aws

User Guide

AWS Transform

AWS Transform: User Guide

Table of Contents

What is AWS Transform?

AWS Transform is a service that helps you accelerate and simplify the transformation of infrastructure, applications, and code with an agentic AI experience. It provides specialized tools to streamline modernization and migration efforts across various workloads.

With AWS Transform, you can:

- Modernize and migrate IBM z/OS and Fujitsu GS21 to AWS
- Migrate VMware workloads to Amazon EC2
- Modernize .NET applications to Linux-ready cross-platform .NET
- Assess workloads for migration readiness

AWS Transform helps you offload labor-intensive, complex transformation tasks across the discovery, planning, and execution phases to AI agents with expertise in languages, frameworks, and infrastructure. This approach allows your teams to focus on innovation while it efficiently transforms complex, large-scale projects through a unified web experience.

There are no additional charge to use AWS Transform.

Terminology

Within this section, italics indicate an official term within the definition of a different term.

Account connection

Account in this context is a generalized reference to a customer-owned container or security boundary for resources in AWS or remote service, for example, an AWS account or GitHub account.

Artifact

An output deliverable produced by AWS Transform.

Administrator

Administrators can read and mutate everything in the workspace. They can begin chats with AWS Transform, start and stop jobs, and upload/download artifacts. Administrators can interact

with running jobs for human-in-the-loop (HITL) actions, and can approve critical HITL actions such as merging to main, performing graph decomposition, or deploying code to production environments. Administrators can mutate workspaces, connectors, and users.

Agent

A task-specific service that executes a specific transformation type. For example, VMware migrations.

Approver

Approvers have permissions similar to Administrators except that they do not have permissions to mutate workspaces, connectors, or users. Approvers cannot mutate workspaces, connectors, or users.

Asset

Input for a transformation job. For example, customer's source code, server, database, network. Assets are accessed via a connector.

Collaborator request

A *task* in which AWS Transform is asking a human to do something.

Connector

Connectors are asset providers that allow access to customer-owned resources in a system external to AWS Transform.

When you set up a connector, the administrator of the account to which you are connecting must accept the connection. In order to accept the connection, they must have permissions given in the connector acceptor policy.

The following two accounts must either be identical, or in the same AWS Organizations organization:

- The account from which the AWS Transform administrator enables the service.
- The account that will be on the receiving end of your transformation. This account must be assigned an IAM role that allows it to use a connector.

Contributor

Contributors can read everything in the workspace. They can begin chats with AWS Transform, start and stop jobs, upload or download artifacts, and interact with running jobs for HITL actions. However, they cannot perform critical HITL actions such as merging to main,

performing graph decomposition, or deploying code to production environments. Contributors also cannot mutate workspaces, connectors, or users.

Objective

A user-defined end state that AWS Transform works to reach. Users write this, and then AWS Transform converts the objective into a series of tasks that it perform in concert with users when required.

Job

A long-running process (weeks/months+) that AWS Transform is working on in order to fulfill an objective defined by a user. Made up of multiple tasks and collaborator requests.

Plan

A list of tasks that AWS Transform undertakes (with help from human users) in pursuit of an objective.

Reader

Readers can view the status and outcomes of AWS Transform jobs, but cannot make any changes. They can read everything in the workspace, download artifacts, view jobs, and view human-in-the-loop (HITL) actions. However, readers cannot perform mutating actions in AWS Transform.

Task

An individual unit of work that is part of a job.

Worklog

A log of what actions AWS Transform and users have performed as part of a job.

Workspace

A AWS Transform resource that contains other resources like connectors and jobs. A workspace serves as a permissions boundary.

Getting started with AWS Transform

Topics

Setting up AWS Transform

Before you begin

Before you set up AWS Transform make sure you have an AWS account with administrator access

> **Note**
>
> If you want to try out AWS Transform as a proof-of-concept or for test environments see Quick start: Trying AWS Transform.

Getting started with AWS Organizations

Follow these steps to set up AWS Transform:

1. Sign in to your AWS Organizations management account.
2. Navigate to the AWS Transform service.
3. Choose **Enable service** for your organization to use AWS Transform.
4. Configure the necessary permissions for organizational member accounts.
5. Access the AWS Transform web experience from your member accounts.

> **Note**
>
> To use the Landing Zone Accelerator (LZA) on AWS solution to build your landing zone together with AWS Transform for migration capabilities, your AWS Transform account and LZA installation must be in the same AWS Organization. Using separate Organizations IDs for LZA and AWS Transform deployments is not supported because this can cause inconsistencies in organizational management and resource deployments. To learn how to set up your LZA installation using Organizations see Deploy a cloud foundation to support highly-regulated workloads and complex compliance requirements in the *Landing Zone Accelerator on AWS Implementation Guide user guide*.

Getting started with AWS IAM Identity Center

Follow these steps to use IAM Identity Center for AWS Transform and to add users and groups.

By default, no users have access to AWS Transform when you first enable it.

> **Note**
>
> IAM Identity Center is not limited to the region in which it is set up. If you already set up IAM Identity Center in a region that is not supported by AWS Transform, you can use it for AWS Transform.

1. Set up IAM Identity Center following the instructions in To enable an instance of IAM Identity Center.

   Configure IAM Identity Center to use an external enterprise identity provider, and replicate its user and group info into IAM Identity Center.
2. In the AWS console, select AWS Transform and choose **Get started**.
3. Choose **Enable service** for your organization to use AWS Transform.
4. Select an encryption key. The default selection is an AWS managed key. To use a custom key:
   a. Under **Encryption key**, choose **Customize encryption settings**.
   b. Select **Use an AWS KMS key**.
   c. Choose an existing key or create a new one.

d. Choose **Submit** to apply your changes, and then choose **Enable AWS Transform**.

Click **View profile** to view the configuration. The Web application URL is used by your users to access the AWS Transform unified web experience.

5. Select **Users** in the navigation pane and select **Assign users or groups**.
6. Search for the name of the user or groups you want to authorize to use AWS Transform. The search references users and groups propagated from your identity provider.
7. Select a group or user, select **Done**, and then, **Assign**. These users are authorized to use the AWS Transform unified web interface.

Using third-party identity providers

AWS Transform supports integration with third-party identity providers (IdPs) such as Azure Active Directory (Entra ID) and Okta Workforce Identity. This allows you to use your existing identity management system for user authentication.

Prerequisites

Before configuring third-party identity provider integration, ensure that users in your identity provider have name, email, and username attributes configured

Stored Information

When you use AWS Transform with IdPs, AWS stores minimal user information that is encrypted and secured:

Stored User Information

AWS Transform stores basic user profile information upon first login, including display name, email address, username (preferred_username), and a unique user identifier. This information is encrypted using either a customer-owned KMS key or a service-owned key, depending on the customer's AWS Transform profile configuration. The data is stored in AWS Transform's authentication database and is only collected during the initial login session. This populates the search results when inviting other users to a workspace.

Data Lifecycle

User information is stored only for users who have logged in to the AWS Transform web app at least once, and may become stale if users update their information in their identity provider

without logging back into AWS Transform. All stored user information is deleted when the AWS Transform profile is deleted.

Client Secret Storage

The client secret provided during setup is stored using AWS Secrets Manager via a Service Linked Secret (SLS) in your account.

User Identifier Handling

Entra

Uses the "oid" (object identifier) claim as the unique user identifier, which is immutable and uniquely identifies users across the Microsoft tenant. This value is visible to customers in the Entra console and appears in CloudTrail logs.

Okta Workforce Identity

Uses different claims for user identification depending on token type - the "sub" claim in ID tokens and the "uid" claim in Access tokens. AWS Transform validates that both claims contain the same value during authentication. This value is visible to customers in the Okta console and appears in CloudTrail logs.

Setting up Azure Active Directory (Entra ID)

To configure Azure Active Directory integration with AWS Transform:

1. Navigate to the Azure portal and select **Azure Active Directory**.
2. In the left navigation pane, choose **Manage** > **App registrations**.
3. Choose **+ New registration**.
4. Enter an application name, choose your supported account type, leave the redirect URI blank, and choose **Register**.
5. In the left navigation, choose **Manage** > **Manifest**.
6. Update `requestedAccessTokenVersion` from `null` to `2` and choose **Save**.
7. Choose **Manage** > **Expose an API** and choose **Add a scope**.
8. Create an Application ID URI using the default structure `api://<client-id>`.
9. Add the scope `transform:read_write`.

10. Choose **Add a certificate or secret** and create a new client secret. Save this value as it's needed for profile creation.
11. Find the Issuer URL by choosing **Endpoints** and selecting the OpenID Connect metadata document. The "issuer" field in the metadata is your Issuer URL.
12. Create a profile in the AWS Transform console using the Client ID, Client Secret, and Issuer URL.
13. After profile creation, add a redirect URI by choosing **Add a platform**, selecting **Web**, and entering `<web-application-url>/login/callback`.

Setting up Okta Workforce Identity

To configure Okta Workforce Identity integration with AWS Transform:

1. Navigate to your Okta Workforce Identity console.
2. Choose **Applications** > **Applications** and select **Create App Integration**.
3. Select **OIDC - OpenID Connect** and **Web Application**, then choose **Next**.
4. Name your application, leave the Grant Type as *Authorization Code*, leave redirect URIs blank, configure user assignments, and choose **Save**.
5. Navigate to the **Sign On** tab and set the Issuer to **Okta URL** instead of Dynamic.
6. Copy the Client ID and configure it as the Audience for your Authorization Server by going to **Security** > **API** and adding an Authorization Server.
7. In the Authorization Server, add the scope `transform:read_write` under the **Scopes** tab.
8. Add an Access Policy that allows the OIDC Application to use this Authorization Server and configure a rule for the policy.
9. On the Authorization Server Settings page, note the Issuer URL for profile creation in AWS Transform.
10. Create a profile in AWS Transform using the Issuer URL, Client ID, and Client Secret from the application settings.
11. After profile creation, add `<web-application-url>/login/callback` as a redirect URL in the application's General tab.

> **Note**
>
> If you would like to be redirected back to the AWS Transform webapp after logout, you'll need to configure your web application URL as a trusted origin under **Security** > **API**.

User onboarding

This section describes the experience for users who have been granted access to AWS Transform.

Accepting the invitation

When a user is added to AWS Transform, they receive an email invitation containing:

- A greeting and information about the invitation
- The AWS Transform web application URL
- Their username
- A link to accept the invitation and set up their password

To set up their account:

1. The user clicks the "Accept invitation" link in the email.
2. On the "New user sign up" page, they enter and confirm a password.
3. The password must meet security requirements, including:
    - At least 8 characters
    - At least one uppercase letter
    - At least one lowercase letter
    - At least one number
    - At least one special character
4. After creating a password, they see a confirmation that their account was successfully created.

Signing in to AWS Transform

To sign in to AWS Transform:

1. Navigate to the AWS Transform web application URL provided in the invitation email.
2. Enter the username.
3. Choose **Next**.
4. Enter the password.
5. Choose **Sign in**.

Welcome experience

Upon first login, users see the AWS Transform welcome page with:

- A personalized greeting
- Available transformation capabilities
- Option to create a workspace

The welcome page provides information about the transformation capabilities available in AWS Transform, including:

- Modernize IBM z/OS migrations to AWS
- Migrate VMware workloads to Amazon EC2
- Modernize .NET applications to Linux-ready cross-platform .NET
- Assess workloads for migration readiness

Users can start by creating a workspace or asking their team to add them to an existing workspace.

Enable AWS Transform

To enable AWS Transform:

1. Sign in to the AWS Management Console.

2. In the search bar at the top of the console, search for *AWS Transform*.

3. Select **AWS Transform** from the search results.
4. Choose **Get started** to enable the service in your current Region.
5. Optional: configure IAM Identity Center. You will also be able to choose to use a third-party identity provider (IdP) in a later step.
6. Select an **Encryption key**: **default AWS managed key** or **Customize encryption settings.**
7. Choose which AWS Transform capabilities you want to enable:
   - **Command line interface** (CLI), needed to create and run custom transformations. To enable the CLI, view and follow the download instructions.
   - **Web application**, the agentic user interface for modernization. Choose **Enable web application** to use it.
8. Choose **Enable AWS Transform**.
9. Optional: choose **Enable View profile** to access the AWS Transform **Users**, **Settings**, and **Connectors** tabs, or **Manage users** to manage users.

   You can access the **Users**, **Settings**, and **Connectors** tabs at any time by choosing the menu icon in the top left corner of the console.
10. Configure User access by choosing an identity provider, either IAM Identity Center or a third-party identity provider (IdP).

    > **Note**
    >
    > This choice is finalized and cannot be changed when you enable AWS Transform.

11. Choose **Enable web application**.
12. The system displays "Enabling AWS Transform" while it creates the necessary resources.

After AWS Transform is enabled, the **Settings** tab displays the following information:

- **Web application URL** - The URL for accessing the AWS Transform web application
- **Start URL for IDE** - The URL for accessing AWS Transform in integrated development environments
- **Region** - The AWS Region where AWS Transform is enabled

Quick start: Trying AWS Transform

The easiest way to try out AWS Transform is with a standalone AWS account. You may want to do this as a proof-of-concept or for test environments. Use this procedure:

1. Sign in to the AWS Management Console.
2. Navigate to the AWS Transform service.
3. Choose **Get started** to enable the service.
4. Select and set up your identity provider.
5. Assign users to AWS Transform service.
6. After the service is enabled, you'll see the AWS Transform web application URL.
7. Open that URL in a new browser window to access the AWS Transform web experience.

Now you're ready to set up your workspace.

Managing users

AWS Transform integrates with IAM Identity Center for user management. This section describes how to add users to IAM Identity Center and grant them access to AWS Transform.

Adding users in IAM Identity Center

To add users in IAM Identity Center:

1. Navigate to the IAM Identity Center console.
2. In the navigation pane, choose **Users**.
3. Choose **Add user**.
4. Enter the required information:
   - **Username** - A unique identifier for the user (cannot be changed later)
   - **Email address** - The user's email address
   - **First name** and **Last name** - The user's name
   - **Display name** - The name that appears in the user list
5. For **Password**, choose how the user receives their password:
   - **Send an email** - Send setup instructions via email

- **Generate a one-time password** - Create a password to share manually

6. Choose **Next** to review the user information.
7. Review the details and choose **Add user**.

After the user is added, they'll receive an email invitation to set up their IAM Identity Center account. The invitation link is valid for 7 days.

You can also learn about working with IAM Identity Center and AWS Transform in this video:

VideoTitle

Adding users to AWS Transform

After adding users to IAM Identity Center, you can grant them access to AWS Transform:

1. Return to the AWS Transform console.
2. In the navigation pane, choose **Users and groups**.
3. Select the **Users** tab or the **Groups** tab.
4. Search for and select the users or groups that you want to add from IAM Identity Center.
5. Choose **Assign users and groups** to grant the selected users or groups access to AWS Transform.

After adding users, they appear in the **Users** list with a status of "Pending" until they accept the invitation and sign in.

Understanding collaborator permissions

AWS Transform uses a workspace-based permission model to control access to resources and actions. Each user is assigned a specific role within a workspace, which determines what actions they can perform. A user can have different roles in different workspaces.

User roles

AWS Transform supports five user roles within each workspace. These roles apply within the context of a workspace, and a user will be assigned roles in each workspace they are a member of. The access permissions defined for each role are workspace agnostic, so user A with the Administrator role in workspace A has the same permissions as user B with the Administrator role in workspace B.

Role permissions

Detailed permissions for each role:

Action	ResourceType	Admin	Approver	Contributor	ReadOnly
Create	Workspace	✓	✓	✓	✓
List	Workspace	✓	✓	✓	✓
Get	Workspace	✓	✓	✓	✓
Update	Workspace	✓	✗	✗	✗
Delete	Workspace	✓	✗	✗	✗
Create	ChatMessage	✓	✓	✓	✗
Read	ChatMessage	✓	✓	✓	✓
Create	RoleAssociation	✓	✗	✗	✗
Read	RoleAssociation	✓	✓	✓	✓
Update	RoleAssociation	✓	✗	✗	✗
Delete	RoleAssociation	✓	✗	✗	✗
Read	CriticalHITLTask	✓	✓	✓	✓
Update	CriticalHITLTask	✓	✓	✗	✗
Delete	CriticalHITLTask	✓	✓	✗	✗
Read	HITLTask	✓	✓	✓	✓
Update	HITLTask	✓	✓	✓	✗
Delete	HITLTask	✓	✓	✓	✗
Create	Job	✓	✓	✓	✗

Action	ResourceType	Admin	Approver	Contributor	ReadOnly
Read	Job	✓	✓	✓	✓
Update	Job	✓	✓	✓	✗
Delete	Job	✓	✓	✓	✗
Read	Worklog	✓	✓	✓	✓
Create	Artifact	✓	✓	✓	✗
Read	Artifact	✓	✓	✓	✓
Update	Artifact	✓	✓	✓	✗
Delete	Artifact	✓	✓	✓	✗
Create	Connector	✓	✓	✓	✗
Read	Connector	✓	✓	✓	✓
Update	Connector	✓	✓	✓	✗
Delete	Connector	✓	✓	✓	✗

Human-in-the-loop (HITL) actions

AWS Transform provides two types of HITL actions - standard and critical:

Standard HITL actions

These are routine actions that can be performed by users with Contributor, Approver, or Administrator roles.

Critical HITL actions

These are actions with significant impact, and thus require higher permission levels. Examples include:

- Merging code to main branches

- Performing graph decomposition
- Deploying code to production environments

Critical HITL actions can only be performed by users with Approver or Administrator roles.

To ensure there's a differentiation between Standard HITL and Critical HITL actions in AuthZ policies, AWS Transform provides two separate HITL APIs, one for completing a standard HITL action, and one for completing a critical HITL action.

AWS Transform environment

AWS Transform is an agentic service that uses natural language processing to help you plan and execute your workload transformations. You interact with AWS Transform primarily through a conversational interface, where the service adapts and responds based on the context of your discussion. For example, you can start by describing your current architecture and transformation goals using everyday language, such as "I need to migrate my on-premises VMs to EC2."

As you converse with AWS Transform, the service builds a customized job plan that aligns with your specific requirements. The conversation flow is dynamic and driven by you, allowing you to refine your transformation plan iteratively. You can modify the job plan during the conversation by making requests like "add a testing phase before production cutover" or "remove the backup step since we already have a solution." You can also go back to a previous task and perform it again. AWS Transform continuously updates the plan based on your input, ensuring the final transformation strategy meets your technical and business needs.

Here's what you see when you open AWS Transform.

- View control pane: This is the narrow pane on the left of AWS Transform. You can choose one of the icons to choose what view is shown to the right of the view control pane. Hover over each icon for a tool tip explaining the view. The give standard views, from top to bottom, are:
  - **Job Plan**
  - **Dashboard**
  - **Approvals**
  - **Artifacts**
  - **Worklog**

  Some workflows provide additional views.

When you are working in a job plan, the views are:

- **Chat**
- **Approvals**
- **Artifact store**
- **Worklog**

When you are in a workspace, the views are at the top:

- **Jobs**
- **Artifacts**
- **Collaborators (Users)**
- **Connectors**
- **Settings**

- The **View** pane is next to the view controls.
- The **Chat** pane is in the center. This is where you conduct your conversation with AWS Transform.

> **Note**
>
> Users with read-only permissions are unable to send messages in the chat.

- To the right is the **Collaboration** pane. This appears when *human in the loop* (HITL) activities are performed, such as:
  - Uploading data files
  - Reviewing information and plans provided by AWS Transform

Start your project

AWS Transform guides you through your modernization and migration projects. To get started:

1. Create a workspace to host your project. On the **Workspaces** page, choose **Create workspace**. Follow the instructions and then open the workspace.
2. In the **Chat** pane, type "create a job"
3. Choose a job, and follow the instructions provide by AWS Transform. If your input is required, the **Collaboration** pane appears and explains what is required.

Workflow flexibility

The AWS Transform workflow environment provides flexibility in the progression of your modernization projects. Using natural language in the chat pane you can:

- Retry a task that AWS Transform has already completed, providing different human-in-the-loop input along the way.
- Rerun a job, for example, if there have been changes in your modernization sources.

Chatting with AWS Transform

AWS Transform chat is available at every stage of your project. To open the chat click the purple hexagonal icon in the lower right corner of the web console.

The chat is there for you to ask anything. For example, you can ask the chat to explain concepts, guide you through a process, explain a AWS Transform request or response, or explain a AWS Transform report.

> **Note**
>
> Users with read-only permissions are unable to send messages in the chat.

AWS Transform chat integrations

AWS Transform chat is integrated with:

Experience-Based Acceleration

Experience-Based Acceleration (EBA) is offered through AWS Transform chat. It enables you to perform EBA assessment for Windows workloads and generate a plan. You can start by importing assessment results from CAST. It helps you discover your application portfolio, after which you can use AWS Transform chat to select applications that meet your business needs (for example, filtering applications based on complexity or lines of code). You can then perform deeper assessment of the selected application and generate a modernization plan.

AWS Countdown Premium

AWS Countdown Premium (CDP) integrated into AWS Transform provides sustained, expert guidance with designated AWS engineering support. Your designated CDP Engineer dives deep

into your tech stack, providing personalized, context-aware support when issues arise. Users can leverage AWS Transform to log a support ticket directly from its chat interface. The support ticket embeds worklog and job plan details to help support understand the customer's context. If a customer or partner is part of the CDP Program, the support ticket automatically gets routed to the designated CDP Engineer who can help debug issues, interpret transformation outputs, and drive progress for expedited issue resolution. For example:

- For .NET workloads, CDP can assist with repository connector issues, dependency resolution, and post-transformation deployment.
- In mainframe scenarios, CDP can help troubleshoot refactored Java code, configure database migration tools, and build CI/CD pipelines.
- In VMware migrations, CDP can accelerate network configuration, Application Migration Service setup, and agent installation.

AWS Skill Builder

AWS Skill Builder provides relevant learning modules through the chat. You can ask AWS Transform chat about your learning needs, and it presents you relevant course catalog. Skill Builder delivers contextual micro-learning experiences as you work through transformation stages.

Migration assessment

The assessment job type in AWS Transform is designed to provide cost estimates and savings for migrating your workloads to AWS.

This job type uses an advanced AI-powered language model specifically trained for AWS infrastructure analysis. The specialized LLM evaluates Amazon EC2 instance recommendations, performs BYOL (Bring Your Own License) analysis, and determines optimal dedicated host mappings by incorporating real-time Amazon EC2 pricing data and host configuration specifications.

The assessment job type currently supports compute-based workloads. While it includes servers deployed to run specialized workloads, it cannot assess the specialized workloads themselves.

When planning to migrate to AWS you can use this assessment to:

- Get cost estimates, including Amazon EC2 and Amazon EBS and storage cost estimates, for your migration
- Identify potential savings opportunities
- Receive Amazon EC2 instance recommendations
- Analyze licensing options (BYOL)
- Determine optimal dedicated host mappings

Creating and starting a job

The first step of a migration project is to create an AWS Transform job.

To create and start a new migration assessment job

1. On your workspace landing page, choose **Create a job with AWS Transform**.
2. Choose the migration assessment option.
3. Review the job details that AWS Transform proposes. You can specify a different name for the job if you'd like.
4. Choose **Create and start a job**.

Migration assessment workflow

1. In the left navigation pane, choose **Share on-premises server data**.
2. Upload data files that can be used by AWS Transform for the assessment. Make sure the file includes all the servers that you want to assess for migration to AWS. You can include 30,000 servers per assessment job. The maximum supported file size is 10 MB. These types of files are supported:

   - The AWS Transform discovery tool enables you to automatically discover server inventory in your organization in preparation for migration. When you configure OS access the discovery tool can help you obtain database assessment and assist in application dependency mapping and wave planning.
   - RVTools: You can upload either a ZIP of .csv files or an excel file that RVTools produces when you choose **Export all to Excel** from the RVTools **File** menu.
   - Migration Portfolio Assessment (MPA) import file
   - Export for vCenter
   - Migration Evaluator Quick Insights. You can download the **Migration Evaluator Quick Insights** file from the Migration Evaluator Console.
   - A Microsoft Excel file created from the AWS Transform Assessment Data template. You can download the AWS Transform Assessment data template from AWS Transform.
3. To include a storage assessment, upload a NetApp Data Infrastructure Insights (DII) file or an Excel file created from the AWS Transform Assessment data template. See Preparing DII files to learn more about DII files. You can download the AWS Transform Assessment Data template from AWS Transform.
4. Specify the AWS Region where you want to host your migrated workloads. All commercial AWS Regions are supported.
5. Choose **Generate business case**. The right pane automatically switches to the **Worklog** tab where you can track the progress of the job.
6. Download and review the business case.
7. (Optional) Use the chat pane to ask AWS Transform questions about your business case.

Preparing DII files

To provide AWS Transform with the necessary data for a storage assessment you have to run NetApp queries and save the results in CSV files. Zip the CSV files and upload the zip to AWS Transform.

For NAS volumes, run these queries:

- Disk
- Internal Volumes
- Storage
- Share

For SAN volumes, run these queries:

- Virtual Machines
- Volume

Follow these steps to run the queries and save the resulting files:

1. From the home page of your NetApp DII tenant choose **Explore** and then choose **New Metric Query**.
2. Select the query type.
3. Select a period of 30 days.
4. Open the settings from the gear icon and choose **Select all the columns**.
5. Save as a CSV file by selecting the **Export to csv** icon.

Tracking the progress of a migration assessment job

The **Worklog** tab provides a detailed log of the actions that AWS Transform takes, along with human input requests and your responses to those requests. AWS Transform adds entries to the worklog to show the progress of the assessments, including processing uploaded files, analyzing files, generating the assessment report, and any errors that occur.

Custom

What is AWS Transform custom?

AWS Transform custom uses agentic AI to perform large-scale modernization of software, code, libraries, and frameworks to reduce technical debt. It handles diverse scenarios including language version upgrades, API and service migrations, framework upgrades and migrations, code refactoring, and organization-specific transformations.

Through continual learning, the agent improves from every execution and developer feedback, delivering high-quality, repeatable transformations without requiring specialized automation expertise.

Key Capabilities

AWS Transform custom provides the following capabilities:

- **Natural language-driven transformation definition** - Create custom transformations using natural language, documentation, and code samples
- **Transformation execution** - Apply transformations consistently and reliably across multiple codebases
- **Continual learning** - Automatically improve transformation quality from every execution
- **AWS-managed transformations** - Use ready-to-use, AWS-vetted transformations for common scenarios

Transformation Patterns

AWS Transform custom supports diverse transformation patterns to address your modernization needs. Each pattern has different complexity characteristics based on the scope and nature of the changes required.

Pattern	Description	Complexity	Examples
API and Service Migrations	Migrating between API versions or equivalent services	Medium	AWS SDK v1→v2 (Java, Python, JavaScript),

Pattern	Description	Complexity	Examples
	while maintaining functionality		Boto2→Boto3, JUnit 4→5, javax→jakarta
Language Version Upgrades	Upgrading to newer versions of the same programming language, adopting new features and replacing deprecated functionality	Low-Medium	Java 8→17, Python 3.9→3.13, Node.js 12→22, TypeScript version upgrades
Framework Upgrades	Upgrading to newer versions of the same framework, addressin g breaking changes	Medium	Spring Boot 2.x→3.x, React 17→18, Angular upgrades, Django upgrades
Framework Migrations	Migrating to entirely different framework s that serve similar purposes	High	Angular→React, Redux→Zustand, Vue.js→React
Library and Dependency Upgrades	Upgrading third-party libraries to newer versions while maintaining the same language and framework	Low-Medium	Pandas 1.x→2.x, NumPy upgrades, Hadoop/HBase/Hive library upgrades, Lodash upgrades
Code Refactori ng and Pattern Modernization	Modernizing code patterns and adopting best practices without changing external functionality	Low-Medium	Print→Logging frameworks, string concatenation→f-strings, type hints adoption, observabi lity instrumentation

Pattern	Description	Complexity	Examples
Script and File-by-File Translations	Translating independent scripts or configuration files where files are mostly self-contained	Low-Medium	AWS CDK→Terraform, Terraform→CloudFormation, Excel→Python notebooks, Bash→PowerShell
Architecture Migrations	Migrating between hardware architectures or runtime environments with minimal code changes	Medium-High	x86→AWS Graviton (ARM), on-premises→Lambda, traditional server→containers
Language-to-Language Migrations	Translating codebases from one programming language to another	Very High	Java→Python, JavaScript→TypeScript, C→Rust, Python→Go
Custom and Organization-Specific Transformations	Unique organizational requirements and specialized modernization needs	Varies	Custom internal library migrations, organization-specific coding standards, proprietary framework migrations

Note

For COBOL/mainframe languages, use AWS Transform for Mainframe. For .NET Framework upgrades to .NET Core, consider AWS Transform for Windows. For VMware migrations to AWS, consider AWS Transform for VMware.

How AWS Transform custom Works

AWS Transform custom is usually used in large-scale projects where multiple codebases or modules are transformed. Teams typically follow a four-phase workflow:

Define Transformation - Provide natural language prompts, documentation, and code samples to the agent, which generates an initial transformation definition. This definition can be iteratively refined through chat or direct edits. This phase can be skipped when using AWS-managed transformations.

Pilot or Proof-of-Concept - Test the transformation on sample codebases and refine based on results. This validation phase helps estimate the cost and effort of the full transformation. Continual learning improves quality during this phase.

Scaled Execution - Set up automated bulk execution using the CLI, with developers reviewing and validating results. Monitor progress using the web application and track transformations across multiple repositories.

Monitor and Review - Continual learning automatically improves the transformation quality. Review and approve knowledge items that have been extracted from executions to ensure they meet quality standards.

Understanding Key Concepts

This section explains key concepts for working with AWS Transform custom.

Transformation Definitions

A **transformation definition** is a package that contains the instructions and knowledge needed to perform a specific code transformation. It consists of:

- `transformation_definition.md` (required) - Contains the core transformation logic and instructions
- `summaries.md` (optional) - Summaries for user-provided reference documentation
- `document_references/` folder (optional) - User-provided documentation and reference materials

AWS Transform CLI automatically downloads transformation definitions to the current directory when needed for execution, inspection, or modification.

> **Important**
>
> When publishing a transformation, the directory must only contain these files and folders. No other files or subdirectories are allowed.

Transformation Registry

The **transformation registry** is your AWS account's centralized repository for storing and managing transformation definitions. Transformations in the registry can be:

- Listed using `atx custom def list`
- Executed across multiple codebases
- Shared with other users in your AWS account
- Version-controlled

> **Important**
>
> Transformation definitions are account-specific. If you want to use a transformation in a different AWS account, you must publish it separately in that account.

Draft vs Published Transformations

AWS Transform custom supports two states for transformations in the registry:

Draft transformations are in-progress or untested transformation definitions. They are saved as specific versions and can be retrieved, updated, and executed by users referring to that specific version. Drafts are useful for iterative development, testing, and refinement before the transformation is ready to be shared with your team. Drafts are also associated to specific conversation. If you restart the CLI, you can use `atx --conversation-id {id}` to restore a previous one.

Published transformations are available in your account's transformation registry for execution by other users with the required IAM permissions. Published transformations can be discovered using `atx custom def list`.

The typical workflow is:

1. Create transformation locally
2. Save as draft for testing (`atx custom def save-draft`)
3. Refine and validate
4. Publish to share with your team (`atx custom def publish`)

You can also publish a transformation directly without saving it as a draft.

References vs. Knowledge Items

AWS Transform custom uses two types of knowledge to improve transformation quality:

References are user-provided documentation that you explicitly add to a transformation definition. References support text files only (maximum 10MB total for all files) and usually contain documentation, API specifications, migration guides, and code samples. You add references when creating or updating a transformation definition in interactive mode.

Knowledge items are automatically extracted learnings from transformation executions. These are created asynchronously by the continual learning system based on execution trajectories, developer feedback, and code fixes encountered during transformations. Knowledge items start in a "not approved" state and must be explicitly approved by transformation owners before they can be used in future executions. Unlike references which you provide upfront, knowledge items accumulate over time as the transformation is executed across different codebases.

Build and Validation Commands

The **build or validation command** is an optional parameter that specifies how to validate your code during the transformation process. This command is executed at various points during the transformation to ensure code integrity.

Providing a command that validates the results and returns issues if validation fails is very important to improve transformation quality through continual learning. If no build or validation is needed, omit from your input.

For examples and detailed guidance, see Build and Validation Commands in the Workflows section.

Continual Learning

Continual learning is the system that automatically captures feedback from every transformation execution and improves transformation quality over time. The system gathers information through:

- Explicit feedback - Comments and code fixes provided in interactive mode
- Implicit observations - Issues the agent encounters while transforming and debugging code

This information is processed to create **knowledge items** that are added to the transformation definition to improve future transformations. The learning process occurs automatically after transformations are completed, requiring no additional user input.

> **Important**
>
> Knowledge items are transformation-specific and are not shared across different transformations or different customer accounts.

Getting Started

This section describes how to set up AWS Transform custom and run your first transformation.

Prerequisites

Before installing AWS Transform custom, ensure you have the following:

Supported Platforms

AWS Transform custom supports the following operating systems:

- **Linux** - Full support for all Linux distributions
- **macOS** - Full support for macOS systems
- **Windows Subsystem for Linux (WSL)** - Supported when running under WSL

> **Important**
>
> Windows native execution is not supported. AWS Transform custom will detect native Windows environments and exit with an error message. For Windows users, install and use Windows Subsystem for Linux (WSL).

Required Software

- **Node.js 20 or later** - Download from https://nodejs.org/en/download
- **Git** - Must be installed and the working directory needs to be a valid Git repo to execute a transformation. Run `git init; git add .; git commit -m "Initial commit"` to initialize a Git repo in your working directory.

Network Requirements

An internet connection with access to the following endpoints is required:

- `transform-cli.awsstatic.com`
- `transform-custom.<region>.api.aws`
- `*.s3.amazonaws.com`

If you are working in an internet-restricted environment, update firewall rules to allowlist these URLs.

Installing the AWS Transform CLI

The recommended installation method is using the installation script.

To install the AWS Transform CLI

1. Run the installation script:

   ```
   curl -fsSL https://transform-cli.awsstatic.com/install.sh | bash
   ```

2. Verify the installation:

   ```
   atx --version
   ```

   The command should display the installed version of the AWS Transform CLI.

Configuring Authentication

AWS Transform custom requires AWS credentials to authenticate with the service. Configure authentication using one of the following methods.

Environment Variables

Set the following environment variables:

```
export AWS_ACCESS_KEY_ID=your_access_key
export AWS_SECRET_ACCESS_KEY=your_secret_key
export AWS_SESSION_TOKEN=your_session_token
```

You can also specify a profile using the `AWS_PROFILE` environment variable:

```
export AWS_PROFILE=your_profile_name
```

AWS Credentials File

Configure credentials in `~/.aws/credentials`:

```
[default]
aws_access_key_id = your_access_key
aws_secret_access_key = your_secret_key
```

IAM Permissions

Your AWS credentials must have permissions to call the AWS Transform service. At minimum, you need `transform-custom:*` permissions.

The following IAM policy provides full access to AWS Transform custom:

```
{
  "Version": "2012-10-17"       ,
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "transform-custom:*"
      ],
      "Resource": "*"
    }
  ]
}
```

For more granular control, please refer to the AWS Transform Custom IAM Service Authorization Reference Guide

Note

- Transformation definitions are AWS resources with ARNs (Amazon Resource Names). You can use IAM policies to control access to specific transformations or groups of transformations by specifying the transformation ARN in your IAM policy resource statements. Tags can be used for grouped access control.
- AWS IAM Identity Center is required to access the AWS Transform, but is not required to use the CLI.

Configuring AWS Region

AWS Transform custom is available in specific AWS regions and automatically detects your region configuration using standard AWS CLI precedence.

Supported Regions

AWS Transform custom is available in the following AWS regions:

- `us-east-1` (US East - N. Virginia)
- `eu-central-1` (Europe - Frankfurt)

How Region is Determined

The CLI checks these sources in priority order to determine which region to use:

1. `AWS_REGION` environment variable (highest priority)
2. `AWS_DEFAULT_REGION` environment variable
3. Selected profile in `~/.aws/config` (via `AWS_PROFILE` environment variable)
4. Default profile in `~/.aws/config`
5. Default fallback to `us-east-1` if no configuration is found

> **Note**
>
> If `ATX_CUSTOM_ENDPOINT` is set, the region is extracted from the endpoint URL and overrides all other settings.

Setting Your Region

Choose one of these methods to configure your region:

Option 1: Environment variable (recommended for temporary use):

```
export AWS_REGION=<your-region>
```

Option 2: AWS CLI configuration (recommended for permanent setup):

```
aws configure set region <your-region>
```

Option 3: Profile-specific configuration:

```
aws configure set region <your-region> --profile your_profile_name
export AWS_PROFILE=your_profile_name
```

Option 4: Direct file editing:

Edit `~/.aws/config` directly:

```
[default]
region = <your-region>

[profile your_profile_name]
region = <your-region>
```

Verifying Your Region Configuration

To check which region AWS Transform custom will use:

Check current region setting:

```
aws configure get region
```

Check environment variables:

```
echo "AWS_REGION: $AWS_REGION"
echo "AWS_DEFAULT_REGION: $AWS_DEFAULT_REGION"
echo "AWS_PROFILE: $AWS_PROFILE"
```

View detailed region resolution in debug logs:

```
tail -f ~/.aws/atx/logs/debug.log
```

Example log output showing region source:

```
2026-01-07 22:34:28 [DEBUG]: Initializing FrontendServiceClient with config:
{
  "endpoint": "https://transform-custom.us-east-1.api.aws",
  "region": "us-east-1",
  "regionSource": "AWS_REGION"
}
```

The `regionSource` field shows where the region was derived from (e.g., "aws-config (profile: default)", "AWS_REGION", "default").

> **Important**
>
> If your region configuration points to an unsupported region, the CLI will display a clear error message with instructions to update your configuration to a supported region.

Running Your First Transformation

The fastest way to get started is to use an AWS-managed transformation. AWS-managed transformations are pre-built, AWS-vetted transformations that are ready to use without any setup.

To run an AWS-managed transformation interactively:

```
atx
```

Then ask the agent to execute a transformation:

```
Execute the AWS/java-aws-sdk-v1-to-v2 transformation on the codebase at ./my-java-
project
```

To run a transformation non-interactively:

```
atx custom def exec -n AWS/java-aws-sdk-v1-to-v2 -p ./my-java-project -c "mvn clean
 install" -x -t
```

For detailed information about execution modes, configuration options, and command flags, see the section called “Command Reference”.

Understanding Transformation Results

When a transformation completes, AWS Transform custom provides a report describing the results. If the transformation was not validated (e.g. successful build for a Java applications), the report provides recommendations for further actions.

Most transformations are performed in multiple steps, using Git to commit intermediate ones. You can review the changes made by the transformation using standard Git commands:

```
git status
git log
git diff <original commit-id>
```

Creating a Custom Transformation

When AWS-managed transformations don't meet your needs, you can create custom transformations tailored to your specific requirements.

To create a custom transformation

1. Start the AWS Transform CLI in interactive mode:

   ```
   atx
   ```

2. Tell the agent you want to create a new transformation and describe your requirements.
3. Provide reference materials such as documentation and code samples to improve quality.
4. Test and refine the transformation iteratively.
5. Publish it to your organization's transformation registry.

For detailed instructions on creating custom transformations, see the section called “Workflows”.

AWS Transform Web Application (Optional)

The AWS Transform web application is an optional interface for monitoring large-scale transformation campaigns across multiple repositories. Before using AWS Transform web application, you need to have a user identity enabled to access AWS Transform in your organization. For information about enabling AWS Transform, see Setting up AWS Transform.

To use the AWS Transform web application:

1. Visit https://aws.amazon.com/transform/ and sign in using AWS IAM Identity Center credentials.
2. Select the "Custom / code" job type in the AWS Transform web application.
3. Tell the agent the name of the transformation definition you would like to use.
4. The web application provides an AWS Transform CLI command that can be shared with your team. This command invokes the transformation definition and logs execution results to the web application.
5. After transformation results are captured, view statistics, time savings measurements, and ask questions about the progress of your job in the Dashboard.

The web application is designed for enterprise-scale operations where you need centralized monitoring of transformations across multiple codebases.

> **Note**
>
> AWS IAM Identity Center is required to access the AWS Transform web application. The CLI does not require IAM Identity Center - only standard AWS credentials are needed.

Introduction to custom transformation commands

Here are a few of the commands you can use with custom transformations. The complete list of commands is in the AWS Transform custom transformations command reference.

- `atx custom`

  - **Executes custom transformation interactive experience, allowing creation, discovery, execution, and refinement of transformations.**
  - This is the default command for `atx`.
  - `--trust-all-tools` (`-t`) is optional and implicitly allows all requested tool requests by the agent. Use with caution especially in production environments. You can configure tool trust for specific tools and commands using the trust settings file.
- `atx custom --help | atx custom -h`
  - **Shows help menu.**
  - Each command also includes a help menu, for example, `atx custom def exec --help`.
- `atx --version | atx -v`
  - **Shows version.**
  - The version number changes with each release.
- `atx custom def list`
  - **Prints list of transformations available in the transformation registry.**
- `atx custom def exec`
  - **Executes a transformation**
- `atx mcp`
  - **Used to manage MCP server configurations**

Workflows

Executing Transformations

This section describes the different ways to execute transformations and options for controlling execution behavior.

Execution Modes

AWS Transform custom supports three execution modes to accommodate different workflows.

Interactive Conversational Mode

Start the CLI with `atx` and ask the agent to execute a transformation through natural language. This mode allows you to have a full conversation with the agent, interrupt execution at any point, and provide feedback during the transformation process.

Use this mode when you want maximum control and the ability to guide the agent through complex scenarios.

Direct Interactive Execution

Use `atx custom def exec -n <transformation-name> -p <path>` to start a specific transformation interactively. This mode allows you to review and interact with the agent at the beginning, during, or at the end of execution. The agent will pause at key decision points and ask for your input.

This is ideal for testing and refining transformations before running them autonomously.

Non-Interactive/Headless Mode

Use `atx custom def exec -n <transformation-name> -p <path> -x -t` for full automation. The `-x` flag enables non-interactive mode, and the `-t` flag automatically trusts all tools without prompting.

This mode is designed for CI/CD pipeline integration and bulk execution where no human intervention is available or desired.

Common Command Flags

When executing transformations with `atx custom def exec`, the following flags are commonly used:

- `-n` or `--transformation-name` - Specifies the name of the transformation to execute
- `-p` or `--code-repository-path` - Specifies the path to your codebase (use "." for current directory)
- `-c` or `--build-command` - Specifies the build or validation command to run
- `-x` or `--non-interactive` - Enables non-interactive mode (no user prompts)
- `-t` or `--trust-all-tools` - Automatically trusts all tools without prompting
- `-d` or `--do-not-learn` - Prevents knowledge item extraction from this execution
- `--tv` or `--transformation-version` - Specifies a specific version of the transformation
- `-g` or `--configuration` - Provides a configuration file or inline configuration

> **Important**
>
> The `-t` or `--trust-all-tools` flag automatically approves all tool executions without prompting and bypasses all security guardrails. This is required for a fully autonomous experience but not required to execute the transformation. Use with caution in production environments.

Using Configuration Files

AWS Transform custom supports optional configuration files in YAML or JSON format. Configuration files allow you to specify execution parameters and provide additional context to the agent.

To use a configuration file:

```
atx custom def exec --configuration file://config.yaml
```

You can also provide configuration as inline key-value pairs:

```
atx custom def exec --configuration "key=value,key2=value2"
```

Example configuration file (config.yaml):

```
codeRepositoryPath: ./my-project
transformationName: my-transformation
buildCommand: mvn clean install
additionalPlanContext: |
  The target Java version to upgrade to is Java 17.
  Ensure compatibility with our internal logging framework version 2.3.
validationCommands: |
  mvn test
  mvn verify
```

The `additionalPlanContext` parameter provides extra context for the agent's execution plan. This is especially useful with AWS-managed transformations to customize their behavior for your specific needs.

Build and Validation Commands

The build or validation command is an optional parameter that specifies how to validate your code during the transformation process. AWS Transform custom will attempt to infer the best build command based on the transformation if not specified, though it is recommended to be specific for quality.

Examples of build and validation commands:

- Java: `mvn clean install` or `gradle build`
- Python: `pytest` or `python -m py_compile`
- Node.js: `npm run build` or `npm test`
- Linters: `eslint .` or `pylint .`

Even for languages or transformations that don't require building, providing a command that validates the results and returns issues if validation fails is very important to improve transformation quality.

If no build or validation is needed, omit from your input.

Controlling Learning Behavior

By default, AWS Transform custom extracts knowledge items from every transformation execution. You can prevent learning for specific executions.

To prevent learning from an execution:

```
atx custom def exec -n my-transformation -p ./my-project -d
```

The `-d` or `--do-not-learn` flag opts out of allowing knowledge item extraction from the current execution.

Resuming Conversations

AWS Transform custom allows you to resume previous conversations within 30 days of creation.

To resume the most recent conversation:

```
atx --resume
```

To resume a specific conversation:

```
atx --conversation-id <conversation-id>
```

> **Important**
>
> Conversations can only be resumed within 30 days of creation. After 30 days, the conversation can no longer be resumed.

Tracking Agent Minutes

AWS Transform custom tracks the agent minutes consumed during a transformation session. Agent minutes accumulate throughout the conversation lifecycle and are displayed when the conversation ends:

```
Agent minutes used: 12.50
```

Agent minutes persist across interruptions. If you interrupt a session with Ctrl+C and resume it later, the previously accumulated minutes carry over and continue accumulating in the resumed session.

To check Agent Minutes during an interactive session:

Type `/usage` at the input prompt to display the current accumulated Agent Minutes without ending the conversation.

To set an Agent Minutes budget limit:

```
atx custom def exec -n my-transformation -p ./my-project --limit 30
```

The `--limit` option sets a maximum Agent Minutes budget for the session. Agent Minutes reflect active agent work time, not wall clock time. When the limit is reached, the CLI displays a message and exits with instructions to resume:

```
## Budget limit reached: 30.00 / 30.00 Agent Minutes. Exiting.
```

You can resume the conversation later with an increased limit:

```
atx --conversation-id <conversation_id> -t --limit <increased_limit>
```

Continual Learning

This section describes how to manage knowledge items created by continual learning.

Understanding Knowledge Items

Knowledge items are automatically extracted learnings from transformation executions. These are created asynchronously by the continual learning system based on:

- Developer feedback provided in interactive mode
- Code issues encountered during transformations

Knowledge items start in a "not approved" state and must be explicitly approved by transformation owners before they can be used in future executions. Unlike references which you provide upfront, knowledge items accumulate over time as the transformation is executed across different codebases.

Listing Knowledge Items

View all knowledge items for a transformation definition.

To list knowledge items:

```
atx custom def list-ki -n my-transformation
```

This displays all knowledge items that have been extracted from executions of the specified transformation definition.

Viewing Knowledge Item Details

View detailed information about a specific knowledge item.

To view knowledge item details:

```
atx custom def get-ki -n my-transformation --id <knowledge-item-id>
```

Enabling and Disabling Knowledge Items

Control which knowledge items are applied to future transformations.

To enable a knowledge item:

```
atx custom def update-ki-status -n my-transformation --id <knowledge-item-id> --status
 ENABLED
```

To disable a knowledge item:

```
atx custom def update-ki-status -n my-transformation --id <knowledge-item-id> --status
 DISABLED
```

Configuring Auto-Approval

You can configure whether knowledge items are automatically enabled or require manual approval.

To enable auto-approval for all knowledge items:

```
atx custom def update-ki-config -n my-transformation --auto-enabled TRUE
```

To disable auto-approval:

```
atx custom def update-ki-config -n my-transformation --auto-enabled FALSE
```

Deleting Knowledge Items

Permanently remove knowledge items that are not useful.

To delete a knowledge item:

```
atx custom def delete-ki -n my-transformation --id <knowledge-item-id>
```

Exporting Knowledge Items

Export all knowledge items for a transformation to markdown format for review or documentation.

To export knowledge items:

```
atx custom def export-ki-markdown -n my-transformation
```

Advanced Configuration

This section describes advanced features and configuration options for AWS Transform custom.

Environment Variables

You can customize CLI behavior using environment variables.

ATX_SHELL_TIMEOUT

Override the default timeout for shell commands (900 seconds/15 minutes).

```
export ATX_SHELL_TIMEOUT=1800  # 30 minutes
```

This is useful for large codebases or long-running build processes.

ATX_DISABLE_UPDATE_CHECK

Disable automatic version checks and update notifications during command execution.

```
export ATX_DISABLE_UPDATE_CHECK=true
```

Trust Settings

Trust settings allow you to pre-approve specific tools and commands to execute without prompts. These settings are configured in the `~/.aws/atx/trust-settings.yaml` file.

The file contains two lists:

- `trustedTools` - Tools that can execute without prompting
- `trustedShellCommands` - Shell commands that can execute without prompting

Default trusted tools:

- `file_read`
- `get_transformation_from_registry`
- `list_available_transformations_from_registry`

Editing trust settings:

You can manually edit the trust-settings.yaml file to add or remove trusted tools and commands. The `trustedShellCommands` list supports wildcard patterns using `*` for flexible command matching.

Examples:

- `cd *` - Matches compound commands starting with cd
- `*&&*` - Trusts all commands with && operators

Session-level trust:

During interactive prompts, you can choose:

- `(y)es` - Execute once
- `(n)o` - Deny
- `(t)rust` - Trust for the current session only

Session-level trust settings are temporary and reset when the CLI restarts, providing temporary approval without permanently modifying trust-settings.yaml.

Model Context Protocol (MCP) Servers

The AWS Transform CLI supports Model Context Protocol (MCP) servers, which extend its functionality with additional tools.

