowsUser mydomain\userA`

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.
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 execution server to verify the CloudEndure replication status.

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

   ```python
   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.

---

Figure 88: Migration factory server list

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.

Figure 89: Copy files to post_launch folder
Verify the replication status

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.

```
*** Getting Server List ***
*****************************
### Servers in Target Account: 515****** , region: us-east-1 ###
  MF-RHEL.mydomain.local
  Server-T1.mydomain.local
  server1.mydomain.local
### Servers in Target Account: 114******* , region: us-east-2 ###
  MF-Ubuntu.mydomain.local
  Server-T15.mydomain.local
  server2.mydomain.local
```

**Figure 90: Migration factory server list**

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

1. Log in to the **Migration Factory** web interface and select **Resource List** on the upper-right corner of the page.
2. Choose the **Server List** tab and enter the wave name in the search field.
3. Check the **replication_status** column.

```
* Verify replication status *
*****************************
Migration Factory : You have successfully logged in

### Replication Status for Account: 515****** , region: us-east-1 ###
Server Server-T1 replication status: Healthy
Server server1 replication status: Healthy

### Replication Status for Account: 114******* , region: us-east-2 ###
Server MF-Ubuntu replication status: Healthy
Server Server-T15 replication status: Healthy
Server server2 replication status: Healthy
```

**Figure 91: Replication status for servers**
Create the admin user

This activity creates a local admin/sudo user on source machines. This user may be needed to troubleshoot issues that may occur after migration cutover from the in-scope source servers to AWS. The migration team will use this user to log in to the target server when the authentication server (for example, the DC or LDAP server) is not reachable.

**Note**
This activity is not required if you have a local admin/sudo user and log in credentials for all source servers.

Use the following procedure to create a local admin user.

1. Use a remote desktop protocol (RDP) client to log in to the migration execution server with domain credentials. This user needs admin permissions to all source Windows machines.
2. Signed in as an administrator, open a command prompt (CMD.exe).
3. Navigate to the `c:\migrations\scripts` folder and run the following Python command to create a local user and add the user to the local administrator group.

   ```
   python 2-UserMgmt.py --Waveid <wave-id>
   ```

   Replace `<wave-id>` with the appropriate Wave ID value to create a local admin user for all the Windows servers in that wave.

**Note**
To use an alternative credential for Windows server other than the current login user, use the `--WindowsUser` argument. For example, `--WindowsUser mydomain\userA`

4. The script generates a list identifying the servers included for the specified wave.
5. Select target OS: Windows or Linux and select option 1 to create an user.
6. Enter a new local admin username and password. A local admin/sudo user for all source servers (Either Windows or Linux) in the specified wave is created.

Note
The migration team will use this user to log in to the target server when the authentication server (for example, the DC or LDAP server) is not reachable, for troubleshooting purposes.

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. Log in to the Migration Factory web interface.
2. Select Tools from the drop-down menu on the upper-right corner of the page.
3. Choose the Application Migration Service tab.
4. On the Application Migration configuration page, take the following actions and choose Submit
   • In the Wave Id field, select the target Wave Id.
In the AWS Account ID field, select either a specific account Id or All Accounts.
For Test and Cutover, select Validate Launch Template.

When validation is successful, you will receive the following message: SUCCESS: Launch templates validated for all servers in this wave.

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 (AWS MGN) in test mode.

Use the following procedure to launch test instances.

1. Log in to the Migration Factory web interface.
2. Select Tools from the drop-down menu on the upper-right corner of the page.
3. Choose the Application Migration Service tab.
4. On the Application Migration configuration page, take the following actions and choose Submit.
   - In the Wave Id field, select the target Wave Id.
   - In the AWS Account ID field, select either a specific account Id or All Accounts.
   - For Test and Cutover, select Launch Test Instances.

When launch is successful, you will receive the following message: SUCCESS: Launch Test instances was completed for all servers in this wave.

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 from the migration execution 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 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 Id if exist.

Figure 99: Migration factory server list

4. The script will then return a list of target instance Id if exist.

Figure 100: Target Instance Id

**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.
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. Log in to the Migration Factory web interface.
2. Select Tools from the drop-down menu on the upper-right corner of the page.
3. Choose the Application Migration Service tab.
4. On the Application Migration configuration page, take the following actions and choose Submit
   - In the Wave Id field, select the target Wave Id.
   - In the AWS Account Id field, select either a specific account Id or All Accounts.
   - For Test and Cutover, select Mark As Ready for Cutover.

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.

Figure 101: Instance status check results

*Verify instance status*