Configuration:

Configure MCP servers in the `~/.aws/atx/mcp.json` file.

Managing MCP servers:

View list of configured MCP servers:

```
atx mcp tools
```

List available tools offered by a specific MCP server:

```
atx mcp tools --server <server-name>
```

Usage tracking:

The CLI automatically tracks MCP tool usage during transformation executions. Usage statistics are persisted as `mcp_usage.json` in the conversation directory alongside `metadata.json`. The file records per-tool metrics for each execution, including:

- Number of invocations per tool
- Number of errors per tool
- Total execution time per tool
- Last error details (if any)

Tags and Organization

You can organize transformations with tags for access control and categorization.

> **Note**
>
> Some of these commands require specifying the Amazon Resource Name (ARN) for a Transformation Definition. The ARN structure is: `arn:aws:transform-custom:<region>:<account-id>:package/<td-name>`

To list tags for a transformation:

```
atx custom def list-tags --arn <transformation-arn>
```

To add tags to a transformation:

```
atx custom def tag --arn <transformation-arn> --tags '{"env":"prod","team":"backend"}'
```

To remove tags from a transformation:

```
atx custom def untag --arn <transformation-arn> --tag-keys "env,team"
```

Tags can be used for grouped access control in IAM policies. You can create policies that grant permissions to all transformations with specific tags (e.g., all transformations tagged with `team:frontend` or `environment:production`).

Logs

AWS Transform CLI maintains two types of logs for troubleshooting and debugging.

Conversation logs:

Location: `~/.aws/atx/custom/<conversation_id>/logs/<timestamp>-conversation.log`

These logs contain the full conversation history for a specific session.

Developer debug logs:

Locations:

- `~/.aws/atx/logs/debug*.log`
- `~/.aws/atx/logs/error.log`

These logs provide advanced troubleshooting information for the CLI itself.

> **Note**
>
> There may be multiple debug log files in the logs directory (i.e. debug1.log, debug2.log). Review and provide all relevant logs for example, ~/.aws/atx/custom/<conversation-id>/* and ~/.aws/atx/logs/*, when opening support tickets for faster resolution.

CLI Updates

Keep your CLI up to date to access new features and improvements.

To check for updates:

```
atx update --check
```

To update to the latest version:

```
atx update
```

To update to a specific version:

```
atx update --target-version <version>
```

Create Custom Transformations

This section describes how to create, modify, and manage custom transformation definitions.

Creating a New Transformation

Use the interactive CLI to create a new transformation definition.

To create a transformation definition

1. Start the AWS Transform CLI:

   ```
   atx
   ```

2. Tell the agent you want to create a new transformation.
3. Provide a clear, detailed description of the transformation objective. Include:
   - The source and target state (e.g., "upgrade from version X to version Y")
   - Specific changes required (e.g., "update import statements, replace deprecated methods")
   - Any special considerations or constraints
4. When the agent requests clarification or additional information, provide specific examples and reference materials.
5. Review the initial transformation definition created by the agent.
6. Test the transformation on a sample codebase.
7. Iterate by providing feedback, code fixes, or additional examples.
8. Save the transformation locally or publish it to the registry.

Best practices for creating transformations:

- Start with simple, well-defined transformations before attempting complex ones
- Provide comprehensive reference materials including migration guides and code samples
- Test on multiple sample codebases before publishing

- Use deterministic build or validation commands to enable continual learning
- Consider breaking complex transformations into multiple smaller steps
- Mark crucial information with "CRITICAL:" or "IMPORTANT:" in your transformation definitions to ensure the agent prioritizes these requirements
- When you need exact requirements followed (like using a specific command or string value), explicitly specify the complete string in your transformation definitions. You can wrap these in bash quotes to clearly indicate they're terminal commands or literal strings, which reduces variability and ensures consistent execution

Providing Reference Materials

You can provide reference files to AWS Transform custom by specifying file paths during the conversation.

Recommended types of reference files:

- Before/after example code
- Documentation for APIs, libraries, or features involved
- Human-readable migration guides

To provide a reference file:

```
Take a look at the documentation here: /path/to/migration-guide.md
```

You can also provide a directory containing multiple reference files:

```
Take a look at the docs we have here: /path/to/docs/
```

> **Note**
>
> Only text-based files (.md, .html, .txt, code files) are supported. Binary files, images, and rich text files (e.g., .pdf, .png, .docx) are not currently supported. It is often possible to extract the text content and use that as reference. If you have many small text files, consider concatenating them into few descriptively-named files. There is a limit of 10MB total for all files.

Modifying an Existing Transformation

You can modify custom transformations both before and after saving them as drafts or publishing them. You cannot modify AWS-managed transformations. If you need to customize them, you can provide additional context using the config file.

To modify an existing transformation

1. Start the AWS Transform CLI:

   ```
   atx
   ```

2. Tell the agent you want to modify an existing transformation.
3. Choose whether to:

   - Provide a file path to a locally stored transformation (i.e. not a saved draft or published)
   - Request the list of transformations from the registry

4. If choosing from the registry, select the transformation you want to modify.
5. Work with the agent to describe the changes you want to make.
6. Test the updated transformation on a sample codebase.
7. Publish your updates to the registry if desired.

Publishing and Managing Transformations

You can publish and manage your transformations using the interactive experience or with the following commands.

To save a transformation as a draft:

```
atx custom def save-draft -n my-transformation --description "Description of the
 transformation" --sd ./transformation-directory
```

To publish a transformation:

```
atx custom def publish -n my-transformation --description "Description of the
 transformation" --sd ./transformation-directory
```

To list available transformations:

```
atx custom def list
```

To download a transformation definition:

```
atx custom def get -n my-transformation
```

This downloads the transformation definition to your current working directory. You can specify a target directory with the `--td` flag and a version with the `--tv` flag.

To delete a transformation definition:

```
atx custom def delete -n my-transformation
```

> **Important**
>
> This permanently deletes the specified transformation definition from your account.

Managing Transformation Versions

AWS Transform custom maintains versions of your transformation definitions. You can specify a version when executing or downloading a transformation.

To execute a specific version:

```
atx custom def exec -n my-transformation --tv v1 -p ./my-project
```

To download a specific version:

```
atx custom def get -n my-transformation --tv v1
```

If no version is specified, the latest version is used.

Command Reference

This section provides a comprehensive reference of all AWS Transform custom CLI commands.

Interactive Commands

atx

Start an interactive conversation with the AWS Transform CLI.

```
atx                          # Start new conversation
atx --resume                 # Resume most recent conversation
atx --conversation-id <id>   # Resume specific conversation
atx -t                       # Start with all tools trusted
```

Interactive session commands

The following commands can be typed at the input prompt during an interactive session:

- `/usage` - Display the accumulated agent minutes for the current session

Transformation Definition Commands

atx custom def exec

Execute a transformation definition on a code repository.

Option	Long Form	Parameter	Description
`-p`	`--code-repository-path`	`<path>`	Path to the code repository to transform. For current directory, use "."
`-c`	`--build-command`	`<command>`	Command to run when building repository
`-n`	`--transformation-name`	`<name>`	Name of the transformation definition in the registry

Option	Long Form	Parameter	Description
`-x`	`--non-interactive`	-	Runs the transformation with no user assistance
`-t`	`--trust-all-tools`	-	Trusts all tools (no tool prompts)
`-d`	`--do-not-learn`	-	Opt out of allowing knowledge item extraction from this execution
`-g`	`--configuration`	`<config>`	Path to config file (JSON or YAML) or key=value pairs
`--tv`	`--transformation-version`	`<version>`	Version of the transformation definition to use
`--limit`	`--limit`	`<limit>`	Set Agent Minutes budget limit. Transformation exits when limit is reached and can be resumed with an increased limit
`-h`	`--help`	-	Display help for command

atx custom def list

List available transformation definitions.

```
atx custom def list
```

```
atx custom def list --json    # Output in JSON format
```

atx custom def get

Get details of a specific transformation definition.

Option	Long Form	Parameter	Description
-n	--transformation-name	<name>	The name of the transformation definition to retrieve
--tv	--transformation-version	<version>	The version of the transformation definition to retrieve
--td	--target-directory	<directory>	The target directory at which to save the transformation definition (default: ".")
--json	--json	-	Output response in JSON format
-h	--help	-	Display help for command

atx custom def delete

Delete a transformation definition permanently.

```
atx custom def delete -n <transformation-name>
```

atx custom def publish

Publish a transformation definition to the registry.

Option	Long Form	Parameter	Description
-n	--transformation-name	<name>	The name of the transformation definition to publish
--tv	--transformation-version	<version>	The version of the transformation definition to use (not applicable with --description or --source-directory)
--sd	--source-directory	<directory>	The source directory from which to read the local transformation definition files (not applicable with --transformation-version)
--description	--description	<description>	A description for the transformation definition (not applicable with --transformation-version)
--json	--json	-	Output response in JSON format
-h	--help	-	Display help for command

atx custom def save-draft

Save a transformation definition as a draft in the registry. Drafts expire after 30 days and do not show up in the registry. This command returns the version number of the draft. You must execute and retrieve drafts explicitly using the transformation name and version number.

```
atx custom def save-draft -n <transformation-name> --description "Description" --sd
 <directory>
```

Knowledge Item Commands

atx custom def list-ki

List knowledge items for a transformation definition.

```
atx custom def list-ki -n <transformation-name>
atx custom def list-ki -n <transformation-name> --json
```

atx custom def get-ki

Retrieve a knowledge item from a transformation definition.

```
atx custom def get-ki -n <transformation-name> --id <id>
```

atx custom def delete-ki

Delete a knowledge item from a transformation definition.

```
atx custom def delete-ki -n <transformation-name> --id <id>
```

atx custom def update-ki-status

Update knowledge item status (ENABLED or DISABLED).

```
atx custom def update-ki-status -n <transformation-name> --id <id> --status ENABLED
```

atx custom def update-ki-config

Update knowledge item configuration for auto-approval.

```
atx custom def update-ki-config -n <transformation-name> --auto-enabled TRUE
```

atx custom def export-ki-markdown

Export all knowledge items for a transformation definition to markdown.

```
atx custom def export-ki-markdown -n <transformation-name>
```

Tag Commands

atx custom def list-tags

List tags for a transformation definition.

```
atx custom def list-tags --arn <transformation-arn>
```

atx custom def tag

Add tags to a transformation definition.

```
atx custom def tag --arn <arn> --tags '{"env":"prod","team":"backend"}'
```

atx custom def untag

Remove tags from a transformation definition.

```
atx custom def untag --arn <arn> --tag-keys "env,team"
```

Update Commands

atx update

Update the AWS Transform CLI.

```
atx update                    # Update to latest version
atx update --check            # Check for updates only
atx update --target-version <version>  # Update to specific version
```

MCP Commands

atx mcp tools

Allows clients to view/confirm their MCP tool configurations. Updates to the configurations must be managed directly via the file ~/.aws/atx/mcp.json

```
atx mcp tools     # View list of MCP servers
atx mcp tools -s    # List tools for the specified server
```

AWS-Managed Transformations

AWS-managed transformations are pre-built, AWS-vetted transformations for common use cases that are ready to use without any additional setup.

Overview

AWS-managed transformations have the following characteristics:

- **Validated by AWS** - These transformations are vetted by AWS to be high quality
- **Ready to use** - No additional setup required
- **Continuously growing** - Additional transformations are continually being added
- **Customizable** - Pre-built transformations can be customized by providing additional guidance or requirements specific to your organization's needs using the `additionalPlanContext` configuration parameter
- **Early access support** - Some transformations may be marked as early access as they undergo further testing and refinement

Available AWS-Managed Transformations

The following AWS-managed transformations are currently available:

Language Version Upgrades:

- `AWS/java-version-upgrade` - Upgrade Java applications using any build system from any source JDK version to any target JDK version with comprehensive dependency modernization including Jakarta EE migration, database drivers, ORM frameworks, and Spring ecosystem updates. You can specify your desired target JDK version either through interactive chat with the agent, or by passing an additionalPlanContext configuration parameter.
- `AWS/python-version-upgrade` - Migrate Python projects from Python 3.8/3.9 to Python 3.11/3.12/3.13, ensuring compatibility with the latest Python features, security updates, and runtime while maintaining functionality and performance. You can specify your desired

target Python version either through interactive chat with the agent, or by passing an additionalPlanContext configuration parameter.

- `AWS/nodejs-version-upgrade` - Upgrade NodeJS applications from any source NodeJS version to any target NodeJS version. You can specify your desired target NodeJS version either through interactive chat with the agent, or by passing an additionalPlanContext configuration parameter.

AWS SDK Migrations:

- `AWS/java-aws-sdk-v1-to-v2` - Upgrade the AWS SDK from V1 to V2 for Java projects using Maven or Gradle.
- `AWS/python-boto2-to-boto3` - Migrate Python applications from boto2 to boto3, based on the official AWS migration documentation.
- `AWS/nodejs-aws-sdk-v2-to-v3` - Upgrade Node.js applications from AWS SDK for JavaScript v2 to v3 to leverage modular architecture, first-class TypeScript support, middleware stack, and improved performance while ensuring all AWS service interactions continue to function correctly, without modifying the underlying Node.js version.

Analysis:

- `AWS/comprehensive-codebase-analysis` - This transformation performs deep static analysis of codebases to generate hierarchical, cross-referenced documentation covering all aspects of the system. It combines behavioral analysis, architectural documentation, and business intelligence extraction to create a comprehensive knowledge base organized for maximum usability and navigation. The transformation places special emphasis on technical debt analysis, providing prominent, actionable insights on outdated components and maintenance concerns at the root level.
- `AWS/java-performance-optimization` - Optimize Java application performance by analyzing JFR profiling data to detect CPU/memory hotspots and anti-patterns, then applying targeted code fixes to reduce resource usage and improve efficiency. For instructions on collecting JFR data, see https://docs.oracle.com/javacomponents/jmc-5-4/jfr-runtime-guide/run.htm.

Early Access Transformations

> **Note**
>
> Early access transformations are functional but might be frequently updated based on customer feedback.

- `AWS/early-access-java-x86-to-graviton` - [Early Access] Validates Java application compatibility with Arm64 architecture for running on AWS Graviton Processors. Identifies and resolves Arm64 incompatibilities by updating dependencies, detecting architecture-specific code patterns, and recompiling native libraries when source code is available. Makes targeted code modifications necessary for Arm64 support, such as architecture detection, and native library loading, but does not perform general code refactoring. Maintains current Java version and JDK distribution and validates compatibility through build and test execution. For optimal results, run in an Arm64-based environment.

  > **Note**
  >
  > Many modern Java applications are already Arm64-compatible.

- `AWS/early-access-angular-to-react-migration` - [Early Access] Transform an Angular application to React.
- `AWS/vue.js-version-upgrade` - [Early Access] An early-access transformation for major version upgrades from Vue.js 2 to Vue.js 3 to modernize components, state management, routing, and global APIs to Vue.js 3 patterns. Minor or patch updates are outside the scope.
- `AWS/angular-version-upgrade` - [Early Access] This is an early-access transformation to transform an older Angular application to a target Angular version by upgrading components, services, templates, and routing to modern Angular patterns.
- `AWS/early-access-log4j-to-slf4j-migration` - [Early Access] This transformation migrates Java applications from Log4j (1.x/2.x) to SLF4J with Logback backend. Handles source code, dependency management (Maven/Gradle), and logging configuration files. Validates via compile, test, and residual import scan.

Customizing AWS-Managed Transformations

You can customize AWS-managed transformations to meet your organization's specific needs by providing additional context through the `additionalPlanContext` configuration parameter.

Example: Customizing Java version upgrade

```
codeRepositoryPath: ./my-project
transformationName: AWS/java-version-upgrade
buildCommand: mvn clean install
additionalPlanContext: |
  The target Java version to upgrade to is Java 17.
  Update all internal library dependencies to versions compatible with Java 17.
  Ensure compatibility with our custom authentication framework.
```

Example: Customizing AWS SDK migration

```
codeRepositoryPath: ./my-project
transformationName: AWS/java-aws-sdk-v1-to-v2
buildCommand: gradle build
additionalPlanContext: |
  Maintain our existing error handling patterns.
  Use our organization's standard credential provider chain.
  Update logging to use our internal logging framework.
```

Common Use Cases

This section provides step-by-step guidance for common transformation scenarios using the AWS Transform CLI.

Lambda Runtime Upgrades

Lambda periodically deprecates older language runtimes, requiring teams to upgrade their Lambda functions to supported versions. AWS Transform custom can automate these upgrades using the AWS-managed language version upgrade transformations. For example, to upgrade a Lambda function written in Python 3.9 to Python 3.13, you would use the `AWS/python-version-upgrade` transformation. Similar transformations are available for Java (`AWS/java-version-upgrade`) and Node.js (`AWS/nodejs-version-upgrade`), covering the most common Lambda runtime languages.

To upgrade a single Lambda function interactively, navigate to the root of your function's source repository and start the AWS Transform CLI with `atx`. In the interactive session, describe the upgrade you need—for example, "Upgrade this Python 3.9 Lambda function to Python 3.13." The agent will select the appropriate AWS-managed transformation, analyze your code, update

language syntax, adjust dependencies, and modernize deprecated API calls. You can provide a build or validation command such as `pytest` or `python -m py_compile *.py` so the agent can verify the transformed code compiles and passes tests. Throughout the interactive session, you can review the agent's proposed changes, provide feedback, and request adjustments before accepting the final result.

When you need to upgrade Lambda runtimes across many repositories, use non-interactive mode to run transformations at scale. For each repository, invoke the CLI with the appropriate flags to run without user input. For example, to upgrade a Node.js 16 Lambda function to Node.js 22:

```
atx custom def exec \
  -n AWS/nodejs-version-upgrade \
  -p ./my-lambda-repo \
  -c "npm test" \
  -g "additionalPlanContext='Upgrade from Node.js 16 to Node.js 22'" \
  -x -t
```

The `-x` flag runs the transformation non-interactively, and `-t` trusts all tool executions so no prompts are required. You can script this command across dozens or hundreds of repositories using a simple shell loop. After each transformation completes, review the Git diff to verify the changes and run your test suite to confirm the upgraded function works correctly. You can also integrate AWS Transform custom into your CI/CD pipeline to identify deprecated Lambda functions on an ongoing basis. This approach not only addresses the immediate problem of deprecated runtimes, but also establishes a continuous process to detect and upgrade functions before they reach end-of-support status.

Central platform teams can create campaigns through the AWS Transform web application to define the transformation, specify target Lambda repositories, and track progress across the organization. For a detailed walkthrough of building a production-grade scaled deployment solution using AWS Batch, see Building a scalable code modernization solution with AWS Transform custom.

AWS Transform custom and interface endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and AWS Transform custom by creating an *interface VPC endpoint*. Interface endpoints are powered by AWS PrivateLink, a technology that enables you to privately access AWS Transform custom services without an internet gateway,

NAT device, VPN connection, or Direct Connect connection. Traffic between your VPC and AWS Transform custom does not leave the Amazon network.

Each interface endpoint is represented by one or more Elastic Network Interfaces in your subnets.

For more information, see Interface VPC endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.

Note

- AWS PrivateLink integration with AWS Transform custom is available in US East (N. Virginia) (us-east-1) and Europe (Frankfurt) (eu-central-1) regions.
- You must enable AWS PrivateLink integration for Amazon S3 since AWS Transform custom makes S3 API calls. For detailed instructions, see the AWS PrivateLink for Amazon S3 documentation. If you encounter S3 access issues while using AWS Transform custom, refer to our troubleshooting guide.
- If you are not using AWS PrivateLink Private DNS feature (see Private DNS), you must:
  - Configure routing to VPC interface endpoints (see the Routing to VPC interface endpoints documentation)
  - Set the `ATX_CUSTOM_ENDPOINT` environment variable to specify your custom domain, for example:

```
ATX_CUSTOM_ENDPOINT=https://transform-custom.<region>.api.aws atx
```

Considerations for AWS Transform custom VPC endpoints

Before you set up an interface VPC endpoint for AWS Transform custom, ensure that you review Interface endpoint properties and limitations in the *Amazon VPC User Guide*.

AWS Transform custom supports making calls to all of its API actions through the interface endpoint.

Prerequisites

Before you begin any of the procedures below, ensure that you have the following:

- An AWS account with appropriate permissions to create and configure resources.

- A VPC already created in your AWS account.
- Familiarity with AWS services, especially Amazon VPC and AWS Transform custom.

Creating an interface VPC endpoint for AWS Transform custom

You can create a VPC endpoint for the AWS Transform custom service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Creating an interface endpoint in the *Amazon VPC User Guide*.

Create the following VPC endpoints for AWS Transform custom using this service name:

- com.amazonaws.*region*.transform-custom

Replace *region* with AWS Region where you desire to use AWS Transform custom CLI, for example, *com.amazonaws.us-east-1.transform-custom*.

For more information, see Accessing a service through an interface endpoint in the *Amazon VPC User Guide*.

Creating a VPC endpoint policy for AWS Transform custom

You can attach an endpoint policy to your VPC endpoint that controls access to AWS Transform custom. The policy specifies the following information:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Example: VPC endpoint policy for AWS Transform custom actions

The following is an example of an endpoint policy for AWS Transform custom. When attached to an endpoint, this policy grants access to the listed AWS Transform custom actions for all principals on all resources.

```
{
```

```
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "transform-custom:*"
         ],
         "Resource":"*"
      }
   ]
}
```

Using an on-premises computer to connect to a AWS Transform custom endpoint

This section describes the process of using an on-premises computer to connect to AWS Transform custom through a AWS PrivateLink endpoint in your AWS VPC.

1. Create a VPN connection between your on-premises device and your VPC.
2. Create an interface VPC endpoint for AWS Transform custom.
3. Set up an inbound Amazon Route 53 endpoint. This will enable you to use the DNS name of your AWS Transform custom endpoint from your on-premises device.

Troubleshooting

This section provides guidance for troubleshooting common issues with AWS Transform custom.

Log Locations

AWS Transform CLI maintains logs in the following locations:

Conversation logs:

```
~/.aws/atx/custom/<conversation_id>/logs/<timestamp>-conversation.log
```

These logs contain the full conversation history for debugging specific transformation executions.

Developer debug logs:

```
~/.aws/atx/logs/debug*.log
~/.aws/atx/logs/error.log
```

These logs provide detailed information about CLI operations and errors. Each log file has a 5MB limit before it rolls over. There may be multiple debug logs in this directory, such as debug1.log and debug2.log.

Common Issues

Installation issues:

If installation fails, ensure you have Node.js 20 or later installed:

```
node --version
```

Download Node.js from https://nodejs.org/en/download if needed.

Authentication issues:

Verify your AWS credentials are configured correctly:

```
aws sts get-caller-identity
```

Ensure your IAM user or role has the required `transform-custom:*` permissions.

Network connectivity issues:

If you encounter connection errors, verify network access to required endpoints:

- `transform-cli.awsstatic.com`
- `transform-custom.<region>.api.aws`
- `*.s3.amazonaws.com`

If working in an internet-restricted environment, update firewall rules to allowlist these URLs.

Region configuration issues:

If you encounter region-related errors:

- Verify your region is supported

- Check for environment variables that may override your configuration: `echo $AWS_REGION $AWS_DEFAULT_REGION`
- Check your region configuration: `aws configure get region`
- Update your region if needed: `aws configure set region <your-region>`
- Check debug logs for region resolution details

Git issues:

Ensure Git is installed and your repository is under git source control:

```
git --version
git status
```

AWS Transform custom requires repositories to be under git source control.

Transformation execution issues:

If a transformation fails:

1. Review the conversation logs at `~/.aws/atx/custom/<conversation_id>/logs/`
2. Check for build or test failures in the transformation output
3. Verify the build command is correct for your project
4. Try running the transformation in interactive mode to provide feedback

Conversation resumption issues:

If you cannot resume a conversation:

- Verify the conversation is less than 30 days old
- Check the conversation ID is correct
- Ensure you have network connectivity

Getting Support

For additional assistance, visit AWS Support through the AWS Console.

When opening a support ticket, include:

- Conversation logs from ~/.aws/atx/custom/<conversation_id>/logs/
- Debug logs from ~/.aws/atx/logs/
- Steps to reproduce the issue
- AWS Transform CLI version (atx --version)

S3 access issues

The AWS Transform custom service vends S3 pre-signed URLs to facilitate uploading and downloading the potentially large transformation files to/from client machines. This means that not only does the CLI interact with the AWS Transform service endpoints but it also interacts with the S3 service endpoints. These interactions use pre-signed URLs so your client's IAM credentials **do not** require any S3 permissions but sometimes your local machine's proxy server configurations and your network's S3 VPC Endpoint Policies can restrict this traffic.

Please examine the the section called “Log Locations” for more detailed error messages to help root cause any tool failures or S3 access denied relating to downloading and uploading transformation files.

If you are using a VPN/Proxy server that leverages the https_proxy and no_proxy environment variables, consider adding the following values to your no_proxy environment variable value to bypass the proxy for the S3 and AWS Transform custom service endpoints, for example:

```
export no_proxy=.s3.amazonaws.com,.transform-custom.<region>.api.aws
```

If you are using an AWS PrivateLink for Amazon S3 in your VPC, this may have a policy defined to restrict S3 traffic. Please ensure your S3 VPC endpoint policy allows GetObject and PutObject operations for the AWS Transform custom service-owned buckets. This can be accomplished by adding the following statement to your VPC endpoint policy:

```
{
    "Effect": "Allow",
    "Principal": "*",
    "Action": ["s3:PutObject","s3:GetObject"],
    "Resource": "arn:aws:s3:::aws-transform-custom-*/*"
}
```

Explicit Deny statements in policies take precedence over Allow statements. Review S3 VPC Endpoint policies for Deny statements that may be restricting access.

Discovery tool

The AWS Transform discovery tool enables you to automatically discover server inventory in your organization in preparation for migration. To use the discovery tool, you configure it, let it run, and then review the results in the Discovered Inventory pane.

After you configure vCenter access the discovery tool begins collecting information. The length of time that the discovery tool needs to run to completely analyze your network depends on the size of your VMware environment. For a directional migration business case you can use the VMware MPA file that the tool generates after the server collection has completed.

Discovery tool workflow

The workflow for the discovery tool consists of two types of activities:

- Configuration activities
- Data review and use

These steps describe the workflow to installing and using the discovery tool and making use of the collected data:

1. Installation of the discovery tool on vCenter
2. Set up vCenter access

   Data discovery begins after this step
3. Set up OS access and then review the collection status of VMware servers, databases, network connection.
   - Adjust OS credentials as needed.
4. To generate a migration business case, upload the ZIP file to Migration assessment or unzip it and upload *vmware_data_mpa.csv* from the *mpa_exports* directory.

Setting up the discovery tool

Installing the discovery tool

Prerequisites

These are the prerequisites for using AWS Transform discovery tool:

- VMware vCenter Server version 6.5, 6.7, 7.0 or 8.0
- You should have permissions to deploy an OVA into your VMware vCenter
- For VMware vCenter Server setup, make sure that you can provide vCenter credentials with Read and View permissions set for the System group
- The tool requires 4 vCPU, 16GB of RAM, and a 35GB hard disk
- DHCP must be available in the network for the discovery tool VM
- The tool collects data using a centralized approach. VMs that are in scope must allow inbound connectivity from the discovery tool VM (default ports, custom port configuration is supported):
  - Linux – SSH TCP/22
  - Windows – TCP/5985 for HTTP, TCP/5986 for HTTPS
  - SNMP – UDP/161
- For Linux, user accounts that can SSH into the server. For SSH discovery, the tool uses ss -tnap.
- The SSH user must be able to execute the ss command using sudo. If ss is not available, the tool will fall back to netstat.

Download the discovery tool

1. Sign in to vCenter as a VMware administrator and switch to the directory where you want to download the discovery tool OVA file.
2. Download the OVA file from this URL: https://s3.us-east-1.amazonaws.com/atx.discovery.collector.bundle/releases/latest/AWS-Transform-discovery-tool.ova

Deploy the discovery tool

1. Sign in to vCenter as a VMware administrator.
2. Use one of these ways to install the OVA file:

a. **Use the UI:** Choose **File**, choose **Deploy OVF Template**, select the discovery tool OVA file you downloaded in the previous section, and then complete the wizard. Ensure the proxy settings in the server management dashboard are configured correctly.

b. **Use the command line:** To install the discovery tool OVA file from the command line, download and use the VMware Open Virtualization Format Tool (ovftool). To download ovftool, select a release from the OVF Tool Documentation page. This is an example of using the ovftool command line tool to install the discovery tool OVA file.

```
ovftool --acceptAllEulas --name='discovery tool' --datastore=datastore1 -
dm=thin ATX-Transform-discovery-tool.ova 'vi://username:password@vcenterurl/
Datacenter/host/esxi/'
```

Descriptions of the replaceable values in the example:

- The name is the name that you want to use for your discovery tool VM.
- The datastore is the name of the datastore in your vCenter.
- The OVA file name is the name of the downloaded discovery tool OVA file.
- The username/password are your vCenter credentials.
- The vcenterurl is the URL of your vCenter.
- The vi path is the path to your VMware ESXi host.

3. Locate the deployed discovery tool in your vCenter. Right-click the VM, and then choose **Power**, **Power On**.
4. After a few minutes, the IP address of the discovery tool displays in vCenter. You use this IP address to connect to the discovery tool.

Discovery tool virtual machine specifications

- **Operating System** – Amazon Linux 2023
- **RAM** – 16 GB
- **CPU** – 4 cores
- **Disks** - 35 GB
- **VMware requirements** – See VMware host requirements for running AL2023 on VMware

Accessing the discovery tool VM

- The discovery tool VM comes by default with a username and password ("discovery", "password"). For strong security users are highly encouraged to update the password using `sudo passwd discovery` after logging into the VM through vSphere Client → Discovery Tool VM → "Launch Web Console".
- SSH access is disabled by default. Users can use preconfigured `enablessh` and `disablessh` aliases to enable/disable SSH access to the discovery tool VM. Users can SSH into the VM via `ssh discovery@<VM-IP>` after enabling SSH access. Users are encouraged to keep SSH access disabled most of the times and enable it only while actively required. Password change is enforced when running `enablessh`.
- To access the discovery tool data directory at `/home/ec2-user/.local/share/DiscoveryTool`, we recommend switching to `ec2-user` by running `sudo su ec2-user`.

Configure krb5.conf For Kerberos Authentication Protocol (optional)

krb5.conf configuration may not be required if your environment has proper DNS SRV records configured for Kerberos service discovery. However, explicit configuration is recommended for: Environments without DNS-based Kerberos discovery, Custom or non-standard Kerberos setups.

To configure the Kerberos authentication protocol on your discovery tool VM:

1. SSH to Discovery tool VM
2. Open krb5.conf configuration file in the `/etc` folder. To do so, you can use the following example `sudo nano /etc/krb5.conf`
3. Update the krb5.conf configuration file with at least the following information.

```
[realms]
<KERBEROS_REALM> = {
kdc = <KDC_hostname>
default_domain = <domain_name>
}
[domain_realm]
.<domain_name> = <KERBEROS_REALM>
<domain_name> = <KERBEROS_REALM>
```

Replace the placeholders with your actual values:

- <KERBEROS_REALM> - Your Kerberos realm (all uppercase, e.g., TOOL.EXAMPLE.COM)
- <KDC_hostname> - Hostname or IP address of your Key Distribution Center (e.g., domain-controller.example.com)
- <domain_name> - Your domain name (e.g., example.com)

For detailed configuration options, refer to the MIT Kerberos krb5.conf documentation and Sample krb5.conf file

Verify kerberos setup is working in the discovery tool VM by running `kinit <principal>` and `klist` to obtain and view the ticket. <principal> = username@DOMAIN (DOMAIN in all caps) e.g., testuser@EXAMPLE.COM).

Upon verifying, provide the principal and password in the discovery tool UI.

Import a self-signed certificate authority into the discovery tool (Optional)

This is required when you use WinRM over HTTPS and target servers using WinRM HTTPS certificates signed by a self-signed Certificate Authority (CA), and you want to enable "Validate server SSL certificate" on the discovery tool.

Prerequisites

1. Self-signed CA certificate that was used to sign the WinRM HTTPS certificates on target servers
2. Certificate in PEM format (.pem or .crt extension)

To import a self-signed certificate authority on the discovery tool VM:

1. Ssh to Discovery tool VM
2. Place the CA certificate(s) that signed your target servers' WinRM certificates into trust store directory `/etc/pki/ca-trust/source/anchors/` on the discovery tool VM. For example: `sudo cp winrm-ca.pem /etc/pki/ca-trust/source/anchors/winrm-ca.pem`. Note: If your target servers use certificates signed by different CAs, copy all relevant CA certificates to this directory.
3. Update the certificate trust store: `sudo update-ca-trust`

4. Reboot the VM
5. (Optional) To verify that certificates have been successfully imported, you can run the following command. `sudo trust list –filter=ca-anchors | grep -A 5 "<certificate_name>"`

See Installation and configuration for Windows Remote Management

Configure discovery tool access to vCenter

To configure discovery tool to access VCenter

1. In a web browser access: `https://ip_address:5000`, where *`ip_address`* is the IP address of the discovery tool from Deploy Discovery Tool. The discovery tool uses a self-signed certificate for HTTPS connection which results in a security warning. Choose **Accept the risk and continue** to continue to the discovery tool console.
2. If you're accessing the discovery tool console for the first time, create a discovery tool login password. Create a password, which you will use for future logins.

   > **Important**
   >
   > Remember this password - there is no password recovery mechanism.

3. On the **Discovery tool** page, under **Step 1. Set up vCenter access**, choose **Set up access**.
4. On the **Set up vCenter access** page, provide the **vCenter URL/IP**, the **vCenter username** and **vCenter password** and choose **Set up and connect**.

   The discovery tool begins to collect vCenter information, as described in Discovered VMware Inventory.

After initial configuration choose **Edit vCenter access** in the **Discovery tool status** frame to change your vCenter access settings.

Configure the discovery tool for OS access

Configure OS access so that the discovery tool can:

- Discover databases to perform database assessment and to assist in VM migration,

- Track network connections, including the process associated with the connection, to assist in application dependency mapping and wave planning.

Enable discovery tool OS Access

1. Navigate to the **Set up OS access** page to provide Windows and Linux credentials.
2. Choose a protocol from the dropdown menu.
3. Provide the required credentials for the selected protocol.
4. Select **Auto-connect** to enable the discovery tool to try all provided credentials on discovered servers until matching credentials are found for each server.

   See the section called "Using Auto-Connect Feature With Caution" for important security recommendations regarding the auto-connect feature.
5. Choose **Set up and connect**.

When the OS matching process is completed, you see a message that the data collection is in progress, and an error regarding servers for which a credentials match was not found.

Supported protocols setup

You must set up WinRM, SSH, and SNMP protocols on target servers for the discovery tool to communicate with them.

Set up WinRM and WMI

WinRM is automatically installed with all currently-supported versions of the Windows operating system.

To verify or edit WinRM configuration, use the `winrm` command line tool:

- Verify installed WinRM listeners: `winrm enumerate winrm/config/listener`
- Verify WinRM configurations: `winrm get winrm/config`
- Example command to set up WinRM: `winrm quickconfig -transport:https`

Listener Ports

Default HTTP port is 5985; HTTPS is 5986. You can use other ports as needed. The ports must be open between the discovery tool and target servers.

Encryption

The discovery tool uses encrypted WinRM communication. We recommend that WinRM listeners on target servers also use encryption: `winrm set winrm/config/service '@{AllowUnencrypted="false"}'`

NTLM vs Kerberos

WinRM authentication protocols Kerberos and NTLM are supported by the discovery tool. NTLM can be used only with HTTPS and Kerberos can be used with both HTTP or HTTPS.

WMI Requirements

Proper WMI access permissions are needed for remote PowerShell WMI query execution.

For network collection, ensure these conditions are met:

- Allow network connectivity via ICMP
- Allow network connectivity via TCP port 135 + ephemeral TCP port range (49152 - 65535)
- Disable UAC
- Remote DCOM permissions are set up
- Create a dedicated service account with minimal required permissions
- WMI namespace permissions are set up for Windows accounts with namespaces: `\\root\\standardcimv2`, `MSFT_NetTCPConnection` class

For database (SQL Server) collection, a Windows account (local or domain) belonging to the **Local Administrator Group** is required due to complex WMI objects permission requirements.

Set up SSH

- Port 22 must be open between the discovery tool and target servers
- For SSH network collection to work properly, provide a user configured for passwordless sudo
- Ensure either the `ss` or `netstat` command are available on the machines (should come installed by default)

Set up SNMP

- Port 161/UDP must be open between the discovery tool and target servers

- For SNMP v2: Provide a read-only community string that can access TCP connection OIDs.
- For SNMP v3: Provide username/password and auth/privacy details with read-only permission that can access TCP connection OIDs

The discovery tool requires access to:

- `"1.3.6.1.2.1.6.13.1.1." (tcpConnState)`
- `"1.3.6.1.2.1.6.19.1.8." (tcpConnectionProcess)`
- `"1.3.6.1.2.1.25.4.2.1.2." (hrSWRunName)`

Updating the discovery tool

The discovery tool does not have an automatic updates feature however you will receive a reminder notification after 30 days of installation to update. It is recommended to keep the application up-to-date to receive the latest features and security patches.

To manually update the tool

1. Obtain the latest discovery tool Open Virtualization Archive (OVA) file by downloading it from the provided link.
2. (Optional) We recommend that you delete the previous discovery tool OVA file, before you deploy the latest one.
3. Follow the steps in the Deploy discovery tool section to deploy the updated version.

Revoking vCenter access

Editing vCenter access to change only the **vCenter URL/IP** or to choose **Revoke access** deletes all of the data discovered by the discovery tool, including OS access configuration and discovered inventory.

Data collection

Scheduling the discovery tool

After your initial discovery collection, the discovery tool continues to run on this schedule:

- VMware discovery - every hour
- Database discovery - once a day
- Network metrics - every 15 seconds, may be less frequent for large environments

To manually run a collection, from the **Actions** menu choose:

- **Start** - enable the discovery module
- **Stop** - disable the discovery module
- **Collect data now** - use this to start discovery right away, for example, after making a change in your network.

OS data collection attempts

When a new server is discovered, the discovery tool attempts each configured credential for each IP address and the hostname. After the discovery tool finds a valid credential, it will continue to use that credential unless a new credential is added.

After a collection failure, the discovery tool attempts to collect networking data for a server after 3 minutes, 30 minutes, 2 hours, and then 6 hours. After 4 failed attempts, the discovery tool continues to try all configured credentials once every 6 hours.

Discovered VMware Inventory

After you set up the vCenter Access, the **Number of discovered VMs** displayed in the **Discovery tool status** frame begins to increment, and the **VMware discovery** status is shown as **Enabled** in the **Collection module** frame.

You can navigate to the **Discovered inventory** page and see the servers that are being discovered by the discovery tool. From this page, you can choose **Download inventory** to download a zip file containing MPA files listing the VMs that have been discovered with performance utilization data, database information, and server to server communication information.

You can download the zip file while the discovery tool continues to work, and obtain partial results. Upload this file to Migration assessment to obtain a business case for migration.

Data Points Collected

The discovery tool gathers comprehensive data across VMware, Database, and Network components. These sections detail the specific data points collected for each component.

VMware Data Collection

This table describes the VMware virtual machine information collected by the discovery tool:

Name	Type	Category	Sample Value
vm_name	String	VM Info	"w2k22-snmpd-v2-en-us-mssql-2022-testcase4-1"
vm_id	String	VM Info	"vm-30920"
vm_uuid	String	VM Info	"4201ecf8-cc44-ee7e-01da-34dfb2acf6c0"
powerstate	String	VM Info	"poweredOn"
host	String	VM Info	"esxi-70-node1.testlab.local"
primary_ip_address	String	VM Info	"192.168.0.52"
cpus	Integer	VM Info	2
memory	Integer	VM Info	4096
total_disk_capacity_mib	Integer	VM Info	32768
os_according_to_the_configuration_file	String	VM Info	"Microsoft Windows Server 2016 or later (64-bit)"
max_cpu_usage_pct_dec	Float	VM Performance	79.33
avg_cpu_usage_pct_dec	Float	VM Performance	45.06
max_ram_usage_pct_dec	Float	VM Performance	63.99
avg_ram_utl_pct_dec	Float	VM Performance	29.27

Discovery tool's OS-related data

Network collection

The Network collection module makes it possible for you to discover dependencies among servers in your on-premises data center. This network data accelerates your migration planning by providing visibility into how applications communicate across servers.

This module collects network data for the server inventory that comes from the VMware collection. It uses WinRM to collect data from Windows servers and uses SSH, SNMPv2, and SNMPv3 to collect data from Linux servers.

Network Data Collection

The Network collection module captures TCP IPv4 connections in ESTABLISHED or TIME_WAIT state. These data points are collected:

- Source IP, port, process ID, and process name
- Target IP, port, process ID, and process name
- State (ESTABLISHED and TIME_WAIT)
- Transport protocol (TCP)
- IP version (IPv4)
- Count (number of times this unique connection was observed)

Database collection

The Database collection module gathers database (SQL Server) information from Windows servers. The module uses WinRM protocol to remotely connect to each Windows server and run PowerShell queries to get information about all installed SQL Server services (components) on the server using WMI namespaces, registry, and file properties.

A SQL Server component is a specific service or feature instance installed as part of a SQL Server deployment on a Windows server. The discovery tool collects Database Engine, Analysis Services, Reporting Services, and Integration Services.

Database Data Collection

The Database collection module gathers SQL Server component information. This table describes key database data points collected:

Name	Type	Category	Sample Value
Engine Type	String	Component	sql_server
Is Engine Component	Boolean	Component	Y
Status	String	Service	Running, Stopped, StartPending
Version	String	Service	2015.131.5026.0
Edition	String	Service	Developer Edition (64-bit)
SQL Service Name	String	Service	MsDtsServer130, Mssql
SQL Service Type	String	Service	SQL Server service, Integration Services service
Instance Name	String	Instance	MSSQLSERVER
Display Name	String	Service	SQL Server (MSSQLSERVER2017)
Start Mode	String	Service	Automatic, Manual, Disabled
Service Account Name	String	Service	NT Service/MsDtsServer130
Is Clustered	Boolean	Configuration	N

Note: Full format includes all service types. MPA format includes only database engine components. Not all fields are available depending on the SQL service type and configuration.

Security considerations

Securing the discovery tool VM

Discovery tool VM security is crucial because the discovery tool stores all credentials, logs, and customer data on the discovery tool VM.

The discovery tool VM has ssh access disabled by default and can only be accessed from client `vCenter UI -> "Launch Web Console"`.

The discovery tool VM comes with a default login password "password" for user "discovery".

- We recommend that you update this password immediately after deployment.
- We **require** you to update the password if you want to enable ssh using the command `enablessh` after logging in using vCenter `Launch Web Console`. Please note, every time this command is called, you need to reset it.

Securing Credentials

General best practices for Credential Management

- Store credentials securely
- Regularly rotate all credentials
- Use password managers or secure vaults
- Monitor credential usage
- Follow the principle of least privilege and only grant the minimum necessary permissions needed

SNMP v2 Credentials

- Use complex, non-default community strings
- Avoid common strings like "public" or "private"
- Treat community strings like passwords

SNMP v3 Credentials

- Enable both authentication and privacy
- Use strong authentication protocols (SHA preferred over MD5)
- Use strong encryption protocols (AES preferred over DES)
- Use complex passwords for both auth and privacy
- Use unique usernames (avoid common names)

WinRM Credentials

- Avoid disabling WinRM certificate check.
- We recommend that you create a dedicated service account with minimal required permissions.
- Avoid using domain administrator or local administrator accounts when Database (SQL Server) Collection is not needed.

Using Auto-Connect Feature With Caution

The discovery tool uses two mechanisms to assign credentials to servers during *OS level collection* (Network and Database modules, need to connect to individual server (VM)): auto-connect and manual.

Manual: a server can be manually associated with a specific credential. In this case, the discovery tool will use that credential only, failure or success. The user has to manually monitor collection status for that server and make adjustment.

Auto-connect: if no credential is manually associated with the server, the discovery tool will use auto-connect mechanism for that server. That means:

- at the start of each collection round, the discovery tool will get a list of credentials available for that server (based on OS types) and also configured to be "auto-connectable".
- The discovery tool will then test all credentials against the server in a loop.
  - If a working credential is found, the collection round for the server is successful, the discovery tool remembers it and will try it first next time.
  - If no working credential is found, the collection round for the server has failed
    - Network module: the server will use a backoff schedule, starting next collection round in 3 min, 30 min, 2 hours, and 6 hours following each failure (similar to exponential backoff).
    - Database collection: there is no retry. The discovery tool will make 1 attempt for each server every day.

Impacts/Risks:

Only use auto-connect when you are sure that risks are mitigated in your system:

Risk 1: With auto-connect configured on multiple wrong credentials, automatically trying them against servers could trigger account lockouts for production environments where lockout policies are configured. For example, a data center can set its VMs to lock down after 3 failed SSH login attempts. In this case, if auto-connect is configured for 3 wrong SSH credentials, legitimate account

lockouts would happen. If lockouts happen across multiple systems, critical business processes could be impacted, potentially causing cascading failures in dependent systems. In addition, Security Operations Centers could experience alert storms from mass authentication failure events, creating false positive security incidents that drain resources and may mask real attacks.

Risk 2: Actor with access to the discovery tool (knows discovery tool password) can brute force OS credentials on all servers, by configuring a large number of test credentials and use auto-connect to find the successful ones.

Mitigations:

Follow these guidelines:

- Make sure that the discovery tool password is properly secured and known only to authorized actors
- Ensure proper credentials are entered for the environment if lockout policies are in place. We recommend only configure known, working credentials even if account lockout policies are absent to ensure minimal operational load on individual VMs.
- "Auto-connect" is an opt-in feature. Do not select it and use manual credential assignment if account lockouts is a concern to the environment.

Troubleshooting

Verifying discovery tool connectivity to vCenter

When you experience VMware module configuration errors follow these steps to verify connectivity:

Access the discovery tool VM

- Log-in to the discovery tool VM, open Remote Console in vCenter
  - Username: discovery
  - Password: password

Test vCenter Connectivity

1. Test vCenter API Access:

```
curl -v --insecure -u <username>:<password> https://<vcenter-ip-or-hostname>:443/
mob
```

2. Expected Success Output:

```
[ec2-user@discoverytool ~]$ curl -v --insecure -u <user>:<password> https://vcsa/
mob > tmp.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0*
 Trying 192.168.2.125:443...
* Connected to vcsa (192.168.2.125) port 443 (#0)
...
</xml>
* Connection #0 to host vcsa left intact
```

Test SSL Certificate

1. Run this command:

```
openssl s_client -showcerts -servername <hostname> -connect <hostname>:443
```

2. Expected Success Output:

   - Should show vSphere certificate details
   - Verifies SSL/TLS connectivity on port 443

```
[ec2-user@discoverytool ~]$ openssl s_client -showcerts -servername vcsa -connect
 vcsa:443
CONNECTED(00000003)
depth=0 CN = vcsa.onpremsim.env, C = US
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 CN = vcsa.onpremsim.env, C = US
verify error:num=21:unable to verify the first certificate
verify return:1
---
Certificate chain
 0 s:/CN=vcsa.onpremsim.env/C=US
```

```
   i:/CN=CA/DC=vsphere/DC=local/C=US/ST=California/O=vcsa.onpremsim.env/OU=VMware
 Engineering
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
---
Server certificate
subject=/CN=vcsa.onpremsim.env/C=US
issuer=/CN=CA/DC=vsphere/DC=local/C=US/ST=California/O=vcsa.onpremsim.env/OU=VMware
 Engineering
---
```

WinRM Troubleshooting

If you're experiencing connectivity issues with WinRM, follow these steps to test the connection:

Test basic WinRM connectivity using ports 5985 (HTTP) and 5986 (HTTPS). We need to make sure that connectivity works on port 5986 (HTTPS)

```
# Check WinRM listener configuration
winrm enumerate winrm/config/listener

# Note: Replace <HOST> with the target computer's hostname or IP address. Adjust the
 username and password as needed.
# Test WinRM connection on port 5985 (HTTP)
$cred = Get-Credential
Test-WSMan -Computer <HOST> -Authentication Negotiate -Credential $cred -Port 5985

# Test WinRM connection on port 5986 (HTTPS)
Test-WSMan -Computer <HOST> -Authentication Negotiate -Credential $cred -Port 5986
```

If the above tests fail, try establishing a PowerShell session with certificate validation disabled:

```
$cred = Get-Credential
$so = New-PsSessionOption -SkipCACheck -SkipCNCheck -SkipRevocationCheck
Enter-PSSession -ComputerName <HOST> -Credential $cred -Port 5985 -SessionOption $so
```

SNMP Troubleshooting

Access the discovery tool VM

- Log-in to the discovery tool VM, open Remote Console in vCenter
  - Username: discovery
  - Password: password

Install SNMP Tools (if needed)

- `sudo yum install net-snmp-utils -y`

Test SNMP Connection to Linux Servers

1. `snmptable -v 2c -c <COMMUNITY_STRING> <REMOTE_SERVER_IP> .1.3.6.1.2.1.6.13.1`
2. Example:

```
#SNMPv2c:
snmptable -v 2c -c public 192.168.1.100 .1.3.6.1.2.1.6.13.1

#SNMPv3 (with authentication):
snmptable -v 3 -u <username> -a MD5 -A <auth_password>
 192.168.1.100 .1.3.6.1.2.1.6.13.1

#SNMPv3 (with privacy):
snmptable -v 3 -u <username> -a MD5 -A <auth_password> -x DES -X <priv_password>
 192.168.1.100 .1.3.6.1.2.1.6.13.1
```

Network collection errors

a terminal is required to read the password

Error:

```
ss command failed on <host>: sudo: a terminal is required to read the password; either
 use the -S option to read from standard input or configure an askpass helper sudo: a
 password is required
```

The ss command is prompting for user password. The configured ssh user must be in the sudoers group and be configured with passwordless sudo for the ss/netstat command. To configure passwordless sudo:

1. Create a new sudoers file:

   ```
   sudo vi -f /etc/sudoers.d/<username>
   ```

2. Add the line:

   ```
   <username> ALL=(ALL) NOPASSWD: /usr/sbin/ss, /usr/bin/netstat
   ```

3. After this change, running `sudo ss -tnap` and `sudo netstat -tnap` should execute without prompting for a password

Access issues in Discovered inventory

If you see a message in **Server collection status** such as Missing credentials, or Access denied:

1. Select the server on the table of discovered servers.
2. Choose **Manage access credential** You can choose to:
   a. Select alternative credentials from the **Select credentials** dropdown.
   b. Select **Use new credentials** and provide new credentials.
3. **Save**.

The discovery tool retries the connection after you save your changes.

Common error messages

This table describes common UI messages and their explanations:

Message	Location	Explanation
One or more credentia ls contain unknown UUIDs	OS access page	Race condition when two users edit OS credentials at the same time; try again
A password has already been created	Create password page	Race condition when two users create passwords at the same time; refresh
Invalid password	Sign-in page	Incorrect password for logging in; contact admin or reach out
An on-demand collection is already in progress	Inventory page	Race condition when two users start manual collections at the same time; try again after the current manual collection is finished
An internal error occurred	Various pages	Retry or send logs
Export failed	Inventory page	Retry or send logs
Your session has expired. Please log in again.	Sign-in page	Session has timed out, need to login again

AWS Transform Connectors

AWS Transform connectors enable you to securely access resources across account boundaries for migration and modernization workflows. This enables you to access and manage resources across account boundaries while maintaining security controls and permissions. A connector represents a connection between your AWS Transform-enabled source account and external resources in AWS target accounts or in third-party systems.

Connectors require requests for access from the *source connector*, and approval from the owner of the target account. You make this request when you are setting up a source connector. Similarly, you may receive requests to approve *destination connectors* to allow other accounts access to your account.

- **Connector creation:** Create connectors in your AWS Transform-enabled source account, specifying the target account and required permissions
- **Permission Setup:** AWS Transform requires specific IAM roles and permissions in the destination account to perform migration actions, including:
  - Managing migration data with AWS KMS
  - Making cross-region API calls
  - Installing replication agents on VMware servers
  - Creating network infrastructure (VPCs, subnets, routing)
  - Launching Amazon EC2 instances and deploying CloudFormation stacks
  - Executing migration workflows through Migration Hub
- **Approval Process:** You must approve destination connector requests before AWS Transform can access your resources.
- **Active Connection:** Once you approve a connector, it enables AWS Transform jobs to securely access and manage resources in the target account.

User traceability for agentic operations

By default, AWS CloudTrail records agent actions on your AWS resources under the AWS Transform service identity. User traceability attributes those actions to the specific AWS IAM Identity Center user who initiated or interacted with the job, enabling security auditing and incident investigation at the individual-user level.

> **Note**
>
> User traceability is supported for IAM Identity Center (IDC) users only.

To enable user traceability, enable **Background session** in your AWS Transform profile settings. When enabled, AWS Transform automatically creates an IAM Identity Center background session when you interact with a job and uses it to produce identity-enhanced IAM role sessions for connector operations. These sessions embed your user identity in downstream AWS service calls and AWS CloudTrail events. When background sessions are disabled, AWS Transform falls back to standard assumed-role sessions without user identity context.

New connectors are automatically configured to support user traceability. For existing connectors, you must manually add `sts:SetContext` to the connector role's trust policy:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": { "Service": ["transform.amazonaws.com"] },
      "Action": ["sts:AssumeRole", "sts:SetContext"],
      "Condition": {
        "StringEquals": { "aws:SourceAccount": "your-account-id" }
      }
    }
  ]
}
```

You can view and revoke your active background sessions from the **Background sessions** tab on the job page. Revoking a session immediately invalidates it and any connector sessions derived from it.

For more information, see Trusted identity propagation overview in the *AWS IAM Identity Center User Guide*.

Managing connectors

On the Connectors page you can see the:

- **Source Connectors Tab:** Lists all connectors you've created to connect to other accounts, showing target accounts and connector status
- **Destination Connectors Tab:** Shows incoming connector requests from other accounts wanting to access your resources, and requiring your approval or rejection

AWS Transform requires confirmation for critical actions such as approving, rejecting, or deleting connectors, to avoid accidental changes that could disrupt your active migration workflows.

To create a connector

1. Navigate to the **Source Connectors** tab.
2. Specify the target account and resource type.
3. Configure required permissions and access scope.

To set up permissions

1. Choose to create a new IAM role with required permissions, or
2. Select an existing role you previously created for connectors.
3. Ensure the role has all necessary permissions for AWS Transform operations.

To manage incoming requests

1. Review connector requests in the **Destination Connectors tab**.
2. Approve legitimate requests to enable cross-account access.
3. Reject unauthorized or unnecessary connection attempts.

To manage your connectors

1. Monitor your active connectors for security and compliance.
2. Delete connectors when you no longer need them. Note that this will cause running jobs using the connector to fail.
3. Update permissions as your requirements change.

VMware migration

AWS Transform can help you migrate your VMware environment to Amazon EC2 by using generative AI. This document provides an overview of AWS Transform and of the workflow of the migration process.

Capabilities and key features

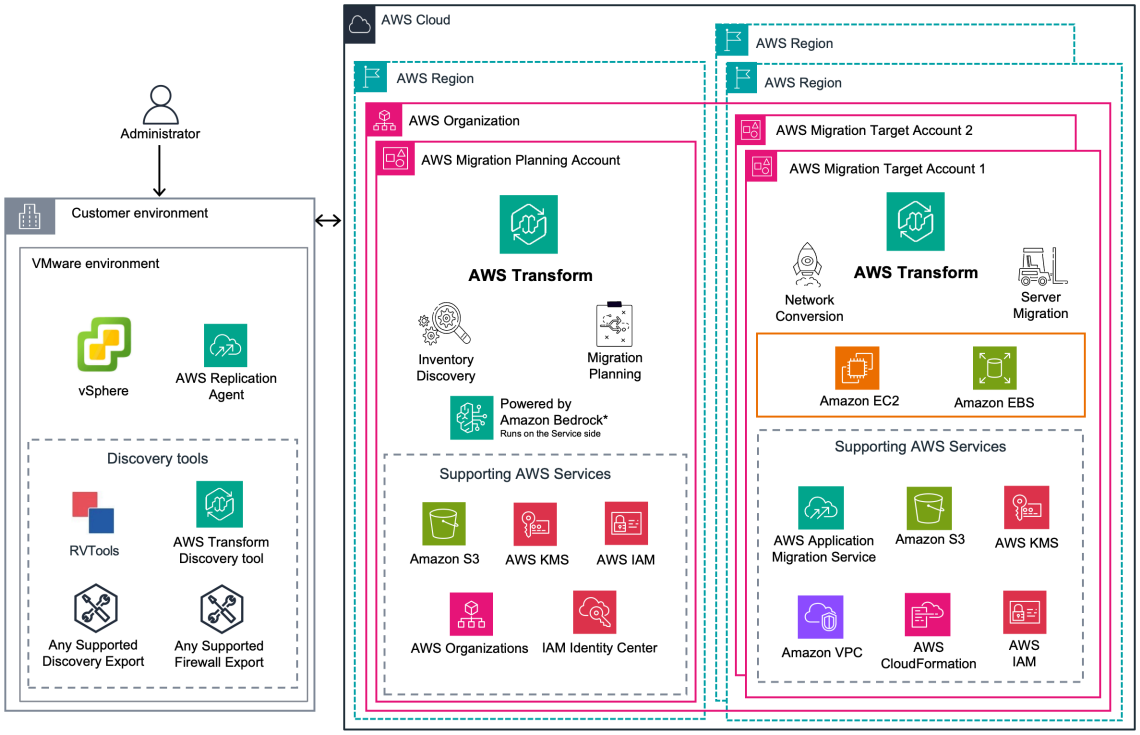
AWS Transform offers the following capabilities and key features for migrating your VMware environment to AWS.

- Three discovery options:
  - Assisted discovery of your VMware environment by using collectors from AWS Application Discovery Service.
  - Use the open-source Export for vCenter tool.
  - Importing independently collected discovery data.
- AI-driven conversion of your source VMware network configuration to an Amazon VPC network architecture.
- AI-driven generation of migration plans, including application grouping and suggested migration waves.
- Rehosting your servers to run natively on Amazon EC2.

AWS Transform supports migrating Windows and Linux servers of supported operating systems. For the full list of supported operating systems, see Supported operating systems in the *AWS Application Migration Service User Guide*.

AWS Transform VMware migration architecture

This diagram displays an overview of AWS Transform VMware migration architecture.



Limitations

AWS Transform has the following limitations:

- If you stop a running migration job, and then ask the agent to restart it, the job will start again from the beginning and you will lose any progress you have made in the job. However, artifacts created in the job before restarting it will still be available.
- You can specify one target AWS Region per VMware migration job. To migrate applications to different target Regions, create multiple VMware migration jobs.
- NSX imports are only supported for end-to-end migration jobs.

> **Important**
>
> AWS Transform generates network configurations and migration strategies based on your environment assessment. Review these configurations with stakeholders to ensure that they meet your organization's security, compliance, and business requirements. While AWS Transform provides automated configuration recommendations, you are responsible for

validating and adjusting the settings to match your security and compliance needs before proceeding with migration.

VMware migration jobs

To use AWS Transform for VMware migrations, you first need a workspace, which is a logical container in which you can create one or more transformation jobs. The sections in this topic describe how to get a workspace and how to create and start a VMware migration job in it.

Getting a workspace

For information about getting a workspace, see Getting started.

The workspace that you use determines the AWS Region where you can create transformation jobs. That is the AWS Region where your jobs will reside. Your discovery data and AWS Transform recommendations will also reside in this AWS Region. To create workspaces and jobs in a different AWS Region, ask your administrator to create a different workspace for you. For information about supported AWS Regions, see *Supported Regions*.

Even though the AWS Region where you can create jobs and store discovery data and recommendations is determined by your AWS Transform administrator, you can specify a different AWS Region as your target for the migration. In other words, you can run discovery and receive AWS Transform recommendations in one AWS Region, but then create your target environment in a different AWS Region. If you do that, you will be transferring your data across AWS Regions. For more information, see the section called “AWS account connectors”.

Job types

AWS Transform offers the following types of VMware migration jobs that you can choose from depending on your migration needs.

End-to-end migration

1. Perform discovery
2. Generate wave plan
3. Generate VPC configuration

4. (Optional) Deploy VPC networks
5. Migrate servers

Network migration only

1. Generate VPC configuration
2. (Optional) Deploy VPC networks

Network-and-server migration

1. Generate VPC configuration
2. (Optional) Deploy VPC networks
3. Migrate servers

Discovery and server migration

1. Perform discovery
2. Generate wave plan
3. Migrate servers

Creating and starting a job

The first step of a migration project is to create an AWS Transform job. For VMware migration projects, you can choose different job types, depending on your goals. The following procedure describes how to create and start a new VMware migration job of any type. For information about the different job types, see the section called “Job types”.

To create and start a new VMware migration job

1. On your workspace landing page, choose **Create a job**.
2. Choose the VMware migration option, and then specify the type of VMware migration job that you want to create. For information about the steps included in each of the four VMware migration job types, see the section called “Job types”.

3. After you answer all the chat questions, choose **Create job**.

VMware migration workflow

The type of VMware migration job you choose determines the workflow. For a description of the different types of VMware migration jobs, see the section called “Job types”.

Discover source data

To discover and collect on-premises data, you can use the AWS Transform discovery tool, AWSMigration Evaluator collector, or upload exports from RVTools and ModelizeIT, as well as AWSMigration Portfolio Assessment (MPA) format exports generated by tools like Cloudamize. AWS Transform can process AWS Transform discovery tool exports as CSV and JSON files in a ZIP file format, RVTools exports as either an Excel file or a ZIP file containing CSV files, and ModelizeIT CSV exports in a ZIP file. For each data source AWS Transform accepts the primary server file individually. However, supplementary files such as connections will only be processed when they are included in a ZIP file with the server files or in an Excel file with the server sheet.

We recommend that you upload the most detailed data available and review your export files before uploading to ensure data completeness and accuracy. Verify that all required files are included in your upload, and confirm that the data reflects your current environment state. This preliminary review helps minimize the need for re-uploads and ensures that AWS Transform can more accurately capture your on-premises environment and better support migration planning, including application grouping and wave planning. You can also incrementally add data as you obtain better data.

AWS Transform automatically detects file format and structure, parses and extracts structured entity records, removes duplicates across multiple files, validates data quality and reports issues, and then prepares a summary ready for downstream migration planning.

After you share and confirm your on-premises data, you can review discovery data by expanding **Discover on-premises** data in the **Job Plan** and choosing **Inventory readiness summary**. You can also ask questions about the data you uploaded to verify that AWS Transform correctly ingested everything or to identify and correct any mistakes in the data processing. For example, you can ask about operating system and versions. If you need to correct mistakes you can re-upload your inventory data from the discovery tool you used, and AWS Transform will automatically process, de-duplicate, and merge the updated records. You can also remove a previously uploaded file if you no longer want to use that inventory data.

Migration planning

The Migration planning job step within AWS Transform for VMware is a collaborative chat-based experience for planning large migrations. AWS Transform agents apply AWS Prescriptive Guidance to guide customers from analysis of on-premises data to finalized migration wave plans.

After the Discover on-premises data job completes successfully, AWS Transform uses the discovery data to group applications into migration waves. AWS Transform guides you through steps to analyze and scope your servers, group them into applications, generate move groups, and build migration waves. When analyzing your on-premises environment, you can ask questions to better understand how AWS Transform analyzed your installed software, for example, server dependencies and network architecture.

AWS Transform supports scope adjustments within application groups and waves. You can re-upload discovery data at any time, and AWS Transform will automatically process, de-duplicate, and merge new records with existing data. When changes are detected—such as newly discovered dependencies or infrastructure additions, AWS Transform will flag impacted dependency groups and provide recommendations for wave plan adjustments. Migration planning can also make use of unstructured text data to enrich the planning process.

There are four migration planning stages:

- In **Scope and analyze**, you can review your discovery data, ask questions about your software and network environment and determine the resources in scope for migration.
- In **Group apps**, you can provide a combination of business and technical rules regarding, for example, hostname analysis, network dependencies, and business rules, so that migration planning can group your infrastructure into applications. If you already have an inventory of your applications, migration planning can use that instead.
- In **Generate move groups**, you can give migration planning your technical and business requirements so that it can determine which applications must be moved together. Technical dependencies include databases, message queues, or other resources shared between multiple apps. Business and operational dependencies include business criticality, RPO and RTO, data center location, and application owners.
- Finally, in **Build waves**, you can give migration planning context about your timelines and priorities so it can build a wave plan that you can migrate. You can select move groups for inclusion in a wave based on factors such as priority score, move group size, user counts, and application complexity.

Migration Planning Terminology:

- *Migration waves* are logical groups that are migrated together. Migration waves are comprised of one or more move groups.
- A *move group* is a set of co-dependent applications that must be moved together. They may have technical dependencies such as a shared database or they have business dependencies such as supporting a shared business function.
- A *dependency* is a relationship between systems. There are several types of dependencies including:
  - Critical or hard dependencies, where systems cannot operate without the dependency. Common examples of this are applications that depend on databases, other applications, or services.
  - Soft dependencies, which are not critical for the operation of the system. Common examples of this include latency insensitive dependencies that can be migrated independently.
  - Non-technical dependencies include business, organizational, operational and compliance dependencies. These are dependencies related to your organization and its priorities. Examples of this include shared business function and organizational ownership.

Workflow

Migration planning is an interactive and iterative workflow. You can go back and make changes to previous steps at any time. A typical migration planning workflow is:

1. Migration planning starts by summarizing the discovery data that is available. Review the available data and return to the discovery step at any time to provide additional data.
2. Within the scoping and analysis step you can ask questions about your on-premises environment to validate the data you have collected. Example questions include:
   a. List my servers by operating system
   b. Summarize my on-premises network topology
   c. List the most common technologies running in my environment.
3. While analyzing your environment, if you identify servers that should not be in scope for migration you can tell AWS Transform to exclude those resources. Examples of this include:
   a. Remove all servers which have *legacy* in their hostname
   b. Remove all servers in the 10.0.2.0/24 subnet
   c. Remove all servers running versions of Windows older than 2022

4. Once you have sufficiently explored your environment and determined your migration scope you can tell AWS Transform to move to the next migration planning step.
5. The next step is application grouping. If you already have your servers mapped to applications, you can tell AWS Transform to use that mapping and skip this step. If you do not have your applications pre-defined you can provide the technical and business logic that defines your applications. AWS Transform will guide you through the application grouping process and suggest the data points that you can provide to effectively group your servers together into applications. The more information you can provide about your on-premises applications, the more effectively AWS Transform can group your servers into apps. Once you have provided sufficient information, you can then instruct AWS Transform to perform application grouping.
6. Once application grouping has been performed, review the application groups. You can instruct AWS Transform to make any necessary changes, for example:
   a. Move server example-server to application-5
   b. Rename application-5 "HR App Test Environment"
   c. Remove all Linux servers from IIS Dev Farm
7. Once your apps are grouped, instruct AWS Transform to move to the next step
8. The next step is move grouping. In the move grouping step you identify applications that must be moved together. Provide context around your technical and non-technical dependencies. AWS Transform will guide you through the process and suggest data points that you can provide to group your apps together. There are several considerations to make at this stage including:
   a. What should be the target size of move group?
   b. Do you want to combine environments, for example dev, test, and prod, for each app, or split them?
   c. How do you want to consider network dependencies? Are all dependencies critical or can some dependencies be considered soft dependencies and be split across move groups?
9. Once you have provided your rules for move grouping, instruct AWS Transform to execute your move grouping strategy. You can then review and modify your move groups. Once you have reviewed your move groups you can instruct AWS Transform to move to the final migration planning step.
10. Wave planning is the final step within migration planning. In this step, you group your move groups into migration waves and prioritize those waves. Within the wave planning step, AWS Transform will guide you through providing the business prioritization required to group your move groups into waves and then prioritize those waves. Considerations within wave planning include:

a. The business criticality of each of your move groups

b. The migration timelines and the timelines for each move group

c. The risks associated with each move group

d. The number of servers to migrate per wave

11. Once you have provided sufficient guidance on how to group into waves, instruct AWS Transform to execute the wave planning. You can then review your waves and modify them.

12. Once you have finalized your wave plan, you can complete migration planning and move to execution. You can return to migration planning at any time to refine and iterate on your plan.

Connect target account

The target account is where your network will be deployed and where your migrated servers and applications will reside in AWS.

> **Important**
>
> - AWS Transform will create an Amazon S3 bucket on your behalf in this target AWS account. This bucket won't have `SecureTransport` enabled by default. If you want the bucket policy to include secure transport, you must update the policy yourself. For more information, see Security best practices for Amazon S3.

To use an existing target account connector

1. In the **Job Plan** pane, expand **Choose target account**, and then choose **Create or select connectors**.
2. In the **Collaboration** tab, select an existing connector if your workspace already has connectors, and then choose **Use connector**. In the list of available connectors, if a connector is grayed out, that means its version isn't compatible with the job type that you selected earlier.

> **Important**
>
> If you specify a connector with a target AWS Region that is different from the AWS Transform Region, that means AWS Transform will be transferring your data across AWS Regions.

3. Choose **Continue**.

To create a new connector

1. In the **Job Plan** pane, expand **Connect target account**, and then choose **Create or select connectors**.
2. Specify the AWS account and AWS Region that you want to use as your target, and then choose **Next**.

> **Important**
>
> If you specify a connector with a target AWS Region that is different from the discovery AWS Region, that means AWS Transform will be transferring your data across AWS Regions.

3. Choose whether you want to use Amazon S3 managed keys for encryption. If you specify your own KMS key, you can use the default key policy. However, if you want a less permissive key policy, the following is an example. For information about how to create a KMS key, see Create a KMS key in the *AWS Key Management Service Developer Guide*.

   AWS Transform uses the `kms:DescribeKey` permission to make sure the key exists. It uses the `kms:GenerateDataKey` and `kms:Decrypt` permissions to encrypt and decrypt the transformation job data in the Amazon S3 bucket.

   AWS Transform uses default Amazon S3 encryption. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys
4. Choose **Continue**.
5. Copy the verification link, share it with an administrator of the target AWS account, and ask them to approve the connection request.

6. After the administrator of the AWS account approves the request, select the newly created connector from the list of connectors in the **Collaboration** tab, and then choose **Use connector**.
7. Choose **Send to AWS Transform**.

If you plan to modify the AWS Application Migration Service template to enable post-launch actions, add the following permission to the target connector role. You can find the name of that role in the **Collaboration** tab after the connector is created. For information about how to add permissions to a role, see Update permissions for a role in the *IAM User Guide*.

```
{
      "Sid": "MGNPostLaunchActions",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::target-account-ID:role/service-role/
AWSApplicationMigrationLaunchInstanceWithSsmRole"
}
```

Migrate network

AWS Transform migrates VMware networks to AWS by translating your source environment configuration into AWS-equivalent network resources. AWS Transform analyzes your source network data and creates VPCs, subnets, security groups, NAT gateways, transit gateways, elastic IPs, routes, and route tables as needed. You can review and modify the generated network configuration before deployment. For deployment, you can either have AWS Transform deploy the configuration for you and analyze deployed network connectivity, or choose self-deployment—in which case AWS Transform generates Infrastructure as Code (IaC) in your preferred format: AWS Cloud Development Kit (AWS CDK), Landing Zone Accelerator (LZA), or HashiCorp Terraform.

Source Network Mapping

The network mapping process requires uploading a configuration file from your source environment. The tool you choose depends on your source network type:

- **Software Defined Networks (SDN):** Import/Export for VMware NSX network virtualization or Cisco ACI config for Cisco Application Centric Infrastructure.

- **VMware vSphere networks:** RVTools. Note: when using RVTools files, AWS Transform will focus on generate Amazon VPC configurations, while security group configurations require additional input. For network security settings, you may upload configuration files from additional sources like firewalls and software-defined networks. See ??? for more details.
- **Networks based on firewall configuration data:** Export files from Palo Alto Networks Firewall or Fortinet FortiGate Firewall.
- **Hybrid networks running both VMware and non-VMware workloads:** Application mapping tools - modelizeIT.

> **Warning**
>
> The official RVTools site is https://www.dell.com/en-us/shop/vmware/sl/rvtools, which is the site that this guide links to in steps that mention RVTools. Beware of the scam site (rvtools)(dot)(org).

AWS Transform creates VPCs from all source network segments, with each detected segment becoming its own distinct VPC. Network segmentation varies by source type:

- **vNetwork:** AWS Transform groups VMs by vSwitch and VLAN. VLANs can appear under multiple vSwitches (except VLAN 0).
- **NSX networks:** AWS Transform segments the network based on Tier-1 routers, grouping the routers and collecting their segments.

The network mapping process generates these key resources:

- **Network topology:** Network deployment best practice for implementation. See the Network topologies section for more details.
- **Workload segment configuration:** Amazon VPC segments with CIDR block definitions for organizing workloads and managing network traffic flow.
- **Security configuration:** Pre-configured access rules for different network segments, supporting ingress and egress traffic control.

> **Note**
>
> AWS Transform tags all generated resources with "CreatedBy": "AWSTransform" along with definition and execution IDs for tracking and management purposes.

Firewall and Software-Defined Network Configuration Files

AWS Transform supports configuration files that enable automated network and security group generation, based on firewall and policies configurations. These files can be used standalone or as a complement to RVTools file.

- Cisco Application Centric Infrastructure (ACI): Network policy configurations
- Palo Alto Networks: Firewall security policies
- Fortinet FortiGate: Firewall security policies

When you upload a firewall or Cisco ACI file, AWS Transform generates network infrastructure and security groups. When you upload an RVTools file, AWS Transform generates network infrastructure only. You can optionally add firewall or Cisco ACI file to generate security groups.

To extract configuration files from firewall and network environments, follow these procedures. Consult vendor documentation for the latest information.

Fortinet FortiGate

- Firmware: v7.0 and up
- Requirements: `super_admin` or `super_admin_readonly` privileges on global level
- Steps:
  - 1. Connect to the firewall via SSH or built-in CLI client
  - 2. Run: `show | grep ""` (`| grep ""` disables pagination)
  - 3. Save all output to a file starting from the `show` command

Palo Alto Networks

- Firmware: 10.1 and up
- Requirements: superadmin role

- Steps: Connect via SSH, run commands below, save outputs to palo-conf.txt and palo-default.txt:
  - `set cli pager off`
  - `set cli config-output-format set`
  - `configure`
  - `show` # Save as palo-conf.txt
  - `show predefined` # Save as palo-default.txt

Cisco ACI

- Firmware: 6.0 and up
- Requirements: Admin role with all privileges; SCP/SFTP/FTP destination configured
- Steps:
  - Connect to APIC controller via browser
  - Go to Admin >> Config Rollbacks
  - In “Take a snapshot” select remote location and push “Create a snapshot now” button
  - After receiving “Transfer successful” message, connect to the remote location server and retrieve the latest snapshot file (.gz file)

Network Topologies

During the migration to the target network you can choose the **Isolated VPCs** topology or the **Hub and Spoke** topology.

> **Important**
>
> For both topologies AWS Transform does not open the communication to the internet. You must open it manually after taking appropriate security precautions.

Isolated VPCs

These are independent network environments that operate as separate units within AWS . VPCs maintain complete network isolation, with no built-in communication pathways between them. This separation provides the highest level of network boundary protection. You can connect the VPCs through specific networking configurations if needed.

Hub and Spoke

In this model, an AWS Transit Gateway created by AWS Transform acts as the hub that connects to multiple workload VPCs (the spokes). During network convergence, AWS Transform creates a spoke VPC for each detected source network segment.

AWS Transform creates three specialized VPCs for traffic management and security:

- Inspection VPC: Where you establish the firewall that inspects the traffic. You can create firewall rule configurations here to modify VPC connections.
- Inbound VPC: For all traffic from the public internet (north-south). Includes an internet gateway.
- Outbound VPC: For all traffic to the public internet. Has an internet gateway, a Network Address Translation (NAT) gateway and an elastic IP address.

AWS Transform automatically associates all spoke VPCs with the default association route table and propagates routes from all spoke VPCs to the default propagation route table. This automation creates routing paths without manual configuration, though traffic flow remains subject to security group permissions.

If you want fine-grained control over the communication between the VPCs, choose the **Isolated VPCs** option and modify the generated network to create the specific communication paths your require.

IP migration approaches

The system offers two key network configuration choices for your migration

Network range selection:

- **Keep Existing Ranges (IP Address Ranges Retention):** Keep original IP address ranges during migration. Ideal for lift-and-shift scenarios with legacy applications that have hard-coded IP dependencies or existing firewall rules.
- **Update to new IP ranges (CIDR update):** You can modify each VPC CIDR range during migration, and AWS Transform automatically propagates changes to subnets, route tables, and security groups.

IP addresses assignment:

- **Fixed IP addresses (Static):** the system assigns static IPs based on the CIDR. This is best for applications requiring predictable network behavior, DNS management, or IP-based access control. IPs persist across instance restarts using Elastic Network Interfaces (ENIs).
- **Dynamic IP assignment (AWS DHCP):** Automatically assign IPs from subnet pools at instance launch. Optimal for cloud-native applications and auto-scaling workloads. Reduces operational overhead but requires applications to use DNS or service discovery.

You can combine either range selection with either IP assignment method.

> **Note**
>
> IP addresses assignment strategy is set at the wave level. You can assign different strategies to specific servers by customizing the wave file. For example if you chose a static IP address approach for the wave, but want to assign a dynamic approach to a specific server, you would use `[RESET_VALUE]` as described in Editing your configuration In the *Application Migration Service user guide*.

> **Important**
>
> When you choose to create security groups, you cannot use Dynamic Host Configuration Protocol (DHCP) for server migrations. Security groups use Classless Inter-Domain Routing (CIDR) configurations, and enabling DHCP could compromise your network's security posture.

Review VPC Configurations

After AWS Transform generates Amazon VPC configurations, it displays the generated VPC networks. You can either use the current configuration or modify VPC CIDRs. Note: You cannot modify the prefix length (the value after the "/") or any other resources.

To modify VPC CIDRs:

1. In the Generated VPCs list, provide your modified CIDRs.
2. Choose **Submit** to apply the changes and rerun the mapping process.

3. Review the results, then either continue with network deployment or repeat the modification steps.

Deploy Network

After reviewing and approving the generated network configuration, choose to deploy using AWS Transform or on your own.

> **Note**
>
> Ensure your target account has the required quotas before beginning deployment.

Deployment Options:

- **AWS Transform-managed deployment:** AWS Transform uses CloudFormation templates to deploy your network and runs Reachability Analyzer to check connectivity between subnets across multiple VPCs and within the same VPC.

  > **Note**
  >
  > Network deployment requests require explicit approval before execution. See Deployment approvals process for more details.

- **Self-deployment:** AWS Transform generates Infrastructure as Code (IaC) templates in the following formats:
  - CloudFormation: Templates for provisioning network resources
  - AWS CDK: TypeScript project for programmatic infrastructure deployment
  - HashiCorp Terraform: HCL templates for managing network resources
  - Landing Zone Accelerator (LZA): A network-config.yaml file for LZA network configuration. See Using configuration files in the Landing Zone Accelerator on AWS Implementation Guide.

> **Note**
>
> When deploying this network configuration via the Landing Zone Accelerator (LZA) pipeline, ensure that your AWS Transform account and LZA installation are in the same AWS Organization. Deployment will fail if there is a mismatch between the Organizations

IDs used in AWS Transform and LZA. To learn how to set up your LZA installation using Organizations see AWS Organizations based installation (without AWS CloudTrail).

After you select a network configuration format, use the link provided to download a zip file containing the generated templates. The zip folder includes a README.md file that explains how to use the generated templates.

To verify the downloaded file hasn't been corrupted or tampered with, generate and download a checksum, then compare it to a locally generated hash using `openssl dgst -sha256 -binary <file.zip> | base64` command.

Deployment approvals process

Network deployment requests require explicit approval before execution. When you submit a deployment request, it automatically routes to authorized approvers through the AWS Transform Approvals tab. Approvers validate both CloudFormation templates and network configurations to ensure compliance with security standards and architectural requirements. Each submission triggers a new review cycle, and deployments proceed only after receiving confirmation. If an approver denies your request, contact them directly to discuss necessary modifications. The system tracks all approval decisions for audit purposes and maintains deployment history.

Security group association

AWS Transform creates security groups based on your source environment configurations when migrating from NSX environments. AWS Transform can generate security groups from RVTools files when combined with Additional configuration files from sources such as firewalls and software-defined networks.

Important

AWS Transform makes a best effort to create security groups that match your source environment. It is your responsibility to review and, if necessary, modify the security groups to ensure that they meet your company's needs and security policies.

AWS Transform converts the following configurations to security groups:

- Security policies and security policy rules

- Gateway policies and gateway policy rules

Tag network resources

AWS Transform automatically tags all generated resources with `"CreatedBy": "AWSTransform"` along with definition and execution IDs for tracking purposes. You can also add custom tags to network resources during migration to track and manage your infrastructure.

You can apply custom tags at two levels:

- **Job-level tags:** Applied to all resources created during the migration job, including VPCs, subnets, security groups, and route tables.
- **VPC-level tags:** Applied to specific VPC resources and their associated components.

> **Note**
>
> If your migration is part of the **AWS Migration Acceleration Program (MAP 2.0)**, you can include the required MAP tag:
>
> - **Key:** `map-migrated` **Value:** `mig`*`MPE_ID`*
>
>   (where *`MPE_ID`* is your Migration Portfolio Evaluation identifier)

AWS Transform automatically apply these tags during network deployment.

To use existing AWS network resources not created by AWS Transform, you must tag the resources (including VPCs and subnets). AWS Transform can tag resources during migration wave execution — it will tag all network resources in the target AWS account and AWS Region. Alternatively, you can manually tag network resources you've created with the following tags:

- **Key:** CreatedFor **Value:** AWSTransform
- **Key:** ATWorkspace **Value:** workspace ID

Find your workspace ID in the AWS Transform web app URL, https:// ... /workspace/*`workspace-id`*/job/job-id

Learn more about how to tag network resources in the section called “VPC and subnet tags”.

Migration to multiple target accounts

AWS Transform supports migrating VMware workloads to multiple AWS accounts simultaneously. This capability enables you to migrate workloads directly to their intended target accounts while maintaining your organization's security boundaries and governance structures.

Benefits

Multi-account migration provides the following benefits:

- *Maintain security boundaries* - Migrate workloads directly to accounts that align with your business units and security requirements
- *Unified management* - Control the entire migration process from a single management interface

Limitations

Multi-account migration has the following limitations:

- *Single region only* - You can migrate to multiple accounts within a single AWS Region. For multi-region migrations, you must create separate projects for each target region
- *One account per wave* - Each migration wave can target only one account. Applications requiring different target accounts must be placed in separate waves
- *AWS Organizations required* - All target accounts must be part of an AWS Organization

Prerequisites

Before you begin a multi-account migration, ensure you have the following:

- An AWS Organization containing all target accounts for your migration
- A designated Management Account with Delegated administrator or an account with *Manager* permissions in the AWS Organization.
- Appropriate permissions initialized across all target accounts. Set up these roles using the link to the Application Migration Service console, provided to you by AWS Transform when it requests that you set up these permissions.
  1. If you're using Delegated administrator permissions, you can select delegated administrators.
  2. You can **View Roles** to learn about the permissions.

3. Select **Initialize Multi Account Network Migration**. Return to AWS Transform and report that you successfully initialized multi account network migration.

Implementation overview

Multi-account migration follows this high-level process:

1. After you've kicked off migration and connected your discovery account tell AWS Transform that you want to start network migration.
2. AWS Transform asks if you want to deploy to **Single Account** or **Multi Account**
3. Create or select a target connector. Learn more in Connect target account. The target account must be an account in your Organizations with Delegated administrator or Manager permissions.
4. Provide your source network file.
5. *Review network translation* - AWS Transform displays the target network configuration, including which resources deploy to which accounts. After your VPC networks are generated you can modify CIDR ranges and select targets across multiple accounts in the table provided in the human-in-the-loop (HITL) pane.
6. Generate Infrastructure as code (IaC) files - The IaC files (CDK, CFT, Terraform and LZA-compatible yaml) that AWS Transform generates include any changes you make to the target accounts and CIDR ranges.
7. *Deploy network infrastructure* - Choose to deploy the network yourself or have AWS Transform deploy it across your target accounts.
8. *Execute server migration* - AWS Transform migrates servers to their assigned target accounts (using Application Migration Service global view).

> **Note**
>
> The inventory file that you provide to AWS Transform should include the target account configuration. The workloads, and the subnets they use, should go to the same target account.

Set up service permissions

In this step, you initialize the AWS Application Migration Service (Application Migration Service) if you haven't already. To learn more about this requirement, see Initializing Application Migration Service with the console or Initializing AWS Application Migration Service with the API.

After you initiate Application Migration Service, AWS Transform helps you add MAP tags to your resources (if you have a MAP 2.0 agreement in place) so that you can get MAP credit. For information about MAP, see AWS Migration Acceleration Program.

Prepare and migrate waves

At this stage, you will see migration waves in the **Job Plan** pane. For each wave, perform the following steps. In some of these steps you will have the option of importing an updated inventory file. AWS Transform allows one import to a given target AWS account and target AWS Region at a time. This means that if you work on more than one wave simultaneously, or if there is more than one migration job running with the same target account, you must wait for an import to finish before you can perform another import in a different wave or job.

Prepare waves

Each wave includes a **Set up migration wave** task. On its **Collaboration** tab you can configure the wave's settings.

Set up migration wave

1. In the **Job Plan** pane, expand the step **Set up migration waves**, and then choose **Set EC2 recommendation preferences**. Follow the instructions in the right pane, and then choose **Continue**. Learn more about Amazon EC2 recommendations in Generating Amazon EC2 recommendations in AWS Migration Hub.
2. In the **Staging area subnet** section, you can choose a staging area subnet from the dropdown menu of the available subnets.

   Only subnets that are tagged in VPCs *that are also tagged* with these tag key-value pairs appear in the list. Learn more in the section called “VPC and subnet tags”.
3. For each wave, choose your IP assigment approach:

   - **Use source IP or the converted IP from the new CIDR**

- **Use new IP using DHCP**

4. In the **Job Plan** pane, choose **Confirm inventory for *`wave-name`***. Download the inventory file and review the list of servers and Amazon EC2 configurations. Modify the file if necessary, but do not remove columns or change the titles of the existing columns. You can control the operating system licensing options (BYOL / LI) and tenancy by specifying the configuration in columns with these headers: mgn:launch:placement:operating-system-licensing and mgn:launch:placement:tenancy. Learn more in Import parameters in the *Application Migration Service user guide*. After you choose whether to continue with the file you downloaded or to upload a version of the file that you updated, choose **Continue**.

> **Note**
>
> AWS Transform provides Amazon EC2 recommendations based on the utilization specification of your source VMs. You can modify the suggested Amazon EC2 instance types to include recommendations from the Migration Evaluator, AWS Optimization and Licensing Assessment (OLA), or a *Migration assessment* job.

VPC and subnet tags

Your VPCs and their subnets are automatically tagged with these tags so that their subnets appear in AWS Transform's list of available subnets:

- **Key:** `CreatedFor` **Value:** `AWSTransform`
- **Key:** `ATWorkspace` **Value:** *`workspace ID`*

Migrate waves

When you migrate a wave, AWS Transform keeps you informed of the progress by providing a table in the **Collaboration** pane. You can also ask AWS Transform about the status of the migration in natural language, for example:

- What is the status of my servers?
- What's the status of my wave?
- What's the status of the step that I'm currently in?

Deploy replication agents

1. In the **Job Plan** pane, expand **Deploy replication agents**, and then choose **Start replication agent deployment**. You have two options:

   - **Use AWS Transform to automate deployment:** To automate the deployment of the agents on the source servers in this wave, AWS Transform uses an MGN connector already deployed in your account. For information about how to deploy an MGN connector in your account, see Set up the MGN Connector in the *Application Migration Service User Guide*.

     To use this option, perform the following steps:

     1. Open the AWS Systems Manager console at https://console.aws.amazon.com/systems-manager/.
     2. In the left navigation pane, under **Node Tools**, choose **Fleet Manager**.
     3. Choose the name of the managed instance of the MGN connector that you want AWS Transform to use for this wave.
     4. Tag the managed instance with the following key-value pairs.
        - Key: `CreatedFor` Value: `AWSTransform`
        - Key: `ATWorkspace` Value: *`workspace ID`*

        Find your workspace ID in the AWS Transform web app URL, https:// ... / workspace/*`workspace-id`*/job/job-id
     5. In AWS Transform, choose **Use AWS Transform to automate deployment**.
     6. Specify the MGN connector that you tagged and the AWS Secrets Manager secret that you want AWS Transform to use for this wave. You must create a single set of credentials for the MGN connector to use for deploying replication agents on all servers in a particular wave. For information about setting up the secret, see Register server credentials.
     7. If AWS Transform encounters errors during the deployment of the agent, you will see those errors in the **Job Plan** pane. Choose each error in the **Job Plan** pane to view its details in the **Collaboration** tab.
     8. After you resolve all errors, you can track the replication status for the wave by choosing **Review replication status** in the **Job Plan** pane.

   - **Deploy replication agents on your own:** You can deploy the replication agents on the source servers manually. Alternatively, you can use the MGN connector or another

automation framework to deploy them on your own. For information about how to set up the MGN connector, see Set up the MGN Connector in the *Application Migration Service User Guide*.

To deploy the replication agents manually, or use an automation framework other than the MGN Connector to deploy them, perform the following steps.

1. Go to the AWS Application Migration Service console, and export a list of your servers. For instructions, see Exporting your data inventory.
2. Filter the list by wave to obtain a list of the servers in the current wave.
3. Follow the instructions under Installing the AWS Replication Agent. Specify the `user-provided-id` parameter, and for every server set its value to the server's `mgn:server:user-provided-id` as it appears in the .csv file that you exported from AWS Application Migration Service. AWS Transform connects the replication agent with the imported server using this parameter. If it's not provided, MGN will create a separate instance of source server for each agent that is installed.

To see the replication agent installation status, check the AWS Systems Manager run command history at agent installation time. For information, see Understanding command statuses in the *AWS Systems Manager User Guide*.

To see the replication status in real-time, go to the AWS Application Migration Service console. Status updates in the AWS Transform web app are delayed.

For quotas related to replication, see AWS Application Migration Service service quota limits in the *Application Migration Service User Guide*.

> **Note**
>
> AWS Transform does not support MGN agentless replication. For information about agentless replication, see Agentless replication overview in the *Application Migration Service User Guide*.

2. When replication is complete, expand **Review the replication status** in the **Job Plan** pane. In the right pane you can see the status of the replication and resolve replication alerts.

> **Note**
>
> To proceed, you must install the MGN replication agent on all servers in a wave. Disconnect and archive servers on which you don't install the replication agent. You can use the disconnect-from-service command to disconnect servers. To archive disconnected servers, use the mark-as-archived command. The archiving command only works for source servers whose lifecycle state is `DISCONNECTED`.

Launch test instances

1. In the **Job Plan** pane, under **Launch test instances**, choose **Confirm instance launch**.
2. Download the inventory file, review it, and choose whether to continue with the current file or upload a modified one, then choose **Launch test instances**. You can change the launch settings within the inventory file, but don't modify the list of source servers and applications.

Mark applications as ready for cutover

1. In the **Job Plan** pane, expand **Mark applications as ready for cutover**, and choose **Mark applications as ready for cutover**.
2. In the **Collaboration** tab, review the replication status of each application, and resolve replication alerts.
3. Choose **Mark for cutover**.

Launch cutover instances

1. In the **Job Plan** pane, under **Launch cutover instances**, choose **Confirm instance launch**.
2. Download and open the inventory file, review the inventory, and choose whether to continue with the current inventory or upload a modified one. At this step, don't modify the list of source servers and applications listed in the inventory file. You can only change the launch settings within the inventory file.
3. Choose whether to continue with the current inventory or upload a modified one, and then choose **Continue**.
4. Choose **Launch cutover instances**.

Finalize cutover

1. (Optional) Review the launched Amazon EC2 cutover instances, validate connectivity and run acceptance tests. If you want to fix anything because there's a connectivity issue or a problem in the testing, you need to revert the cutover. This is the time to revert it.
2. In the **Job Plan** pane, expand **Finalize cutover**, and then choose **Start finalizing cutover**. Finalizing the cutover removes the replication agents. After you finalize the cutover you cannot make any changes, you cannot fix any connectivity issues, or anything else, and you cannot revert the cutover.
3. Choose **Finalize cutover**.

Manage server status

During wave migration you can ask AWS Transform to update or change the status of a server. For example, if 9 out of 10 of the servers in your wave passed the test phase but one failed, you can allow AWS Transform to continue to move the 9 into the next phase, and ask to put re-run the test on the tenth.

Lifecycle states include:

- **Not ready** – The server is undergoing the initial sync process and is not yet ready for testing. Data replication can only commence once all of the initial sync steps have been completed.
- **Ready for testing** – The server has been successfully added and data replication has started. test or cutover instances can now be launched for this server.
- **Test in progress** – A Test instance is currently being launched for this server.
- **Ready for cutover** – This server has been tested and is now ready for a cutover instance to be launched.
- **Cutover in progress** – A cutover instance is currently being launched for this server.
- **Cutover complete** – This server has been cutover. All of the data on this server has been migrated to the AWS cutover instance.
- **Disconnected** – This server has been disconnected.

AWS account connectors for VMware migrations

To perform a VMware migration, you need an AWS account target account connector.

The target AWS account connector connects your migration job to your new AWS environment where your workloads will reside after the migration. It's important to ensure that the target AWS account that you specify for this connector is properly set up with the necessary permissions, quotas, and configurations to support your migrated infrastructure.

When you create your target AWS account connector, AWS Transform will ask you to specify a target AWS Region. That is the AWS Region where your target environment with all your servers will reside. You can specify any one of the following AWS Regions for the target account connector:

- US East (N. Virginia)
- US East (Ohio)
- US West (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacific (Tokyo)
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- Asia Pacific (Sydney)
- Asia Pacific (Singapore)
- Canada (Central)
- Europe (Frankfurt)
- Europe (London)
- Europe (Paris)
- Europe (Ireland)
- Europe (Stockholm)
- South America (São Paulo)

> **Important**
>
> If you specify a target AWS Region that is different from the AWS Transform AWS Region, that means AWS Transform will be transferring your data across AWS Regions.

The target connector connects your migration job to the target AWS account and target AWS Region for the following purposes:

- **Network-infrastructure setup** – The target account is where you will create new Amazon VPCs and associated network resources to host your migrated applications in the target AWS Region that you specify when you create the target connector.
- **Amazon EC2-instance setup** – The target AWS account is where you will migrate your VMware virtual machines and run them as Amazon EC2 instances in the target AWS Region.
- **Testing and validation:** – Before final cutover, you will use the target AWS account for testing the migrated servers and ensuring they function correctly in the AWS environment.
- **Cost management** – The target AWS account will be where the costs for running your migrated infrastructure are incurred and where you can track those costs.
- **Long-term operations** – Post-migration, this target AWS account becomes your primary account for operating and managing your former source workloads in AWS.

> **Note**
>
> AWS Transform may update connector types when introducing features requiring permission changes within your AWS accounts. You can use a connector version that is compatible with your VMware migration job. New connectors are created with the latest version for that connector type. The current version for the discovery account connector type is 1.0. The current version for target account connector type is 2.0.

Tracking the progress of a migration job

You can track the progress of the transformation in two ways:

- **Worklog** – Provides a detailed log of the actions that AWS Transform takes, along with human input requests, and your responses to those requests.
- **Dashboard** – Provides a high-level summary of the VMware migration.

Modernization of mainframe applications

AWS Transform is designed to accelerate the modernization of legacy mainframe applications. It orchestrates the analysis of mainframe codebases, generate documentation, extract business logic, decompose monolithic structures, transform legacy code, and manage the overall journey with human inputs (HITL) when needed. The transformation capabilities of AWS Transform for modernizing and migrating mainframe applications empower you to modernize your critical mainframe application faster, while preserving your business-critical logic throughout the transformation process.

Topics

Capabilities and key features

AWS Transform provides the following capabilities for mainframe modernization:

- Supports modernization of zOS mainframe applications written in COBOL (Common Business-Oriented Language) with associated JCL (Job Control Language), CICS (Customer Information Control System) transactions, BMS (Basic Mapping Support) screens, Db2 databases, and VSAM (Virtual Storage Access Method) data files.
- Supports refactoring of Fujitsu GS21 mainframe applications with PSAM (Presentation Service Access Method), Japanese character sets, and NDB (Network Data Base) data files.
- Performs goal-driven reasoning, analysis, decomposition, planning, documentation generation, and code refactoring.

- Automatically refactors COBOL-based mainframe workloads into modern, cloud-optimized Java applications.
- Orchestrates and integrates seamlessly with underlying tools executing analysis, documentation, decomposition, planning, and code refactoring.
- Helps you set up cloud environments for modernized mainframe applications by providing ready-to-use Infrastructure as Code (IaC) templates.

High-level walkthrough

Here's a high-level walkthrough of AWS Transform for modernizing and migrating mainframe applications.

1. Start a chat with AWS Transform, and enter an objective.
2. Based on your objective, AWS Transform proposes a modernization plan—breaking down the high-level goal into intermediate steps.
3. Depending on the goal you provided, AWS Transform can:
   - Analyze the codebase
   - Generate technical documentation
   - Extract business logic from your mainframe applications
   - Decompose the monolithic application into functional domains
   - Plan waves for code modernization
   - Refactor the application assets, including transforming the COBOL codebase to Java-based architecture, and optionally Reforge to improve the quality of refactored code
   - Re-run your jobs as needed

Along the way, AWS Transform might request information from you to execute the tasks.

Human in the loop (HITL)

Throughout the transformation of mainframe applications, you can monitor the progress and status of the transformation tasks through the AWS Transform web experience.

AWS Transform will gather additional information from you in the following scenarios:

- When additional information is needed to execute tasks.

- When approval is required for intermediate artifacts (for example, domains decomposition or wave planning).
- When issues arise that AWS Transform cannot automatically resolve.

Supported file types for transformation of mainframe applications

The supported file types for zOS include:

- COBOL artifacts and related CPY (Copybooks)
- JCL (Job Control Language) and JCL Procedure (PROC)
- CICS System Definition (CSD)
- BMS (Basic Mapping Support)
- Db2 databases
- VSAM (Virtual Storage Access Method)
- IMS TM (Transaction Manager)
- PL/I support for business logic extraction and technical documentation generation

The supported file types for Fujitsu GS21 include:

- PSAM (Presentation Service Access Method)
- ADL (AIM Definition Language)
- NDB (Network Data Base)

For more information about Fujitsu GS21 see these topics in the *AWS Transform for mainframe* migration guide:

- GS21
- Capture & Replay - GS21 Terminals
- Mainframe, AS400, Open VMS and GS21

Supported Regions and quotas for AWS Transform mainframe

For a list for supported Regions, see Supported Regions for AWS Transform.

> **Note**
>
> Your data might be processed in a different Region from the Region where you use AWS Transform. For information on cross-region processing, see the section called "Cross-region processing".

For the quota limits, see *Quotas*.

High-level overview of mainframe modernization journey

Modernizing mainframe applications to AWS is typically achieved in four phases: (a) assess, (b) mobilize, (c) migrate and modernize, and (d) operate, optimize, and innovate. Each of these phases are further described in detail with goals and associated activities to support your modernization journey.

Phases

- Phase 1: Assess
- Phase 2: Mobilize
- Phase 3: Migrate and modernize
- Phase 4: Operate, optimize, and innovate

Phase 1: Assess

Goal: To understand your modernization readiness and develop initial modernization plans.

With AWS Transform you can evaluate your mainframe applications, infrastructure, and operations to determine the right modernization approach for your business needs.

> **Note**
>
> Each phase contains a list of all possible activities you can do when modernizing your mainframe application. You can skip some of the activities that don't align with your use-case or goals.

Activities

- Start with an informal conversation about the modernization and work on opportunity qualification by getting the right people in the room
- Determine opportunity qualification
- Do a first-call presentation with application dependency mapping plan and application architecture
- Determine ROM (Rough order of magnitude) that includes infrastructure cost during migration, migration labor, post-production costs, project duration
- Rapid portfolio detection:
  - Execute pattern & tool guidance assessment
  - Do a source code analysis
  - Identify proof of concept (POC) candidates
- Create technical enablement training:
  - Immersion days
  - Workshops with general topics
- Create a Migration readiness assessment (MRA) report for large mainframe modernization projects or new AWS customers
- Create a statement of work for Mobilize phase

Phase 2: Mobilize

Goal: To build foundation and validate approach through successful proof of concept or pilot modernizations.

With the mobilize phase, you can establish the necessary frameworks, tools, and processes to support mainframe modernization at scale.

Activities

- Kick off the modernization plan with a pilot and the identified POC:
  - Analyze the pilot and set up the code for modernization
  - Build and test the pilot modernized data
  - Deliver results and review the pilot to create a detailed solution for larger data modernization
- For platform:
  - Create a security, compliance, and monitoring plan
  - Set up landing zone environments with AWS mainframe landing zone and account structure
  - Determine architecture infrastructure, application, integration, and more
- Think about people and processes:
  - Cloud Center of Excellence is a good resource
  - Operation model
- Create a comprehensive portfolio analysis for different application scenarios:
  - Comprehensive discovery that includes high-level documentation
  - Decomposition plan (for all applications)
  - Pattern and tool guidance for all applications
  - Come up with an initial modernization and migration plan
- Create a detailed solution considering pilot, platform, people and processes, and comprehensive portfolio analysis
- Determine the business case
- Complete successful pilot with solution for the statement of work for next phase

Phase 3: Migrate and modernize

Goal: To complete modernization and migration within planned timeline and budget.

During this phase, you can migrate and modernize your mainframe applications and data to AWS.

Activities

- Allocate resources to the production-ready application code
- Review and analyze the design using existing documentation or by interviewing subject matter experts (SMEs)

- Define test scenarios
- Set up modernization environments with appropriate runtime, databases, and third-party software
- Migrate and modernize source code and data:
  - You can modernize your application iteratively based on business domains in the application codebase
  - Scale modernization environments as needed to support your modernization workloads
- Establish CI/CD (continuous integration, continuous delivery) pipeline
- Migrate selected workloads to initiate your modernization.
- Transfer application programs and adjust environment scaling as needed:
  - You can scale the environments up or down depending on your need
  - Modernize work packages by running code sanity tests
  - Resynchronize with source code as necessary
- Run comprehensive testing and verify your migration:
  - Collect initial state data
  - Record test scenarios on mainframe
  - Stage environment, application, and data
  - Execute test scenarios on the cloud
  - Compare source and target test results
  - Build automated test scenario execution pipelines
  - Generate KPIs and reports
- Validate your data for production setup
- Validate your progress by:
  - Running integration and system testing
  - Running performance testing
  - Running user acceptance testing
  - Running HA/DR testing
- Define a roll-out plan with application owner and create a plan for training and knowledge transfer
- Consider the cutover and execute production cutovers

- Establish sustained support for post-production activities such as operations, monitoring, capability planning, and warranty for application functions

Phase 4: Operate, optimize, and innovate

Goal: To support operational excellence and continuous improvement of modernized applications.

After modernization, you can optimize your applications' performance, security, and costs while leveraging AWS services for innovation.

Activities

- Monitor application for:
  - Performance metrics
  - Security practices
  - Compliance guidelines
  - Costs
- Logging details
- Manage application operations:
  - Infrastructure management
  - Transactions and job management
  - Runtime environment support
- Application optimization and modernization
- Application development and maintenance:
  - Coding IDE
  - Build
  - Integration
  - Testing
  - Deployment

Transformation of mainframe applications

AWS Transform accelerates the transformation of your mainframe modernization applications. This topic describes the available capabilities.

Topics

Prerequisite: Prepare project inputs in S3

AWS Transform is capable of handling complex mainframe codebases. To use codebase, make sure you have all the assets in your S3 location.

Key project inputs:

- **Source code:** You must upload your mainframe source code files to S3. This includes COBOL programs, JCL scripts, copybooks, and any other relevant source files.
- **Data files**: If you have any VSAM files or other data files that your mainframe applications use, these need to be uploaded to S3.
- **Configuration files**: Include configuration files specific to your mainframe environment.

Other project inputs

- System Management Facility (SMF) records: If applicable, upload SMF records to a new folder in the S3 bucket where the source code is stored, these records should be in a .zip file format.
- Formatting: For technical documentation generation, you can leverage an optional configuration file to generate PDF documents which aligns with your required formats and standards, including headers, footers, logos, and customized information.
- Glossary: AWS Transform AWS Transform leverages automation with Generative AI for documentation generation and business rule extraction. Including a glossary CSV file with information about important abbreviations and terminologies in the root directory of your zip file will help improve the generated documentation quality.
- Test data: If available, upload test data sets that can be used to validate the modernized application. This data should be stored in a new folder in the S3 bucket where the source code is stored.

Download modernization tools

For a test environment modernizing mainframe applications, beyond the automation necessary to run test cases at scale, AWS Transform makes tools available to achieve specific testing tasks. These tools include Data Migrator, designed to facilitate the migration of database schemas and data from legacy systems, Compare tool, to automate the verification that a modernized application produces the same results as reference data, and Terminals, to provide capabilities to connect to legacy mainframe and midrange screen interfaces to capture scenario scripts and videos, in the context of capturing online test cases. These tools are downloaded in your S3 bucket.

Optional configuration of S3 vector bucket.

In Regions where S3 vector buckets are available, AWS Transform will store searchable vector encodings of job output in this S3 vector bucket in your account to provide an AI powered search and chat experience. Data is not used outside of this job, and is not used to train models. T o enable this, you must create and provide S3 vector bucket. AWS Transform automatically creates and attaches a role with required permissions to write to this bucket.

Sign-in and create a job

To sign into the AWS Transform web experience, follow all the instructions in the Getting started with AWS Transform section of the documentation.

To create and start a job: Follow the steps in Start your project.

When setting up your workspace for mainframe transformation, you can optionally set up an Amazon S3 bucket to be used with the S3 connector. After creating the bucket and uploading the desired input files into the bucket, save that S3 bucket ARN for use later. Or you can set up the S3 bucket when setting up the connector as well. For more information, see the section called “Set up a connector”.

Create workspace: Name and describe your workspace where jobs, collaborators, and associated artifacts will be stored.

Create job: Create a job by selecting from preconfigured job plans, or customize the job plan based on your objective by selecting from the list of supported capabilities.

When you set up your workspace for mainframe transformation, you must set up an Amazon S3 bucket to be used with the S3 connector. After creating the bucket and uploading the desired input files into the bucket, save that S3 bucket ARN for use later. Alternatively, you can set up the S3 bucket when setting up the connector as well.

> **Important**
>
> AWS Transform will refuse operations from you if you don’t have the proper permissions. For example, a contributor cannot cancel a job transformation of mainframe applications or delete a job. Only an administrator can perform these functions.

When you create your job you can select from the capabilities below, but the Kickoff step is always required as it is where the location of the source code for the project is located. In the first job set up in a workspace you are required to set up a connector to your Amazon S3 bucket.

Tracking transformation progress

You can track the progress of the transformation throughout the process in two ways:

- **Worklog** – This provides a detailed log of the actions AWS Transform takes, along with human input requests, and your responses to those requests.
- **Dashboard** – This provides high-level summary of the mainframe application transformation. It shows metrics on number of jobs transformed, transformation applied, and estimated time to complete the transformation of mainframe applications. You can also see details of each step including, lines of code by file types, generated documentation by each file type, the decomposed code, migration plan, and the refactored code.

Set up a connector

Set up a connector with your Amazon S3 bucket so that AWS Transform can access your resources and perform transformation functions.

This is some of the information that AWS Transform might ask you to provide:

- AWS account ID for performing mainframe modernization capabilities
- Amazon S3 bucket ARN where your transformation resources are stored
- Amazon S3 bucket path for the input resources you want to transform
- Whether to enable AWS Transform chat to learn from your job progress (optional)

Once you have set up your S3 connector and S3 project inputs, and shared an S3 vector bucket for indexed output, you can create a job based on the objective of the mainframe modernization project. Here is the full list of capabilities you can select from with dependencies noted For example, code analysis is required for most steps.

> **Important**
>
> Your data is stored and persisted in the AWS Transform's artifact store in your workspace and is used only for running the job.

S3 bucket CORS permissions

When setting up your S3 bucket to view artifacts in AWS Transform, you need to add this policy to the S3 bucket's CORS permission. If this policy is not set up correctly, you may not be able to use the inline viewing or file comparison functionalities of AWS Transform.