```
* Verify instance status *
*******************************************************************************
### In Account: 515256.02.0 , region: us-east-1  
*******************************************************************************
- The following instances PASSED 2/2 status checks -
  Server-T1.mydomain.local

- WARNING: the following instances FAILED 2/2 status checks ----
  server1.mydomain.local

*******************************************************************************
### In Account: 114.----.----.0 , region: us-east-2  
*******************************************************************************
- The following instances PASSED 2/2 status checks -
  MF-Ubuntu.mydomain.local
  Server-T15.mydomain.local
  server2.mydomain.local
```
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 execution 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 folder and run the following Python command.

   python 4-Shutdown-all-servers.py --Waveid <wave-id>

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

**Note**
To use an alternative credential for Windows server other than the current login user, use the --WindowsUser argument. For example, --WindowsUser mydomain\userA.

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

   ```
   # Getting Server List#
   Servers in Target Account: 515/588903, region: us-east-1
   server1.mydomain.local
   server2.mydomain.local
   
   Servers in Target Account: 114/23453, region: us-east-2
   M1.16xlarge.mydomain.local
   server1.mydomain.local
   server2.mydomain.local
   ```

   **Figure 102: Migration factory server list**

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.
Launch instances for Cutover

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

Use the following procedure to launch test instances.

1. Log in to the Migration Factory web interface.
2. Select Tools from the drop-down menu on the upper-right corner of the page.
3. Choose the Application Migration Service tab.
4. On the Application Migration configuration page, take the following actions and choose Submit.
   - In the Wave Id field, select the target Wave Id.
   - In the AWS Account ID field, select either a specific account Id or All Accounts.
   - For Test and Cutover, select Launch Cutover Instances.

When launch is successful, you will receive the following message: SUCCESS: Launch Cutover instances was completed for all servers in this wave.

Note
This action will also update the migration_status for the server launched.
Retrieve the target instance IPs

This activity retrieves the target instance IPs. 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 execution 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` folder and run the following Python command.

   ```bash
   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.
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 execution server to verify the connections for the target server.

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

```python
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.

```
*** Getting Server List ****

## Servers in Target Account: 5155macht, region: us-east-1
Server-T1.mydomain.local
server1.mydomain.local

## Servers in Target Account: 1147macht, region: us-east-2
MF-Ubuntu.mydomain.local
Server-T15.mydomain.local
server2.mydomain.local
```

*Figure 108: Migration factory Server List*

4. The script returns the test results for both RDP and SSH access.

```
*Checking RDP Access for Windows servers*
RDP test to Server Server-T1.mydomain.local : Pass
RDP test to Server server1.mydomain.local : Pass
RDP test to Server Server-T15.mydomain.local : Pass
RDP test to Server server2.mydomain.local : Pass

*Checking SSH connections for Linux servers*
SSH test to server MF-Ubuntu.mydomain.local : Pass
```

*Figure 109: Verify connection result*
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. 13) 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

This solution is configured to retain the solution-created Amazon S3 bucket (for deploying in an opt-in Region) if you decide to delete the AWS CloudFormation stack 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 CloudEndure with this solution
- **Region**: The AWS Region where the solution is deployed

Note that 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'
   ```

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. 13) 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

<table>
<thead>
<tr>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 2020</td>
<td>Initial release</td>
</tr>
<tr>
<td>February 2021</td>
<td>Release v1.1.0: Added optional migration tracker component; for more information on new features, refer to the CHANGELOG.md file in the GitHub repository.</td>
</tr>
<tr>
<td>May 2021</td>
<td>Release v1.1.1: Updated AWS Lambda functions to support Python v3.7; for more information, refer to the CHANGELOG.md file in the GitHub repository.</td>
</tr>
<tr>
<td>August 2021</td>
<td>Release v2.0.0: New feature to integrate with AWS Application Migration Service (AWS MGN); for more information, refer to the CHANGELOG.md file in the GitHub repository.</td>
</tr>
<tr>
<td>September 2021</td>
<td>Release v2.0.1: Bug fixes; for more information, refer to the CHANGELOG.md file in the GitHub repository.</td>
</tr>
<tr>
<td>October 2021</td>
<td>Release v2.0.2: Bug fixes; for more information, refer to the CHANGELOG.md file in the GitHub repository.</td>
</tr>
</tbody>
</table>
Contributors

The following individuals contributed to this document:

- Wally Lu
- Dev Kar
- Lyka Segura
- Chris Baker
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.
For the latest AWS terminology, see the AWS glossary in the AWS General Reference.