```
[
    {
        "AllowedHeaders": [],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
              "https://*.transform.eu-central-1.on.aws",
              "https://*.transform.ap-south-1.on.aws",
```

```
            "https://*.transform.ap-northeast-1.on.aws",
            "https://*.transform.ap-northeast-2.on.aws",
            "https://*.transform.ap-southeast-2.on.aws",
            "https://*.transform.ca-central-1.on.aws",
            "https://*.transform.eu-west-2.on.aws",
            "https://*.transform.us-east-1.on.aws"
        ],
        "ExposeHeaders": [],
        "MaxAgeSeconds": 0
    }
]
```

Analyze code

After you share the Amazon S3 bucket path with AWS Transform, it will analyze the code for each file with details such as file name, file type, lines of code, and their paths.

> **Note**
>
> You can download the Analyze code results through the artifacts tab at the job or workspace level. At the job level go to the 'Artifacts' option in the left navigation menu, and open the 'results' folder, or at the workspace level find the name of the job, and open the 'results' folder. This will download a zip file that contains the classification file for manual classification workflow, assets list, dependencies JSON file, and list of missing files.

In the job plan select Analyze code in the left navigation pane to view your results. You can view your code analysis results in several ways:

- **List view** – All files in the Amazon S3 bucket you want to transform for mainframe
- **File type view** – All files in the Amazon S3 bucket displayed per file type. For a list of supported file types, see Supported files.
- **Folder view** – All files in the Amazon S3 bucket displayed in folder structure.

Within the file results, AWS Transform provides the following information depending on what file view you choose:

- Name
- File type

- Total lines of code
- File path
- Comment lines
- Empty lines
- Effective lines of code
- Number of files
- Cyclomatic Complexity - Cyclomatic complexity represents the number of linearly independent paths through a program's source code. AWS Transform will show a cyclomatic complexity for each of the files. With this metric, you can evaluate code maintainability and identify areas that need refactoring.

Missing files– Missing files from the mainframe modernization code analysis. These files ideally, should be added as a part of the source input in Amazon S3 bucket, and the analysis step should be re-run for better and cohesive results.

Identically named – AWS Transform gives you a list of files that share the same name, and possibly the same characteristics (e.g., number of lines of code). It will not have the ability to compare the difference between the contents of any two files at one time.

Duplicated IDs – With Cobol program, the **Program ID** field serves as the unique identifier of the file. This ID must be unique because it's used to call the program throughout your project. However, some projects might have COBOL files with different names but the same Program ID. Getting the list of those files during the assessment can help understand the dependencies among all programs.

> **Note**
>
> This is specific to COBOL code and files.

When you have programs with duplicated IDs, it's suggested to change the Program IDs of these files to have a unique identifier for each of these in the COBOL code. You can then re-run your job to get more accurate and comprehensive code analysis results.

By resolving duplicate Program IDs, you can:

- Improve code clarity and maintainability

- Reduce potential conflicts in program calls
- Enhance the accuracy of dependency mapping
- Simplify future modernization efforts

Codebase issues – Potential issues detected within the codebase that you should resolved before continuing with the modernization project. These issues could include missing references with associated statements or unsupported links in the code.

Update classification – With manual reclassification, you can reclassify files using the bulk update feature by uploading the JSON file with the new classification.

> **Important**
>
> This is only available for `UNKNOWN` and `TXT` files.

After reclassification, AWS Transform will:

1. Updates the classification results
2. Re-runs dependency analysis with the new file types
3. Refreshes all affected analysis results

> **Note**
>
> You can reclassify files only after the initial analysis loop completes.

Inline viewer and file comparison

The Inline viewer is a feature in the AWS Transform for mainframe capabilities that provides two key visualization capabilities:

- **File view**: View content of selected legacy files from jobs
- **File comparison**: Compare content of two legacy files side-by-side

Input file viewing

To view your files in the Analyze code step

- Under **View code analysis results**, select a file using the check box in the list.

  Choose the **View** action button (enabled when 1 item is selected).

  File content will be rendered on screen in the **File View** component.

File comparison

To compare files in the Analyze code step

1. Under **View code analysis results**, select two files using the check boxes in the list.
2. Choose the **Compare** action button (enabled only when 2 items are selected).
3. Files will be displayed side-by-side in the **File comparison** component.

> **Note**
>
> You can't select more than two files to compare files.

> **Important**
>
> If you're having issues with inline viewer or file comparison make sure that the S3 bucket is set up correctly. For more information on S3 bucket's CORS policy, see the section called "S3 bucket CORS permissions".

Data analysis

AWS Transform provides data analysis to help understand the impact of data relationships and elements to the mainframe modernization project. The two outputs provided are:

- Data lineage: Traces the lifecycle of data by mapping relationships between data sources, jobs, and programs
- Data dictionary: Serves as a repository documenting the structural metadata of legacy data elements

On completion of the analysis, there will be two tabs present for each output, and then multiple views available described below for both data lineage and dictionary.

> **Note**
>
> When you request data anlysis. AWS Transform performs code analysis, which is required in order to perform data analysis.

Data lineage

AWS Transform provides multiple views based on the data relationships that need to be understood across the code base being modernized. The four table views available in data lineage include:

- **Data sets**: This view provides a comprehensive impact analysis, including operation tracking to help distinguish between read, write, update, and delete
- **Db2 tables**: Provide the impact analysis related to Db2 tables and the operations
- **Program-to-data**: Identify which COBOL programs reference each dataset
- **JCL-to-data relationships**: Identify which JCL scripts reference each dataset

Summary provides an overview of data sources, and their relationships to programs and JCLs, as well as summary information about how the data sources are leveraged and total operations found in the codebase.

Data dictionary

Understanding the data elements within the data sources present in the codebase is the next step to realize how the various data sources are dependent. Data dictionary is a data catalog providing field level metadata with business language descriptions for accurate transformation mapping.

- **COBOL data structure**: Provides field information across COBOL copybooks present in the codebase, including field properties and business definition
- **Db2 tables**: Provides column and table properties across the Db2 tables present in the codebase, including primary and foreign key, schema information, and data types

Relationship between data lineage and dictionary is available by selecting the data source, and then utilizing the data lineage or data dictionary button in order to dive deeper on the relationship between the data source and the data elements. The navigation between data lineage and dictionary, provides integrated data visibility with data lineage providing the "what" - structure and meaning, alongside the "where" - usage and relationship.

Activity metrics analysis

Activity metrics analysis allows you to analyze System Management Facility (SMF) records type 30 and type 110. The analysis provides insights into batch jobs and CICS transactions that used within your mainframe application, and can help with retiring unused code or decisions around target architecture for modernized application. If you choose to include activity metrics analysis in your job, then we recommend completing the code analysis step first to provide richer SMF outputs, though this is not required.

> **Note**
>
> When you negotiate the job plan for mainframe modernization, we recommend completing the code analysis and decomposition steps first to provide richer outputs.

Onboard SMF records for analysis

You must provide the SMF record location within your S3 bucket as the initial step for SMF analysis. We recommend providing a minimum of 13 months of records to capture annual occurrences that shorter time frames might miss. While 13 months is recommended, any time frame with detectable SMF records will produce analysis results.

Your SMF extract must meet these format requirements:

- Include all types 30 and 110
- Use raw binary file in EBCDIC format
- Include RDW bytes

Provide a link to the SMF records within your S3 bucket, ensuring the records are in a folder separate from your source code. Records must be in .zip file format. If you provide a link to a folder where SMF records are not detected, you will receive an error message and no analysis will be completed.

Analysis output

The analysis output contains three components: tabular view, .csv output (available in your S3 bucket), and visualization within the dashboard. The header displays the time range of your provided records so you can identify any missing key dates. For example, if your records span May 1 - October 31st but exclude the busy day after Thanksgiving, you can easily understand that key records are missing. Timestamps in the web application reflect your system's time zone and the SMF records.

Tabular view

The tabular view contains up to three main components for batch jobs and CICS transactions:

- **Summary** - Provides key jobs and transactions
- **Batch job/CICS transactions analysis** - Provides aggregated analysis across jobs and transactions
- **Code analysis comparison** (batch only) - Available when code analysis step is executed prior to SMF analysis

The discovery summary provides three groups of jobs and transactions that help you quickly identify items for deeper analysis.

Batch job and CICS transaction analysis provides aggregated analysis across jobs and transactions. Analysis results have some columns hidden by default - use the gear icon to display additional fields.

The transaction key generates unique data aggregation for CICS transactions, combining four fields into a single key value:

- Transaction ID
- Program name
- SysPlex ID
- SysID

You can search by the complete key or any component of the key in the analysis output.

Code analysis comparison highlights jobs found in either SMF records or the analyze code step but not present in both. This shows jobs that haven't run during your SMF record timeframe or weren't

present in the analyze code step. This output is only available when you execute the code analysis step before SMF analysis in your job.

Best practices

To get comprehensive outputs, for **Batch jobs (type 30)**, ensure your JCL file name matches the job name to get meaningful outputs in the code analysis comparison.

Generate technical documentation

You can generate technical documentation for your mainframe applications undergoing modernization. By analyzing your code, AWS Transform can automatically create detailed documentation of your application programs, including descriptions of the program logic, flows, integrations, and dependencies present in your legacy systems. This documentation capability helps bridge the knowledge gap, enabling you to make informed decisions as you transition your applications to modern cloud architectures.

> **Note**
>
> When you request generation of technical documentation, AWS Transform performs code analysis, including code dependency analysis, which are required in order to generate the documentation.

To generate technical documentation

1. In the left navigation pane, under **Generate technical documentation**, choose **Select files and configure settings**.
2. Select the files in the Amazon S3 bucket that you want to generate documentation for, and configure the settings in the **Collaboration** tab.

   > **Note**
   >
   > Selected files should have the same encoding type (that is, all in the same CCSID - UTF8 or ASCII). Otherwise, generated technical documentation might have empty fields or sections.

3. Choose the documentation detail level:

- **Summary** – Provides a high-level overview of each file in the scope. Also, gives a one-line summary of each file.
- **Detailed functional specification** – Provides comprehensive details for each file in the mainframe application transformation scope. Some details include logic and flow, dependencies, input and output processing, and various transaction details.

> **Note**
>
> Documentation can be generated only for COBOL and JCL files.

4. Choose **Continue**.
5. Once AWS Transform generates documentation, review the documentation results by following the Amazon S3 bucket path in the console, where the results are generated and stored.
6. Once the documentation is generated, you can also use AWS Transform chat to ask questions about the generated documentation and decide the next steps.

Add user information into the documentation

```
ARTIFACT_ID.zip
### app/
    ### File1.CBL
    ### File2.JCL
    ### subFolder/
    #   # File3.CBL
### glossary.csv
### pdf_config.json
### header-logo.png
### footer-logo.png
# ...
```

Optional files can be added in the zip file to help improve the generated documentation quality and provide customized PDF cover page. Some of these can be:

- **glossary.csv file**: You can choose to provide and upload an optional glossary in the zip file in the S3 bucket. The glossary is in CSV format. This glossary helps create documentation with relevant descriptions in line with the customer vocabulary. A sample `glossary.csv` file looks like:

```
LOL,Laugh out loud
ASAP,As soon as possible
WIP,Work in progress
SWOT,"Strengths, Weaknesses, Opportunities and Threats"
```

- **pdf_config.json**: You can leverage this optional configuration file to generate PDF documents which align with their company's formats and standards, including headers, footers, logos, and customized information. A sample `pdf_config.json` looks like:

```
{
    "header": {
    "text": "Acme Corporation Documentation",
    "logo": "header-logo.png"
  },
  "customSection": {
    "variables": [
      {
        "key": "business Unit",
        "value": "XYZ"
      },
      {
        "key": "application Name",
        "value": "ABC"
      },
      {
        "key": "xxxxxxxxxx",
        "value": "yyyyyyyyyyy"
      },
      {
        "key": "urls",
        "value": [
          {
            "text": "Product Intranet Site",
            "url": "https://example.com/intranet"
          },
          {
            "text": "Compliance Policies",
            "url": "https://example.com/policies"
          }
        ]
      }
    ]
```

```
  },
  "footer": {
    "text": "This document is intended for internal use only. Do not distribute
 without permission.",
    "logo": "footer-logo.png",
    "pageNumber": true
  }
}
```

- **Header:**
  - For the cover page PDF file, the default text will be the project name.
  - For each program PDF file, the default text will be the program name.
  - There is no default logo. If a header logo is not configured, no logo will be displayed.
  - The font size and logo size shall be dynamically changed based on the number of words or logo file size.
- **Custom section:**
  - If the custom section is not configured, it will be omitted from the PDF.
  - The link has to be click able.
- **Footer:**
  - There is no default text or logo for the footer.
  - The page number will be displayed in the footer by default, unless explicitly configured otherwise.
  - The font size and logo size shall be dynamically changed based on the number of words or logo file size.

Generate documentation inline viewer

You can view the PDF files in the generate technical documentation step.

To view the PDF files

1. Navigate to the **Review documentation results** tab.
2. Locate the PDF in the table listing generated PDFs.
3. Either select the rile and then select **View** or select the link element overlaid on the file name.

The PDF opens in AWS Transform, with the option to expand the screen in the top right.

> **Note**
>
> AWS Transform also gives you the ability to download either an XML of PDF version of the generated technical documentation.

> **Important**
>
> If you're having issues with documentation inline viewer, make sure that the S3 bucket is set up correctly. For more information on S3 bucket's CORS policy, see the section called "S3 bucket CORS permissions".

Extract business logic

You can extract essential business logic from your mainframe applications that are undergoing modernization. AWS Transform automatically analyzes your code to identify and document critical business elements, including detailed process flows, and business logic that is embedded in your applications. This capability serves multiple stakeholders in your modernization journey. Business analysts can leverage extracted logic to create precise business requirements and identify gaps or inconsistencies in current implementations. Developers gain the ability to quickly comprehend complex legacy system functionality without extensive mainframe expertise.

> **Note**
>
> When you request business logic extraction. AWS Transform performs code analysis, including code dependency and entry point analysis, which are required in order to perform business logic extraction.

To extract business logic

1. In the left navigation pane, under **Extract business logic**, choose **Configure settings**.
2. In the **Collaboration tab** select how you want to extract business logic:

- **Application level**: Generates business documents for all business functions, transactions, batch jobs, and files. This selects all of the files in the application.
- **File level**: Generates business documents only for files you select from the file table.

> **Note**
>
> - For either option you can select **Include detailed functional specification** so that AWS Transform includes control flow and comprehensive business rules for the selected files.
> - Selected files should have the same encoding type (that is, all in the same CCSID - UTF8 or ASCII). Otherwise, generated documentation might have empty fields or sections.
> - Documentation can be generated only for COBOL and JCL files.
> - For application level, programs used by CICS transactions and batch jobs are grouped together, while all other programs are categorized as *Unassigned*.

3. Choose **Continue**.
4. Once AWS Transform extracts business logic it stores the results in an Amazon S3 bucket in JSON format so that you can view them online.

> **Note**
>
> The number of generated business rule files might be larger than your initial selection. Some selected files may trigger business rule extraction to include additional dependent files, which will also appear in the results table.

View the extracted business documentation inline

You can view the business logic in the Extract business rule step. To do this,

1. Navigate to **Review business logic extraction results**.
2. Select the document you want to view from the table, and then click the **View result** button.

The business documentation page opens in a new browser tab.

Decomposition

You can decompose your code into domains that account for dependencies between programs and components. This helps the related files and programs to be grouped appropriately within the same domain. It also helps maintain the integrity of the application logic during the decomposition process.

> **Note**
>
> When you request decomposition, AWS Transform performs code analysis, including code dependency analysis, which are required in order to perform decomposition. We also recommend that you perform business logic extraction before decomposition for better results.

To get started with decomposing your application:

1. Select **Decompose code** from the left navigation pane.

   > **Note**
   >
   > One domain, Unassigned, is automatically created for all files not associated with a domain. On initial navigation, all files should be associated to Unassigned, unless domains were proposed from application level business logic extraction.

2. Create a new domain through the Actions menu, and then choose Create domain.
3. In Create domain, provide domain name, optional description, and mark some files as seeds.
   - CICS configured files (CSD) and scheduler configured files (SCL) can be used for automatic seed detection.
   - You can also set one domain only as a common component. The files in this domain are common to multiple domains.
4. Choose Create.

> **Note**
>
> You can create multiple domains with different files as seeds.

5. After confirming all domains and seeds, choose Decompose.
6. AWS Transform will check the source code files and then decompose into domains with programs and data sets with similar use cases and high programming dependencies.

   AWS Transform gives you a tabular and graph view of decomposed domains as dependencies. Graph view has three options:

   - **Domain view** – Can view how different domains are related to each other in visual format.
   - **Dependency view** – Can view all files in each domain as a complex dependency graph. If a node that was added to a domain didn't receive information from a seed in the same domain, then this node will either be predicted into unassigned (node didn't receive any information), disconnected (in a sub graph that didn't receive seed information) or into another domain (node received information from at least that domain).
   - **Subgraph view** - User can create subgraphs to visualize a subset of nodes to more clearly understand the relational impact and boundaries of that collection of nodes.
     - To create a subgraph select a group of nodes and select the Extract subgraph option from the tool bar
     - Merging subgraphs is available, when merging a new third subgraph is created in addition to the two subgraphs you are merging.

   > **Note**
   >
   > Repeat these steps to add more domains or to reconfigure your already created domains with a different set of seeds if you don't like current domain structure.

7. When completed, choose **Continue**.

Seeds

Seeds are the foundational inputs for the decompose code phase. Each component or file (e.g., JCL, COBOL, Db2 tables, CSD, and scheduler files) can be assigned as a seed to only one domain, ensuring clear boundaries and alignment during the decomposition process.

The identification of the seeds depends on the structure of the application or portfolio. In the case of a typical mainframe legacy application, seeds can often be determined by adhering to established naming conventions, batch-level grouping in the scheduler, and transaction-level grouping defined in the CICS system. Additionally, database tables can also serve as seeds, providing another layer of structure for decomposition.

Import and/or update dependencies files

During decomposition, you can upload a JSON file for the dependencies that replaces the existing files generated by the dependencies analysis AWS Transform performs.

Export dependencies function allows you to download the dependencies json file generated in the decomposition step. After downloading, you can modify the file per your requirement. Then, you can **import dependencies** using the AWS Transform's upload functionality which allows you to upload the new JSON file of the dependencies that replaces the file generated by the dependencies analysis. After that, the graph in the decomposition step will be updated.

To export, modify, and import dependencies

1. On the **View decomposition results** page, choose **Actions**.
2. In the dropdown list, choose **Update dependencies file** option under **Other actions**.
3. In the **Update dependencies file** modal,
    a. Download the dependency file AWS Transform created from the existing analysis results.
    b. In the downloaded file, modify the dependencies based on what you want to achieve.
    c. After modifying, save and upload this file using the **Upload dependency file** button.

    > **Note**
    >
    > The only accepted file format is JSON file.

4. Next, choose **Import**.

AWS Transform will import the dependency file and create a new dependencies graph based on your input.

Import and/or Update Domains

For customers that have domains, seed, and/or file relationships mapped prior to the decomposition step, you can upload this domain definition through the Import domains file function available through the Actions menu. Some examples of when this function may be utilized:

- Bring forward decomposition from another job
- Subject matter experts providing this mapping

Once the domains file has been imported, the user can either run decomposition against the domain definition, or if satisfied can Save and then Submit the domain definition.

Parent/child/neighbor files

In a dependencies graph, programs relate to each other through different types of connections. Understanding these relationships helps you analyze program dependencies during transformation of your mainframe applications. It also helps with understanding the boundaries of a domain. For example, if you select a domain, and then select parent one level, it will show you the connected nodes.

Parent relationships – A parent file calls or controls other programs. Parents sit above their dependent programs in the hierarchy. You can select parent at one level or at all levels.

Children relationships – A child file is called or controlled by the parent program. Children sit below their parent in the file hierarchy.

Neighbor relationships – Neighbors are files at the same hierarchical level. They share the same parent program and might interact with each other directly.

Migration wave planning

Based on the domains you created in the previous step, AWS Transform generates a migration wave plan with recommended modernization order.

> **Note**
>
> Decomposition is a required step that must be completed in your job plan prior to running migration wave planning. If migration wave planning is not selected, but added to the job plan due to the inclusion of decomposition, this step will auto complete.

1. To view the planning results, choose **Plan Migration Wave**, and then choose **Review Planning Results**.
2. Review the domain wave plan (either in a table view or a chart view).
3. You can either choose to go with the recommended migration wave plan generated by AWS Transform or add your preference manually by importing a JSON file.

   > **Note**
   >
   > You can choose to migrate multiple domains in a single wave.

4. (Optional) If you decide to manually adjust migration wave plan, AWS Transform generates a new migration wave plan per your preference. You can also adjust the domains in each wave as required by choosing **Add preference** and then, **Add and regenerate**.
5. After verifying, choose **Continue**.

If you're satisfied with this migration plan, you can move next step for refactoring the code. If you need to adjust the preference, you can follow these steps again.

Refactor code

In this step, AWS Transform refactors the code in all or selected domain files into Java code. The goal of this step is to preserve the critical business logic of your application while refactoring it to a modernized cloud-optimized Java application.

> **Note**
>
> When you request refactoring, AWS Transform performs code analysis, including code dependency analysis, which are required in order to perform refactoring. We also recommend that you perform decomposition and wave migration planning before refactoring, for better results.

1. Navigate to **Refactor code** in the left navigation pane, and choose **Domains to migrate**.
2. Select the domains you want to refactor.
3. Choose **Continue**. You can track the status of refactoring domains (and files in it) using the worklog. AWS Transform will do the transformation of the mainframe code, and generate results without any manual input.
4. After refactoring completes, it will change the status to `Completed` in the worklog. You can view the results of refactored code by going to the Amazon S3 bucket where the results are stored. Each domain will provide a status for **Transform** (with each file), and **Generate** and will be marked as `Done`.

> **Note**
>
> Along with the refactored code, your S3 bucket will also have the AWS Transform for mainframe Runtime to be compiled.

You might also see certain domains that have a `Done with issues` status. Expand those to see files showing a `Warning` status or an `Error` status. You can view the issues for the `Warning` and `Error` files, and choose to fix them for better refactoring results. Additional guidance for fixing these errors and warnings can be found in the console by viewing each of these files.

File transformation status

After your refactoring completes, AWS Transform will give you transformation status for all your files. These may include:

Ignored – AWS Transform will also give you the `Ignored files` after the code refactor. These are the files that are ignored during refactoring and haven't been included in the transformation.

Missing – `Missing files` are not included during the refactoring and transformation. These should be added again as a part of the source input in Amazon S3 bucket for better and cohesive results. AWS Transform will give you the number and information of missing files in the console.

Pass through – `Pass through` files are not modified during the refactoring step, and do not go through any transformation. This status is useful for the Refactoring action which may not have changed the file depending on the configured refactoring.

Fatal – An unexpected error occurred during the transformation of this file.

Error – An error occurred during the transformation of this file and these files need to go through refactoring again.

Warning – The transformation generated all expected outputs for this file, but some elements might be missing or need additional input. Fixing these and running the refactoring steps again would give you better transformation results.

Success – The transformation generated all expected outputs for this file and it has detected nothing suspicious.

Custom transformation configuration

Refactor transformation allows you to change and/or modify configuration to improve the results of transformation.

To customize your transformation configuration

1. In **Refactor code** section, go to **Configure transformation** under Select domains.
2. In **Configure refactor** modal, specify the **Refactor engine version** (e.g. `4.6.0`) which will be used to compile and run the generated application. For more information on available engine versions, see AWS Transform for mainframe Runtime release notes.
3. Add your project name, root package, and target database. The target database is target RDMS for the project.
4. Under **Legacy encoding**, define the default encoding for your files (e.g., `CP1047`). And mark the check boxes next to **Export Blusam masks** and **Specify generate file format**. You can also choose to specify conversion table encoding file format.
5. Review all you changes. Then, choose **Save and close**.

This will allow you to reconfigure your code with the new specified properties.

Reforge code

Reforge uses Large Language Models (LLMs) to improve the quality of refactored code. The initial COBOL-to-Java transformation preserves functional equivalence while retaining COBOL-influenced data structures and variable names from the legacy system. Reforge restructures this code to follow modern Java practices and idioms, replacing COBOL-style constructs with native Java collections and naming conventions. This makes the code more readable and maintainable for Java developers.

Note

Quotas for reforge are:

- 3,000,000 lines of code per job
- 50,000,000 lines of code per user per month

Reforge your code after refactoring by following these steps:

1. Choose **Reforge java code** in the left navigation pane and then select **Configure code reforge**.
2. Provide the S3 location to your zipped buildable source project and choose **Continue**. Use this zip structure:

   ```
   input.zip
           ### PROJECT-pom
           ### PROJECT-entities
           ### PROJECT-service
           ### PROJECT-tools
           ### PROJECT-web (optional)
           ### pom.xml
   ```

   AWS Transform analyzes your zip package to locate files within the PROJECT-service directory so that it can provide a selectable list of classes that you can reforge. These classes have the suffix `ProcessImpl.java`.
3. Complete the **Select classes to reforge** page and choose **Continue**. Track the reforge status on the **Worklog** tab.
4. View the results of your completed reforge on the **View results** page, which displays the reforge status per class. It also specifies where to find the Reforge result in your S3 bucket.

Once AWS Transform gets this input from you it gives you a downloadable file with the **Reforge results**.

This is the zip structure resulting from a successful reforge:

```
reforge.zip
### maven_project
### reforge.log
```

```
###tokenizer_map.json
```

- **maven_project** contains the reforged source code.
  - Files that have been refactored but whose compilation was not successfully finalized are located at `/src/main/resources/reforge/originalClassName.java.incomplete` and are named `originalClassName.java.incomplete`. Compare these to the original versions of the files to choose reforged functions you want to save.
  - Source files provided to AWS Transform that were refactored successfully are backed up to `src/main/resources/reforge/originalClassName.java.original` and are named `originalClassName.java.original`. The refactored versions of the files replace the source files provided to AWS Transform.

> **Note**
>
> The `originalClassName.java` files are replaced with the reforged files only if the reforging process is successful. Otherwise, they retain the original content.

- **reforge.log** contains logs that you can use to diagnose job failures or provide to AWS support in case of an issue.
- **tokenizer_map.json** contains a mapping of token IDs to your data, such as file paths and class/method names, that are tokenized in the logs for privacy protection. You can provide this file to AWS support in case of an issue.

Plan your modernized applications testing

You can create and manage test plans for your mainframe modernized applications based on extracted code attributes, job complexity, and scheduler paths. AWS Transform helps prioritize which jobs to test and identifies the specific artifacts needed for each test case. The test planning process is divided into three main phases: configuration, scoping, and review.

To create a test plan

1. **Configure test plan settings**

   a. In the left navigation pane, under **Plan your testing**, choose **Configure settings**.

b. (Optional) Provide S3 paths for your Business Logic Extraction (BLE). These artifacts enhance test plan quality. Without BLE artifacts some fields in the test plan may remain incomplete.

2. **Define test plan scope**

   a. Select entry points, such as batch jobs, to include in your test plan.

   b. Filter and sort jobs based on multiple attributes:

      - Business functions (extracted from BRE BLE)
      - Domains (from decomposition phase)
      - File paths and locations
      - Custom search criteria

   c. Select individual jobs or entire groups for testing.

   d. Review job relationships and dependencies.

3. **Review and adjust the test plan**

   The generated test plan provides comprehensive information including:

   - Preferred execution order based on dependencies
   - Job group assignments from scheduler
   - Complexity scores, which are aggregate scores for test cases
   - Business domain associations
   - Cyclomatic complexity metrics
   - Dataset and table dependencies
   - Lines of code metrics
   - Business function mappings

Test plan customization options

Your test plan can be customized to address specific needs. For example, you can:

- Create new test cases by selecting multiple entry points
- Merge existing test cases to combine related functionality
- Split test cases for more granular testing

- Remove unnecessary test cases
- Add or remove entry points for existing test cases
- Modify test case descriptions and attributes
- Adjust execution order

Detailed test case information

Each test case provides details that describe its content and the related dataset or datafiles:

- Comprehensive description of test scope
- Complete list of entry points included
- Aggregate metrics showing complexity and size
- Business rules and automated test case guidance
- Dataset and table dependencies with direction (input/output)
- Interactive dependency graph visualization
- Execution prerequisites and requirements

Data management features

These details help you understand the data related to the test case. You can:

- Filter data sets by input/output direction
- Identify required artifacts for test execution
- Track data dependencies between test cases
- Monitor data set usage across test plans

> **Note**
>
> AWS Transform automatically analyzes scheduler dependencies and assigns complexity scores to help prioritize testing efforts. Higher complexity scores indicate jobs that may require more thorough testing or isolation during the testing process.

Business rules and test case guidance

The AWS Transform test plan provides recommendations for your test case plan based on Business Rule Extraction:

- Automatic processing of business rules using LLM
- Generation of synthetic test cases
- Testing guidance for specific business scenarios
- Traceability between rules and test cases

The final test plan is stored in the specified S3 location and includes all of the necessary information to execute your testing strategy effectively. You can export the test plan for integration with other testing tools or documentation systems.

> **Note**
>
> While synthetic test case guidance is provided, the actual test artifacts must be created separately based on the guidance. The test plan serves as a comprehensive blueprint for your testing strategy but does not generate the test data or execution scripts.

Test case creation rules - Summary

General rules

- A test case contains 1 to many JCLs in schedule execution order
- Test cases are created from valid supported schedulers (CA7 and Control-M)
- The test executes JCLs, not the scheduler itself or scheduled tasks
- A scheduled task is unique and executes only one JCL
- One JCL can be executed by multiple scheduled tasks
- One JCL can exist without being in a schedule

Default test case creation rules

- If a single JCL is in a test case:
  - JCL is not involved with a schedule

  - JCL is executed by a scheduled task that diverges into multiple branches
  - A branch in the schedule contains only this JCL
- If multiple JCL in a test case: represents a linear execution path sequence within a schedule branch
- If JCL is on a divergent scheduled task: JCL becomes its own separate test case
- If JCL is on a convergent scheduled task: JCL can start a new test case but won't be included in previous branches
- Missing JCLs are skipped and execution continues (with planned future enhancement to cut test case at missing JCL point)

User operations rules

- **Create Test Case**: user selects from available JCLs
  - Succeeds if JCLs exist in schedule execution branch sequence
  - Follows schedule execution order
- **Add a JCL to Test Case**: user selects from available JCLs
  - Succeeds if JCL can exist in the test case's schedule execution branch/path
- **Remove JCL from Test Case**: user can remove any JCL from a test case
  - Allowed even if it causes execution path gaps
- **Merge Test Cases**: user selects two test cases to combine
  - Succeeds if JCLs can exist together in same schedule execution branch
  - Maintains schedule execution order
- **Split Test Case**: user selects one JCL in a test case for split
  - Creates new test case from split point forward
  - Original test case modified to exclude JCLs past split point
- **Delete Test Case**: user can delete any created test case

> **Note**
>
> You cannot create, add, or merge test cases from different scheduler branches/path. A future enhancement is planned to allow operations beyond divergent/convergent tasks in the scheduler.

Generate test data collection scripts

You can generate JCL scripts to collect test data from your mainframe systems based on the test plan created in the previous step. AWS Transform automatically creates data collection scripts for datasets, database tables, and sequential files needed for comprehensive testing. The data collection process is divided into four main phases: input configuration, test case selection, script configuration, and script generation.

To generate test data collection scripts

1. **Provide test plan input**

   a. In the left navigation pane, under **Test data collection**, choose **Provide test plan input**.

   b. Specify the S3 path to your test plan JSON file from Plan your modernized applications testing.

   c. The input field is pre-populated if the test plan was generated in a previous job step.

   d. You can also select test plan from other jobs by specifying the appropriate S3 location.

2. **Select test cases for data collection**

   a. Review the complete list of test cases from your test plan.

   b. Filter and sort test cases based on multiple attributes:

      - Business functions and domains
      - Database table dependencies
      - Data set requirements
      - Complexity metrics
      - Custom search criteria

   c. Select individual test cases or use bulk selection options.

   d. Review test case details including entry points, metrics, and business rules by clicking on one Test Case to see details.

3. **Configure data collection scripts**

   a. Download sample templates and configuration files for reference.

      - AWS Transform provides sample templates for Db2 database unloads, VSAM file REPRO and sequential dataset processing to be used as guidance on the kinds of template expected by the process.

- Standards may vary from site to site so the expectation is that customers will modify or replace these templates that conform to their own standards.
- These modified templates need to be uploaded to a S3 bucket where the test data collection can process them.

b. Provide variable configuration file (JSON format) containing:

- User prefixes and environment-specific constants
- Database configuration parameters
- Destination endpoint settings and data transfer parameters
- Other required parameters defined by user and to be used in JCL templates

c. Upload JCL templates for different data collection methods:

- **Db2 template**: For database table unloading (customize for BMC, IBM DSN, or other unload utilities)
- **VSAM template**: For VSAM file processing (typically uses REPRO utility)
- **Sequential datasets template**: For processing sequential datasets, partitioned datasets, GDGs etc.

4. **Review and manage generated scripts**

The generated scripts provide comprehensive data collection capabilities including:

- Separate scripts for "before" and "after" test execution data collection
- Organized script structure by test case and data type
- Scripts are automatically stored into S3 bucket for easy access and transfer
- Generated JCL scripts are ready for mainframe execution
- Variable substitution is based on user configuration defined in the templates

Script generation features

Generated scripts are automatically customized based on your templates and configuration:

- **Template-based generation**: Uses your provided JCL templates with variable substitution
- **Environment**: Incorporates your specific mainframe configuration
- **Data type handling**: Creates appropriate scripts for sequential datasets, VSAM files, and database tables

- **Collection for test cases**: Generates both "before" and "after" data collection scripts
- **Sequential dataset processing**: AWS provided sample provides for a file transfer function, but this can be customized to compression utilities available at your site or utilities such as Connect Direct or managed file transfer etc.

Data collection strategy

The generated scripts support comprehensive data collection strategies:

- **Sequential dataset collection**: REPRO and copy utilities for VSAM and flat files
- **Database table unloading**: Customizable Db2 unload processes
- **Sequential dataset processing**: Customizable post processing of sequential datasets such as compression, managed file transfer services etc.
- **Dependency management**: Coordinated collection based on test case definition

> **Note**
>
> AWS Transform generates scripts based on your templates and configuration. Review all generated JCL before executing on your mainframe environment to ensure compatibility with your specific system configuration and security requirements.

Template customization and best practices

ATX Test Data Collection provides flexible template customization capabilities:

- **Multi-utility support**: Adapt templates for different mainframe utilities (BMC, IBM, DSN)
- **Variable-driven configuration**: Use constants for environment-specific parameters
- **Reusable templates**: Create standardized templates for consistent script generation
- **Data handling**: Incorporate organization-specific data handling requirements
- **Security integration**: Include appropriate security and access controls
- **Performance optimization**: Configure for efficient data collection and transfer

Generated output structure

The generated scripts are organized in your S3 bucket with the following structure:

- **Test case organization**: Scripts grouped by associated test cases
- **Collection timing**: Separate folders for "before" and "after" data collection
- **Data type classification**: Scripts organized by sequential datasets, database tables, and transfers
- **Metadata files**: Summary information and execution guidance
- **Ready-to-transfer format**: JCL formatted for direct mainframe deployment

The final script collection is stored in the specified S3 location and includes all necessary JCL to execute your data collection strategy effectively. You can download the scripts for transfer to your mainframe environment or integrate with automated deployment processes.

> **Note**
>
> While comprehensive JCL scripts are generated, actual execution must be performed on your mainframe environment. The scripts serve as ready-to-use data collection tools but require appropriate mainframe access and execution permissions.

Test automation script generation

You can generate test automation scripts to execute test cases on your modernized application based on the test plan created in the previous step. AWS Transform automatically creates comprehensive test scripts that utilize the data collected from the test data collection process. The test automation script generation process consists of three main phases: input configuration, test case selection, and script generation results.

To generate test scripts

1. **Provide test plan input**

   a. In the left navigation pane, under **Test automation script generation**, choose **Provide test plan input**.

   b. Specify the S3 path to your test plan JSON file from Plan your modernized applications testing.

c. The input field is pre-populated if the test plan was generated in a previous job step.

d. You can also select test plan from other jobs by specifying the appropriate S3 location.

e. The system uses this test plan as the foundation for generating automation scripts.

2. **Select test cases for script generation**

a. Review the complete list of test cases from your test plan.

b. Filter and sort test cases based on multiple attributes:

- Business functions and domains
- Database table dependencies
- Data set requirements
- Complexity metrics
- Custom search criteria

c. Select individual test cases or use bulk selection options with the **Check All** and **Uncheck All** buttons.

d. Review test case details including entry points, metrics, and business rules by clicking on individual test cases.

The selected test cases will have automation scripts generated for execution on the modernized application.

3. **Review and manage generated test automation scripts:**

- The system displays a success message confirming script generation completion.
- Generated test scripts are automatically stored in your specified S3 bucket location.
- Access the complete list of generated test scripts with their respective S3 locations.
- Each test case has its corresponding automation script stored in individual S3 locations.
- Scripts are ready for deployment and execution on your modernized application environment.

Test automation script features

The generated automation scripts provide comprehensive testing capabilities:

- **Modernized application testing**: Scripts are specifically designed to execute test cases on your transformed application

- **Data integration**: Utilizes the test data collected from the previous test data collection step, these data need to be copied to folders for each test cases
- **Automated execution**: Scripts can be used to set up data sinks, run test cases, and compare results, some parameters will have to be set according to your deployment environment
- **Organized structure**: Scripts are systematically organized by test case in your S3 bucket
- **Ready-to-deploy format**: Scripts are formatted for direct deployment to your testing environment

Generated output structure

The generated test automation scripts are organized in your S3 bucket with the following structure:

- **Test case organization**: Each test case has its dedicated script stored in individual S3 folder
- **Execution-ready format**: Scripts are formatted for immediate deployment and execution after setting up some variable depending from your environment
- **Centralized access**: All scripts are accessible from a single S3 bucket location for easy management

Test execution strategy

The generated scripts support comprehensive test execution workflows:

- **Environment setup**: Scripts include capabilities for setting up initial data to execute the test
- **Data preparation**: Integration with collected test data from test case data collection step
- **Test case execution**: Automated execution of individual test cases on the modernized application
- **Result comparison**: Built-in capabilities for comparing test results and validating application behavior

> **Note**
>
> AWS Transform generates test automation scripts based on your test plan and selected test cases. The scripts are designed for execution on your modernized application environment and utilize the test data collected in the previous step. Review all generated scripts before

deployment to ensure compatibility with your specific application configuration and testing requirements.

Best practices for test automation

- **Environment validation**: Ensure your modernized application environment is properly configured before script execution
- **Data verification**: Validate that the required test data from the collection phase is available and accessible
- **Script customization**: Review and customize generated scripts as needed for your specific testing requirements
- **Execution monitoring**: Implement appropriate monitoring and logging during test script execution
- **Result analysis**: Establish processes for analyzing test results and identifying application issues

The final collection of test automation scripts provides a complete testing framework for validating your modernized application functionality. The scripts can be integrated into your continuous testing processes or executed as part of your application validation workflow.

Deployment capabilities in AWS Transform

AWS Transform helps you set up cloud environments for modernized mainframe applications by providing ready-to-use Infrastructure as Code (IaC) templates. Through the AWS Transform chat interface, you can access pre-built templates that create essential components like compute resources, databases, storage, and security controls. The templates are available in popular formats including CloudFormation (CFN), AWS Cloud Development Kit (AWS CDK), and Terraform, giving you flexibility to deploy your infrastructure.

These templates serve as building blocks that reduce the time and expertise needed to configure environments for your modernized mainframe applications. You can customize these templates to fit your needs, giving you a foundation to build your deployment environment.

To retrieve the IaC templates, ask in the AWS Transform chat for the Infrastructure-as-Code templates clarifying your preferred modernization pattern (such as AWS Transform for mainframe refactor), your preferred topology (standalone vs high availability), and your preferred format (CloudFormation vs Cloud Development Kit vs Terraform).

Build and deploy your modernized application post-refactoring

After you complete the refactoring process with AWS Transform, you can build and deploy your modernized Java application. This guide walks you through retrieving your modernized code, configuring your environment, and deploying and testing your application.

> **Note**
>
> In addition to the guidance provided here, the AWS Transform generated code package will include an *Set up the AWS Automated Refactor Development Environment* document which provides instructions to set up a IDE (Integrated Development Environment) with Developer Runtime.

Topics

Prerequisites

Before you begin, make sure you have:

- Successfully completed a refactoring job with AWS Transform.
- Access to the Amazon S3 bucket containing your modernized code. You can find this path on the console under **Refactor code** → **View results** or see Step 1: Retrieve the modernized code.
- Installed and configured build software tool stack on your development machine, such as Apache Maven or Apache Tomcat. For more information on Runtime versioning, see AWS Transform for mainframe Runtime release notes.
- Installed and configured Amazon Corretto or a version of Java runtime. For more information on installing Amazon Corretto, see Amazon Corretto 24.

- Access to create and configure Amazon Aurora PostgreSQL databases for Runtime components, if necessary. For more information on Creating the Aurora PostgreSQL database, see Working with Amazon Aurora PostgreSQL.
- Administrative access to deploy applications to your runtime environment.
- Reviewed the AWS Transform for mainframe Runtime concepts for fundamental concepts on applications modernized with AWS automated refactoring solution.

Step 1: Retrieve the modernized code

To retrieve the modernized code

1. Navigate to your **Refactor code** → **View results** page on the console and locate the S3 path containing your generated code.
2. Download and extract the generated code package.
3. Open `codebase/app-pom/pom.xml` and note the required runtime engine version. For example `<gapwalk.version>4.6.0</gapwalk.version>`.
4. Locate the *Set up the AWS Automated Refactor Development Environment* document from the downloaded code package for reference.

Step 2: Build the modernized application

To build your modernized application

1. Access the runtime version from a dedicated Amazon S3 bucket on the AWS account used with AWS Transform:
2. Download and install the appropriate runtime version (identified in the section called “Step 1: Retrieve the modernized code”) on your local development machine.

   > **Note**
   >
   > Additional information for installing the runtime dependencies on your local machine is available in section 3.1 of the *Set up the AWS Automated Refactor Development Environment* document.

3. Open the command prompt and navigate to your application's root directory.
4. To build deployable packages for the modernized application run the Maven build command:

```
mvn package
```

Refer to the Application Organization page for details on the basic organization of the modernized code. For instance, for modernized application containing a front-end web application, you may expect at-least the following deployable `.war` aggregates in addition to the runtime components:

- **Service project**: Contains legacy business logic modernization elements

```
<business-app>-service.*.war
```

- **Web project**: Contains the modernization of user interface-related elements

```
<business-app>-web.*.war
```

Step 3: Configure the test environment

To configure your test environment

1. Configure your modernized application runtime. For more information, see the Set up configuration for Runtime section in the *AWS Mainframe Modernization user guide*.

> **Note**
>
> Refer section 5 of the *Set up the AWS Automated Refactor Development Environment* guide for runtime component specific configuration examples.

2. Prepare input and output (I/O) data sets for modernized applications. Modernized applications may process sequential I/O data sets, VSAM data sets, or others.

> **Note**
>
> Refer section 6 of the *Set up the AWS Automated Refactor Development Environment* for examples.

3. A runtime environment - You can use your existing runtime environment or create a new runtime environment.

- To configure a non-managed runtime environment, see Set up a non-managed application.
- To configure a managed runtime environment, see Set up a managed application.

After configuring the test environment, you move to the next step of deploying the modernized application.

Step 4: Deploy the modernized application

Deploy the application artifacts in the runtime you created and/or configured in the the section called "Step 2: Build the modernized application" and the section called "Step 3: Configure the test environment" sections.

Additional guidance for deploying the modernized application can be found using these links:

- Deploy on Amazon EC2
- Deploy on containers on Amazon ECS and Amazon EKS
- Create an AWS Mainframe Modernization application

Step 5: Test the modernized application

After deployment,

1. Review the available Runtime APIs for ways to interact with the modernized applications.
2. Test your application to align its functional equivalence with legacy application. For example, see Test a sample application in the *AWS Mainframe Modernization user guide*.

Additional example

For a specific example of modernizing mainframe application with AWS Transform, see Modernize the CardDemo mainframe application.

Tutorial: Reimagining mainframe applications with exported artifacts from AWS Transform for mainframe

This tutorial walks you through how to leverage modernization outputs from your AWS Transform and use Kiro or any Spec-Driven IDE to generate modernized applications from mainframe COBOL and JCL code.

The example showcases the modernization of a simplified Customer Account Processing application, transforming COBOL code into equivalent Java microservices, complete with REST APIs, entity mappings, and automatically translated business rules. In this tutorial, we use Kiro as our IDE to demonstrate how to use AWS Transform output for reimagining the application.

Overview

In this tutorial, you will:

- Prepare your modernization outputs and Kiro workspace
- Generate target application specifications using methodology steering files and prompts
- Generate source code, tests, and optionally boost quality with Kiro Hooks

Prerequisites

Before beginning this tutorial, ensure you have:

- Mainframe application source code
- Kiro IDE installed locally
- Access to outputs from AWS Transform for mainframe, including:
  - Dependency and data analysis results
  - Technical documentation
  - Business documentation
- Java 17+ and Maven 3.9+ installed (for validating generated code)

Step 1: Prepare modernization outputs

Upon completion of your mainframe modernization job within AWS Transform, the service generates outputs stored in your designated S3 bucket. Download these artifacts to a local workspace that Kiro can access for the modernization process.

The S3 bucket stores the results of each Transform job in a hierarchical layout that makes it easy to locate the artifacts produced by a particular run. The structure is:

```
/your-s3-bucket/transform-output/your-jobid/
### 1/
    ### business-documentation/
    ### data_analysis/
    ### decomposition/
    ### documentation/
### inputs/
```

- `your-jobid` – A unique identifier that groups together everything generated for a single logical job (e.g., a migration or conversion request).
- `1/` – The first execution of that job. If you need to run the job again, a new numeric folder (2/, 3/, …) will be created alongside the existing one, preserving the output of each execution attempt.
- Sub-folders under `1/` – These folders hold the outputs from the first run; select only the documents that are relevant to your re-imagined project, as you may not need every file.
  - `business-documentation` – contains the extracted business logic for the entire application, in a zip archive.
  - `data_analysis` – provides data-lineage visibility for data modernization and generates data dictionaries with business meanings.
  - `decomposition` – holds workloads grouped into business domains as identified by the BLE agent.
  - `documentation` – includes technical documents that explain the mainframe application.
  - `inputs` – The inputs folder stores the original source artifacts of the mainframe application—COBOL code, JCL scripts, copybooks, and related configuration files.

Import all these files into the root of your Kiro project workspace.

Prepare your Kiro workspace

Kiro is an agentic IDE that helps you do your best work by bringing structure to AI coding with spec-driven development with features such as specs, steering, and hooks. Specs or specifications are structured artifacts that formalize the development process for complex features in your application. Steering gives Kiro persistent knowledge about your workspace through markdown files. Agent hooks are powerful automation tools that streamline your development workflow by automatically executing predefined agent actions when specific events occur in your IDE.

Steering files give Kiro persistent knowledge of your forward engineering workspace through markdown files. Instead of explaining your conventions in every chat, these files ensure Kiro consistently follows your established patterns, libraries, and standards throughout your project.

Place the steering files in the root of your workspace under `.kiro/steering/`.

Product Overview (product.md)

Defines your product's purpose, target users, key features, and business objectives. This helps Kiro understand the "why" behind technical decisions and suggest solutions aligned with your product goals. A sample product.md looks like:

```
Product Requirements — COBOL/JCL Business Rules Only

CRITICAL PRINCIPLE: COBOL/JCL BUSINESS RULES ONLY

ABSOLUTE REQUIREMENT: Implement ONLY the business rules explicitly defined in COBOL/JCL
 programs for CustomerAccountManagement Business Function ONLY.
- NO industry standards
- NO assumptions about validation logic
- NO inferences from incomplete COBOL/JCL rules
- ONLY explicit COBOL/JCL business rule specifications

COBOL/JCL Business Rule Implementation

Rule Discovery & Implementation
1. Locate Entry Points: ApplicationLevelAnalysis/*/entrypoint-*/
2. Locate Program Paths: Search cbl/ jcl programs mentioned in the Entry point files.
 cbl/jcl programs are located at program_paths.
3. Locate COBOL Program Rules: Search carddemo-v2-main/app/cbl/*.json
4. Locate JCL Program Rules: Search carddemo-v2-main/app/jcl/*.json
5. Extract Components: Rule_Id, Rule_Name, Rule_Description, Acceptance_Criteria,
 Rule_Type
```

```
6. Implementation: Group related COBOL/JCL rules into single methods with exact
 specification

MANDATORY Code Documentation:
/**
 * <Rule_Id>COBOL_OR_JCL_RULE_ID_1</Rule_Id>
 * <Rule_Description>Brief description from COBOL/JCL rule</Rule_Description>
 * <Acceptance_Criteria>Given/When/Then criteria from COBOL/JCL rule</
Acceptance_Criteria>
 *
 * <Rule_Id>COBOL_OR_JCL_RULE_ID_2</Rule_Id>
 * <Rule_Description>Brief description from related COBOL/JCL rule</Rule_Description>
 * <Acceptance_Criteria>Given/When/Then criteria from related COBOL/JCL rule</
Acceptance_Criteria>
 */
```

Technology Stack (tech.md)

Documents your chosen frameworks, libraries, development tools, and technical constraints. When Kiro suggests implementations, it will prefer your established stack over alternatives. A sample tech.md looks like:

```
Technology Stack & Implementation Standards

Technology Stack

Backend (LTS Only)
- Java: Eclipse Temurin LTS (Java 17)
- Spring Boot: Latest LTS 3.x
- Database: H2 (testing), Aurora PostgreSQL (production)
- Build: Maven wrapper (./mvnw) only

Frontend
- Framework: React + TypeScript
- State Management: Redux Toolkit
- API Client: Axios with error handling
- Styling: AWS Cloudscape Design System

AWS Infrastructure
- Database: Amazon RDS Aurora PostgreSQL
- Compute: ECS Fargate
- Load Balancing: ALB
- Registry: Amazon ECR
```

```
- Monitoring: CloudWatch
- CI/CD: AWS CodeBuild
- IaC: AWS CDK (v2)
```

Project Structure (structure.md)

Outlines file organization, naming conventions, import patterns, and architectural decisions. This ensures generated code fits seamlessly into your existing codebase. A sample structure.md looks like:

```
Project Structure & Data Organization

Project Structure

COBOL Artifact Locations
- Extracted business rules for each COBOL/JCL file: carddemo-v2-main/app/*
- Copybooks: source-code/cpy/
- Entry Points: ApplicationLevelAnalysis/*/entrypoint-*/

Data Loading Strategy
- Use DataInitializationService with CSV parsing
- Match CSV headers to copybook field names
- Active only with appropriate testing profile

COBOL/JCL Business Rule Discovery Process

Step-by-Step Rule Location
1. Locate Entry Points: Navigate to ApplicationLevelAnalysis/*/entrypoint-*/
2. Find Program References: Search cbl/ jcl programs mentioned in the Entry point files
3. Extract COBOL Rules: Search carddemo-v2-main/app/cbl/*.json for business rules
4. Extract JCL Rules: Search carddemo-v2-main/app/jcl/*.json for business rules
5. Parse Rule Components: Extract Rule_Id, Rule_Name, Rule_Description,
 Acceptance_Criteria, Rule_Type
```

These foundation files are included in every interaction by default, forming the baseline of Kiro's project understanding.

The Kiro specification workflow

Using the steering files and prompts you provide, Kiro automatically creates three core documents that become the backbone of every specification:

- `requirements.md` - Captures user stories and acceptance criteria in structured EARS notation
- `design.md` - Documents technical architecture, sequence diagrams, and implementation considerations
- `tasks.md` - Provides a detailed implementation plan with discrete, trackable tasks

These specifications are living documents. You can edit any of them at any time—adding, removing, or refining details as your understanding of the project evolves. Kiro's design encourages continuous refinement, so the specs remain synchronized with shifting requirements and design decisions, giving you a reliable foundation for development.

Next: the section called “Architecture specifications with Kiro”

Architecture specifications with Kiro

After you configure your Kiro workspace and steering files, the next step is to generate application specifications. You create methodology steering files that guide Kiro through business logic analysis, domain-driven design, and microservices decomposition for your target architecture.

Step 2: Generate application specifications

Tailor specifications generation to your architecture design target

The specification generation process can be tailored to align with your target application architecture and organizational requirements. For applications targeting a microservices architecture, we follow a Domain-Driven Design (DDD) approach, as demonstrated in this tutorial, where Kiro analyzes the business logic extracted by AWS Transform to identify bounded contexts, aggregate roots, entities, and domain events, ultimately generating comprehensive microservice specifications with clear service boundaries and integration patterns.

Alternatively, for applications designed with a layered architecture, we can adopt a more traditional Software Requirements Specification (SRS) standard, focusing on functional and non-functional requirements organized by system layers and components.

Additionally, organizations may opt to define a custom approach that reflects their specific methodologies, architectural patterns, or industry standards, ensuring the specification generation process aligns with their unique development practices and governance requirements.

Methodology steering files for a microservice architecture

In addition to the shared steering files, the specification generation step uses methodology-specific steering files. Place these alongside the other steering files under `.kiro/steering/`.

Business logic analysis methodology

This document describes the standardized methodology for analyzing Business Logic Extraction results from mainframe applications.

```
# Business Rules Extraction (BRE) Analysis Methodology
## Overview
This document describes the standardized methodology for analyzing Business Logic
Extraction results from mainframe applications, specifically for the CardDemo
application BRE output.
## Analysis Process
### Step 1: Identify the Entry Point
Location: input/bre_output/index.html
- This is the main navigation file for all extracted business logic
- Contains hierarchical structure of business functions and components
- Use this to understand the overall application structure
### Step 2: Locate Business Function Documentation
Location: input/bre_output/ApplicationLevelAnalysis/[BusinessFunctionName]/
Each business function folder contains:
- [BusinessFunctionName].json - Overview with key capabilities and components list
- [BusinessFunctionName].html - Human-readable overview
- entrypoint-[COMPONENT]/ - Subfolders for each component
### Step 3: Analyze Component Entry Points
Location: input/bre_output/ApplicationLevelAnalysis/[BusinessFunctionName]/entrypoint-
[COMPONENT]/
Each entrypoint folder contains:
- entrypoint-[COMPONENT].json - Component summary with:
  - Business functions performed
  - Program flow (functionality_flow)
  - Datasource summary with access types (READ/WRITE/UPDATE)
  - Environment summary (workload type, database types, integration components)
  - Program paths
- entrypoint-[COMPONENT].html - Human-readable version
Key Information to Extract:
1. Business Functions: List of operations performed
2. Program Flow: Call hierarchy and program relationships
3. Datasources:
   - Name and original name
```

```
   - Type (VSAM_KSDS, CICS_FILE, MQ_QUEUE, SEQUENTIAL, etc.)
   - Access mode (READ, WRITE, UPDATE)
   - Business purpose
   - Programs that use it
4. Environment: Workload type (Transaction/Batch), database types, integration
 components
### Step 4: Review Detailed Program Documentation
Location: input/bre_output/aws-mainframe-modernization-carddemo-main/app/
Detailed program files are organized by type:
- cbl/[PROGRAM]-cbl.json - COBOL programs
- jcl/[JCL]-jcl.json - JCL jobs
- app-vsam-mq/cbl/[PROGRAM]-cbl.json - MQ-enabled COBOL programs
Each detailed JSON file contains:
1. Description: High-level program purpose
2. Flow Diagram Code: Mermaid diagram showing detailed process flow
3. Rules: Array of business rules with:
   - Rule_Id: Unique identifier
   - Rule_Name: Descriptive name
   - Rule_Description: What the rule does
   - Rule_Type: Process Rules, Validation Rules, Decision Rules, Action Rules,
     Computation Rules, Definitional Rules
   - Acceptance_Criteria: Given/When/Then format
Note: These JSON files are single-line formatted and may appear truncated.
They contain 40-100+ business rules per program.
```

Domain Driven Design methodology

This document describes the standardized methodology for applying Domain-Driven Design principles to business functions extracted from mainframe applications. This process transforms technical mainframe analysis into a modern DDD model with bounded contexts, aggregates, entities, value objects, and domain events.

```
# Domain-Driven Design (DDD) Analysis Methodology
## Overview
This document describes the standardized methodology for applying Domain-Driven
Design principles to business functions extracted from mainframe applications.
## Prerequisites
- Completed BRE (Business Rules Extraction) analysis
- Business function analysis document available
- Understanding of the business domain
- Knowledge of DDD tactical and strategic patterns
## DDD Analysis Process
### Step 1: Identify Bounded Contexts
```

```
Input: Business function analysis with components, datasources, and business
 capabilities
Criteria for Bounded Context Identification:
- Cohesion: Related concepts that change together
- Autonomy: Can be developed and deployed independently
- Business Alignment: Matches business organizational structure
- Data Ownership: Clear ownership of data entities
- Language Consistency: Consistent terminology within context
### Step 2: Define Ubiquitous Language
Input: Bounded contexts, business function documentation, datasource field definitions
### Step 3: Identify Aggregates and Aggregate Roots
Input: Bounded contexts, datasources, entity relationships, business rules
### Step 4: Define Entities
Input: Aggregates, datasource structures, business rules
```

Microservices specification methodology

This document describes the standardized methodology for generating comprehensive microservice specifications from DDD (Domain-Driven Design) analysis. This process transforms DDD design artifacts (bounded contexts, aggregates, entities, value objects, domain services, and use cases) into detailed, implementation-ready microservice specifications.

```
# Microservice Specification Generation Methodology
## Overview
This document describes the standardized methodology for generating comprehensive
microservice specifications from DDD analysis.
## Prerequisites
- Completed BRE (Business Rules Extraction) analysis
- Completed DDD analysis with traceability matrix
- Understanding of microservices architecture patterns
- Knowledge of the target technology stack
## Microservice Specification Generation Process
### Step 1: Identify Microservices from Bounded Contexts
### Step 2: Define Service Overview
### Step 3: Define Service Boundaries
### Step 4: Define Data Ownership
```

Prompt for specification generation

Kiro specs are structured artifacts that formalize the development process for complex features in your application. They provide a systematic approach to transform high-level ideas into detailed

implementation plans with clear tracking and accountability. From the Kiro pane, click the + button under Specs and describe your project idea using prompts.

Below is a sample prompt for Kiro to generate application specifications:

```
Ultimate AWS Microservices Implementation Prompt

Context:
You are tasked with implementing the customer-management-service microservice
architecture based on specifications in the microservices-specs folder. This project
requires both backend microservices development and a frontend implementation. The
system will follow modern microservices best practices on AWS, including proper
service isolation, communication patterns, deployment strategies, and AWS-native
integration.

Role:
You are a Senior AWS Solutions Architect and Full-Stack Developer with over 20
years of experience designing and implementing cloud-native applications. You have
deep expertise in Java Spring microservices, Angular frontend development, AWS
services integration, and microservices design patterns.

Action:
1. Begin by analyzing the provided microservices specifications from the
   microservices-specs folder, identifying each required microservice, its
   responsibilities, data models, and integration points.
2. Design the overall architecture for the customer-management-service microservices
   system, including:
   - Service boundaries and responsibilities
   - Data ownership and sharing approach
   - Communication patterns (synchronous vs asynchronous)
   - AWS service selection for each component
3. For the customer-service microservice identified in the specifications:
   - Create a backend project structure with appropriate configuration
   - Implement the data model mapping the customer DynamoDB table definition
     located under the datamodel folder
   - Develop RESTful API controllers following REST best practices
   - Implement service layer business logic as specified
   - Add appropriate exception handling, validation, and logging
   - Configure AWS service integrations (DynamoDB, SQS, SNS, etc. as appropriate)
   - Write unit tests for the service
```

Generate specifications

Once you have reviewed the three specs and are satisfied that they accurately reflect the desired output, you have two options for execution:

- **Run tasks individually** – Execute each entry in tasks.md one-by-one, allowing you to monitor progress and intervene as needed.
- **Run all tasks at once** – Ask the Kiro agent to "Execute all tasks in the spec." Kiro will then generate the complete specification.

Next steps before source code generation

The generated specifications provide a solid starting point, but before you generate the source code, a software architect / application subject matter expert should:

- Review and validate the generated specification
- Adapt those specifications according to business requirement changes

Only after this human-in-the-loop verification and validation should you move to the next step which is the code generation.

Next: the section called “Code generation with Kiro”

Code generation with Kiro

After you define your application specifications, the next step is to generate source code and tests. You provide a code generation prompt to Kiro and optionally configure Hooks to improve code and test quality.

Step 3: Generate source code and tests

Prompt for code generation

Using the same shared steering files and the Kiro specification workflow described in the section called “Prepare your Kiro workspace”, create a new spec with a prompt tailored for code generation. From the Kiro pane, click the + button under Specs and describe your project idea using prompts.

Below is a sample prompt for Kiro spec generation:

```
Mainframe-to-AWS Modernization

Context
You are assisting in a comprehensive modernization of the Customer Account Management
 business function from the CardDemo mainframe application.

Project Structure
- source-code/ — COBOL copybook file definitions
- ApplicationLevelAnalysis/CustomerAccountManagement/ — Business function analysis with
 entrypoint documentation
- carddemo-v2-main/app/ — Extracted business rules per COBOL/JCL file

Target
Produce a compilable, deployable Java Spring Boot microservice that faithfully
 implements all mainframe business logic, following the steering guidelines exactly.

Initial Instructions
- Read and strictly follow the steering guidelines at .kiro/steering/mainframe-
modernization.md before any code generation or analysis.
- Fully understand business requirements, data flows, and program paths from the
 provided analysis files.
- Preserve every business rule and data structure exactly as specified — no assumptions
 or inference allowed.
- Generate code that complies with AWS managed services and architecture principles
 defined in the steering file.
- Produce a clean project structure with all dependencies declared and ready to build
 and deploy.
- Follow the AWS Well-Architected Tool Framework best practices for security,
 reliability, and operational excellence.

Deliverables
- Java Spring Boot microservice source code (fully working and deployable)
- Infrastructure as Code using AWS CDK (v2) with multi-stack architecture
- Granular tasks mapped to business rules and program entrypoints
- Documentation for all generated components

Notes
- If any information is missing or ambiguous, stop and explicitly request clarification
 before proceeding.
- Do not start code generation until the above steps are fully confirmed.
```

Execute code generation

Once you have reviewed the three specs and are satisfied that they accurately reflect the desired solution, you have two options for execution:

- **Run tasks individually** – Execute each entry in tasks.md one-by-one, allowing you to monitor progress and intervene as needed.
- **Run all tasks at once** – Ask the Kiro agent to "Execute all tasks in the spec." Kiro will then generate the complete codebase, unit tests, and functional tests for your new AWS-hosted application in a single pass.

Next steps before deployment

The generated code and test suite provide a solid starting point, but they are not yet production-ready. Before you ship the application, a developer should:

- Review the generated source code for correctness, security, and style compliance.
- Run the automatically generated unit and functional tests, fixing any failures.
- Perform additional manual testing (e.g., integration, performance, and user-acceptance testing) to validate the system in real-world scenarios.

Only after this human-in-the-loop verification should the application be considered ready for deployment.

Boost code and test quality with Kiro Hooks (optional)

Agent hooks are lightweight automation utilities that let you trigger predefined Kiro actions directly from your IDE whenever a specific event occurs. By wiring these hooks into your workflow you can automatically improve the quality of the code and tests that Kiro generates.

Why use a Hook for AWS Transform projects?

- **Complex business logic** – AWS Transform extracts and modernizes mainframe business rules.
- **Risk of omissions** – When Kiro produces new code it may miss subtle but critical rules.
- **Continuous feedback** – A hook can detect missing rules, invoke Kiro to regenerate the relevant sections, and automatically update the test suite.

Below is a sample of predefined hook prompt:

```
{
  "enabled": true,
  "name": "Business Rules Coverage Analyzer",
  "description": "Trigger for comprehensive business rules coverage analysis and
 implementation",
  "version": "1",
  "when": {
    "type": "userTriggered",
    "patterns": [
      "account-management-service/scripts/analyze-business-rules-coverage.py",
      "account-management-service/scripts/analyze-business-rules-coverage.json"
    ]
  },
  "then": {
    "type": "askAgent",
    "prompt": "COMPREHENSIVE BUSINESS RULES COVERAGE ANALYSIS\n\nYou are automatically
 analyzing and improving business rule coverage for account-management-service.
 Execute these steps in sequence:\n\n## STEP 1: Run Coverage Analysis\n• Execute: cd
 account-management-service & python3 scripts/analyze-business-rules-coverage.py\n•
 Capture the current coverage percentage from console output\n\n## STEP 2: Extract
 Specific Uncovered Rules\n• Execute: cd account-management-service &  python3 scripts/
extract_uncovered_rules.py\n• Do the Deep Analysis of the output File: account-
management-service/scripts/uncovered_rules_details.json\n• This section contains
 ONLY the rules that need implementation - focus exclusively on these\n• Do not
 implement rules that are already covered or not in this list\n• Prioritize from
 uncovered_rules_details by:\n  - Missing tagging (business logic exists but not tagged
 per steering guide)\n  - Missing implementation (business logic entirely missing)\n
  - High-impact rules (validation, process rules)\n\n## STEP 3: \n- Check where to
 implement this missing rule. \n- Scan ALL files under the Backend directories: \n1.
 account-management-service/src/main/java/com/enterprise/carddemo/controller\n2.
 account-management-service/src/main/java/com/enterprise/carddemo/service\n3. account-
management-service/src/main/java/com/enterprise/carddemo/repository\n - Scan ALL
 files under the Frontend directories:\n1. account-management-frontend/src\n\n###
 STEP 4:\n• After analyzing, Focus on 5-10 rules from uncovered_rules_details per
 automation cycle\n\n## STEP 5: Implement Improvements\n• DO NOT implement business
 rules without the ACTUAL Logic like using logger.debug without Actual Implemenation.
 \n• For missing tags: Add proper COBOL/JCL rule documentation to existing methods
\n• For missing logic: Implement new methods following the steering guide patterns
\n• Ensure all new code is:\n  - Properly integrated into service flows. Your newly
 Generated Code should be getting invoked from some methods.\n  - Follows existing
 patterns in Backend files like ValidationService, CobolAccountUpdateRulesService,
 etc. OR in frontened files.\n  - Includes proper error handling and logging\n  - Has
 appropriate COBOL/JCL rule documentation headers\n\n## STEP 6: CRITICAL - Verify No
```

```
 Dead Code (Integration Verification)\n• MANDATORY: Verify that ALL newly implemented
 business rule methods are actually invoked from somewhere in the application\n•
 For each new method created, trace the call path to ensure it's reachable:\n  -
 Backend: Check that new validation/service methods are called from Controllers, other
 Services, or Components\n  - Frontend: Check that new utility functions are called
 from React components, hooks, or other utilities\n• Examples of proper integration:\n
  - New validation methods should be called in Controllers or Services during request
 processing\n  - New formatting methods should be called when preparing data for
 display\n  - New business rule methods should be integrated into existing workflows\n
 - New UI utilities should be used in actual form fields or components\n• Fix dead code
 immediately: If any method is not invoked, either:\n  - Add proper integration points
 (recommended)\n  - Remove the dead code (if not needed)\n• Document the integration
 points in your summary\n\n## STEP 7: Validate Improvements\n• Re-run: cd account-
management-service & python3 scripts/analyze-business-rules-coverage.py`\n• Confirm
 coverage percentage increased\n• Report the improvement: \"Coverage improved from
 X% to Y% (+Z rules)\" with specific rule IDs moved from uncovered to covered\n\n##
 STEP 8: Verify and FIX compilation issues\n• Compile the Backend Code and Frontend
 Code to verify if the build is successful. If there are compilation issues, Fix the
 errors, but DO NOT implement quick fix or workarounds.\n\n## SUCCESS CRITERIA:\n#
 Coverage analysis runs successfully\n# At least 5-10 new rules implemented or tagged
\n# Coverage percentage increases\n# No compilation errors\n# All new code follows
 COBOL/JCL business rule patterns\n# NO DEAD CODE: All new methods have verified call
 paths\n# Integration points documented and tested\n\n## OUTPUT REQUIRED:\n• Summary
 of changes made\n• Before/after coverage statistics\n• List of newly covered rules\n•
 Integration verification: Document call paths for each new method\n• Confirmation that
 all code compiles and integrates properly\n• Verification that no dead code exists"
  }
}
```

Summary – What You've Accomplished

- Exported the artifacts from AWS Transform and imported them into a Kiro workspace.
- Defined project-wide conventions with steering files (product, technology, and structure) and a clear prompt to guide the IDE.
- Driven the three-phase workflow (Requirements → Design → Implementation) to produce living specifications.
- Generated a Java-Spring-Boot microservice, infrastructure-as-code, and a full test suite, with an optional Hook-based feedback loop to catch missing business rules.

Next Steps

- Code Review & Quality Assurance
- Test Execution & Enhancement
- CI/CD Integration
- Production Deployment & Monitoring

Full-stack Windows modernization

AWS Transform provides full-stack Windows modernization, including .NET modernization and SQL Server modernization.

The AWS Transform agent for .NET can help you modernize your .NET applications to be compatible with cross-platform .NET. You can modernize .NET using the .NET web app or the .NET IDE.

AWS Transform for SQL Server modernization is an AI-powered service that automates the full-stack modernization of Microsoft SQL Server databases and their associated .NET applications to Amazon Aurora PostgreSQL. The service orchestrates the entire migration journey from schema conversion, data migration and modifying application code to match the new target PostgreSQL, making your teams more productive by automating complex and labor-intensive tasks.

Topics

- Modernizing .NET with AWS Transform
- SQL Server modernization

Modernizing .NET with AWS Transform

The AWS Transform agent for .NET can help you modernize your .NET applications to be compatible with cross-platform .NET. This capability is called .NET modernization. After AWS Transform environment in AWS Transform, you can create a .NET modernization transformation job.

Capabilities and key features

- Analyze .NET Framework codebases from your source control systems, which includes private NuGet support, identifying cross repository dependencies, and providing an analysis report.
- Automated transformation of legacy .NET Framework applications to cross-platform .NET, email notifications, and a transformation summary report.
- Easy integration with the source control platforms (BitBucket, GitHub, and GitLab) to ingest existing code and commit transformed code to a new branch.
- Validation of transformed code through unit tests.

Supported versions and project types

AWS Transform supports transformation for these versions of .NET:

- .NET Framework 3.5+
- .NET Core 3.1
- .NET 5.x+
- .NET 8

AWS Transform can transform to these target .NET versions:

- .NET 8
- .NET 10

AWS Transform supports transformation of these types of projects (C# only):

- Class libraries
- Console applications
- ASP.NET:
  - Model View Controller (MVC), including front-end Razor Views
  - Single Page Application (SPA) back-ends (business logic layers)
  - Web API
  - Web Forms
- Unit test projects (NUnit, xUnit, and MSTest)
- Windows Communication Foundation (WCF) services
- Projects with provided cross-platform versions for third-party or private NuGet packages. If a cross-platform equivalent is missing or unavailable, AWS Transform for .NET will attempt a best-effort conversion.

AWS Transform can also transform the following project types to modern .NET. This is a preview feature, available only in the US East (N. Virginia) Region.

- WinForms desktop projects. This is a preview feature, available only in the US East (N. Virginia) Region.

- WPF desktop projects. This is a preview feature, available only in the US East (N. Virginia) Region.
- Xamarin mobile projects. This is a preview feature, available only in the US East (N. Virginia) Region.
- Projects written in VB.NET. This is a preview feature, available only in the US East (N. Virginia) Region.

Limitations

For more information on quotas and limitations for AWS Transform, see Quotas for AWS Transform.

AWS Transform does not transform the following:

- Blazor UI components
- Win32 DLLs that don't have core compatible libraries
- Repositories that do not contain any solutions.

AWS Transform will not modify the original repo branches, and can only write to a separate target branch specified in your transformation plan.

Human intervention

During the porting of .NET Framework applications to cross-platform .NET, you may be requested to provide input or approvals in the following scenarios:

- Set up a connector to your source code and permissions
- Validate the proposed modernization plan
- Upload missing package dependencies as NuGets
- Review and accept the transformed code

More information

You can modernize your .NET code by using either the AWS Transform web application or the AWS Toolkit for Visual Studio.

- Modernizing your .NET code by using the AWS Transform web application

- Modernizing .NET in the IDE

Modernizing your .NET code by using the AWS Transform web application

AWS Transform for .NET is a new generative AI-powered agent designed to modernize legacy .NET applications. You can modernize your legacy .NET code by using the AWS Transform web application or the Visual Studio AWS Toolkit extension.

To modernize your .NET code using the AWS Transform web application, navigate to the web application follow these detailed steps:

1. Creating the AWS Transform .NET job plan
2. Creating a source code repository connector
3. Confirming your repositories to prepare for transformation
4. Resolving package dependencies to prepare for transformation
5. Reviewing your plan to prepare for transformation
6. Transforming your .NET code

Creating the AWS Transform .NET job plan

After you create your workspace, on the **Jobs** tab, select **Create job**. Then follow the prompts from AWS Transform in the chat pane using natural language. These are the typical steps for creating a .NET modernization job.

1. AWS Transform will ask you which type of transformation job you would like to create. In the chat, enter *.NET modernization*.
2. AWS Transform will suggest a job name and ask you if you want to change the job name. If you would like to change the job name, tell AWS Transform in natural language, for example, *change the job name to ExampleCorpDotNet1*. Otherwise, in the chat, you can accept the suggested job name. After you accept the job name, AWS Transform notifies you in the chat window that it is creating the job.
3. AWS Transform creates the transformation job.

Components of the AWS Transform .NET job plan

The AWS Transform .NET job display has 5 tabs which you select from the vertical icons on the far left: **Tasks**, **Dashboard**, **Approvals**, **Artifacts**, and **Worklog**. Each tab has a left pane and a center chat pane. A right collaboration pane may appear at times to show additional details and to request human input.

Tasks (Job Plan)

This tab displays your job plan in the left pane. A .NET job has 5 steps:

1. *Get resources to be transformed*: In this phase, you create a connector to your code repository using AWS CodeConnections. Depending on your repository permissions, an admin of the code repository may need to approve the connector and give AWS Transform access to the repository.
2. *Discover resources for transformation*: In this phase, AWS Transform discovers repositories in source control, and you select some or all of them for assessment.
3. *Assess code for transformation*: Selected repositories are assessed, and you can view assessment reports.
4. *Prepare for transformation*: In this phase, AWS Transform notifies you if any dependencies are missing from your repositories. You can upload the missing dependencies or ignore them. If you are not an admin for the repo, an admin may need to approve the final transformation plan.
5. *Transform*: In this phase, AWS Transform transforms your repo and provides you the ongoing status during the transformation until it's completed. You can review transformation reports to understand what was changed and why.

You can see the status for each step:

- Not started
- Await user input
- In Progress
- Completed

Dashboard

The **Dashboard** tab provides a high level summary of the transformation. It displays metrics for the number of jobs transformed, transformation applied, and estimated time to complete the

transformation. Below the dashboard is a table of repositories and their status - In-progress, Failed, or Success.

Approvals

Approval requests for the job are displayed and completed on this tab.

Artifacts

Jjob-related artifacts are uploaded or downloaded from this tab.

Worklog

AWS Transform logs its actions in the **Worklog** tab. The **Worklog** provides a detailed log of the actions AWS Transform takes, along with human input requests, and your responses to those requests.

Creating a source code repository connector

After Creating the AWS Transform .NET job plan, on the **Tasks** tab, the left pane of the AWS Transform window lists the phases of the transformation job. The first phase is *Get resources to be transformed*. In this phase, you can *Invite collaborators* and *Connect a source code repository*. The right pane of the AWS Transform window is where you specify the details.

Each transformation job is required to have only one source code repository connector. The AWS Transform agent uses this connector to download your .NET codebase from GitHub, GitLab, or Bitbucket by using AWS CodeConnections. You must set up AWS CodeConnections in the same Region as your AWS Transform job. Alternatively, you can Connect to source code in Amazon S3 instead of a source code repository.

> **Note**
>
> If you have an existing source code repository connector, AWS Transform will notify you. Select **Use existing connector** to use this connector. If you do not wish to use the existing connector, see Deleting an existing connector before Adding a new connector. You can only have one source repository connector per job.

Adding a new connector

To create a new source code repository connector:

1. Enter the AWS account number that you would like to use for the AWS CodeConnections connector.
2. If you have not set up AWS CodeConnections for the same AWS account, Set up AWS CodeConnections.
3. Use the AWS Developer Tools Console to create a connection to your Bitbucket, GitHub, GitHub Enterprise Server, GitLab, or AzureDevOps repository.
4. Copy the Amazon Resource Name (ARN) of the connection you created.
5. Return to AWS Transform, and paste the ARN of the connection you created.
6. Enter a name for the connector.

> **Note**
>
> Do not enter personal information as part of the connector name.

7. Select **Initiate connector creation.**
8. AWS Transform makes the request to create a connector for AWS account you entered, in the same Region as the current transformation job, and notifies you that an approval request is ready to be approved by your AWS administrator in the AWS Management Console. Select **Copy the verification link**.
9. If you are the AWS administrator, sign into the AWS account and go to the verification link to approve the connection request. If you are not the AWS administrator, provide the verification link to your AWS administrator.
10. After the AWS administrator approves the connector request, select **Finalize connector** to complete the connector creation process. Should you wish to use a different connector, select **Restart**, to restart the connector creation process.

The Collaboration tab also includes details about your connector, which include:

- Status -- Approved or Pending approval
- AWS account ID
- AWS Region
- Connection ARN

Deleting an existing connector

> **Note**
>
> If you have an existing source code repository connector, AWS Transform will notify you. Select either **Use existing connector** or **Delete and create a new connector**.

If you choose to delete an existing source code repository connect, AWS Transform will warn you before actually deleting the connector.

1. You can delete an existing connector a couple of ways:
   a. If AWS Transform prompts you to, you can select **Delete and create a new connector**.
   b. You can also select **restart** in the prompt, **To modify this connector, you must restart**.
2. AWS Transform warns you that restarting will delete the connector. To delete the connector, select **Restart** again.
3. AWS Transform warns you again that deleting the connector will remove the connection to your third party repository, such as Bitbucket, GitHub, and GitLab. Also, any AWS Transform jobs that are using this connector will fail if the connector is deleted. To confirm deletion, type *delete* and select **Delete**.
4. To add a new source code repository connector, see Adding a new connector.

Connect to source code in Amazon S3

You can connect AWS Transform to source code in Amazon S3 as an alternative to connecting a source code repo.

S3 bucket organization

Original source code and transformed code are stored in a common S3 bucket, organized as shown below. You must set up your S3 bucket in the same region as your AWS Transform job. Upload your source code to the bucket as a zip file at root level. The zip file must contain a top level folder for each repository.

During transformation, AWS Transform creates a transform-output folder, and stores the transformation results in that folder. Transformation creates a zip file named *transformed-code.zip* containing the transformed code. This includes a code differences report file name diff.txt that highlights file changes at a project level.

```
<customer-bucket>/
### source-code.zip
         ### repo 1
         ### repo 2
         ### ......
         ### repo n
### transform-output/
      ### transformed-code.zip
```

Adding a new S3 Connector

To create a new S3 source code repository connector:

1. In the AWS console, create an S3 bucket.
2. Upload your source code as a zip file to the S3 bucket.
3. In the AWS Transform web app, start a .NET transformation job. In the **Connect a source code repository** step, select **Connect your source code in S3 bucket** and choose **Save**.
4. On the next page, enter your S3 bucket details and choose **Submit**.
   a. Connector name
   b. AWS Account ID
   c. S3 Bucket Arn
   d. S3 Bucket Encryption Key (optional)
5. Approve the connector:
   a. On the next page, copy the verification link.
   b. Have an approver browse to the link to reach the AWS Transform connector creation request page.
   c. Choose to create a new role or use an existing role.
   d. After reviewing the request, choose **Save** and **Approve** to approve it.
6. Specify the S3 zip file location:
   a. In the AWS Transform web app, wait for status to show **Approved**.
   b. On the **Specify asset location** page, enter an S3 URL for the code zip file in the format `s3://bucket-name/zip-file-name` and choose **Send to AWS Transform**.
   c. The job proceeds to the **Discovery** step to continue the transformation.

Deleting an S3 Connector

To delete an existing S3 connector:

1. In the AWS Transform web app, select **Manage connectors** at the top right.
2. In the **Manage connectors** window, select the connector.
3. Choose **Delete**.

Confirming your repositories to prepare for transformation

After Creating a source code repository connector, confirm your repositories as the first step of preparing for transformation. You can:

- Review and change Default Settings that affect the plan contents.
- Review discovered repositories and select which ones to assess for transformation.
- Review the repository assessment report and chat with AWS Transform about its contents.
- Use the plan generated by AWS Transform.
- Customize the transformation plan using the web app or by downloading a JSON file, modifying it, and uploading it to AWS Transform.

Default Settings

In the **Job details** section of the **Collaboration** tab, you can set the following:

On the Tasks tab, in the Prepare for transformation - Confirm the repositories to transform step, you can set the following:

- Set the target branch name where the transformed code is written to your repository.
- Set the target .NET version:
  - **.NET 8**: transform all projects to .NET 8
  - **.NET 10**: transform all projects to .NET 10
- You can update the default Job Settings, or leave them as-is. These include:
  - **Exclude .NET Standard projects from the transformation plan** This setting is selected by default. When selected, AWS Transform excludes any .NET Standard projects from the transformation plan. If you deselect this setting, .NET Standard projects will be transformed, which will make them no longer compatible with the .NET Framework.

- **Check the NuGet sources and get .NET compatible package versions** Allow AWS Transform to search for and get .NET compatible package versions for the code that you would like to transform.

Review the repository assessment report

The repository assessment report provides a tabular view of information about each repository detected by AWS Transform. This includes:

- Repository Owner
- Assessed Branch
- No. of solutions
- No. of projects
- Total Lines of Code
- Detected .NET Versions
- Detected Project Types
- No. of Private Nuget Dependencies
- No. of Public Nuget Dependencies
- Transformation Complexity

To download the file:

1. In your workspace, select the **Collaboration** tab.
2. In the **Choose repositories** pane, select **Download**, and choose **Download as HTML** or **Download as JSON**

Chat about the repository summary report

Your repository summary may be very long. Use chat to better understand its contents. Your report is available for chat when you see this message on the **Worklog** tab: *Assessment report is now available in chat for queries*. To open the chat click the purple hexagonal icon in the lower right corner of the web console. Here are some example prompts:

- List all repositories discovered or assessed.
- How many .NET 6 projects were discovered in the assessment?

- Which projects have the highest complexity?

Use the plan generated by AWS Transform

You can choose to use the AWS Transform plan created from discovering repositories with a higher probability of transformation success, which includes the last modified, root, and cross-dependent repositories.

AWS Transform lists some high level details about the transformation plan, which include:

- *Total number of repositories discovered* This number of legacy .NET repositories that AWS Transform discovered using your source code repository connector. AWS Transform can scan a maximum of 1,000 repositories in each job. If you have more repositories to scan or transform, you can create multiple transformation jobs.
- *Selected repositories* The number of legacy .NET repositories that AWS Transform selected for transformation.
- *Selected dependencies* The number of dependencies that the selected repositories require, which AWS Transform auto-detected and added to the transformation plan.

The *Selected repositories* table lists the selected repositories. You can search repositories by name, navigate to additional pages of repositories, and download a JSON formatted file that lists the selected repositories and their dependencies. You can make any changes in this view.

The *Selected repositories* table lists the following details about the repositories that AWS Transform selected for transformation:

- Repository name
- Source branch
- Last modified date and time
- Lines of code This allows you to see if you're approaching your quota limits. For more information, see Quotas for AWS Transform.
- Dependent repositories You can click on the number of dependent repositories to view more details about these dependencies.

If you are happy with the AWS Transform generated transformation plan, select **Confirm repositories**.

If you would like to make changes to the AWS Transform generated transformation plan, select Customize the transformation plan.

Customize the transformation plan

If you choose not to Use the plan generated by AWS Transform, you can customize the list of repositories to transform using one of the following options:

1. Select repositories in the **Selected repositories** table.
   a. When you add a repository to the plan, AWS Transform will select the repository's dependencies for you and automatically add them to the transformation plan.
   b. You can also deselect repositories in the table.
2. Download the AWS Transform generated JSON list of repos and branches.
   a. Select **Download** and then select **Download as JSON**.
   b. Review and modify the JSON. You can remove repositories, or solutions within repositories, that you don't want to transform.
   c. Upload the revised JSON by selecting **Upload customized plan**.
   d. If the JSON uploads successfully, the **Selected repositories** table displays the selections you made.
3. You can select **Show dependent repos** to display the list of repositories that AWS Transform discovered are dependencies for the repositories you selected.
4. You can repeat these steps if needed. When you are satisfied with your customized plan, select **Confirm repositories**.

Resolving package dependencies to prepare for transformation

After Confirming your repositories to prepare for transformation, if AWS Transform finds missing package dependencies, you must complete this step. You can run a Windows PowerShell script to get the missing package dependencies from the same device as your Visual Studio development environment, or you can retrieve the missing packages manually. Then, upload the missing packages.

The Missing Package Dependencies can be updated in two ways:

- **Resolve using Artifact Connector:** Resolve by configuring an Artifact Repository (ADO NuGet Connector) through a dedicated Connect Artifact Repository HITL workflow, which is displayed when an Artifact Connector is not configured and packages are missing.

- **Update missing package:** AWS Transform lists the missing packages in the Missing package dependencies table. You can search for a missing package by name in the search box. This table includes the following details about the missing packages:
  - Name
  - Associated repositories
  - Framework version status
  - Core version status

To resolve the missing package dependencies, Upload the missing packages.

Upload the missing packages

1. If you choose, you can download a Windows PowerShell helper script to retrieve the missing package dependencies from within your Visual Studio development environment. Or you can find the missing packages manually.

   To use the Windows PowerShell script:

   a. Select **Download Windows PowerShell script**.

   b. Run the script locally with an active connection to the repositories that contain the missing package dependency files.

   c. This script allows you to download the missing package dependencies to your local environment.
2. The script will create a single zip file for you to upload which includes all of the dependencies in one archive. You can also upload individual `.nupkg` or zip files for each dependency.
3. Select **Upload package files.**
4. In the **Upload dependency files** modal, select **Choose files** and browse to the location of the compressed missing package files on your device.
5. Select **Upload**.
6. AWS Transform validates the files you uploaded. During validation, you cannot make any updates. AWS Transform reports the validation status above the *Missing package dependencies* table.
7. AWS Transform also updates the status columns in the *Missing package dependencies* table from *Missing* to *Resolved*. If a package fails validation, its status becomes *Invalid*. For invalid files do the following:

a. In the *Missing package dependencies* table, select the invalid package using the check box.

b. Select **Remove uploaded file**.

c. This changes its status back to *Missing*.

8. After you have uploaded the missing packages and resolved the package dependencies, select **Proceed to review**.

   If you select **Proceed to review** without resolving the missing package dependencies, AWS Transform asks if you would like to start the transformation job without the missing packages. If you select **Ignore the missing package dependencies**, AWS Transform will use assembly references to transform the code. Proceeding with this action can affect related resources.

Folder Structure Requirements

For NuGet uploads AWS Transform requires either:

- The `.nuspec` file to be placed at the root level for manually uploaded packages, or
- Use of the provided PowerShell script to generate the correct structure

Version matching requirements

- Framework status: AWS Transform attempts to match the version of the dependency specified in the `.csproj` file (source code) with the version of the uploaded dependency. An exact match results in a **Success** status.
- Core status: AWS Transform expects a core version specification in the dependencies section of the uploaded `.nuspec` file. If one is found then the core status is **Success**.

  Example format:

```
<dependencies>
    <group targetFramework="net8.0" />
</dependencies>
```

  Common versions include:

  - .NET 5.0 (net5.0)
  - .NET 6.0 (net6.0)
  - .NET 7.0 (net7.0)

- .NET 8.0 (net8.0)

Reviewing your plan to prepare for transformation

After Confirming your repositories to prepare for transformation, and Resolving package dependencies to prepare for transformation, the AWS account administrator must review the transformation plan and approve it in AWS Transform.

AWS Transform displays the job's list of repositories, dependent repositories, and dependent packages that were selected for transformation.

Reviewing the transformation plan

AWS Transform displays the job's list of repositories, dependent repositories, and dependent packages that were selected for transformation.

> **Note**
>
> AWS Transform can transform a maximum of 100 dependencies and repositories per transformation plan.

1. If you are not the AWS account administrator, review the job plan, and if you accept it, select **Send for approval**.
2. If you are the AWS account administrator, you must review the plan, and when ready, approve the plan to start the transformation. After you review the job plan, select either:
   a. *Reject* If the job was created by a user who is not the AWS account administrator, we suggest you notify the job creator to restart the job.
   b. *Approve and start transformation*.

The job review includes the following details:

1. *Job summary* This includes:
   a. The target branch where AWS Transform will place the transformed code.
   b. The target .NET version, .NET 8.0 or .NET 10.
   c. The job settings:
      i. Exclude .NET standard projects

d. Number of repositories selected for transformation

e. Number of dependent repositories

f. Number of private NuGet packages

g. Total lines of code for the job

2. *Repositories selected* These are the repositories selected for transformation. They must be either MVC, Web, Windows Communication Foundation (WCF), Console, class library, UI framework - Razor pages, or unit test packages. This table includes the following information:

   a. Name

   b. Source branch

   c. Supported projects

   d. Lines of code

   e. Projects detected

   f. Projects skipped

   g. Dependencies detected

3. *Dependent repositories added* These are the dependent repositories added for transformation. They must be either MVC, Web, Windows Communication Foundation (WCF), Console, class library, UI framework - Razor pages, or unit test packages. This table includes the following information:

   a. Name

   b. Needed by

   c. Source branch

   d. Supported projects

   e. Lines of code

   f. Projects detected

   g. Projects skipped

4. *Dependent packages* These are the dependent packages added for transformation. They must be either MVC, Web, Windows Communication Foundation (WCF), Console, class library, UI framework - Razor pages, or unit test packages. This table includes the following information:

   a. Name

   b. Associated repositories

   c. Framework version status

d. Core version status

Transforming your .NET code

After the AWS account administrator finishes Reviewing your plan to prepare for transformation, you can view the transformation progress on the *Dashboard* tab of your transformation job.

> **Note**
>
> In addition to transforming repositories and dependencies, AWS Transform can execute fully transformed (zero build errors) unit test projects. AWS Transform doesn't have access to unit test results prior to the transformation and can only share post-transformation results. You can then compare your baseline data, prior to transformation, with the post-transformation results to understand any potential gaps.

In the top right corner, you can see the job status, which has one of the following values:

- Awaiting user input
- Time elapsed
- Running

You can also see the following icons:

- A stop transformation icon
- A refresh icon
- A settings icon

The *Dashboard* provides a high level summary of the transformation. It shows metrics for the number of jobs transformed and transformation applied, and the estimated time to complete the transformation.

The *Dashboard* includes:

1. The transformation *Job details* section, which lists the default settings and details of the transformation job, including:

a. *Target branch destination* To transform your code, AWS Transform creates a new branch for the transformed code in your code repo.

b. *Target .NET version*, .NET 8.0 or .NET 10

c. The AWS Transform *job ID.*

d. The job settings:

   i. Exclude .NET standard projects

2. The *Transformation summary* section contains:

   a. The number of repositories selected for transformation

   b. The number of projects to be transformed

   c. The total lines of codes in these repositories and projects

After the transformation starts, pie charts appear in the *Repository status*, *Package status*, and *Unit test status* sections displaying progress in real time. The *Unit test status* shows the status of unit tests located in your repositories that AWS Transforms runs after transformation to test the transformed code. AWS Transform shares the executed test results, along with individual test name for customers to review the list of unit tests passed and failed.

> **Note**
>
> The *Repositories* section lists the repositories that AWS Transform recommends or that you selected for transformation. Select **Download JSON** to download a list of repositories, dependencies, and packages in your transformation plan.

Review the transformation reports

After a transformation job completes, you can download Transformation Reports:

- Transformation summary report: This report provides an overview of the transformation in HTML or JSON format. Download the transformation summary report from the **Dashboard tab**. Select **Download**, and choose **Download as HTML** or **Download as JSON**.
- Transformation detail report: This report provides in-depth information about a transformation in HTML format. Download transformation detail reports from the **Dashboard tab**. Select a **Download detailed report** link in the repository list to download a detail report.

Chat with AWS Transform about the transformation report

You can chat with AWS Transform about the transformation report after a repository has been completely processed or after the transformation job is completed. The **Worklog** tab displays this message for each repository that is available for chat: *Repository `repository_name` transformation details are now available in chat for queries*. It displays this message when the transformation job is completed: *AWS Transform transformation job is completed*.

To open the chat click the purple hexagonal icon in the lower right corner of the web console.

Here are some example prompts:

- Which projects were successfully transformed?
- Which projects were partially ported?
- What changes were made to the `repo_name` repository?
- What packages were upgraded in the `project_name` project?

Modernizing .NET in the IDE

AWS Transform for .NET is a new generative AI-powered agent designed to modernize legacy .NET applications. For more information, see Modernizing .NET with AWS Transform. You can modernize your legacy .NET code by using the AWS Transform web application or the Visual Studio AWS Toolkit extension. Use AWS Transform in integrated development environments (IDEs) to get assistance with your software development needs. In IDEs, AWS Transform includes capabilities to provide guidance and support across various aspects of .NET code modernization.

To transform a .NET solution or project, the AWS Transform agent analyzes your codebase, determines the necessary updates to port your application, and generates a transformation plan before the transformation begins. During this analysis, the AWS Transform agent divides your .NET solution or project into code groups that you can view in the transformation plan. A code group is a project and all its dependencies that together generate a buildable unit of code such as a dynamic link library (DLL) or an executable.

During the transformation, the AWS Transform agent provides step-by-step updates in the AWS **Transformation Hub** window where you can monitor progress. After transforming your application, AWS Transform generates a summary with the proposed changes in a diff view for you to optionally verify the changes before you accept them. When you accept the changes, AWS Transform makes in-place updates to your .NET solution or project.

AWS Transform performs four keys tasks to port .NET applications to Linux:

- Upgrades language version – Replaces outdated C# versions of code with Linux-compatible C# versions.
- Migrates from .NET Framework to cross-platform .NET – Migrates projects and packages from Windows dependent .NET Framework to cross-platform .NET compatible with Linux.
- Rewrites code for Linux compatibility – Refactors and rewrites deprecated and inefficient code components.
- Generates a Linux compatibility readiness report – For open-ended tasks where user intervention is needed to make the code build and run on Linux, AWS Transform provides a detailed report of actions needed to configure your application after transformation.

For more information about how AWS Transform performs .NET transformations, see How AWS Transform modernizes .NET applications.

Quotas

For AWS Transform .NET transformation quotas in the IDE, see Quotas for AWS Transform.

To modernize your .NET code using the AWS Transform, part of the Visual Studio AWS Toolkit extension, see the following:

- Modernizing .NET using AWS Transform in Visual Studio
- How AWS Transform modernizes .NET applications
- Troubleshooting issues with .NET transformations in the IDE

Modernizing .NET using AWS Transform in Visual Studio

Complete these steps to port a Windows-based .NET application to a Linux-compatible cross-platform .NET application with AWS Transform in Visual Studio 2026 or 2022.

Step 1: Prerequisites

Before you continue, make sure you've downloaded and installed the AWS Toolkit extension with Amazon Q from the Visual Studio Marketplace. Review the AWS Transform capabilities and limitations for .NET modernization to confirm that your code can be transformed.

Step 2: Authenticate in Visual Studio

To connect to your AWS accounts from the Toolkit for Visual Studio, open the Getting Started with the AWS Toolkit User Interface (connection UI):

1. From the Visual Studio main menu, navigate to **Extensions > AWS Toolkit > Getting Started** to open the **Getting Started: AWS Toolkit page**.
2. Select either **AWS Transform** or **Amazon Q Developer** authentication.
3. Choose an existing profile or create a new one. When creating a new profile, enter a name for the profile and your AWS Transform start URL, which was emailed to you after you were added to IAM Identity Center by your administrator.
4. Choose **Connect**. Follow the prompts to sign in and grant access permissions through your web browser. In Visual Studio, you should see **Connected with IAM Identity Center**.

Step 3: Transform your application

To transform your .NET solution or project, complete the following procedure:

1. Open any C# based solution or project in Visual Studio that you want to transform.
2. Open any C# code file in the editor.
3. Choose **Solution Explorer**.
4. From the **Solution Explorer**, right click a solution or project you want to transform, and then choose to port or transform the solution or project with AWS Transform.
5. The AWS Transform window appears.

   The solution or project you selected will be chosen in the **Choose a solution or project to transform** dropdown menu. You can expand the menu to choose a different solution or project to transform.
6. In the **Choose a workspace** dialog, create a workspace or select an existing one. Workspaces make your activity visible in the AWS Transform web application, where you can invite collaborators to view and chat about transformation jobs.
7. In the **Choose a .NET target** dropdown menu, choose the .NET version you want to upgrade to.
8. You can update the Default Settings, or leave them as-is. These include:

   - Exclude .NET Standard projects from the transformation plan

This setting is selected by default. When selected, AWS Transform excludes any .NET Standard projects from the transformation plan. If you deselect this setting, .NET Standard projects will be transformed, which will make them no longer compatible with the .NET Framework.

- Check the NuGet sources and get .NET compatible package versions

  Allow AWS Transform to search for and get .NET compatible package versions for the code that you would like to transform.

9. Choose **Start** to begin the transformation.
10. Wait while AWS Transform analyzes your code and generates an assessment report and a code transformation plan.

    You will be prompted with **View Transformation Plan? Choose Edit Plan** to review or edit the transformation plan, or **Start transformation** to transform with the existing plan.
11. If you choose to edit the plan, a *transformation-plan.md* markdown file will open in the code editor. The transformation plan describes the objective and details of the transformation in a series of steps. Refer to the instruction comments at the end if the plan. You can modify this plan in the following ways:

    - **Transformation steps and substeps**: change, add, or remove steps and substeps to control structural transformation and package update details
    - **Preview Project Transformation**: Add experimental transformation of WinForms desktop projects, WPF projects, Xamarin projects, or projects written in VB.NET

      > **Note**
      >
      > Other project transformations are a preview feature, available only in the US East (N. Virginia) Region.

    - **Customization section**: add customization instructions, such as dependency injection instructions or coding pattern examples.

      > **Note**
      >
      > Coding pattern examples are a preview feature, available only in the US East (N. Virginia) Region.

After saving any plan changes, choose **Start Transformation** to start the transformation.

12. AWS Transform begins transforming your code.

    An **AWS Transformation Hub** opens where you can monitor progress for the duration of the transformation. Here you can see estimated time remaining, a Job ID you can copy to the clipboard, and transformation steps. Choose a step to see the worklog for that step, which explains the current transformation activity.

    Once AWS Transform has started the **Transforming code** step you can navigate away from the project or solution for the duration of the transformation.

13. After the transformation is complete, navigate to the **Transformation Hub** and choose **View diffs** to review the proposed changes in a diff view. To update your files in place, choose **Accept changes** from the Actions dropdown menu.

14. Choose **View code transformation summary** for a summary of the changes AWS Transform made. For details about transformation, including changes made, the reason for changes, and actionable error details, you can download a transformation detail report by choosing **Download transformation report**. Refer to Transformation Reports for more information. The transformation detail report includes a Linux readiness report section that identifies compatibility issues in moving from Windows to Linux that require human attention.

    If any of the items in the **Code groups** table require input under the Linux porting status, you must manually update some files to run your application on Linux. The transformation detail report includes a Linux readiness report section that identifies compatibility issues in moving from Windows to Linux that require human attention.

15. AWS Transform generates a *NextSteps.md* file, which describes any remaining porting work needed. To download the report, choose the **Download next steps** button at the top of the transformation detail report. You can use the NextSteps.md file to transform the project again with AWS Transform and a revised plan, or use it as input to an AI code companion.

16. To update your files in place, choose **Accept changes** from the **Actions** dropdown menu.

How AWS Transform modernizes .NET applications

Review the following sections for details about how .NET transformation with AWS Transform works.

Analyzing your application and generating a transformation plan

Before a transformation begins, AWS Transform builds your code locally to verify that it is buildable and configured correctly for transformation. If the code does not build, AWS Transform will prompt whether to continue with transformation. AWS Transform then uploads your code to a secure and encrypted build environment on AWS, analyzes your codebase, and determines the necessary updates to port your application

During this analysis, AWS Transform divides your .NET solution or project into code groups. A code group is a project and all its dependencies that together generate a buildable unit of code such as a dynamic link library (DLL) or an executable. Even if you didn't select all project dependencies to be transformed, AWS Transform determines the dependencies needed to build your selected projects and transforms them too, so that your transformed application will be buildable and ready for use.

After analyzing your code, AWS Transform generates a transformation plan that outlines the proposed changes that it will make, including a list of code groups and their dependencies that will be transformed.

Transforming your application

To start the transformation, AWS Transform builds your code again in the secure build environment to verify if it is buildable remotely. AWS Transform then begins porting your application. It works from the bottom up, starting with the lowest level dependency. If AWS Transform runs into an issue with porting a dependency, it stops the transformation and provides information about what caused the error.

The transformation includes the following updates to your application:

- Replacing outdated C# versions of code with Linux-compatible C# versions
- Upgrading .NET Framework to cross-platform .NET, including:
  - Identifying and iteratively replacing packages, libraries, and APIs
  - Upgrading and replacing NuGet packages and APIs
  - Transitioning to cross-platform runtime
  - Setting up middleware and updating runtime configurations
  - Replacing private or third-party packages
  - Handling IIS and WCF components
  - Debugging build errors

- Rewriting code for Linux compatibility, including refactoring and rewriting deprecated and inefficient code to port existing code

Reviewing transformation report and accepting changes

After the transformation is complete, AWS Transform provides a transformation report with information about the proposed updates it made to your application, including the number of files changed, packages updated, and APIs changed. It flags any unsuccessful transformations, including affected files or portions of files and the errors encountered during an attempted build. You can also view a build summary with build logs to learn more about what changes were made.

The transformation report also provides a Linux porting status, which indicates whether or not additional user input is needed to make the application Linux compatible. If any of the items in a code group require input from you, you download a Linux readiness report that contains Windows-specific considerations that AWS Transform could not address at build time. If input is needed for any code groups or files, review the report for details about what type of change still needs to be made and, if applicable, for recommendations for how to update your code. These changes must be made manually before your application can be run on Linux.

You can review the proposed changes AWS Transform made in a diff view before accepting them as in-place updates to your files. After updating your files and addressing any items in the Linux readiness report, your application is ready to run on cross-platform .NET.

You can download a *Next Steps* markdown file with prompts for continued transformation with AWS Transform or an AI code companion. To download the file, choose the **Download Next Steps** button at the top right of the report.

Troubleshooting issues with .NET transformations in the IDE

Use the following sections to troubleshoot common issues with .NET transformations in the IDE with AWS Transform.

How do I know if a job is progressing?

If AWS Transform appears to be spending a long time on a step in the **AWS Transformation Hub**, you can check whether the job is still active in the output logs. If diagnostic messages are being generated, the job is still active.

To check the outputs, choose the **Output** tab in Visual Studio. In the **Show output from:** menu, choose **Amazon Q Language Client**.

The following screenshot shows an example of the outputs AWS Transform generates during a transformation.

Why are some projects not selected for transformation?

AWS Transform can only transform supported .NET project types. For a list of supported project types and other prerequisites for transforming your .NET projects, see Step 1: Prerequisites.

How can I get support if my project or solution isn't transforming?

If you aren't able to troubleshoot issues on your own, you can reach out to Support or your AWS account team to submit a support case.

To get support, provide the transformation job ID so AWS can investigate a failed job. To find a transformation job ID, choose the **Output** tab in Visual Studio. In the **Show output from:** menu, choose **Amazon Q Language Client**.

How can I prevent my firewall from interfering with transformation jobs?

If your organization uses a firewall, it might interfere with transformations in Visual Studio. You can temporarily disable security checks in Node.js to troubleshoot or test what is preventing the transformation from running.

The environment variable `NODE_TLS_REJECT_UNAUTHORIZED` controls important security checks. Setting `NODE_TLS_REJECT_UNAUTHORIZED` to "0" disables Node.js's rejection of unauthorized TLS/SSL certificates. This means:

- Self-signed certificates will be accepted
- Expired certificates will be allowed
- Certificates with mismatched hostnames will be permitted
- Any other certificate validation errors will be ignored

If your proxy uses a self-certificate, you can set the following environment variables instead of disabling `NODE_TLS_REJECT_UNAUTHORIZED`:

```
NODE_OPTIONS = –use-openssl-ca
NODE_EXTRA_CA_CERTS = Path/To/Corporate/Certs
```

Otherwise, you must specify the CA certs used by the proxy to disable NODE_TLS_REJECT_UNAUTHORIZED.

To disbale NODE_TLS_REJECT_UNAUTHORIZED on Windows:

1. Open the Start menu and search for **Environment Variables**.
2. Choose **Edit the system environment variables**.
3. In the **System Properties** window, choose **Environment Variables**.
4. Under **System variables**, choose **New**.
5. Set **Variable name** to NODE_TLS_REJECT_UNAUTHORIZED and **Variable value** to 0.
6. Choose **OK** to save the changes.
7. Restart Visual Studio.

Porting UI layer code

AWS Transform can port the UI layer of ASP.NET websites to UI frameworks compatible with ASP.NET Core, which can run on Linux. The following transformations are available:

- MVC Razor UI porting
- Web Forms to Blazor UI porting

MVC Razor UI porting

ASP.NET MVC projects with Razor Views UI can be ported to MVC Razor Views on ASP.NET Core, remaining on the same UI framework. While Razor views are similar between ASP.NET and ASP.NET Core, there are differences that require porting code changes:

- ASP.NET MVC Razor Views use the System.Web.Mvc namespace and are closely tied to the .NET Framework.
- ASP.NET Core Razor Views are modular, built on Microsoft.AspNetCore.Mvc, with improved dependency injection.
- ASP.NET Core has new features including Tag Helpers and View Components.

AWS Transform detects and ports common incompatible components and constructs in Razor Views.

Web Forms to Blazor UI porting

ASP.NET Web Forms UI does not have a counterpart on ASP.NET Core. AWS Transform can port Web Forms UI layer code to Blazor Server on ASP.NET Core. UI layer code is rewritten for the target UI framework.

To enable or disable Web Forms to Blazor UI porting, check or clear the Transform UI ASP.NET Web Forms to Blazor job setting.

Transformation details

Your Web Forms project is converted to a Blazor project with server-side hosting. The following project and file changes are made during transformation:

File transformation mapping

From	To	Description
*.aspx, *.ascx	*.razor	.aspx pages and .ascx custom controls become .razor files
Web.config	appsettings.json	Web.config settings become appsettings.json settings
Global.asax	Program.cs	Global.asax code becomes Program.cs code
*.master	*layout.razor	Master files become layout.razor files

Post-transformation project structure

After transformation, the transformed files reside in the following project locations:

- Components folder: All razor components (*.razor) are placed in a Components folder created at the same level as the project's .csproj file.
- Pages mapping: Web Forms pages (*.aspx) are converted to Components/Pages folder structure.
- Controls mapping: Web Forms controls (*.ascx) are placed directly under the Components folder.

Limitations and unsupported features

Projects with up to 1000 pages (.aspx + .ascx files) can be transformed.

The following features are not supported at present. If your workloads use these features, that code will need to be ported manually or with an AI code companion post-transformation.

- .asmx files (XML web service)
- .svc files (service files)
- .ashx files (generic handler files)
- Service references & web references (reference.map, reference.cs, .wsdl files)
- Mixtures of .aspx (Web Forms) files and .cshtml (Razor) files in the same project
- Dependencies on project types not yet supported by AWS Transform for .NET
- Third-party vendor UI libraries/controls
- Web Site projects

Only Web Application projects (containing a .csproj file) are supported. To convert a Web Site project to a Web Application project, see Microsoft guidance Converting a Web Site Project to a Web Application Project.

Transformation Reports

After your .NET transformation job completes these transformation reports are available:

- Transformation Summary Report
- Transformation Detail Report

Transformation Summary Report

After a transformation job completes, a transformation summary report is available in the web console. Download the transformation summary report from the **Dashboard** tab. Select **Download Transformation Summary Report**, and choose **Download as HTML** or **Download as JSON**.

The Transformation Summary Report provides an overview of the transformation and information specific to each repository that was transformed, including:

- Job Details
  - Transformation job overview
  - Transformation results overview
  - Files overview
- Repositories list, and for each repository:
  - Repository Name Overview
  - Solution Transformation Summary
  - Project Details, including Status, Type, .NET Version, Total Lines of Code, Transformed Lines of Code, and Files Changed

Transformation Detail Report

After a transformation job finishes, transformation detail reports are available for both web console and IDE users.

- Web console users: a detail report is available for each transformed repository. On the **Dashboard** tab, use the **Download detailed report** links in the repository list to download a detail report.
- IDE users: a detail report is available for the transformed solution or project. On the **Transformation summary pane**, use the **Download report button** to download a detail report.

The transformation detail report is an interactive HTML report. It contains in-depth information about the transformation, including the reasons for file changes and actionable error details. The information is arranged in the following hierarchy:

- Job Information
- For each solution:
  - Solution Transformation Summary
  - Transformation Overview
  - Projects
  - For each project:
    - Project Summary
    - Build Errors Summary
    - NuGet package changes

    - File Changes
    - Build Errors
- Unit Test Results
- Linux Readiness Report

For a walk-through of the transformation detail report sections with examples, refer to the blog post Announcing AWS Transform for .NET detailed transformation reports.

SQL Server modernization

AWS Transform for SQL Server Modernization is an AI-powered service that automates the full-stack modernization of Microsoft SQL Server databases and their associated .NET applications to Amazon Aurora PostgreSQL. The service orchestrates the entire migration journey from schema conversion, data migration and modifying application code to match the new target PostgreSQL, making your teams more productive by automating complex and labor-intensive tasks.

Supported regions

AWS Transform for SQL Server is available in US East (N. Virginia) - us-east-1

Cross-Region Usage: For databases in unsupported regions, you can clone the database to a supported region for transformation, then deploy the results back to your target region.

Capabilities and key features

Database transformation

- **Schema conversion:** Automatically converts SQL Server schemas to Aurora PostgreSQL, including tables, views, indexes, constraints, and relationships
- **Stored procedure transformation:** Converts T-SQL stored procedures to PL/pgSQL with AI-enhanced accuracy
- **Data migration:** Migrates data with integrity validation using AWS Database Migration Service (DMS)
- **Database objects:** Supports triggers, functions, views, computed columns, and identity columns
- **Validation:** Automated data integrity verification and referential integrity checks

Application transformation

- **Entity Framework transformation:** Updates Entity Framework 6.3-6.5 and EF Core 1.0-8.0 configurations for PostgreSQL
- **ADO.NET transformation:** Converts ADO.NET data access code from SQL Server to PostgreSQL providers.
- **Connection string updates:** Automatically updates all database connection strings to the new target PostgreSQL database
- **Database provider changes:** Replaces SQL Server providers with Npgsql (PostgreSQL provider)
- **ORM configuration updates:** Modifies data type mappings, identity columns, and database-specific configurations

Orchestration & validation

- **Wave-based modernization:** Organizes large estates into logical migration phases
- **Dependency mapping:** Identifies relationships between applications and databases
- **Human-in-the-loop (HITL) checkpoints:** Provides review and approval gates at critical stages
- **Automated validation:** Tests schema compatibility, data integrity, and application functionality
- **CI/CD integration:** Integrates with existing development pipelines

Deployment

- **Amazon ECS and Amazon EC2 deployment:** Automated containerized deployment with auto-scaling support
- **Infrastructure-as-code generation:** Creates CloudFormation or AWS CDK templates
- **Automated deployment validation:** Verifies successful deployment with health checks
- **Rollback capabilities:** Supports rollback procedures if issues arise

Supported versions and project types

SQL Server versions

AWS Transform supports the following SQL Server versions:

SQL Server Version	Support Status
SQL Server 2022	Supported
SQL Server 2019	Supported
SQL Server 2017	Supported
SQL Server 2016	Supported
SQL Server 2014	Supported
SQL Server 2012	Supported
SQL Server 2008 R2	Supported

Note

All SQL Server editions are supported (Express, Standard, Enterprise). SQL Server can be hosted on AWS (Amazon RDS for SQL Server or SQL Server on Amazon EC2) or hosted outside of AWS.

.NET versions

.NET Version	Support Status
.NET 10	Supported
.NET 8	Supported
.NET 7	Supported
.NET 6 (Core)	Supported
.NET Framework 4.x and earlier	Not Supported

Important

Legacy .NET Framework 4.x and earlier versions are not supported. If your application uses .NET Framework, you must first upgrade to .NET Core 6+ using AWS Transform for .NET modernization before using SQL Server transformation capabilities.

Entity Framework versions

Framework	Supported Versions
Entity Framework 6	6.3, 6.4, 6.5
Entity Framework Core	1.0 through 8.0
ADO.NET	All versions (GA)

Source code repositories

AWS Transform supports the following repository platforms via AWS CodeConnections:

- GitHub and GitHub Enterprise
- GitLab.com
- Bitbucket Cloud
- Azure Repositories

Target database

AWS Transform targets Amazon Aurora PostgreSQL (PostgreSQL 15+ compatible) with support for the latest Aurora features and optimizations.

Technical requirements

Database requirements

- Microsoft SQL Server version 2008 R2 through 2022

- SQL Server hosted on AWS (Amazon RDS for SQL Server or SQL Server on Amazon EC2) or hosted outside of AWS
- For AWS-hosted databases, the database and AWS Transform must be in the same AWS Region.
- For databases hosted outside of AWS, network connectivity to the AWS Transform service is required.
- Database user with VIEW DEFINITION and VIEW DATABASE STATE permissions
- Database passwords using printable ASCII characters only (excluding '/', '@', '"', and spaces)
- VPC containing the source SQL Server must have subnets in at least 2 different Availability Zones (required for DMS replication subnet groups)

Application requirements

- .NET Core 6, 7, or 8 applications
- Entity Framework 6.3-6.5 or Entity Framework Core 1.0-8.0, or ADO.NET
- Database connections discoverable in source code
- Applications successfully build and run
- Source code in supported repository platforms

AWS account requirements

- AWS account with administrator access
- IAM Identity Center enabled
- Required service roles created (see setup instructions below)
- VPC with appropriate network configuration

Data processing and storage

Processing location

- Schema processing occurs in a DMS instance within your VPC
- Data migration is optional and can be excluded if required
- Transformation artifacts are stored in the AWS Transform service region

Stored artifacts

The following items are stored in the service region:

- Agent logs
- Assessment results
- SQL schema files
- DMS output artifacts

> **Important**
>
> **Important for Data Residency:** Even when data migration is opted out, metadata and processing artifacts are stored in the service region. This is important for organizations with strict data residency requirements.

Artifact management

- Customer option for encryption using your own KMS keys
- Defined TTL (time-to-live) period for all artifacts
- Artifacts can be downloaded for offline storage

SQL Server requirements

Database requirements

Externally hosted databases

Databases hosted outside of AWS are supported. To use an externally hosted database, you must ensure network connectivity between the database and the AWS Transform service.

SSIS workloads

- **Limitation:** SQL Server Integration Services (SSIS) transformation is not supported.
- **Workaround:** Use AWS Glue or other ETL services for SSIS migration.

Linked servers

- **Limitation:** External database connections via linked servers require human configuration.
- **Workaround:** Reconfigure linked servers after migration or use alternative approaches such as database links or application-level integration.

Complex T-SQL patterns

- **Limitation:** Some advanced T-SQL patterns may need human intervention.
- **Workaround:** Use Amazon Kiro CLI for complex SQL conversions. AWS Transform flags these for review during the transformation process.

Application requirements

Legacy .NET Framework

- **Limitation:** .NET Framework 4.x and earlier versions are not supported.
- **Workaround:** Use AWS Transform for .NET to upgrade to .NET Core 6+ first, then use SQL Server transformation.

Entity Framework versions

- **Limitation:** Only Entity Framework 6.3-6.5 and EF Core 1.0-8.0 are supported.
- **Workaround:** Upgrade to a supported Entity Framework version before transformation.

VB.NET applications

- **Limitation:** VB.NET is not supported.
- **Workaround:** Convert to C# or use AWS Transform custom to convert from VB.NET to C#.

Cross-database dependencies

- **Limitation:** Challenges when database schemas interact across multiple databases.
- **Workaround:** Review and refactor cross-database queries before migration. Consider consolidating databases or using PostgreSQL schemas.

- **Impact:** May require human intervention for complex cross-database scenarios.

Repository-database coupling

- **Limitation:** Challenges when a single repository serves multiple databases.
- **Workaround:** Consider repository restructuring or phased migration approach.
- **Impact:** May require additional planning for wave-based migrations.

Infrastructure requirements

Single account/region per job

- **Limitation:** Each transformation job targets one AWS account and region.
- **Workaround:** Create multiple transformation jobs for multi-account or multi-region deployments.

Deployment targets

- **Limitation:** Amazon ECS and Amazon EC2 deployments are supported.

Repository requirements

Private NuGet packages

- **Limitation:** Private NuGet packages require additional configuration.
- **Workaround:** Configure private NuGet feeds in transformation settings before starting the job.

SQL Server setup

Perform these steps in your SQL Server environment to enable AWS Transform modernization.

Database setup and configuration

Step 1: Create database user with required permissions

Create a dedicated database user for AWS Transform with the necessary permissions. If you already have a DMS Schema Conversion user, you can reuse it.

Connect to your SQL Server instance and run the following commands:

```
-- Create the login in master database
USE master;
CREATE LOGIN [atx_user] WITH PASSWORD = 'YourStrongPassword123!';

-- Switch to your application database
USE [YourDatabaseName];
CREATE USER [atx_user] FOR LOGIN [atx_user];

-- Grant required permissions
GRANT VIEW DEFINITION TO [atx_user];
GRANT VIEW DATABASE STATE TO [atx_user];
ALTER ROLE [db_datareader] ADD MEMBER [atx_user];

-- Grant master database permissions
USE master;
GRANT VIEW SERVER STATE TO [atx_user];
GRANT VIEW ANY DEFINITION TO [atx_user];
```

> **Note**
>
> Repeat the database-specific commands (USE, CREATE USER, GRANT) for each database you want to modernize.

The db_datareader role is only necessary for data migration, not for schema conversion alone.

- The db_datareader role grants read access to all tables in the database
- This role is required ONLY when performing data migration.
- For schema conversion only (without data migration), the db_datareader role is NOT required
- The other permissions (VIEW DEFINITION, VIEW DATABASE STATE, etc.) are sufficient for schema conversion

Step 2: Store credentials in AWS Secrets Manager

Store your database credentials securely in AWS Secrets Manager. Skip this step if you already have a secret created for DMS.

1. Navigate to AWS Secrets Manager in the console
2. Choose **Store a new secret**
3. Configure the secret:
   - **Secret type:** Credentials for other database
   - **Database**: Microsoft SQL Server
   - **Username**: atx_user (or your chosen username)
   - **Password**: The password you created
   - **Server name**: Your SQL Server endpoint
   - **Database name**: Your database name
   - **Port**: 1433 (or your custom port)
4. Choose **Next**
5. Enter secret name: atx-db-modernization-sqlserver
6. Add required tags (these tags are mandatory):
   - Key: Project, Value: atx-db-modernization
   - Key: Owner, Value: database-connector
7. Choose **Next** through remaining screens
8. Choose **Store**
9. Note the Secret ARN for use in the next step

> **⚠ Important**
>
> Database passwords must use printable ASCII characters only, excluding '/', '@', '"', and spaces. Secrets scheduled for deletion can cause transformation failures.

Step 3: Create required DMS roles

AWS Transform requires specific IAM roles for DMS operations. Deploy these roles using the CloudFormation template below.

Note

If your AWS account already has existing DMS-related roles, modify this template to reuse those resources rather than creating duplicates.

Create a file named dms-roles.yaml with the following content:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: 'DMS Service Roles for AWS Transform SQL Server Modernization'

Resources:
  DMSCloudWatchLogsRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: dms-cloudwatch-logs-role
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - dms.amazonaws.com
                - schema-conversion.dms.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonDMSCloudWatchLogsRole

  DMSS3AccessRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: dms-s3-access-role
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - dms.amazonaws.com
                - schema-conversion.dms.amazonaws.com
            Action: sts:AssumeRole
      Policies:
        - PolicyName: S3TaggedAccess
```

```
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action:
                  - s3:GetBucketLocation
                  - s3:GetBucketVersioning
                  - s3:PutObject
                  - s3:PutBucketVersioning
                  - s3:GetObject
                  - s3:GetObjectVersion
                  - s3:ListBucket
                  - s3:DeleteObject
                Resource: arn:aws:s3:::atx-db-modernization-*
                Condition:
                  StringEquals:
                    aws:ResourceAccount: !Ref AWS::AccountId

  DMSSecretsManagerRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: dms-secrets-manager-role
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - dms.amazonaws.com
                - schema-conversion.dms.amazonaws.com
            Action: sts:AssumeRole
      Policies:
        - PolicyName: SecretsManagerTaggedAccess
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: Allow
                Action:
                  - secretsmanager:GetSecretValue
                  - secretsmanager:DescribeSecret
                Resource: '*'
                Condition:
                  StringEquals:
                    secretsmanager:ResourceTag/Project: atx-db-modernization
```

```
                    secretsmanager:ResourceTag/Owner: database-connector

  DMSVPCRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: dms-vpc-role
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - dms.amazonaws.com
                - schema-conversion.dms.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AmazonDMSVPCManagementRole

  DMSServerlessRole:
    Type: AWS::IAM::ServiceLinkedRole
    Properties:
      AWSServiceName: dms.amazonaws.com
      Description: 'Service Linked Role for AWS DMS Serverless'

Outputs:
  DMSCloudWatchLogsRoleArn:
    Description: ARN of the DMS CloudWatch Logs Role
    Value: !GetAtt DMSCloudWatchLogsRole.Arn
  DMSS3AccessRoleArn:
    Description: ARN of the DMS S3 Access Role
    Value: !GetAtt DMSS3AccessRole.Arn
  DMSSecretsManagerRoleArn:
    Description: ARN of the DMS Secrets Manager Role
    Value: !GetAtt DMSSecretsManagerRole.Arn
  DMSVPCRoleArn:
    Description: ARN of the DMS VPC Role
    Value: !GetAtt DMSVPCRole.Arn
    Export:
      Name: !Sub ${AWS::StackName}-VPCRole

  DMSServerlessRoleArn:
    Description: ARN of the DMS Serverless Role
    Value: !Sub 'arn:aws:iam::${AWS::AccountId}:role/aws-service-role/
dms.amazonaws.com/AWSServiceRoleForDMSServerless'
```

```
    Export:
      Name: !Sub ${AWS::StackName}-ServerlessRole
```

Deploy the CloudFormation stack using AWS CLI:

```
aws cloudformation create-stack \
  --stack-name dms-roles \
  --template-body file://dms-roles.yaml \
  --capabilities CAPABILITY_NAMED_IAM \
  --region us-east-1
```

Or deploy using the AWS Console:

1. Navigate to CloudFormation in the AWS Console
2. Choose **Create stack**
3. Select **Upload a template file**
4. Upload the dms-roles.yaml file
5. Enter stack name: dms-roles
6. Acknowledge IAM capabilities
7. Choose **Create stack**

Step 4: Configure network security

Ensure proper network connectivity between AWS Transform, your SQL Server database, and other AWS services.

Security group configuration (recommended approach)

Recommended Approach: Use security group-based access control rather than IP-based rules. This provides better security, easier management, and works seamlessly with AWS Transform's architecture.

Why Security Group-Based Access Control?

- DMS Schema Conversion creates Elastic Network Interfaces (ENIs) within your VPC
- Your databases do not need to be publicly accessible
- AWS Transform does not expose private IP addresses, making IP-based rules complex
- Security group references provide dynamic, automatic updates as resources scale

Configure your SQL Server security group

When you configure the DMS Schema Conversion Instance Profile in AWS Transform, you specify a Security Group for the DMS SC instance. Your database security group should allow inbound traffic from this DMS SC security group.

Step-by-step configuration:

1. Identify the DMS Schema Conversion Security Group:
   - This is specified when creating the Instance Profile in AWS Transform
   - Note the Security Group ID (e.g., sg-0123456789abcdef0)
2. Update your SQL Server Security Group inbound rules:
   - **Type**: Custom TCP
   - **Port**: 1433 (or your custom SQL Server port)
   - **Source**: The DMS Schema Conversion Security Group ID
   - **Description**: "Allow DMS Schema Conversion access"
3. For Aurora PostgreSQL target (after creation):
   - **Type**: PostgreSQL
   - **Port**: 5432 (or your custom PostgreSQL port)
   - **Source**: The DMS Schema Conversion Security Group ID
   - **Description**: "Allow DMS Schema Conversion access"

> **Important**
>
> **Important for Least Privileged Security Models:** If your organization uses a "least privileged" security model that blocks all traffic by default, you must explicitly allow inbound traffic from the DMS Schema Conversion Security Group to your database port. Do not open port 1433 to all sources or IP ranges.

Required AWS service connectivity

Ensure your VPC can communicate with:

- AWS Transform service endpoints
- AWS DMS endpoints

- Aurora PostgreSQL endpoints
- S3 endpoints for artifact storage
- AWS Secrets Manager endpoints
- AWS CodeConnections endpoints

VPC Endpoints: For private networks, configure VPC endpoints for required AWS services to avoid internet gateway dependencies.

Requirements for externally hosted databases

If your SQL Server database is hosted outside of AWS, ensure the following prerequisites are met and then complete the setup steps before you begin the modernization.

Prerequisites

- An AWS account with a VPC
- Network connectivity between the VPC and the external database. For information about configuring network connectivity, see Configuring network connectivity in the AWS DMS User Guide.

Setup steps

1. Create a secret in AWS Secrets Manager with the connection details for the external database. For more information, see the section called “Step 2: Store credentials in AWS Secrets Manager”.
2. When prompted, provide the VPC ID and security group ID for connecting to the external database. AWS Transform prompts you for this information because the database hostname in the secret cannot be resolved within the AWS account.

AWS Transform setup

Begin your modernization journey by accessing the AWS Transform console and initiating a new modernization project. This step establishes your workspace and sets up the foundational configuration for your full-stack transformation.

Actions to Complete:

- Navigate to AWS Transform service in the AWS Management Console

- Select **Microsoft Modernization** from the service options
- Choose **Create new modernization project**
- Provide project name and description
- Select your target AWS region (must match your SQL Server location)
- Review and confirm project settings

> **Note**
>
> **Best Practice:** Use descriptive project names that include the application name and target environment (e.g., 'CustomerPortal-Production-Modernization') to easily identify projects in multi-wave scenarios.

Create SQL Server modernization job

Connect database and source code repo

Configure database connector

Establish secure connectivity to your SQL Server databases by configuring database connectors. The connector performs environment analysis, dependency discovery, and enables AWS Transform to access your database schemas and metadata for assessment and conversion.

Actions to Complete:

- Select **Configure Database Connector** from the setup wizard
- Choose connection method: **New Connection** or **Existing Connection**
- Provide SQL Server connection details (endpoint, port, database names)
- Configure authentication and test connectivity
- Review discovered databases and confirm selection

> **Note**
>
> **Network Connectivity:** Ensure your SQL Server security groups and NACLs allow inbound connections from AWS Transform service endpoints.

Connect source code repository

Enable AWS Transform to access your .NET application source code through AWS CodeConnections. This integration allows the service to analyze your application code, identify database dependencies, and perform automated code transformations for PostgreSQL compatibility.

Actions to Complete:

- Navigate to **Connect Source Code Repository** section
- Select your source code platform (GitHub, GitLab, Bitbucket, Azure Repos)
- Configure AWS CodeConnections and authorize access
- Select repositories containing .NET applications
- Specify branch for analysis (typically main/master or development)
- Validate repository access and code structure

> **Note**
>
> **Repository Discovery:** AWS Transform automatically scans your repositories to identify .NET projects, Entity Framework configurations, database connection strings, and SQL dependencies.

> **Note**
>
> **Security Note:** AWS Transform only requires read access to your repositories and creates new feature branches for transformed code. Your main branch remains untouched.

Human in the loop

AWS Transform uses human-in-the-loop (HITL) mechanisms to ensure quality and allow you to review and approve critical transformation decisions. The following checkpoints require your attention:

Wave plan review and approval

What you review: The proposed migration waves, including which databases and applications are grouped together and the sequence of waves.

You can:

- Approve the wave plan
- Customize waves by moving databases between waves
- Modify wave sequence
- Split or merge waves

After approval, AWS Transform proceeds with schema conversion.

Schema conversion review

What you review: Converted database objects including tables, stored procedures, functions, and triggers. Action items highlight objects that require attention.

You can:

- Accept converted code
- Modify converted code
- Flag for human review after transformation
- View side-by-side comparison of original and converted code

What happens after approval: AWS Transform applies the schema to Aurora PostgreSQL and proceeds with data migration (if configured).

Application code review

What you review: All application code changes including Entity Framework configurations, connection strings, data access code, and stored procedure calls.

You can:

- Accept changes for each file
- Modify transformed code
- Reject changes (not recommended)
- Add comments for your team
- Download transformed code for local review

What happens after approval: AWS Transform commits the changes to a new branch in your repository and proceeds with validation.

Validation results review

What you review: Automated validation results including schema compatibility, data integrity checks, and application build status.

You can:

- Review passed tests
- Investigate failed tests
- Address warnings
- Proceed to deployment or return to fix issues

What happens after approval: AWS Transform prepares for deployment to your target environment.

Deployment approval

What you review: Deployment configuration including infrastructure-as-code templates, ECS service configuration, and deployment settings.

You can:

- Review and customize deployment settings
- Approve deployment to proceed
- Delay deployment for additional testing
- Download infrastructure code for review

After approval, AWS Transform deploys your modernized application and database to the target environment.

Assessment and wave planning

Set up landing zone

Configure the target infrastructure environment where your modernized applications will be deployed. The landing zone setup includes Aurora PostgreSQL provisioning, networking configuration, and reference deployment configurations.

Actions to Complete:

- Select **Set up Landing Zone** from the configuration menu
- Configure Aurora PostgreSQL target (version, instance class, Multi-AZ)
- Configure network and security settings (VPC, subnets, security groups)

> **Note**
>
> **Cost Consideration:** Aurora PostgreSQL instances incur ongoing costs. Consider using Aurora Serverless v2 for development/testing environments.

Assessment

Conduct comprehensive analysis of your database schemas, application code, and dependencies. The assessment phase provides transformation complexity ratings, identifies potential challenges, and generates detailed reports.

Actions to Complete:

- Select Repository Branch for code analysis
- Select Repositories and Databases for transformation
- Review Assessment Results and complexity ratings
- Generate Assessment Report with effort projections
- Export report for stakeholder review and approval

> **Note**
>
> **Assessment Insights:** The assessment provides detailed analysis of schema objects, stored procedures, application dependencies, and estimates the level of human intervention required.

Wave planning

Organize your modernization into manageable waves based on transformation complexity, business priorities, and application dependencies. AWS Transform generates intelligent wave recommendations that you can customize.

Actions to Complete:

- Review AI-Generated Wave Plan and recommendations
- Customize Wave Configuration based on business priorities
- Validate Dependencies between applications and databases
- Finalize Wave Plan using conversational interface
- Set wave execution priorities and assign responsibilities

> **Note**
>
> **AI-Powered Recommendations:** You can interact with the AI through natural language to adjust recommendations: 'Move CustomerDB to Wave 1' or 'Combine these two applications into the same wave.'

> **Note**
>
> **Best Practice:** Start with a pilot wave containing 1-2 low-complexity databases to establish processes and build team confidence.

Data migration for each wave

Schema conversion for each wave

Transform SQL Server database schemas to PostgreSQL-compatible equivalents using AI-enhanced conversion capabilities. This process converts tables, stored procedures, functions, triggers, and other database objects.

Actions to Complete:

- Provide Schema Conversion Targets and configure preferences
- Execute Automated Conversion using DMS Schema Conversion service
- Review Conversion Results and identify human intervention items
- Handle Manual Interventions through HITL prompts
- Validate human corrections and re-run conversion if needed

> **Note**
>
> **Common Manual Interventions:** Linked servers, user-defined types, advanced T-SQL patterns, and vendor-specific SQL features typically require human review and correction.

Data migration for each wave

Transfer data from SQL Server to Aurora PostgreSQL using AWS Database Migration Service (AWS DMS) integration. Choose between production data migration or synthetic data generation for testing purposes.

Actions to Complete:

- Choose **Production Data Migration** or **Synthetic Data Generation**
- Configure DMS replication instance for data transfer
- Monitor migration progress and handle data inconsistencies
- Validate data integrity and referential constraints
- Confirm successful completion before proceeding

> **Note**
>
> **Automated DMS Integration:** AWS Transform abstracts the complexity of DMS configuration, automatically handling endpoint creation, task configuration, and data type mappings.

> **Note**
>
> **Data Validation:** The service performs empirical testing by running identical queries against both source and target databases to ensure data consistency.

Code migration for each wave

Transform .NET application code to work with Aurora PostgreSQL, including Entity Framework configuration updates, connection string modifications, SQL query adaptations, and framework-specific adjustments.

Actions to Complete:

- Execute Automated Code Analysis for SQL Server dependencies
- Perform Framework Transformation (Entity Framework provider updates)
- Adapt SQL Query syntax to PostgreSQL-compatible SQL
- Configure Target Branch Management for transformed code
- Handle Manual Interventions for complex patterns

SQL Server modernization workflow

This section provides a step-by-step walkthrough of the complete SQL Server modernization process using AWS Transform.

Step 1: Create SQL Server modernization job

Begin your modernization journey by creating a new transformation job in the AWS Transform console.

1. Sign in to the AWS Transform console
2. Choose **Create modernization job**
3. Select **Windows modernization** job and then select **SQL Server modernization**
4. Enter job details:
   - **Job name**: Descriptive name for your project
   - **Description**: Optional description
   - **Target region**: AWS region for deployment
5. Choose **Create job**

> **Important**
>
> Do not include personally identifiable information (PII) in your job name.

Step 2: Connect to SQL Server database

Connect AWS Transform to your SQL Server database to enable schema analysis and conversion.

Create a database connector

1. In your SQL Server modernization job, navigate to Connect to resources
2. Choose **Connect to SQL Server database**
3. Choose **Create new connector**
4. Enter the connector information:
   - C**onnector name**: Descriptive name
   - **AWS account ID**: Account where SQL Server is hosted
5. Once confirmed, you will receive a link for approval. Copy the approval link to get approval from your AWS admin for the account. Once they have approved, you can proceed to the next step.
6. After your administrator has approved the connector request, click Submit to proceed to the source code connection setup.

Step 3: Connect source code repository

AWS Transform needs access to your .NET application source code to analyze and transform the code that interacts with your SQL Server database.

Set up AWS CodeConnections

1. In your SQL Server modernization job, navigate to **Connect to resources**
2. Choose **Connect source code repository**
3. If you don't have an existing connection, choose **Create connection**
4. Select your repository provider:
   - GitHub / GitHub Enterprise
   - GitLab.com
   - Bitbucket Cloud
   - Azure Repositories
5. Follow the authorization flow for your provider
6. After authorization, choose **Connect**

Select your repository and branch

1. Select your repository from the list
2. Choose the branch you want to transform (typically main, master, or develop)
3. (Optional) Specify a subdirectory if your .NET application is not in the repository root
4. Choose **Continue**

> **Note**
>
> AWS Transform will create a new branch for the transformed code. You can review and merge the changes through your normal code review process.

Repository access approval

For GitHub and some other platforms, the repository administrator must approve the connection request:

1. AWS Transform displays a verification link
2. Share this link with your repository administrator
3. The administrator reviews and approves the request in their repository settings

4. Once approved, the connection status changes to *Approved*

> **Important**
>
> The approval process can take time depending on your organization's policies. Plan accordingly.

Step 4: Create deployment connector (optional)

If you want to deploy the transformed applications into your AWS account, you have the option to select a deployment connector.

Set up deployment connector

1. Select **Yes** if you want to deploy your applications. Selecting **No** will skip this step.
2. Add your AWS account where you want to deploy the transformed applications.
3. Add a name that helps you remember the connector easily
4. Submit the connector request for approval.

Deployment connector approval

Your AWS account administrator must approve the connection request for deployment connector.

1. AWS Transform displays a verification link
2. Share this link with your AWS account administrator
3. The administrator reviews and approves the request in their repository settings
4. Once approved, the connection status changes to *Approved*

> **Important**
>
> The approval process can take time depending on your organization's policies. Plan accordingly.

Step 5: Confirm your resources

After connecting to your database and repository, AWS Transform verifies that all required resources are accessible and ready for transformation.

What AWS Transform verifies

- **Database connectivity:** Connection is active, user has required permissions, databases are accessible, version is supported
- **Repository access:** Repository is accessible, branch exists, .NET project files detected, database connections discoverable
- **Environment readiness:** VPC configuration supports DMS, required AWS service roles exist, network connectivity established, region compatibility confirmed

Review the pre-flight checklist

1. Navigate to **Confirm your resources** in the job plan
2. Review the checklist items:
   - # Database connection verified
   - # Repository access confirmed
   - # .NET version supported
   - # Entity Framework or ADO.NET detected
   - # Network configuration valid
   - # Required permissions granted
3. If all items show as complete, choose Continue
4. If any items show warnings or errors, address them before proceeding

Step 6: Discovery and assessment

AWS Transform analyzes your SQL Server database and .NET application to understand the scope and complexity of the modernization.

What gets discovered

- **Database objects:** Tables, views, indexes, stored procedures, functions, triggers, constraints, data types, computed columns, identity columns, foreign key relationships

- **Application code:** .NET project structure, Entity Framework models and configurations, ADO.NET data access code, database connection strings, stored procedure calls, SQL queries in code
- **Dependencies:** Which applications use which databases, cross-database dependencies, shared stored procedures, common data access patterns

Discovery process

- AWS Transform begins discovery automatically after resource confirmation
- Discovery typically takes 5-15 minutes depending on database size and application complexity
- Monitor progress in the worklog
- AWS Transform displays real-time updates as objects are discovered

Review discovery results

After discovery completes, navigate to **Discovery and assessment** to review:

Database Analysis:

- **Object count**: Number of tables, views, stored procedures, functions, triggers
- **Complexity score**: Assessment of transformation complexity (Low, Medium, High)
- **Action items**: Objects that may require human attention
- **Supported features**: Database features that will convert automatically
- **Unsupported features**: Features that require workarounds

Application Analysis:

- **Project type**: ASP.NET Core, Console App, Class Library, etc.
- **.NET version**: Detected .NET Core version
- **Data access framework**: Entity Framework version or ADO.NET
- **Database connections**: Number of connection strings found
- **Code complexity**: Assessment of transformation complexity

Dependency Map:

- Visual representation of application-to-database relationships

- Cross-database dependencies
- Shared components

Understanding complexity assessment

AWS Transform classifies your modernization into three categories:

Complexity	Characteristics	Expected Outcome
Low (Class A)	Standard SQL patterns (ANSI SQL), simple stored procedures, basic data types, Entity Framework with standard configurations	Minimal human intervention expected, high automation success rate
Medium (Class B)	Advanced T-SQL patterns, complex stored procedures with business logic, user-defined functions, computed columns	Some human intervention required, expert review recommended
High (Class C)	CLR assemblies, linked servers, Service Broker, complex full-text search	Significant human refactoring required, consider phased approach

Assessment report

AWS Transform generates a detailed assessment report that includes:

- Executive summary with high-level overview
- Complete database inventory
- Application inventory
- Transformation readiness percentage
- Effort estimation
- Risk assessment and mitigation strategies
- Recommended approach

You can download the assessment report for offline review and sharing with stakeholders.

Step 7: Generate and review wave plan

For large estates with multiple databases and applications, AWS Transform generates a wave plan that sequences the modernization in logical groups.

What is a wave plan?

A wave plan organizes your modernization into phases (waves) based on:

- Dependencies between databases and applications
- Business priorities
- Risk tolerance
- Resource availability
- Technical complexity

Each wave contains a group of databases and applications that can be modernized together without breaking dependencies.

Review the wave plan

1. Navigate to **Wave planning** in the job plan
2. Review the proposed waves
3. For each wave, review:
   - Databases included
   - Applications included
   - Dependencies on other waves
   - Estimated transformation time
   - Complexity level
   - Deployable applications

Customize the wave plan

You can customize the wave plan to match your business needs in 2 ways:

Using JSON:

1. Choose **Download all waves** to get a JSON file with all the waves
2. Modify waves in the JSON by:
   - Moving databases between waves
   - Splitting waves into smaller groups
   - Merging waves together
   - Changing wave sequence
   - Adding or removing databases from scope
3. Upload the JSON file back to the console by choosing **Upload wave plan**
4. AWS Transform validates your changes and warns if dependencies are violated
5. Choose **Confirm waves** to update the wave plan

Using Chat:

You can modify the wave plans by chatting with the agent and asking it to move the repositories and databases to specific waves. This approach works well if you need to make minor edits to the waves.

> ⚠ **Important**
>
> Ensure that dependencies are respected when customizing waves. Transforming a dependent application before its database can cause issues.

Single database modernization

If you're modernizing a single database and application, AWS Transform creates a simple plan with one wave. You can proceed directly to transformation without wave planning.

Approve the wave plan

1. After reviewing and customizing (if needed), choose **Approve wave plan**
2. AWS Transform locks the wave plan and proceeds to transformation
3. You can still modify the plan later by choosing **Edit wave plan**

Step 8: Schema conversion

AWS Transform converts your SQL Server database schema to Aurora PostgreSQL, including tables, views, stored procedures, functions, and triggers.

How schema conversion works

AWS Transform uses AWS DMS Schema Conversion enhanced with generative AI to:

- Analyze SQL Server schemas and relationships
- Map data types from SQL Server to PostgreSQL equivalents
- Transform T-SQL to PL/pgSQL
- Handle identity columns, computed columns, and constraints
- Validate conversion and referential integrity
- Generate action items for objects requiring human review

Supported conversions

Automatically converted:

- Tables, views, and indexes
- Primary keys and foreign keys
- Check constraints and default values
- Most common data types
- Simple stored procedures
- Basic functions and triggers
- Identity columns (converted to SERIAL or GENERATED)
- Most computed columns

May require human review:

- Complex stored procedures with advanced T-SQL
- SQL Server-specific functions (GETUTCDATE, SUSER_SNAME, etc.)
- Computed columns with complex expressions
- Full-text search indexes

- XML data type operations
- HIERARCHYID data type (requires ltree extension)

Not automatically converted:

- CLR assemblies
- Linked servers
- Service Broker
- SQL Server Agent jobs

Start schema conversion

1. Navigate to Schema conversion in the job plan
2. Review the conversion settings:
   - Target PostgreSQL version
   - Extension options (ltree, PostGIS, etc.)
   - Naming conventions
3. Choose **Start conversion**
4. Monitor progress in the worklog
5. Conversion typically takes 10-30 minutes depending on the number of database objects

Review conversion results

After conversion completes, navigate to Review schema conversion:

Conversion Summary:

- **Objects converted**: Count of successfully converted objects
- **Action items**: Objects requiring human attention
- **Warnings**: Potential issues to review
- **Errors**: Objects that could not be converted

Review by Object Type:

- **Tables**: Data type mappings, constraints, indexes

- **Stored procedures**: T-SQL to PL/pgSQL conversion
- **Functions**: Function signature and logic changes
- **Triggers**: Trigger syntax and timing changes

Review action items

1. Choose **View action items**
2. For each action item, review:
   - **Object name**: The database object
   - **Issue type**: What requires attention
   - **Severity**: Critical, Warning, or Info
   - **Recommendation**: Suggested resolution
   - **Original code**: SQL Server version
   - **Converted code**: PostgreSQL version
3. For each action item, you can:
   - **Accept**: Use the converted code
   - **Modify**: Edit the converted code
   - **Flag for later**: Mark for human review after transformation

Example: Stored procedure conversion

SQL Server T-SQL:

```
CREATE PROCEDURE GetProductsByCategory
    @CategoryId INT,
    @PageSize INT = 10
AS
BEGIN
    SET NOCOUNT ON;

    SELECT TOP (@PageSize)
        ProductId,
        Name,
        Price,
        DATEDIFF(DAY, CreatedDate, GETUTCDATE()) AS DaysOld
    FROM Products
```

```
    WHERE CategoryId = @CategoryId
    ORDER BY Name
END
```

Converted PostgreSQL PL/pgSQL:

```
CREATE OR REPLACE FUNCTION get_products_by_category(
    p_category_id INTEGER,
    p_page_size INTEGER DEFAULT 10
)
RETURNS TABLE (
    product_id INTEGER,
    name VARCHAR(255),
    price NUMERIC(18,2),
    days_old INTEGER
) AS $$
BEGIN
    RETURN QUERY
    SELECT
        p.product_id,
        p.name,
        p.price,
        EXTRACT(DAY FROM (NOW() - p.created_date))::INTEGER AS days_old
    FROM products p
    WHERE p.category_id = p_category_id
    ORDER BY p.name
    LIMIT p_page_size;
END;
$$ LANGUAGE plpgsql;
```

Changes made:

- Procedure converted to function returning TABLE
- Parameter names prefixed with p_
- TOP converted to LIMIT
- DATEDIFF converted to EXTRACT
- GETUTCDATE() converted to NOW()
- Column names converted to lowercase (PostgreSQL convention)

Approve schema conversion

1. After reviewing all action items and making necessary modifications
2. Choose **Approve schema conversion**
3. AWS Transform prepares the converted schema for deployment to Aurora PostgreSQL

> **Note**
>
> You can download the converted schema as SQL scripts for offline review or version control.

Step 9: Data migration (optional)

AWS Transform provides options for migrating data from SQL Server to Aurora PostgreSQL. Data migration is optional and can be skipped if you only need schema and code transformation.

Data migration options

Option 1: Production Data Migration

Migrate your actual production data using AWS DMS:

- Full initial load of all data
- Continuous replication during testing (CDC)
- Minimal downtime cutover
- Data validation and integrity checks

Option 2: Skip Data Migration

Transform schema and code only:

- Useful for development/testing environments
- When data will be migrated separately
- For proof-of-concept projects

Configure data migration

1. Navigate to **Data migration** in the job plan
2. Choose your migration option:
   - **Migrate production data**
   - **Skip data migration**
3. If migrating production data, configure:
   - **Migration type**: Full load, or Full load + CDC
   - **Validation**: Enable data validation
   - **Performance**: DMS instance size
   - Choose **>Start migration**

Production data migration process

If you choose to migrate production data:

1. **Initial sync**: AWS DMS performs full load of all tables
2. **Continuous replication**: (If CDC enabled) Keeps data synchronized
3. **Validation**: Verifies row counts and data integrity
4. **Cutover preparation**: Prepares for final synchronization

Migration Timeline:

- Small databases (< 10 GB): 30 minutes - 2 hours
- Medium databases (10-100 GB): 2-8 hours
- Large databases (> 100 GB): 8+ hours

Data validation

AWS Transform validates migrated data with the following checks:

- Row count comparison (source vs target)
- Primary key integrity
- Foreign key relationships

- Data type compatibility
- Computed column results
- Null value handling

Step 10: Application code transformation

AWS Transform transforms your .NET application code to work with Aurora PostgreSQL instead of SQL Server. It asks for a target branch name in your repositories to commit the transformed source code. Once you enter the branch name, AWS Transform will create a new branch and initiate the transformation to match the PostgreSQL database.

What gets transformed

Entity Framework Changes:

- **Database provider**: UseSqlServer() → UseNpgsql()
- **Connection strings**: SQL Server format → PostgreSQL format
- **Data type mappings**: SQL Server types → PostgreSQL types
- **DbContext configurations**: SQL Server-specific → PostgreSQL-specific
- **Migration files**: Updated for PostgreSQL compatibility

ADO.NET Changes:

- **Connection classes**: SqlConnection → NpgsqlConnection
- **Command classes**: SqlCommand → NpgsqlCommand
- **Data reader**: SqlDataReader → NpgsqlDataReader
- **Parameters**: SqlParameter → NpgsqlParameter
- S**QL syntax**: T-SQL → PostgreSQL SQL

Configuration Changes:

- Connection strings in appsettings.json
- Database provider NuGet packages
- Dependency injection configurations
- Startup/Program.cs configurations

Start code transformation

1. Navigate to Application transformation in the job plan
2. Review the transformation settings:
   - **Target .NET version** (if upgrading)
   - **PostgreSQL provider version**
   - **Code style preferences**
3. Choose **Start transformation**
4. Monitor progress in the worklog
5. Transformation typically takes 15-45 minutes depending on codebase size

Step 11: Review transformation results

Before proceeding to deployment, review the complete transformation results to ensure everything is ready for testing.

You can download the transformed code from the repository branch for:

- Local testing and validation
- Code review in your IDE
- Integration with your CI/CD pipeline
- Version control commit

You can also download the transformation summary to review the natural language changes that are made by AWS Transform as part of the transformation.

Transformation summary

1. Navigate to **Transformation summary** in the job plan
2. Review the overall results:
   - **Schema conversion**: Objects converted, action items, warnings
   - **Data migration**: Tables migrated, rows transferred, validation status
   - **Code transformation**: Files changed, lines modified, issues resolved
   - **Readiness score**: Overall readiness for deployment

Generate transformation report

AWS Transform generates a comprehensive transformation report:

1. Choose **Generate report**
2. Select report type:
   - **Executive summary**: High-level overview for stakeholders
   - **Technical details**: Complete transformation documentation
   - **Action items**: List of human tasks required
3. Choose **Download report**

The report includes:

- Transformation scope and objectives
- Objects and code transformed
- Issues encountered and resolutions
- Validation results
- Deployment readiness assessment
- Recommendations for testing

Step 12: Validation and testing

Before deploying to production, validate that the transformed application works correctly with Aurora PostgreSQL.

Validation types

Automated Validation: AWS Transform performs automated checks:

- Schema validation against source database
- Data integrity verification
- Query equivalence testing
- Connection string validation
- Configuration validation

Human Validation: You should perform additional testing:

- Functional testing of application features
- Integration testing with other systems
- Performance testing and benchmarking
- User acceptance testing
- Security testing

Run automated validation

1. Navigate to **Validation** in the job plan
2. Choose **Run validation**
3. AWS Transform executes validation tests:
   - Database connectivity
   - Schema compatibility
   - Data integrity
   - Application build
   - Basic functionality
4. Review validation results:
   - Passed: Tests that succeeded
   - Failed: Tests that need attention
   - Warnings: Potential issues to review

Testing checklist

Database Functionality:

- All tables accessible
- Stored procedures execute correctly
- Functions return expected results
- Triggers fire appropriately
- Constraints enforced properly
- Indexes improve query performance

Application Functionality:

- Application starts successfully
- Database connections established
- CRUD operations work correctly
- Stored procedure calls succeed
- Transactions commit/rollback properly
- Error handling works as expected

Data Integrity:

- Row counts match source
- Primary keys unique
- Foreign keys valid
- Computed columns correct
- Null handling appropriate
- Data types compatible

Performance:

- Query response times acceptable
- Connection pooling configured
- Indexes optimized
- No N+1 query issues
- Batch operations efficient
- Resource utilization reasonable

Step 13: Deployment

After successful validation deploy your modernized application and database to production.

Deployment options

- Amazon ECS and Amazon EC2 Linux

Pre-deployment checklist

Before deploying to production:

- All validation tests passed
- Performance testing completed
- Security review completed
- Backup and rollback plan documented
- Monitoring and alerting configured
- Team trained on new environment
- Stakeholders informed of deployment
- Maintenance window scheduled

Deploy to Amazon ECS

1. Navigate to **Deployment** in the job plan
2. Choose **Deploy to ECS**
3. Configure deployment settings:
   - **Cluster**: Select or create ECS cluster
   - **Service**: Configure ECS service
   - **Task definition**: Review generated task definition
   - **Load balancer**: Configure ALB/NLB
   - **Auto-scaling**: Set scaling policies
4. Review infrastructure-as-code (CloudFormation template or AWS CDK code)
5. Choose **Deploy**

Monitor deployment

AWS Transform deploys your application:

1. Creates Aurora PostgreSQL cluster
2. Applies database schema
3. Loads data (if applicable)
4. Deploys application containers

5. Configures load balancer
6. Sets up auto-scaling

Monitor deployment progress and verify:

- Infrastructure provisioning
- Database initialization
- Application deployment
- Health checks passing
- Application accessible
- Database connections working
- Logs showing normal operation

Post-deployment validation

After deployment:

Smoke Testing:

- Verify critical functionality
- Test key user workflows
- Check integration points
- Monitor error rates

Performance Monitoring:

- Track response times
- Monitor database queries
- Check resource utilization
- Review application logs

User Validation:

- Conduct user acceptance testing

- Gather feedback
- Address any issues
- Document lessons learned

Rollback procedures

If issues arise after deployment:

Immediate Rollback:

- Revert to previous application version
- Switch back to SQL Server (if still available)
- Restore from backup if needed

Partial Rollback:

- Roll back specific components
- Keep database changes
- Revert application code only

Forward Fix:

- Apply hotfix to Aurora PostgreSQL version
- Deploy updated application code
- Monitor for resolution

> **Important**
>
> Keep your SQL Server database available for a period after cutover to enable rollback if needed.

Post-deployment optimization

After successful deployment:

Performance Tuning:

- Optimize slow queries
- Adjust connection pool settings
- Fine-tune Aurora PostgreSQL parameters
- Review and optimize indexes

Cost Optimization:

- Right-size Aurora instance
- Configure auto-scaling appropriately
- Review storage settings
- Optimize backup retention

Monitoring Setup:

- Configure CloudWatch dashboards
- Set up alerting
- Enable Enhanced Monitoring
- Configure Performance Insights

Documentation:

- Update runbooks
- Document architecture changes
- Train operations team
- Create troubleshooting guides

Test and deploy after transformation

Deploy your transformed applications to the landing zone and validate the connectivity between your applications and the transformed database, and optionally set up CI/CD pipelines. Sample CI/CD pipeline configurations are generated as reference to help you implement your deployment automation. This final phase validates the complete modernization.

Configure resources

You need to configure resources used in deployment such as IAM roles, instance profiles and S3 bucket. AWS Transform does not have permissions to create S3 buckets or to make changes in your IAM. This action is needed to be performed by a user who has permissions to create new roles. Usually it is account administrator.

To simplify the process, we provide a setup script that automatically:

- Executes the necessary CloudFormation templates
- Creates the ECS Service-Linked Role
- Sets up all required IAM resources

> **Important**
>
> Run the provided setup script as is - do not modify any resource names in the templates or script as this may impact the deployment process.

For Linux:

```
./setup.sh --region <YOUR-Region>
```

For Windows:

```
./setup.ps1 -Region <YOUR-Region>
```

Check security groups in your VPC

AWS Transform deploys the applications to the same subnet and same security group as Aurora PostgreSQL instance.

In order for the application to connect to the Aurora instance, make sure that this security group has outbound rules allowing connection to the same security group and inbound rules allowing connection from the same security group.

Select and configure applications

The AWS Transform UI shows a list of applications that can be deployed. Select application and click **Configure and Commit** to open a configuration window where you can specify:

- Target infrastructure for the applications: select ECS or one of the EC2 instances
- Application or instance parameters

After you complete and close the configuration window, AWS Transform:

- Generates deployment artifacts based on your configuration
- Commits these artifacts to your repository, including:
  - Deployment scripts
  - CloudFormation templates
  - Sample CI/CD pipeline configurations (which you can use as reference for setting up your deployment automation)

Once the commit is complete, the application status changes to *Configured and committed*. You can then select the applications and use the **Deploy** button in the top right to initiate deployment.

ECS deployment

All applications with target ECS are deployed into a new ECS cluster that is created in your account. Each application is deployed in its own dedicated container - multiple applications per container are not supported. You configure container parameters for each application individually, such as required CPU and Memory.

EC2 deployment

You can select one of proposed EC2 instances and can configure the its parameters such as instance type. Multiple applications can be deployed to the same EC2 instance:

- To deploy multiple applications to a single instance, select the same target EC2 instance for each of these applications during configuration.
- All applications that have the same instance selected as a target are deployed to that instance and assigned different random ports.

- Note that if you edit instance parameters for one application, and then edit another application targeting the same instance, the changes affect all applications deployed to that instance.

Deployment success

Upon successful deployment, your applications are running on Aurora PostgreSQL with reduced licensing costs, improved performance, and cloud-native scalability. The job stays open unless explicitly stopped. This allows the option for a possible re-deployment with updated deployment parameters.

Best practices and troubleshooting

Best practices, and common issues and their resolutions during the modernization process.

ECS application logs

CloudWatch logs

- All ECS container logs are automatically sent to CloudWatch Logs
- Access logs in the CloudWatch console under Log Groups
- Log group naming format: /aws/ecs/{application-name}
- Each container instance creates a new log stream within the group

Viewing logs

Through AWS Console:

- Navigate to CloudWatch > Log Groups
- Select your application's log group
- Choose the relevant log stream to view container logs

Using AWS CLI:

```
aws logs get-log-events --log-group-name /aws/ecs/your-app-name --log-stream-name your-
stream-name
```

Common log locations

- Application logs: CloudWatch Logs
- ECS Service Events: ECS Console > Cluster > Service > Events tab
- Container health/status: ECS Console > Cluster > Service > Tasks tab

Database connection management

Applications use environment variables for database connection settings

If you experience connectivity issues:

- Verify the current connection settings in your environment variables
- Update environment variables to modify database connection strings as needed
- Connection string changes can be made through environment variable updates without application redeployment

Database connection issues

Problem: Cannot connect AWS Transform to SQL Server

Solutions:

- Verify network connectivity between AWS Transform and SQL Server
- Check security group rules for proper port access (1433)
- Confirm database credentials in Secrets Manager
- Test database permissions with the created user
- Ensure SQL Server is configured for mixed mode authentication
- Verify secret has required tags (Project: atx-db-modernization, Owner: database-connector)

Firewall and security group issues

Problem: Connection timeout or "cannot reach database" errors

Root Cause: Security groups or network ACLs blocking traffic

Solutions:

1. Verify Security Group Configuration:
   - Confirm your SQL Server security group has an inbound rule allowing port 1433 from the DMS Schema Conversion security group
   - Check that the source is the security group ID (e.g., sg-0123456789abcdef0), not an IP address
   - Verify the DMS Schema Conversion security group is correctly specified in the Instance Profile
   - Ensure there are no conflicting deny rules
2. Check Network ACLs:
   - Verify subnet-level Network ACLs allow inbound traffic on port 1433
   - Ensure Network ACLs allow outbound ephemeral ports for return traffic
   - Check both the database subnet and DMS subnet Network ACLs
3. Verify VPC Configuration:
   - Confirm the DMS Schema Conversion instance and SQL Server are in the same VPC or have proper VPC peering
   - Check route tables allow traffic between subnets
   - Verify no firewall appliances are blocking traffic
4. Test Connectivity:
   - Launch a test EC2 instance in the same subnet as DMS Schema Conversion
   - Attach the same security group as DMS Schema Conversion
   - Test connection to SQL Server using telnet or SQL Server Management Studio
   - If test succeeds, the issue is with AWS Transform configuration; if it fails, the issue is network/firewall

Common Mistake: Opening port 1433 to 0.0.0.0/0 (all sources) is a security risk. Always use security group-based access control to limit access to only the DMS Schema Conversion security group.

Schema conversion issues

Problem: Schema conversion shows many action items

Solutions:

- Review action items in conversion report
- Prioritize based on impact

- Use Amazon Q Developer for complex SQL conversions
- Consult AWS Support for guidance
- Consider phased approach for complex databases

Application transformation issues

Problem: Application transformation fails to build

Solutions:

- Review build errors in transformation report
- Configure private NuGet feeds if needed
- Update package references if required
- Check for Windows-specific dependencies
- Review transformation logs for detailed errors

Data migration issues

Problem: Data migration validation fails

Solutions:

- Review validation report for specific failures
- Check data type mappings
- Verify identity column configuration (GENERATED BY DEFAULT vs GENERATED ALWAYS)
- Review computed column expressions
- Contact AWS Support for complex data issues

Resource cleanup issues

Problem: Transformation job fails with resource errors

Solutions:

- Check for existing DMS resources (migration projects, data providers, instance profiles)
- Clean up failed or incomplete resources from previous attempts

- Verify secrets are not scheduled for deletion
- Check service quotas for DMS and Aurora PostgreSQL
- Contact AWS Support if cleanup doesn't resolve the issue

Deployment issues

Problem: Transformed application cannot connect to Aurora PostgreSQL

Solutions:

- Verify connection string format for PostgreSQL
- Check security group rules
- Verify database credentials in Secrets Manager
- Ensure SSL/TLS is properly configured
- Test connection using psql or pgAdmin

Getting additional help

When contacting AWS Support, please provide:

- Transformation job ID
- AWS account ID
- Region
- Error messages and screenshots
- Transformation logs (available in AWS Transform console)

Security in AWS Transform

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to AWS Transform, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using AWS Transform. The following topics show you how to configure AWS Transform to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your AWS Transform resources.

Topics

Data protection in AWS Transform

The AWS shared responsibility model applies to data protection in AWS Transform. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the

AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We recommend TLS 1.2 or later.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with AWS Transform or other AWS services using the AWS Management Console, API, AWS Command Line Interface (AWS CLI), or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. For more information about how AWS Transform uses content, see AWS Transform service improvement.

AWS Transform stores your questions, its responses, and additional context, such as console metadata and code in your IDE, to generate responses to your questions. For information about how data is encrypted, see Data encryption in AWS Transform. For information about how AWS may use some questions that you ask AWS Transform and its responses to improve our services, see AWS Transform service improvement.

In AWS Transform, your data is stored in the AWS Region where your AWS Transform profile was created.

With cross-inferencing, your requests to AWS Transform may be processed in a different Region within the geography where your data is stored. For more information, see Cross-Region inference.

Topics

Cross-region processing in AWS Transform

The following sections describe how cross-region inference and cross-region calls are used to provide the AWS Transform service.

Cross-region inference

AWS Transform is powered by Amazon Bedrock, and uses cross-region inference to distribute traffic across different AWS Regions to enhance large language model (LLM) inference performance and reliability. With cross-region inference, you get:

- Increased throughput and resilience during high demand periods
- Improved performance
- Access to newly launched AWS Transform capabilities and features that rely on the most powerful LLMs hosted on Amazon Bedrock

Cross-region inference requests are kept within the AWS Regions that are part of the geography where the data originally resides. For example, a request made from a AWS Transform configuration in the US is kept within the AWS Regions in the US. Although cross-region inferencing doesn't change where your data is stored, your requests and output results may move outside of the Region where the data originally resides. All data will be encrypted while transmitted across Amazon's secure network. There's no additional cost for using cross-region inference.

Cross region inference doesn't affect where your data is stored. For information on where data is stored when you use AWS Transform, see Data protection in AWS Transform.

Supported regions for AWS Transform cross-region inference

Certain requests you make to AWS Transform might require cross-region calls. The following table describes what Regions your requests may be routed to depending on the geography where the request originated.

Source Region	Destination Regions
US East (N. Virginia) (us-east-1)	US East (N. Virginia) (us-east-1) US East (Ohio) (us-east-2) US West (Oregon) (us-west-2)
Europe (Frankfurt) (eu-central-1)	Europe (Frankfurt) (eu-central-1) Europe (Stockholm) (eu-north-1) Europe (Milan) (eu-south-1) Europe (Spain) (eu-south-2) Europe (Ireland) (eu-west-1) Europe (Paris) (eu-west-3)
Asia Pacific (Mumbai) (ap-south-1)	Asia Pacific (Tokyo) (ap-northeast-1) Asia Pacific (Seoul) (ap-northeast-2) Asia Pacific (Osaka) (ap-northeast-3) Asia Pacific (Mumbai) (ap-south-1) Asia Pacific (Hyderabad) (ap-south-2) Asia Pacific (Singapore) (ap-southeast-1) Asia Pacific (Sydney) (ap-southeast-2) Asia Pacific (Melbourne) (ap-southeast-4)
Asia Pacific (Tokyo) (ap-northeast-1)	Asia Pacific (Tokyo) (ap-northeast-1)

Source Region	Destination Regions
	Asia Pacific (Osaka) (ap-northeast-3)
Asia Pacific (Seoul) (ap-northeast-2)	Asia Pacific (Tokyo) (ap-northeast-1) Asia Pacific (Seoul) (ap-northeast-2) Asia Pacific (Osaka) (ap-northeast-3) Asia Pacific (Mumbai) (ap-south-1) Asia Pacific (Hyderabad) (ap-south-2) Asia Pacific (Singapore) (ap-southeast-1) Asia Pacific (Sydney) (ap-southeast-2) Asia Pacific (Melbourne) (ap-southeast-4)
Asia Pacific (Sydney) (ap-southeast-2)	Asia Pacific (Sydney) (ap-southeast-2) Asia Pacific (Melbourne) (ap-southeast-4)
Europe (London) (eu-west-2)	Europe (Frankfurt) (eu-central-1) Europe (Stockholm) (eu-north-1) Europe (Milan) (eu-south-1) Europe (Spain) (eu-south-2) Europe (Ireland) (eu-west-1) Europe (London) (eu-west-2) Europe (Paris) (eu-west-3)
Canada (Central) (ca-central-1)	Commercial AWS Regions + Canada (Central) (ca-central-1)

For a complete list of Regions where you can use AWS Transform, see Supported Regions for AWS Transform.

Cross-Region knowledge

When you ask a general question about AWS Transform services, transformation workflows, or related AWS documentation, AWS Transform might make cross-region requests to US East (Virginia) (us-east-1) for US regions or Europe (Frankfurt) (eu-central-1) for all other regions to retrieve documentation and fulfill your request. For example, when you ask questions about how to use other AWS services such as Lambda, AWS Transform might make a cross-region call to retrieve relevant AWS documentation to respond to your question. The following table describes what Regions your requests may be routed to depending on the geography where the request originated.

Source Region	Destination Regions
US East (N. Virginia) (us-east-1)	US East (N. Virginia) (us-east-1)
Europe (Frankfurt) (eu-central-1)	Europe (Frankfurt) (eu-central-1)
Europe (London) (eu-west-2)	Europe (Frankfurt) (eu-central-1)
Asia Pacific (Tokyo) (ap-northeast-1)	Europe (Frankfurt) (eu-central-1)
Asia Pacific (Sydney) (ap-southeast-2)	Europe (Frankfurt) (eu-central-1)
Asia Pacific (Seoul) (ap-northeast-2)	Europe (Frankfurt) (eu-central-1)
Asia Pacific (Mumbai) (ap-south-1)	Europe (Frankfurt) (eu-central-1)
Canada (Central) (ca-central-1)	Europe (Frankfurt) (eu-central-1)

This setting is enabled by default. An account administrator can modify this setting. Disabling this feature results in the loss of access to features that require AWS Transform to retrieve knowledge from other regions. This might result in less accurate responses.

To disable cross-region calls made by AWS Transform:

1. When first setting up AWS Transform, navigate to the **Get Started** page and complete the configuration. For for an existing AWS Transform configuration, navigate to the **Settings** page.

2. Toggle **Enable cross-region calls to answer general AWS related questions** to the *off* position.

AWS Transform for Windows cross-region inference

The following table shows the source Regions from which you can call the inference profile and the destination Regions to which the requests can be routed:

Source Region	Inference Destination Regions
US East (N. Virginia) (us-east-1)	US East (N. Virginia) (us-east-1) US East (Ohio) (us-east-2) US West (Oregon) (us-west-2)
Europe (Frankfurt) (eu-central-1)	Europe (Frankfurt) (eu-central-1) Europe (Stockholm) (eu-north-1) Europe (Milan) (eu-south-1) Europe (Spain) (eu-south-2) Europe (Ireland) (eu-west-1) Europe (Paris) (eu-west-3)
Asia Pacific (Tokyo) (ap-northeast-1)	Asia Pacific (Tokyo) (ap-northeast-1) Asia Pacific (Osaka) (ap-northeast-3)
Asia Pacific (Sydney) (ap-southeast-2)	Asia Pacific (Sydney) (ap-southeast-2) Asia Pacific (Melbourne) (ap-southeast-4)
Asia Pacific (Seoul) (ap-northeast-2)	All commercial regions
Asia Pacific (Mumbai) (ap-south-1)	All commercial regions
Canada (Central) (ca-central-1)	All commercial regions

AWS Transform for Migrations cross-region inference

The following table shows the source Regions from which you can call the inference profile and the destination Regions to which the requests can be routed:

Source Region	Inference Destination Regions
US East (N. Virginia) (us-east-1)	All commercial regions
Europe (Frankfurt) (eu-central-1)	All commercial regions
Europe (London) (eu-west-2)	All commercial regions
Asia Pacific (Tokyo) (ap-northeast-1)	Asia Pacific (Tokyo) (ap-northeast-1) Asia Pacific (Osaka) (ap-northeast-3)
Asia Pacific (Sydney) (ap-southeast-2)	Asia Pacific (Sydney) (ap-southeast-2) Asia Pacific (Melbourne) (ap-southeast-4)
Asia Pacific (Seoul) (ap-northeast-2)	All commercial regions
Asia Pacific (Mumbai) (ap-south-1)	All commercial regions
Canada (Central) (ca-central-1)	All commercial regions

AWS Transform Custom

AWS Transform custom is available in US East (N. Virginia) (us-east-1) and Europe (Frankfurt) (eu-central-1) and uses Amazon Bedrock geographic cross-region inference. This means that some of your calls might be routed to AWS Regions outside of your source region within the same geographic region. You can access AWS Transform custom features from workspaces deployed in supported regions.

Source Region	Inference Destination Regions
US East (N. Virginia) (us-east-1)	US East (N. Virginia) (us-east-1) US East (Ohio) (us-east-2)

Source Region	Inference Destination Regions
	US West (Oregon) (us-west-2)
Europe (Frankfurt) (eu-central-1)	Europe (Frankfurt) (eu-central-1) Europe (Stockholm) (eu-north-1) Europe (Milan) (eu-south-1) Europe (Spain) (eu-south-2) Europe (Ireland) (eu-west-1) Europe (Paris) (eu-west-3)
Europe (London) (eu-west-2)	n/a
Asia Pacific (Tokyo) (ap-northeast-1)	n/a
Asia Pacific (Sydney) (ap-southeast-2)	n/a
Asia Pacific (Seoul) (ap-northeast-2)	n/a
Asia Pacific (Mumbai) (ap-south-1)	n/a
Canada (Central) (ca-central-1)	n/a

Data encryption in AWS Transform

This topic provides information specific to AWS Transform about encryption in transit and encryption at rest.

AWS Transform provides encryption by default to protect sensitive customer data at rest with encryption using AWS owned keys.

Encryption types

Encryption in transit

Communication between customers and AWS Transform and between AWS Transform and its downstream dependencies is protected using TLS 1.2 or higher connections.

Important

When you Connect a discovery account, AWS Transform creates an Amazon S3 bucket on your behalf in that account. That bucket does not have `SecureTransport` enabled by default. If you want the bucket policy to include secure transport you must update the policy. For more information, see Security best practices for Amazon S3.

Encryption at rest

AWS Transform stores data at rest using Amazon DynamoDB and Amazon Simple Storage Service (Amazon S3). The data at rest is encrypted using AWS encryption solutions by default. AWS Transform encrypts your data with encryption using AWS owned keys from AWS Key Management Service (AWS KMS). You don't have to take any action to protect the AWS managed keys that encrypt your data. For more information, see AWS owned keys in the *AWS Key Management Service Developer Guide*.

Data type	AWS-owned key encryption	Customer managed key encryption (Optional)
Customer bucket data Customer inputs and outputs such as code and documentation stored in an Amazon S3 bucket	Enabled	Enabled
Artifact Store Intermediate artifacts as part of code transformation stored in an S3 bucket	Enabled	Enabled
Job Objective The customer's intent for the job stored in an Amazon S3 bucket	Enabled	Enabled

Data type	AWS-owned key encryption	Customer managed key encryption (Optional)
Chat messages Messages between the customer and AWS Transform stored in an Amazon S3 bucket	Enabled	Enabled
Chat Knowledge Base Indexed data relevant to AWS Transform and customer chat stored in Amazon OpenSearch and processed via AWS Bedrock	Enabled	Enabled

Note: The customer can register their own Customer Managed Key (CMK) to be used for encrypting all of the above data types.

Customer managed keys are KMS keys in your AWS account that you create, own, and manage to directly control access to your data by controlling access to the KMS key. Only symmetric keys are supported. For information on creating your own KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

When you use a customer managed key, AWS Transform makes use of KMS grants, allowing authorized users, roles, or applications to use a KMS key. When an AWS Transform administrator chooses to use a customer managed key for encryption during configuration, a grant is created for them. This grant is what allows the end user to use the encryption key for data encryption at rest. For more information on grants, see Grants in AWS KMS.

AWS Owned Keys (Default)

> **Note**
>
> **AWS owned keys** — AWS Transform uses these keys by default to automatically encrypt personally identifiable data. You can't view, manage, or use AWS owned keys, or audit their use. However, you don't have to take any action or change any programs to protect the

keys that encrypt your data. For more information, see AWS owned keys in the AWSKey Management Service Developer Guide.

Encryption of data at rest by default helps reduce the operational overhead and complexity involved in protecting sensitive data. At the same time, it enables you to build secure applications that meet strict encryption compliance and regulatory requirements.

While you can't disable this layer of encryption or select an alternate encryption type, you can add a second layer of encryption over the existing AWS owned encryption keys by choosing a customer managed key when you create your transformation:

Customer managed keys (Optional)

Customer managed keys — AWS Transform supports the use of a symmetric customer managed key that you create, own, and manage to add a second layer of encryption over the existing encryption using AWS owned keys. Because you have full control of this layer of encryption, you can perform such tasks as:

- Establishing and maintaining key policies
- Establishing and maintaining IAM policies and grants
- Enabling and disabling key policies
- Rotating key cryptographic material
- Adding tags
- Creating key aliases
- Rotating key cryptographic material
- Adding tags
- Creating key aliases
- Scheduling keys for deletion
- For more information, see customer managed key in the *AWS Key Management Service Developer Guide*.

Note

AWS Transform automatically enables encryption at rest using AWS owned keys to protect personally identifiable data at no charge. However, AWS KMS charges apply for using a

customer managed key. For more information about pricing, see the AWS Key Management Service pricing.

For more information on AWS KMS, see What is AWS Key Management Service?

How AWS Transform uses grants in AWS Key Management Service

AWS Transform requires a grant to use your customer managed key.

When you create a [Resource Name] encrypted with a customer managed key, AWS Transform creates a grant on your behalf by sending a CreateGrant request to AWS Key Management Service. Grants in AWS Key Management Service are used to give AWS Transform access to a KMS key in a customer account.

AWS Transform requires the grant to use your customer managed key for the following internal operations:

- Send [KMS API] requests to AWS KMS to verify that the symmetric customer managed KMS key ID entered when creating [Resource Name] is valid.
- Send [KMS API] requests to AWS KMS to generate data keys encrypted by your customer managed key.
- Send [KMS API] requests to AWS KMS to decrypt the encrypted data keys so that they can be used to encrypt your data.

You can revoke access to the grant, or remove the service's access to the customer managed key at any time. If you do, AWS Transform won't be able to access any of the data encrypted by the customer managed key, which affects operations that are dependent on that data. For example, if you attempt to get [Resource Name] from an encrypted [Resource Name]that AWS Transform can't access, then the operation would return an AccessDeniedException error.

Create a customer managed key

You can create a symmetric customer managed key by using the AWS Management Console, or the AWS Key Management Service APIs.

To create a symmetric customer managed key, follow the steps for Creating symmetric customer managed key in the *AWS Key Management Service Developer Guide*.

To use your customer managed key with your AWS Transform resources, the following API operations must be permitted in the key policy:

kms:CreateGrant – Adds a grant to a customer managed key. Grants control access to a specified KMS key, which allows access to grant operations AWS Transform requires. For more information about Using Grants, see the *AWS Key Management Service Developer Guide*.

This allows AWS Transform to do the following:

- Call [KMS API (GenerateDataKeyWithoutPlainText/GenerateDatakey) to generate an encrypted data key and store it, because the data key isn't immediately used to encrypt.
- Call [KMS API (Decrypt)] to use the stored encrypted data key to access encrypted data.
- Set up a retiring principal to allow the service to RetireGrant.
- kms:DescribeKey – Provides the customer managed key details to allow [Service Name] to validate the key.

Encryption for SQL modernization

During the SQL Server modernization process AWS Transform creates resources in your account, such as EC2 instances, ECS clusters, ECR images and S3 objects. You have the option of providing a customer managed key to encrypt data in your account. You will need to tag your KMS key with key **CreatedFor** and value **AWSTransform** . This example policy lists the minimum permissions required in your key policy if you don't have a default key policy. If you have a default key policy, you still need to manually update the KMS key in addition to your default key policy to allow ECS to use your key.

```
"Statement": [
   {
     "Effect": "Allow",
     "Principal": {
       "AWS": "arn:aws:iam::111122223333:root"
     },
     "Action": [
       "kms:Decrypt",
       "kms:GenerateDataKey"
     ],
     "Resource": "*",
     "Condition": {
       "StringLike": {
         "kms:ViaService": "s3.*.amazonaws.com",
```

```
          "kms:EncryptionContext:aws-transform": "*"
        },
        "ArnLike": {
          "aws:PrincipalArn": "arn:aws:iam::*:role/*AWSTransform*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:ReEncrypt*"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "kms:ViaService": "ec2.*.amazonaws.com"
        },
        "ArnLike": {
          "aws:PrincipalArn": "arn:aws:iam::*:role/*AWSTransform*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": "kms:CreateGrant",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "kms:ViaService": [
            "ec2.*.amazonaws.com",
            "ecr.*.amazonaws.com"
          ]
        },
        "Bool": {
          "kms:GrantIsForAWSResource": "true"
```

```
        },
        "ArnLike": {
          "aws:PrincipalArn": "arn:aws:iam::*:role/*AWSTransform*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fargate.amazonaws.com"
      },
      "Action": "kms:GenerateDataKeyWithoutPlaintext",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:ecs:clusterName": "AWSTransform*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fargate.amazonaws.com"
      },
      "Action": "kms:CreateGrant",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:ecs:clusterName": "AWSTransform*"
        },
        "ForAllValues:StringEquals": {
          "kms:GrantOperations": "Decrypt"
        }
      }
    }
  ]
```

Encryption for database migration

During the database migration process, AWS Transform uses AWS Database Migration Service (DMS) to migrate your database data. You can optionally provide a customer managed key to encrypt your database data during migration.

The following policy contains three statements. The first statement grants root account permissions for the AWS KMS key. The second statement allows the connector role to validate the key during discovery. The third statement allows the connector role to use the key for DMS operations.

```
"Statement": [
   {
     "Sid": "Enable IAM User Permissions",
     "Effect": "Allow",
     "Principal": { "AWS": "arn:aws:iam::111122223333:root" },
     "Action": "kms:*",
     "Resource": "*"
   },
   {
     "Sid": "Allow connector role to validate key during discovery",
     "Effect": "Allow",
     "Principal": { "AWS": "connector-role-arn" },
     "Action": "kms:DescribeKey",
     "Resource": "*"
   },
   {
     "Sid": "Allow connector role to use key via DMS",
     "Effect": "Allow",
     "Principal": { "AWS": "connector-role-arn" },
     "Action": [
       "kms:CreateGrant",
       "kms:Decrypt",
       "kms:DescribeKey",
       "kms:Encrypt",
       "kms:GenerateDataKey*",
       "kms:ReEncrypt*"
     ],
     "Resource": "*",
     "Condition": {
       "StringEquals": {
         "kms:ViaService": "dms.us-east-1.amazonaws.com"
       }
     }
   }
 ]
```

Using customer managed KMS keys

After creating a customer managed KMS key, an AWS Transform administrator must provide the key in the AWS Transform console to use it to encrypt data.

To set up a customer managed key to encrypt data in AWS Transform, administrators need permissions to use AWS KMS.

To use functionality that is encrypted with a customer managed key, users need permissions to allow AWS Transform to access the customer managed key.

If you see an error related to KMS grants while using AWS Transform, you likely need to update your permissions to allow AWS Transform to create grants. To automatically configure the needed permissions, go to the AWS Transform console and choose **Update permissions** in the banner at the top of the page. In order to allow AWS Transform to create grants, you need to update the permissions on the AWS Transform console page.

Allow AWS Transform access to customer managed keys

The following example policy statement grants users permissions to access features encrypted with a customer managed key by allowing AWS Transform access to the key. This policy is required to use AWS Transform if an administrator has set up a customer managed key for encryption.

> **Note**
>
> This information does not apply to SQL Server modernization deployment workflow using CMK.

```
"Statement": [
   {
     "Sid": "QKMSDecryptGenerateDataKeyPermissions",
     "Effect": "Allow",
     "Action": [
       "kms:Decrypt",
       "kms:GenerateDataKey",
       "kms:GenerateDataKeyWithoutPlaintext",
       "kms:ReEncryptFrom",
       "kms:ReEncryptTo"
     ],
```

```
      "Resource": [
        "arn:aws:kms:arn:aws::111122223333:key/[[key_id]]"
      ],
      "Condition": {
        "StringLike": {
          "kms:ViaService": [
            "q.us-east-1.amazonaws.com"
          ]
        }
      }
    }
  ]
```

AWS Transform service improvement

To help AWS Transform provide the most relevant information, we may use certain content from AWS Transform, such as questions that you ask AWS Transform and its responses, for service improvement. This page explains what content we use and how to opt out.

AWS Transform content used for service improvement

We may use certain content from AWS Transform for service improvement. AWS Transform may use this content, for example, to provide better responses to common questions, fix AWS Transform operational issues, for de-bugging, or for model training.

Content that AWS may use for service improvement includes, for example, your questions to AWS Transform and the responses and code that AWS Transform generates.

How to opt out

The way you opt out of AWS Transform using content for service improvement depends on the environment where you use AWS Transform.

For AWS Transform configure an AI services opt-out policy in AWS Organizations. For more information, see AI services opt-out policies in the *AWS Organizations User Guide*.

To opt out of sharing your content in Visual Studio IDE, use the following procedure.

Go to **Tools** -> **Options** -> **AWS Toolkit** -> **Amazon Q**

Toggle **Share Amazon Q Content with AWS** to **True** or **False**.

To opt out of sharing your telemetry data in the AWS Toolkit for Visual Studio, use this procedure:

1. Under **Tools**, choose **Options**.
2. In the **Options** pane, choose **AWS Toolkit**, and then choose **General**.
3. Deselect **Allow AWS Toolkit to collect usage information**.

Identity and access management for AWS Transform

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use AWS Transform resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting AWS Transform identity and access)
- **Service administrator** - determine user access and submit permission requests (see How AWS Transform works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policy examples for AWS Transform)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-

based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How AWS Transform works with IAM

Before you use IAM to manage access to AWS Transform, learn what IAM features are available to use with AWS Transform.

IAM feature	AWS Transform support
Identity-based policies	Yes

IAM feature	AWS Transform support
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
ACLs	No
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Principal permissions	Yes
Service roles	Yes
Service-linked roles	Yes

To get a high-level view of how AWS Transform and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for AWS Transform

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for AWS Transform

To view examples of AWS Transform identity-based policies, see Identity-based policy examples for AWS Transform.

Resource-based policies within AWS Transform

Supports resource-based policies: No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Policy actions for AWS Transform

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

To see a list of AWS Transform actions, see Actions, resources, and condition keys for AWS Transform in the *Service Authorization Reference*.

To see a list of actions for AWS Transform *custom*, see Actions, resources, and condition keys for AWS Transform custom in the *Service Authorization Reference*.

Policy actions in AWS Transform use the following prefix before the action:

```
transform
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "transform:action1",
      "transform:action2"
         ]
```

To view examples of AWS Transform identity-based policies, see Identity-based policy examples for AWS Transform.

Policy resources for AWS Transform

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of AWS Transform resource types and their ARNs, and with which actions you can specify the ARN of each resource, see Actions, resources, and condition keys for AWS Transform in the *Service Authorization Reference*.

To see a list of resource types and their ARNs, and with which actions you can specify the ARN of each resource for AWS Transform *custom*, see Actions, resources, and condition keys for AWS Transform custom in the *Service Authorization Reference*.

To view examples of AWS Transform identity-based policies, see Identity-based policy examples for AWS Transform.

Policy condition keys for AWS Transform

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of AWS Transform condition keys, and with which actions and resources you can use a condition key, see Actions, resources, and condition keys for AWS Transform in the *Service Authorization Reference*.

To see a list of condition keys, and with which actions and resources you can use a condition key for AWS Transform *custom*, see Actions, resources, and condition keys for AWS Transform custom in the *Service Authorization Reference*.

To view examples of AWS Transform identity-based policies, see Identity-based policy examples for AWS Transform.

ACLs in AWS Transform

Supports ACLs: No

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

ABAC with AWS Transform

Supports ABAC (tags in policies): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using temporary credentials with AWS Transform

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

Cross-service principal permissions for AWS Transform

Supports forward access sessions (FAS): Yes

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

Service roles for AWS Transform

Supports service roles: Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break AWS Transform functionality. Edit service roles only when AWS Transform provides guidance to do so.

Service-linked roles for AWS Transform

Supports service-linked roles: Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS

account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing AWS Transform service-linked roles, see Using service-linked roles for AWS Transform.

Identity-based policy examples for AWS Transform

By default, users and roles don't have permission to create or modify AWS Transform resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by AWS Transform, including the format of the ARNs for each of the resource types, see Actions, Resources, and Condition Keys for AWS Transform in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete AWS Transform resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies

that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.

- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the AWS Transform console

To access the AWS Transform console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the AWS Transform resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Allow administrators to accept a connector request from the account with AWS Transform

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "transform:GetConnector",
                "transform:AssociateConnectorResource",
                "transform:RejectConnector"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketPublicAccessBlock",
                "s3:GetAccountPublicAccessBlock"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:AttachRolePolicy",
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/service-role/
AWSTransform-*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreatePolicy"
            ],
            "Resource": "arn:aws:iam::111122223333:policy/service-role/
AWSTransform-*"
```

```
        }
    ]
}
```

Allow administrators to assign existing IAM Identity Center users and create new IAM Identity Center users to assign to AWS Transform

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
  "Sid": "AllowKmsAccessViaIdentityCenter",
  "Effect": "Allow",
  "Action": [
  "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
  "ArnLike": {
  "kms:EncryptionContext:aws:sso:instance-arn": "arn:*:sso:::instance/*"
  },
  "StringLike": {
  "kms:ViaService": "sso.*.amazonaws.com"
  }
  }
  },
  {
  "Sid": "AllowKmsAccessViaIdentityStore",
  "Effect": "Allow",
  "Action": [
  "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
  "ArnLike": {
  "kms:EncryptionContext:aws:identitystore:identitystore-arn":
 "arn:*:identitystore::*:identitystore/*"
  },
  "StringLike": {
  "kms:ViaService": "identitystore.*.amazonaws.com"
```

```
  }
  }
  }
  ]
  }
```

Allow administrators to enable AWS Transform

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sso:ListInstances",
        "sso:CreateInstance",
        "sso:CreateApplication",
        "sso:PutApplicationAuthenticationMethod",
        "sso:PutApplicationGrant",
        "sso:PutApplicationAssignmentConfiguration",
        "sso:ListApplications",
        "sso:GetSharedSsoConfiguration",
        "sso:DescribeInstance",
        "sso:PutApplicationAccessScope",
        "sso:DescribeApplication",
        "sso:DeleteApplication",
        "sso:UpdateApplication",
        "sso:DescribeRegisteredRegions",
        "sso:GetSSOStatus"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sso-directory:GetUserPoolInfo",
        "sso-directory:DescribeUsers",
```

```
        "sso-directory:DescribeGroups",
        "sso-directory:SearchGroups",
        "sso-directory:SearchUsers",
        "sso-directory:DescribeDirectory"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:DisableAWSServiceAccess",
        "organizations:EnableAWSServiceAccess"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "kms:ListAliases",
        "kms:CreateGrant",
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey*",
        "kms:RetireGrant",
        "kms:DescribeKey"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
```

```
        "arn:aws:iam::*:role/aws-service-role/transform.amazonaws.com/
AWSServiceRoleForAWSTransform"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "transform:UpdateProfile",
        "transform:ListProfiles",
        "transform:CreateProfile",
        "transform:DeleteProfile"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Troubleshooting AWS Transform identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with AWS Transform and IAM.

Topics

- I am not authorized to perform an action in AWS Transform
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my AWS Transform resources

I am not authorized to perform an action in AWS Transform

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `AWS Transform:GetWidget` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: AWS
 Transform:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `AWS Transform:GetWidget` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to AWS Transform.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in AWS Transform. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my AWS Transform resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether AWS Transform supports these features, see How AWS Transform works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Using service-linked roles for AWS Transform

AWS Transform uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to AWS Transform. Service-linked roles are predefined by AWS Transform and include all the permissions that the service requires to call other AWS services on your behalf.

Using service-linked roles for AWS Transform

AWS Transform uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to AWS Transform. Service-linked roles are predefined by AWS Transform and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up AWS Transform easier because you don't have to manually add the necessary permissions. AWS Transform defines the permissions of its service-linked roles, and unless defined otherwise, only AWS Transform can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy cannot be attached to any other IAM entity.

You can delete a service-linked role only after first deleting their related resources. This protects your AWS Transform resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS services that work with IAM and look for the services that have **Yes** in the **Service-linked roles** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for AWS Transform

AWS Transform uses the service-linked role named AWSServiceRoleForAWSTransform – This Service-Linked Role provides AWS Transform with the ability to provide usage information.

The AWSServiceRoleForAWSTransform service-linked role trusts the following services to assume the role:

- `transform.amazonaws.com`

The role permissions policy named AWSServiceRoleForAWSTransform allows AWS Transform to complete the following actions on the specified resources:

- cloudwatch:PutMetricData
  - Send custom metrics to CloudWatch for AWS Transform operations
  - Track transformation progress, success rates, and performance metrics
  - Enable monitoring and alerting on transformation workflows
- sso:DescribeApplication
  - View details about a specific application in Identity Center
  - Get application metadata like name, description, status, and configuration
- sso:GetApplicationAssignmentConfiguration
  - Retrieve assignment configuration settings for an application
  - See how users/groups are configured to be assigned to the application
- sso:ListApplicationAssignmentsForPrincipal
  - List all applications assigned to a specific user or group (principal)
  - View which applications a particular identity has access to
- Enables decryption of KMS-encrypted data when accessed through IAM Identity Center. Only works when the encryption context contains a valid IAM Identity Center instance ARN and must be accessed via IAM Identity Center service endpoints.
- Allows decryption of KMS-encrypted data when accessed through Identity Store. Only works when the encryption context contains a valid Identity Store ARN and must be accessed via Identity Store service endpoints.
- secretsmanager:GetSecretValue
  - Access AWS Transform service-linked secrets used to store client secrets for external identity providers

  - Resource: arn:aws:secretsmanager:*:*:secret:transform-preprod!*
  - Condition: Must be owned by transform-preprod service and accessed from same account
- support:CreateCase, support:DescribeCases, support:DescribeCommunications, support:AddCommunicationToCase, support:ResolveCase
  - Create and manage premium support cases from the AWS Transform web application
  - View case details and communications
  - Add communications and resolve support cases

You must configure permissions to allow your users, groups, or roles to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating a service-linked role for AWS Transform

You don't need to manually create a service-linked role. When you create a profile for AWS Transform in the AWS Management Console, AWS Transform creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you update the settings, AWS Transform creates the service-linked role for you again.

You can also use the IAM console or AWS CLI to create a service-linked role with the `transform.amazonaws.com` service name. For more information, see Creating a service-linked role in the *IAM User Guide*. If you delete this service-linked role, you can use this same process to create the role again.

Editing a service-linked role for AWS Transform

AWS Transform does not allow you to edit the AWSServiceRoleForAWSTransform service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for AWS Transform

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored

or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If the AWS Transform service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForAWSTransform service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported Regions for AWS Transform service-linked roles

AWS Transform does not support using service-linked roles in every Region where the service is available. You can use the AWSServiceRoleForAWSTransform role in the following Regions. For more information, see AWS Regions and endpoints.

Region name	Region identity	Support in AWS Transform
US East (N. Virginia)	us-east-1	Yes
Europe (Frankfurt)	eu-central-1	Yes

Using service-linked roles for AWS Transform Custom

AWS Transform Custom uses the service-linked role named **AWSServiceRoleForAWSTransformCustom** to publish metrics to your account on your behalf. These metrics let you monitor transformation counts, latencies, and status codes directly in your dashboards.

This service-linked role is predefined by AWS Transform Custom and includes only the permissions the service needs. You don't have to manually add any permissions, and the role can only be assumed by AWS Transform Custom. For general information about service-linked roles, see Using service-linked roles in the *IAM User Guide*.

Service-linked role permissions for AWS Transform Custom

The AWSServiceRoleForAWSTransformCustom service-linked role trusts the following services to assume the role:

- `transform-custom.amazonaws.com`

The role permissions policy named AWSServiceRoleForAWSTransformCustom allows AWS Transform Custom to complete the following actions on the specified resources:

- `cloudwatch:PutMetricData` on all AWS resources
  - Publish operational metrics to under the `AWS/TransformCustom` namespace
  - Track transformation counts, latencies, and status codes
  - Scoped to the `AWS/TransformCustom` namespace via the `cloudwatch:namespace` condition key

You must configure permissions to allow your users, groups, or roles to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating a service-linked role for AWS Transform Custom

You don't need to manually create a service-linked role. When you run a transformation using the AWS Transform Custom CLI, AWS Transform Custom creates the service-linked role for you.

If you delete this service-linked role, and then need to create it again, you can use the same process to recreate the role in your account. When you run a transformation using the AWS Transform Custom CLI, AWS Transform Custom creates the service-linked role for you again.

You can also use the IAM console to create a service-linked role with the **AWSServiceRoleForAWSTransformCustom** use case. In the AWS CLI or the AWS API, create a service-linked role with the `transform-custom.amazonaws.com` service name. For more information, see Creating a service-linked role in the *IAM User Guide*. If you delete this service-linked role, you can use this same process to create the role again.

Editing a service-linked role for AWS Transform Custom

AWS Transform Custom does not allow you to edit the AWSServiceRoleForAWSTransformCustom service-linked role. After you create a service-linked role, you cannot change the name of the role

because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for AWS Transform Custom

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained.

The AWSServiceRoleForAWSTransformCustom service-linked role does not create persistent resources in your account. You can delete it directly without any resource cleanup.

If you delete this role and later run a transformation using the AWS Transform Custom CLI, the service-linked role is automatically recreated.

To manually delete the service-linked role using IAM

Use the IAM console, the AWS CLI, or the AWS API to delete the AWSServiceRoleForAWSTransformCustom service-linked role. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported Regions for AWS Transform Custom service-linked roles

AWS Transform Custom does not support using service-linked roles in every Region where the service is available. You can use the AWSServiceRoleForAWSTransformCustom role in the following Regions. For more information, see AWS Regions and endpoints.

Region name	Region identity	Support in AWS Transform Custom
US East (N. Virginia)	us-east-1	Yes
Europe (Frankfurt)	eu-central-1	Yes

AWS managed policies for AWS Transform

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS Transform updates for AWS managed policies

View details about updates to AWS managed policies for AWS Transform since March 1, 2021.

Change	Description	Date
AWSServiceRoleForAWSTransformCustom – New policy	Added a new AWS managed policy for the AWS Transform custom service-linked role. This policy allows AWS Transform custom to publish CloudWatch metrics to your account.	March 23, 2026
DBModProvisioningAndMigration – New policy	This policy grants database provisioning and migration capabilities.	March 24, 2026
DBModDiscoveryAndAssessment – New policy	Added a new AWS managed policy that provides comprehensive database modernization discovery and assessment capabilities.	March 24, 2026
AWSTransformCustomFullAccess – New policy	Added a new AWS managed policy that provides full	December 5, 2025

Change	Description	Date
	access to AWS Transform custom.	
AWSTransformCustomExecuteTransformations – New policy	Added a new AWS managed policy that provides access to execute transformations in AWS Transform custom.	December 5, 2025
AWSTransformCustomManageTransformations – New policy	Added a new AWS managed policy that provides access to create, update, read, and delete transformation resources in AWS Transform custom, as well as execute transformations.	December 5, 2025
AWSServiceRoleForAWSTransform – Updated policy	Added permissions to access the AWS Transform service-linked secret used to store the client secret for external identity providers. Added permissions to create a premium support case from the AWS Transform web app.	December 1, 2025
AWSTransformApplicationECSDeploymentPolicy – Updated policy	Added IAM role inspection permissions, ECS service-linked role creation, and KMS permissions for ECR encryption support.	November 22, 2025

Change	Description	Date
AWSTransformApplicationDeploymentPolicy – Updated policy	Added EC2 networking permissions, IAM role inspection permissions, S3 bucket listing permissions, and KMS encryption support for enhanced deployment capabilities.	November 22, 2025
AWSServiceRoleForAWSTransform – Updated policy	Added support for customer managed keys in IAM Identity Center.	September 17, 2025
AWSTransformApplicationDeploymentPolicy – New policy	Added a new AWS managed policy that enables AWS Transform to deploy transformed .NET applications by creating and managing Amazon EC2 instances, CloudFormation stacks, and associated resources.	August 28, 2025
AWSServiceRoleForAWSTransform – Updated policy	Added a new policy.	May 15, 2025

AWS managed policy: AWSServiceRoleForAWSTransform

This policy is attached to the AWSServiceRoleForAWSTransform service-linked role (SLR).

Permissions details

To view the policy permission details see AWSServiceRoleForAWSTransform in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSServiceRoleForAWSTransformCustom

This policy is attached to the AWSServiceRoleForAWSTransformCustom service-linked role (SLR). This role allows AWS Transform custom to publish CloudWatch metrics to your account on your behalf.

Description

This policy includes the following permissions:

- **Amazon CloudWatch** – Allows publishing metrics to CloudWatch under the `AWS/TransformCustom` namespace. This enables monitoring of transformation counts, latencies, and status codes in your CloudWatch dashboards.

AWS managed policy: AWSTransformApplicationDeploymentPolicy

This policy enables AWS Transform to deploy transformed .NET applications by creating and managing Amazon EC2 instances, CloudFormation stacks, and associated resources.

Description

This policy includes the following permissions:

- **CloudFormation** – Allows creating, updating, deleting, and describing CloudFormation stacks with names that start with AWSTransform. Stack operations are restricted to resources tagged with CreatedBy: AWSTransform and limited to the same AWS account.
- **Amazon EC2** – Allows describing VPCs, subnets, security groups, images, instances, route tables, and internet gateways. Permits running, starting, stopping, terminating, and modifying EC2 instances, but only when called through CloudFormation. Tag creation is restricted to specific allowed tag keys and only during CloudFormation operations.
- **AWS Identity and Access Management (IAM)** – Allows getting and passing specific IAM roles for AWSTransform deployment instances. Includes permissions to inspect role policies and attachments. Access is restricted to the same AWS account.
- **Amazon EC2 Systems Manager (SSM)** – Allows retrieving Amazon Linux AMI parameters from the AWS-managed parameter store and sending commands to AWSTransform-tagged instances.
- **Amazon S3** – Allows managing objects in AWSTransform deployment buckets, including listing buckets and getting bucket locations within the same AWS account.

- **AWS Key Management Service (KMS)** – Allows encryption and decryption operations using KMS keys tagged for AWSTransform, with restrictions to S3 and EC2 service usage.

The policy implements least-privilege access through resource-level permissions, tag-based conditions, service control restrictions using `aws:CalledVia`, account-level restrictions, and explicit deny statements to prevent unauthorized tag modifications outside of CloudFormation operations.

Permissions details

To view the policy permission details see AWSTransformApplicationDeploymentPolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSTransformApplicationECSDeploymentPolicy

This policy enables AWS Transform to deploy transformed applications to Amazon ECS by creating and managing ECS clusters, services, tasks, and associated resources.

Description

This policy includes the following permissions:

- **CloudFormation** – Allows creating, updating, deleting, and describing CloudFormation stacks with names that start with AWSTransform. Stack operations are restricted to resources tagged with CreatedBy: AWSTransform and limited to the same AWS account.
- **Amazon ECS** – Allows creating, updating, and deleting ECS clusters, services, and task definitions. Permits running tasks, listing tasks, and describing task status. All operations are restricted to resources with names starting with AWSTransform and tagged with CreatedBy: AWSTransform.
- **AWS Identity and Access Management (IAM)** – Allows getting and passing specific IAM roles for ECS tasks (AWSTransform-Deploy-ECS-Task-Role and AWSTransform-Deploy-ECS-Execution-Role). Includes permissions to inspect role policies and create the ECS service-linked role when needed.
- **Amazon CloudWatch Logs** – Allows creating, deleting, and managing log groups with names starting with /aws/ecs/AWSTransform. Permits retrieving log events for troubleshooting deployed applications.
- **Amazon ECR** – Allows creating container repositories with names starting with awstransform for storing application container images.

- **AWS Key Management Service (KMS)** – Allows creating grants and generating data keys for ECR encryption when using customer-managed KMS keys.

The policy implements least-privilege access through resource-level permissions, tag-based conditions, and account-level restrictions to ensure operations are limited to AWSTransform-managed resources within the same AWS account.

Permissions details

To view the policy permission details see AWSTransformApplicationECSDeploymentPolicy in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSTransformCustomFullAccess

This policy provides full access to AWS Transform custom.

Description

This policy includes the following permissions:

- **AWS Transform Custom** – Allows all actions on all AWS Transform custom resources. This provides complete administrative access to the service.

Permissions details

To view the policy permission details see AWSTransformCustomFullAccess in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSTransformCustomExecuteTransformations

This policy provides access to execute transformations in AWS Transform custom.

Description

This policy includes the following permissions:

- **AWS Transform Custom** – Allows streaming conversations, executing transformations, and managing campaigns. Includes permissions to get campaign details, update campaign repository status, and update campaigns.

Permissions details

To view the policy permission details see AWSTransformCustomExecuteTransformations in the AWS Managed Policy Reference Guide.

AWS managed policy: AWSTransformCustomManageTransformations

This policy provides access to create, update, read, and delete transformation resources in AWS Transform custom, as well as execute transformations.

Description

This policy includes the following permissions:

- **AWS Transform Custom** – Allows comprehensive management of transformation resources including streaming conversations, executing transformations, and managing transformation packages. Permits creating, getting, and deleting transformation package URLs and completing package uploads.
- **Knowledge Management** – Allows listing, getting, deleting, and updating knowledge items and their configurations and status.
- **Campaign Management** – Allows getting campaign details, updating campaign repository status, and updating campaigns.
- **Resource Tagging** – Allows listing, adding, and removing tags for AWS Transform custom resources.

Permissions details

To view the policy permission details see AWSTransformCustomManageTransformations in the AWS Managed Policy Reference Guide.

AWS managed policy: DBModDiscoveryAndAssessment

This policy provides comprehensive database modernization discovery and assessment capabilities for AWS Transform.

Description

This policy includes the following permissions:

- **Amazon EC2** – Allows describing infrastructure components including instances, VPCs, subnets, security groups, availability zones, VPC endpoints, and internet gateways.

- **Amazon RDS** – Allows describing database instances, clusters, and subnet groups. Allows modifying DB subnet groups within the same AWS account for migration preparation.
- **Amazon RDS Data API** – Allows enabling and disabling HTTP endpoints and executing SQL statements on database clusters tagged for the database modernization project.
- **AWS DMS** – Allows describing endpoints, replication instances, tasks, subnet groups, and orderable instances. Allows listing data providers, instance profiles, and migration projects. Allows describing table statistics, assessment runs, and metadata model operations. Detailed describe and metadata operations are restricted to resources tagged for the database modernization project.
- – Allows listing secrets for discovering database credentials.
- **AWS Identity and Access Management (IAM)** – Allows inspecting specific AWS DMS service roles and their attached policies. Includes access to read AWS-managed AWS DMS policies.
- **AWS Key Management Service (KMS)** – Allows listing key aliases and describing keys. Allows decryption of secrets through integration, restricted to the same AWS account.

The policy implements least-privilege access through resource-level permissions, tag-based conditions, and account-level restrictions to ensure operations are limited to database modernization project resources within the same AWS account.

Permissions details

To view the policy permission details see DBModDiscoveryAndAssessment in the AWS Managed Policy Reference Guide.

AWS managed policy: DBModProvisioningAndMigration

This policy provides database provisioning and migration capabilities for AWS Transform database modernization projects. It includes permissions to create and manage migration infrastructure, provision target databases, and store migration data.

Description

This policy includes the following permissions:

- **AWS DMS** – Allows creating and managing replication subnet groups, instance profiles, data providers, migration projects, endpoints, replication instances, and replication tasks. Includes schema conversion operations such as metadata model import, conversion, export, and

assessment. Allows lifecycle management of replication instances (create, delete, modify, reboot) and replication tasks (delete, start, stop, assess). All write operations are restricted to resources tagged for the database modernization project.

- **Amazon RDS** – Allows creating database subnet groups, database clusters, and database instances. All resources must be tagged for the database modernization project.
- – Allows creating, updating, and tagging secrets with the database modernization naming prefix. Allows retrieving secret values and describing secrets for tagged resources.
- **Amazon S3** – Allows creating and managing S3 buckets and objects for migration data storage. Includes bucket tagging, versioning, and object lifecycle operations. Restricted to buckets with the database modernization naming prefix.
- **AWS Identity and Access Management (IAM)** – Allows passing specific AWS DMS service roles to AWS DMS and schema conversion services. Allows creating the Amazon RDS service-linked role required for database operations.

The policy implements least-privilege access through resource-level permissions, tag-based conditions, and account-level restrictions to ensure operations are limited to database modernization project resources within the same AWS account.

Permissions details

To view the policy permission details see DBModProvisioningAndMigration in the AWS Managed Policy Reference Guide.

AWS Transform permissions reference

This section provides information about the APIs used by AWS Transform, and what they do.

Topics

- AWS Transform APIs

AWS Transform APIs

- transform:BatchGetMessage
- transform:BatchGetUserDetails
- transform:CompleteArtifactUpload
- transform:CreateArtifactDownloadUrl

- transform:CreateArtifactUploadUrl
- transform:CreateAssetDownloadUrl
- transform:CreateConnector
- transform:CreateFeedback
- transform:CreateJob
- transform:CreateSession
- transform:CreateWorkspace
- transform:DeleteConnector
- transform:DeleteJob
- transform:DeleteSelfRoleMappings
- transform:DeleteUserRoleMappings
- transform:DeleteWorkspace
- transform:DetectIsAllowedForOperation
- transform:GetConnector
- transform:GetFeedbackQuestions
- transform:GetHitlTask
- transform:GetJob
- transform:GetLoginRedirectUri
- transform:GetUserDetails
- transform:GetUserPreferences
- transform:GetWorkspace
- transform:ListArtifacts
- transform:ListConnectors
- transform:ListHitlTasks
- transform:ListJobPlanSteps
- transform:ListJobs
- transform:ListMessages
- transform:ListPlanUpdates
- transform:ListUserRoleMappings

- transform:ListWorklogs
- transform:ListWorkspaces
- transform:PutUserPreferences
- transform:PutUserRoleMappings
- transform:RevokeSession
- transform:SearchUsers
- transform:SearchUsersTypeahead
- transform:SendMessage
- transform:StartJob
- transform:StopJob
- transform:SubmitCriticalHitlTask
- transform:SubmitStandardHitlTask
- transform:TestReleaseTraitPreProd
- transform:TestReleaseTraitProd
- transform:UpdateHitlTask
- transform:UpdateJob
- transform:UpdateWorkspace
- transform:VerifySession

Compliance validation for AWS Transform

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. For more information about your compliance responsibility when using AWS services, see AWS Security Documentation.

Resilience in AWS Transform

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, AWS Transform offers several features to help support your data resiliency and backup needs.

AWS Transform and interface endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and AWS Transform by creating an *interface VPC endpoint*. Interface endpoints are powered by AWS PrivateLink, a technology that enables you to privately access the AWS Transform console without an internet gateway, NAT device, VPN connection, or Direct Connect connection. Traffic between your VPC and AWS Transform does not leave the Amazon network.

Each interface endpoint is represented by one or more Elastic Network Interfaces in your subnets.

For more information, see Interface VPC endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.

> **Note**
>
> The AWS Transform WebApp requires additional network configuration to access from a VPC. For instructions, see Accessing the AWS Transform WebApp from a VPC.

> **Note**
>
> For AWS Transform custom PrivateLink documentation, see AWS Transform custom and interface endpoints (AWS PrivateLink).

Considerations for AWS Transform VPC endpoints

Before you set up an interface VPC endpoint for AWS Transform, ensure that you review Interface endpoint properties and limitations in the *Amazon VPC User Guide*.

Prerequisites

Before you begin any of the procedures below, ensure that you have the following:

- An AWS account with appropriate permissions to create and configure resources.
- A VPC already created in your AWS account.
- Familiarity with AWS services, especially Amazon VPC and AWS Transform.

Creating an interface VPC endpoint for AWS Transform

You can create a VPC endpoint for the AWS Transform service using either the Amazon VPC console or the AWS Command Line Interface (AWS CLI). For more information, see Creating an interface endpoint in the *Amazon VPC User Guide*.

Create the following VPC endpoints for AWS Transform using this service name:

- com.amazonaws.*region*.transform

Replace *region* with AWS Region where your AWS Transform profile is installed, for example, *com.amazonaws.us-east-1.transform*.

For more information, see Supported Regions for AWS Transform and Accessing a service through an interface endpoint in the *Amazon VPC User Guide*.

Using an on-premises computer to connect to a AWS Transform endpoint

This section describes the process of using an on-premises computer to connect to AWS Transform through a AWS PrivateLink endpoint in your AWS VPC.

1. Create a VPN connection between your on-premises device and your VPC.
2. Create an interface VPC endpoint for AWS Transform.

3. Set up an inbound Amazon Route 53 endpoint. This will enable you to use the DNS name of your AWS Transform endpoint from your on-premises device.

Accessing the AWS Transform WebApp from a VPC

When you use AWS PrivateLink to access the AWS Transform API privately from your VPC, the webapp requires additional network configuration. It serves static content (HTML, JavaScript, CSS) through CloudFront, which requires internet connectivity. API calls from the webapp go through your VPC endpoint and remain fully private.

This guide shows you how to configure controlled internet egress from your VPC so the webapp can load while keeping your VPC locked down to only the required domains.

How it works

The AWS Transform WebApp uses two network paths:

- **API calls** – When you interact with the webapp (starting jobs, viewing workspaces, and so on), the browser sends API requests to `api.transform.region.on.aws`. With AWS PrivateLink configured, these requests resolve to a private IP address in your VPC and never leave the AWS network.
- **Static content** – The webapp's HTML, JavaScript, and CSS files are served through CloudFront via `tenant-id.transform.region.on.aws`. Loading these files requires internet connectivity because CloudFront content delivery is only available over the public internet.
- **Authentication** – AWS IAM Identity Center sign-in flows use `region.signin.aws`, which also requires internet connectivity.

To enable the webapp while maintaining security, you create a controlled egress path using AWS Network Firewall with domain-based filtering. This allows your VPC to reach *only* the specific domains required by the webapp while blocking all other internet traffic.

Architecture

The following diagram shows the network path for webapp traffic:

```
EC2 Instance / Workspace (Private Subnet)
```

```
    |
    | Route: 0.0.0.0/0 # Network Firewall Endpoint
    v
AWS Network Firewall (Firewall Subnet)
    | Allows: *.cloudfront.net, *.transform.<region>.on.aws,
    |         <region>.signin.aws, SSO domains, S3 presigned URLs
    | Blocks: everything else (TLS SNI inspection)
    |
    | Route: 0.0.0.0/0 # NAT Gateway
    v
NAT Gateway (Public Subnet)
    |
    | Route: 0.0.0.0/0 # Internet Gateway
    v
Internet Gateway # CloudFront Edge Locations
```

Important

The Network Firewall must see both directions of traffic (symmetric routing) to perform TLS SNI inspection. You must configure a return route in the NAT Gateway subnet that sends traffic destined for the private subnet back through the firewall. Without this, domain-based filtering rules will not work.

Prerequisites

Before you begin, ensure you have:

- An AWS account with permissions to create VPC resources, Network Firewall, and NAT Gateways.
- A VPC with a private subnet where your instances or workloads run.
- An internet gateway attached to the VPC (or permissions to create one).
- A AWS Transform VPC endpoint configured. For instructions, see AWS Transform and interface endpoints (AWS PrivateLink).

Setting up controlled internet egress

Complete the following steps to configure Network Firewall with domain-based filtering for the AWS Transform WebApp.

Step 1: Create the firewall subnet

Create a small /28 subnet dedicated to the Network Firewall endpoint. This subnet must be in the same Availability Zone as your private subnet.

```
aws ec2 create-subnet \
  --vpc-id your-vpc-id \
  --cidr-block firewall-subnet-cidr \
  --availability-zone your-az \
  --region region
```

Step 2: Create a public subnet for the NAT Gateway

```
aws ec2 create-subnet \
  --vpc-id your-vpc-id \
  --cidr-block public-subnet-cidr \
  --availability-zone your-az \
  --region region
```

Step 3: Configure the public subnet route table

Create a route table for the public subnet that routes internet traffic to the internet gateway.

```
# Create route table
PUB_RTB=$(aws ec2 create-route-table \
  --vpc-id your-vpc-id \
  --region region \
  --query 'RouteTable.RouteTableId' --output text)

# Add default route to internet gateway
aws ec2 create-route \
  --route-table-id $PUB_RTB \
  --destination-cidr-block 0.0.0.0/0 \
  --gateway-id your-igw-id \
  --region region

# Associate with public subnet
```

```
aws ec2 associate-route-table \
  --route-table-id $PUB_RTB \
  --subnet-id public-subnet-id \
  --region region
```

Step 4: Create the NAT Gateway

```
# Allocate an Elastic IP
EIP=$(aws ec2 allocate-address --domain vpc \
  --region region --query 'AllocationId' --output text)

# Create NAT Gateway in the public subnet
NAT_ID=$(aws ec2 create-nat-gateway \
  --subnet-id public-subnet-id \
  --allocation-id $EIP \
  --region region \
  --query 'NatGateway.NatGatewayId' --output text)

# Wait for NAT Gateway to become available (~2 minutes)
aws ec2 wait nat-gateway-available \
  --nat-gateway-ids $NAT_ID --region region
```

Step 5: Create the Network Firewall rule group

Create a stateful rule group that allows traffic only to the domains required by the AWS Transform WebApp.

```
aws network-firewall create-rule-group \
  --rule-group-name transform-webapp-domains \
  --type STATEFUL \
  --capacity 100 \
  --rule-group '{
    "StatefulRuleOptions": {
      "RuleOrder": "STRICT_ORDER"
    },
    "RulesSource": {
      "RulesSourceList": {
        "Targets": [
```

```
          ".cloudfront.net",
          ".transform.region.on.aws",
          "region.signin.aws",
          ".s3.region.amazonaws.com",
          "oidc.region.amazonaws.com",
          "portal.sso.region.amazonaws.com",
          "assets.sso-portal.region.amazonaws.com",
          "directory-id.awsapps.com"
        ],
        "TargetTypes": ["TLS_SNI", "HTTP_HOST"],
        "GeneratedRulesType": "ALLOWLIST"
      }
    }
}' \
--region region
```

Replace *region* with the AWS Region where your AWS Transform profile is installed (for example, us-east-1). Replace *directory-id* with your IAM Identity Center directory ID (for example, d-1234567890). You can find your directory ID in the IAM Identity Center console.

The following table explains the allowed domains.

Domain	Purpose
.cloudfront.net	CloudFront CDN – serves webapp static assets (JavaScript, CSS, images)
.transform. *region*.on.aws	Webapp tenant URL – the browser loads the initial page from this domain via CloudFront
region.signin.aws	SSO sign-in redirect page
.s3.*region*.amazonaws.com	S3 presigned URLs – artifact uploads and downloads
oidc.*region*.amazonaws.com	OIDC token exchange for SSO authentication
portal.sso. *region*.amazonaws.com	SSO portal login page

Domain	Purpose
`assets.sso-portal.region.amazonaws.com`	SSO portal static assets (CSS, JavaScript)
`directory-id .awsapps.com`	IAM Identity Center portal for your organization

Note

The `.cloudfront.net` wildcard allows traffic to any CloudFront distribution, not only the AWS Transform WebApp's. A narrower domain filter is not possible because CloudFront edge IPs are shared across distributions and TLS SNI inspection cannot distinguish individual distributions behind the same domain.

Note

API calls to `api.transform.region.on.aws` go through AWS PrivateLink and do not require internet egress. They are not affected by the firewall.

Step 6: Create the firewall policy

The policy must use `STRICT_ORDER` rule evaluation with `drop_established` as the default action. This ensures that any traffic not matching the allowlist is dropped.

```
# Get the rule group ARN
RG_ARN=$(aws network-firewall describe-rule-group \
  --rule-group-name transform-webapp-domains \
  --type STATEFUL --region region \
  --query 'RuleGroupResponse.RuleGroupArn' --output text)

# Create the firewall policy
aws network-firewall create-firewall-policy \
  --firewall-policy-name transform-webapp-policy \
  --firewall-policy "{
    \"StatelessDefaultActions\": [\"aws:forward_to_sfe\"],
    \"StatelessFragmentDefaultActions\": [\"aws:forward_to_sfe\"],
```

```
    \"StatefulRuleGroupReferences\": [
      {
        \"ResourceArn\": \"$RG_ARN\",
        \"Priority\": 1
      }
    ],
    \"StatefulEngineOptions\": {
      \"RuleOrder\": \"STRICT_ORDER\"
    },
    \"StatefulDefaultActions\": [\"aws:drop_established\", \"aws:alert_established\"]
  }" \
  --region region
```

Step 7: Create the Network Firewall

```
# Get the policy ARN
FW_POLICY_ARN=$(aws network-firewall describe-firewall-policy \
  --firewall-policy-name transform-webapp-policy \
  --region region \
  --query 'FirewallPolicyResponse.FirewallPolicyArn' --output text)

# Create the firewall
aws network-firewall create-firewall \
  --firewall-name transform-webapp-firewall \
  --firewall-policy-arn $FW_POLICY_ARN \
  --vpc-id your-vpc-id \
  --subnet-mappings SubnetId=firewall-subnet-id \
  --region region

# Wait for the firewall to become READY (3-5 minutes)
while true; do
  STATUS=$(aws network-firewall describe-firewall \
    --firewall-name transform-webapp-firewall \
    --region region \
    --query 'FirewallStatus.Status' --output text)
  echo "Status: $STATUS"
  if [ "$STATUS" = "READY" ]; then break; fi
  sleep 15
done
```

Step 8: Get the firewall endpoint ID

```
FW_ENDPOINT=$(aws network-firewall describe-firewall \
  --firewall-name transform-webapp-firewall \
  --region region \
  --query "FirewallStatus.SyncStates.\"your-az\".Attachment.EndpointId" \
  --output text)
echo "Firewall endpoint: $FW_ENDPOINT"
```

Step 9: Configure the firewall subnet route table

Route internet-bound traffic from the firewall subnet to the NAT Gateway.

```
FW_RTB=$(aws ec2 create-route-table \
  --vpc-id your-vpc-id \
  --region region \
  --query 'RouteTable.RouteTableId' --output text)

aws ec2 create-route \
  --route-table-id $FW_RTB \
  --destination-cidr-block 0.0.0.0/0 \
  --nat-gateway-id $NAT_ID \
  --region region

aws ec2 associate-route-table \
  --route-table-id $FW_RTB \
  --subnet-id firewall-subnet-id \
  --region region
```

Step 10: Update the private subnet route table

Route all internet-bound traffic from the private subnet through the firewall.

```
aws ec2 create-route \
  --route-table-id private-subnet-route-table-id \
  --destination-cidr-block 0.0.0.0/0 \
```

```
  --vpc-endpoint-id $FW_ENDPOINT \
  --region region
```

If a default route already exists, use `replace-route` instead of `create-route`.

Step 11: Add the symmetric return route (required)

> **Important**
>
> This step is critical. Network Firewall uses TLS SNI inspection and must see both directions of a TCP connection. Add a route in the NAT Gateway subnet's route table that sends return traffic destined for the private subnet back through the firewall.

```
aws ec2 create-route \
  --route-table-id $PUB_RTB \
  --destination-cidr-block private-subnet-cidr \
  --vpc-endpoint-id $FW_ENDPOINT \
  --region region
```

Replace *private-subnet-cidr* with the CIDR block of your private subnet (for example, `10.0.144.0/20`).

Without symmetric routing, the firewall only sees one direction of traffic. The TLS inspection engine cannot extract the Server Name Indication (SNI) from the TLS handshake, and all domain-based rules will fail silently.

Step 12: Remove the IPv6 default route (if present)

If the private subnet route table has an IPv6 default route (`::/0`) pointing directly to an internet gateway, it will bypass the firewall for IPv6-capable destinations. Remove it:

```
aws ec2 delete-route \
  --route-table-id private-subnet-route-table-id \
  --destination-ipv6-cidr-block ::/0 \
```

```
  --region region
```

Verification

From an instance in the private subnet, verify the configuration:

```
# Should SUCCEED - webapp content via CloudFront (allowed)
curl -vL --connect-timeout 15 \
  'https://tenant-id.transform.region.on.aws'

# Should SUCCEED - API via PrivateLink (does not use firewall)
curl -v --connect-timeout 15 \
  'https://api.transform.region.on.aws/'

# Should FAIL - non-allowlisted domain (blocked by firewall)
curl -v --connect-timeout 15 'https://www.example.com'
# Expected: TLS connection error (firewall drops after SNI inspection)
```

Troubleshooting

The webapp does not load (connection timeout)

- Verify the private subnet route table has a `0.0.0.0/0` route pointing to the firewall endpoint.
- Verify the firewall subnet route table has a `0.0.0.0/0` route pointing to the NAT Gateway.
- Verify the NAT Gateway is in an `available` state.

Non-allowlisted domains are not blocked

- Check for an IPv6 `::/0` route pointing to the internet gateway. This bypasses the firewall. Remove it (Step 12).
- Verify symmetric routing is configured. The NAT Gateway subnet route table must have a return route through the firewall for the private subnet CIDR (Step 11).
- Verify the firewall policy uses `STRICT_ORDER` with `drop_established` and `alert_established` as default actions.

Cost considerations

Resource	Approximate cost
Network Firewall	~$0.395/hr (~$288/month) per Availability Zone
NAT Gateway	~$0.045/hr (~$33/month) + data processing fees
Elastic IP (public IPv4)	~$0.005/hr (~$3.60/month)
Network Firewall data processing	$0.065/GB
NAT Gateway data processing	$0.045/GB

Estimated base cost is approximately $325/month for a single Availability Zone deployment. For production deployments, deploy the firewall, NAT Gateway, and associated subnets in each Availability Zone where private subnets exist.

Cleanup

To remove all resources created by this guide, run the following commands in reverse order. Replace the placeholders with the resource IDs from your deployment. You can find these values in the AWS Management Console or by using the `describe` commands from the setup steps.

```
# Remove return route from NAT Gateway subnet
aws ec2 delete-route --route-table-id pub-rtb-id \
  --destination-cidr-block private-subnet-cidr \
  --region region

# Remove route from private subnet
aws ec2 delete-route \
  --route-table-id private-subnet-route-table-id \
  --destination-cidr-block 0.0.0.0/0 --region region

# Delete Network Firewall (takes ~5 minutes)
aws network-firewall delete-firewall \
  --firewall-name transform-webapp-firewall \
```

```
  --region region

# Delete firewall policy and rule group
aws network-firewall delete-firewall-policy \
  --firewall-policy-name transform-webapp-policy \
  --region region
aws network-firewall delete-rule-group \
  --rule-group-name transform-webapp-domains \
  --type STATEFUL --region region

# Delete NAT Gateway (wait ~5 minutes for full deletion)
aws ec2 delete-nat-gateway --nat-gateway-id nat-gateway-id \
  --region region

# Release Elastic IP
aws ec2 release-address --allocation-id eip-allocation-id \
  --region region

# Delete subnets
aws ec2 delete-subnet --subnet-id firewall-subnet-id \
  --region region
aws ec2 delete-subnet --subnet-id public-subnet-id \
  --region region
```

Monitoring AWS Transform

Monitoring is an important part of maintaining the reliability, availability, and performance of AWS Transform and your other AWS solutions. AWS provides the following monitoring tools to watch AWS Transform, report when something is wrong, and take automatic actions when appropriate:

- *Amazon CloudWatch* monitors your AWS resources and and the applications you run on AWS in real time. You can collect and track metrics, create customized dashboards, and set alarms that notify you or take actions when a specified metric reaches a threshold that you specify. For example, you can have CloudWatch track CPU usage or other metrics of your Amazon EC2 instances and automatically launch new instances when needed. For more information, see the Amazon CloudWatch User Guide.
- *Amazon CloudWatch Logs* enables you to monitor, store, and access your log files from Amazon EC2 instances, CloudTrail, and other sources. CloudWatch Logs can monitor information in the log files and notify you when certain thresholds are met. You can also archive your log data in highly durable storage. For more information, see the Amazon CloudWatch Logs User Guide.
- *Amazon EventBridge* can be used to automate your AWS services and respond automatically to system events, such as application availability issues or resource changes. Events from AWS services are delivered to EventBridge in near real time. You can write simple rules to indicate which events are of interest to you and which automated actions to take when an event matches a rule. For more information, see Amazon EventBridge User Guide.
- *AWS CloudTrail* captures API calls and related events made by or on behalf of your AWS account and delivers the log files to an Amazon S3 bucket that you specify. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. For more information, see the AWS CloudTrail User Guide.

Logging AWS Transform API calls using AWS CloudTrail

AWS Transform is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in AWS Transform. CloudTrail captures all API calls for AWS Transform as events. The calls captured include calls from the AWS Transform console and code calls to the AWS Transform API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for AWS Transform. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was

made to AWS Transform, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

AWS Transform information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in AWS Transform, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing events with CloudTrail Event history.

For an ongoing record of events in your AWS account, including events for AWS Transform, create a trail. A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for creating a trail
- CloudTrail supported services and integrations
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple regions and Receiving CloudTrail log files from multiple accounts

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or AWS Identity and Access Management (IAM) user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity element.

Understanding AWS Transform log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

Notifications

AWS Transform provides email notifications to keep you informed about workspace access changes, job status updates, and pending collaborator requests.

Email notifications

Email notifications are enabled by default, and are sent to users who have read-only or higher permissions on a job. Users can modify their notification preferences. There are three categories of email notifications:

- **Workspace access updates**

  You receive a notification when you are added to a new workspace, and when your role is changed within a workspace.
- **Daily digest emails**

  You receive a daily digest email at 9:00 AM Pacific time summarizing your ongoing jobs if you have active jobs with one or more open required collaborator requests. The email includes a table showing workspace name, job name, job status, collaborator request details, and required actions (limited to 10 rows)

  > **Note**
  >
  > Daily digest emails are only sent for jobs that have required collaborator requests. Optional collaborator requests do not trigger digest emails.
- **Job status updates**

  You receive an email when a job you have access to completes or fails.
- **Collaborator request updates**

You receive an email whenever a task requires your input, review, or approval.

Managing email notifications

You can modify your email notification preferences in the web application by clicking the **Settings** cog in the top right corner of the application, and choosing **Notification settings**.

The notification settings page provides the following controls:

Notification Type	Description	Default Setting
Workspace Access Updates	Receive emails when added to a workspace, or when your role is changed within a workspace	Enabled
Daily Collaborator Request Digest	Receive a daily digest of open collaborator requests for jobs in accessible workspaces	Enabled
Job Completion Updates	Receive emails when a job is completed or fails	Enabled
Collaborator Request Updates	Receive emails when an input or approval is required from you	Disabled

You can use a general setting to enable or disable all notifications, or configure individual notification types separately.

Quotas for AWS Transform

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas cannot be increased.

To view the quotas for AWS Transform, open the Service Quotas console. In the navigation pane, choose **AWS services** and select **AWS Transform**.

To request a quota increase, see Requesting a Quota Increase in the *Service Quotas User Guide*. If the quota is not yet available in Service Quotas use the limit increase form.

Your AWS account has the following quotas related to AWS Transform.

Workload	Dimension	Quota	Adjustable	Description
Platform	Workspaces per AWS Region per AWS Transform-enabled account	100 workspaces for each supported AWS Region	Yes	The maximum number of workspaces per AWS Transform-enabled account per AWS Region
VMWare Migration	Server count for each wave	Each wave: 150 servers	No	The maximum number of servers in a wave
Mainframe	Concurrent jobs for each IAM Identity Center account	10 jobs for each supported Region	Yes	The maximum number of concurrent Mainframe jobs in an IAM Identity Center account
	Lines of code for each job	3,000,000 lines of code for	No	The maximum number of lines

Workload	Dimension	Quota	Adjustable	Description
		each supported region		of code for each job.
	Analysis monthly lines of code	100,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Analyze Code job objective in a calendar month
	Business document generation monthly lines of code	50,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Business Documentation Generation job objective in a calendar month
	Technical document generation monthly lines of code	50,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Technical Documentation Generation job objective in a calendar month
	Transform of lines of code	10,000,000 lines of code for each supported region	Yes	The maximum number of lines of code transformed in a calendar month

Workload	Dimension	Quota	Adjustable	Description
	Decomposition monthly lines of code	50,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Decomposition job objective in a calendar month
	Refactor monthly lines of code	50,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Refactor job objective in a calendar month
	Reforge monthly lines of code	50,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Reforge job objective in a calendar month
	Extracted business logic monthly lines of code	50,000,000 lines of code for each supported region	Yes	The maximum number of lines of code for Extracted Business Logic job objective in a calendar month

Workload	Dimension	Quota	Adjustable	Description
.NET	.NET monthly lines of code (web & IDE)	1,000,000 lines of code for each supported Region	Yes	The maximum lines of .NET application code that can be transform ed in a calendar month in each supported AWS Region of an IAM Identity Center account
	Concurrent .NET jobs with the Microsoft Visual Studio Extension for each IAM Identity Center account	10 jobs for each supported Region	Yes	The maximum number of concurrent .NET transformation jobs in each supported AWS Region of an IAM Identity Center account when using the Microsoft Visual Studio Extension .

Workload	Dimension	Quota	Adjustable	Description
	Concurrent .NET web jobs for each IAM Identity Center account	5 jobs for each supported Region	Yes	The maximum number of concurrent .NET transformation jobs in each supported AWS Region of an IAM Identity Center account when using the AWS Transform web app.
Custom transformations	Custom transformation definitions	1,000	Yes	The maximum number of custom transformation definitions
	Custom transformation definition size	10 MB	No	The maximum total size of all files included in the transformation definition, including document references.

Workload	Dimension	Quota	Adjustable	Description
	Custom transformation monthly CLI conversation requests	1,000,000	Yes	The maximum number of custom transformation conversation requests made in the CLI allowed in a calendar month
Assessment	Server count limit for each assessment	Each assessment: 30,000 servers	No	The maximum number of servers in a single assessment job

Supported Regions for AWS Transform

> **Note**
>
> If you make a request that requires AWS Transform to retrieve information from an opt-in Region not listed on this page, AWS Transform can make calls to that Region. To manage access to Regions AWS Transform can make calls to, see Security in AWS Transform.

This topic describes the AWS Regions where you can use AWS Transform. For more information about AWS Regions, see Specify which AWS Regions your account can use in the *AWS Account Management Reference Guide*.

Your data might be processed in a different Region from the Region where you use AWS Transform. For information on cross-region processing in AWS Transform, see Cross-region processing. For information on where data is stored during processing, see Data protection.

Supported AWS Regions (enabled by default)

You can create AWS Transform workspaces in the following AWS Regions. These Regions are enabled by default - you don't need to enable them before use. For more information, see Regions that are enabled by default.

- US East (N. Virginia)
- Europe (Frankfurt)
- Asia Pacific (Mumbai)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Europe (London)
- Asia Pacific (Seoul)
- Canada (Central)

The workspace in which you create a job determines the AWS Region of the job. To create a job in a different Region, you must use a different workspace that is in your desired Region.

For VMware projects, the Region of the workspace is used for discovery. However, you can specify a different Region as the migration target. That target Region is where your workloads are hosted when the migration is complete. For more information about AWS Region considerations for VMware migrations, including the list of possible target Regions, see the section called “AWS account connectors”.

AWS support for AWS Transform

AWS Transform enables you to create and manage AWS Support cases through the chat if you encounter issues during your transformation jobs. Your cases are automatically enriched with job context and routed to the appropriate support team. If you are an active Countdown Premium customer, your designated engineer will reach out to manage issue resolution. Learn more in AWS Countdown Premium. Otherwise, your case will be investigated by the AWS Transform support team in their support queue.

Prerequisites

Before you can use support case management, you need:

- An active AWS Support plan - Business, Enterprise On-Ramp, or Enterprise
- An existing AWS Transform application

Enabling Support Case Management

1. Open the AWS Transform console.
2. In the navigation pane, choose **Settings**.
3. Under **Support**, choose **Enable Support case management**.
4. Choose **Save changes**.

AWS Transform uses a service-linked role (SLR) to create and manage support cases on your behalf. No additional IAM permissions are required.

Creating a Support Case

To create a support case, use the chat interface:

1. Ask to create a support case. For example:
   - "Create a support case for my failed transformation job"
   - "I need help with job errors, create a case"
2. The chat prompts you for required information:

- A brief description of your issue
- A detailed description

3. Confirm the case creation when prompted.

The case is automatically populated with context from your transformation job, including:

- Job ID and status
- Job metadata
- Job execution logs (work logs)
- Job plan

The case is automatically assigned based on your job type and issue category.

Viewing Support Cases

To view your support cases, prompt the chat:

- "List my support cases"
- "Show me open support cases"
- "What's the status of my cases?"

The chat will display:

- Case ID
- Subject
- Status, such as Open, Pending, Resolved, or Closed
- Last updated date
- Associated job ID

To view details of a specific case:

- "Show me details for case [case-id]"
- "What's the latest update on case [case-id]?"

Adding Communications to a Case

To add a message to an existing case:

1. Ask the chat: "Add a message to case [case-id]"
2. Provide your message when prompted.
3. The chat will confirm the message was added.

You'll receive notifications when AWS Support responds to your case.

Resolving a Case

To resolve a support case:

1. Ask the chat: "Resolve case [case-id]"
2. Confirm the resolution when prompted.

> **Note**
>
> You can only resolve cases that have been addressed by AWS Support.

Disabling Support Case Management

1. Open the AWS Transform console.
2. In the navigation pane, choose **Settings**.
3. Under **Support**, choose **Disable Support case management**.
4. Choose **Save changes**.

> **Note**
>
> Disabling Support case management removes the chat-based case management interface but does not close existing cases. You can still access your cases through the AWS Support Center.

Document history for the AWS Transform User Guide

The following table describes the document history for the AWS Transform User Guide.

Change	Description	Date
Added KMS key policy for database migration	Added Encryption for database migration section.	March 26, 2026
Documented new AWS managed policy	Added DBModProvisioningAndMigration section.	March 24, 2026
Documented new AWS managed policy	Added DBModDiscoveryAndAssessment section.	March 24, 2026
Added support documentation	Added AWS support for section.	December 23, 2025
Documented new AWS managed policies	Added AWSTransformCustomFullAccess, AWSTransformCustomExecuteTransformations, and AWSTransformCustomManageTransformations sections.	December 7, 2025
VMware migration	Updated the VMware migration section.	December 1, 2025
Updated AWS managed policy	Updated the AWSServiceRoleForAWSTransform documentation to reflect added permissions.	December 1, 2025
Supported regions	Updated the Supported regions topic.	December 1, 2025

Mainframe modernization	Updated the Mainframe modernization section.	December 1, 2025
Getting Startedy	Updated the Getting Started topic.	December 1, 2025
Full-stack Windows modernization	Added the Full-stack Windows modernization topic, which includes the updated .NETsection and the new SQL Server modernization section.	December 1, 2025
Discovery tool	Added the Discovery tool section.	December 1, 2025
Updated AWS managed policy	Updated the AWSTransformApplicationECSDeploymentPolicy with expanded IAM role inspection permissions, ECS service-linked role creation, and KMS permissions for ECR encryption support.	November 22, 2025
Updated AWS managed policy	Updated the AWSTransformApplicationDeploymentPolicy with additional EC2 networking, IAM role inspection, S3 bucket management, and KMS encryption permissions.	November 22, 2025
Discovery collector	Added the AWS Transform discovery tool> documentation.	October 31, 2025

VMware migration architecture diagram added	Added a VMware migration architecture diagram in VMware transformation documentation.	October 23, 2025
VMware migration results update	Updated migration results information in VMware transformation documentation.	October 12, 2025
VMware migration results update	Updated migration results information in VMware transformation documentation.	October 12, 2025
S3 connector documentation	Added S3 connector documentation for .NET web app.	October 8, 2025
New user connectors topic	Added user connectors topic.	September 21, 2025
Updated AWS managed policy	Updated the AWSServiceRoleForAWSTransform documentation.	September 17, 2025
Storage assessment support	Added NetApp Data Infrastructure Insights (DII) file support for storage assessments.	September 2, 2025
Target regions update	Added three supported target regions.	August 31, 2025
Managed policy update	Added IAM Identity Center read-only permisions to be used by AWSServiceRoleForAWSTransform.	August 29, 2025

Documented new AWS managed policy	Added AWSTransformApplicationDeploymentPolicy section.	August 28, 2025
interface endpoints	Added interface endpoints topic.	July 9, 2025
Initial release	Initial release of the AWS Transform User Guide.	May 15, 2025