AMS Advanced Application Developer's Guide

AMS Advanced Application Deployment Options

Version September 30, 2020

AMS Advanced Application Developer's Guide: AMS Advanced Application Deployment Options

Table of Contents

Application Onboarding to AMS Introduction

Welcome to AWS Managed Services (AMS), AMS operations plan. The purpose of this document is to describe the various methods you can use when onboarding your applications to AMS once initial networking and access management has been set up, and the issues you should consider when choosing those methods.

This document is intended for system integrators and application developers to assist in determining and crafting application processes for new AMS customers.

What is Application Onboarding?

AMS application onboarding refers to the deployment of resources and applications, as needed, into your AMS infrastructure. Architecting applications and infrastructure on the AMS platform is very similar to doing so on native AWS. Following AWS application and infrastructure design best practices while considering the capabilities that are provided by AMS will yield capable and operable applications hosted in the AMS environment.

Note

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

New regions are added frequently. To learn more, see AWS regions and availability zones.

What we do, what we do not do

AMS gives you a standardized approach to deploying AWS infrastructure and provides the necessary ongoing operational management. For a full description of roles, responsibilities, and supported services, see Service Description.

Note
To request that AMS provide an additional AWS service, file a service request. For more information, see Making Service Requests.

- **What we do**:

  After you complete onboarding, the AMS environment is available to receive requests for change (RFCs), incidents, and service requests. Your interaction with the AMS service revolves around the lifecycle of an application stack. New stacks are ordered from a preconfigured list of templates, launched into specific virtual private cloud (VPC) subnets, modified during their operational life through requests for change (RFCs), and monitored for events and incidents 24/7.

  Active application stacks are monitored and maintained by AMS, including patching, and require no further action for the life of the stack unless a change is required or the stack is decommissioned. Incidents detected by AMS that affect the health and function of the stack generate a notification and may or may not need your action to resolve or verify. How-to questions and other inquiries can be made by submitting a service request.

  Additionally, AMS allows you to enable compatible AWS services that are not managed by AMS. For information about AWS-AMS compatible services, see Self-service provisioning mode.

- **What we DON'T do**:

  While AMS simplifies application deployment by providing a number of manual and automated options, you're responsible for application development, deployment, testing and tuning, and management. AMS provides troubleshooting assistance for infrastructure issues that impact applications, but AMS can't access or validate your application configurations.

AMS Amazon Machine Images (AMIs)

AMS produces updated Amazon Machine Images (AMIs) every month for a variety of operating systems. The AMS AMIs are based on updated Amazon Machine Images that are modified for AMS.

To receive alerts when new AMS AMIs are released, you can subscribe to an Amazon Simple Notification Service (Amazon SNS) notification topic called "AMS AMI". For details, see AMS AMI notifications with SNS.

The AMS AMI naming convention is: `customer-ams-<operating system>-<release date> - <version>`. (for example, `customer-ams-rhel6-2018.11-3`)

Only use AMS AMIs that start with `customer`.

AMS recommends always using the most recent AMI. You can find the most recent AMIs by either:

- Viewing the latest AMS AMI CSV file, available from your CSDM.
- Running this AMS `SKMS` command:

```
aws amsskms list-amis --vpc-id VPC_ID --query "Amis.sort_by(@,&Name)[?
 starts_with(Name,`customer`)].[Name,AmiId,CreationTime]" --output table
```

- Looking in the AMS console, on the **AMIs** page.

AMS AMI content added to base AWS AMIs, by operating system (OS)

- Linux AMIs:
  - AWS CLI Tools

  - NTP
  - Trend Micro Endpoint Protection Service Agent
  - Code Deploy
  - PBIS / Beyond Trust AD Bridge
  - SSM Agent
  - Yum Upgrade for critical patches
  - AMS custom scripts / management software (controlling boot, AD join, monitoring, security, and logging)
- Windows Server AMIs:
  - Microsoft .NET Framework 4.5
  - PowerShell 5.1
  - AWS Tools for Windows PowerShell
  - AMS PowerShell Modules controlling boot, AD join, monitoring, security, and logging
  - Trend Micro Endpoint Protection Service Agent
  - SSM Agent
  - CloudWatch Agent
  - EC2Config service (through Windows Server 2012 R2)
  - EC2Launch (Windows Server 2016 and later)

Linux-based AMIs:

- Amazon Linux 2 (Latest Minor Release)
- Amazon Linux (Latest 2018.03 Release)
- Red Hat Enterprise 7 (Latest Minor Release)
- Red Hat Enterprise 8 (Latest Minor Release)
- SUSE Linux Enterprise Server 12 SP4
- SUSE Linux Enterprise Server 15 SP1
- CentOS 7 (Latest Minor Release)

> **Note**
> To use the CentOS AMIs, you must opt in to the no cost Cent OS license from the AWS Marketplace. To do this, go to AWS Marketplace and follow the instructions for opting in.

- Amazon Linux: For product overview, pricing information, usage information, and support information, see Amazon Linux AMI (HVM / 64-bit) and Amazon Linux 2.

  For more information, see Amazon Linux 2 FAQs.
- RedHat Enterprise Linux (RHEL): For product overview, pricing information, usage information, and support information, see Red Hat Enterprise Linux (RHEL) 7 (HVM).
- CentOS: To use the CentOS AMIs, you must opt in to the no cost Cent OS license from the AWS Marketplace. To do this, go to AWS Marketplace and follow the instructions for opting, or re-opting, in. You do not incur software charges for using this product, but you are responsible for other AWS charges, including EC2 usage.
- SUSE Linux Enterprise Server for SAP applications 15:
  - Run the following steps once per account:
    1. Navigate to the **AWS Marketplace**.
    2. Search for the respective SUSE 15 SAP product. We currently support:
       - SUSE Linux Enterprise Server for SAP Applications 15

    - SUSE Linux Enterprise Server for SAP Applications 15 SP1
  3. Click **Continue to subscribe**.
  4. Click **Accept terms**.
- Complete the following steps **every time** you need to launch a new **SUSE Linux Enterprise Server for SAP Applications 15** instance:
  1. Note the AMI ID for the subscribed **SUSE Linux Enterprise Server for SAP Applications 15** AMI.
  2. Create a manual (Management | Other | Other | Create) RFC with the following wording; replace *`AMI ID`* with the AWS Marketplace AMI ID you have subscribed to.

Windows-based AMIs:

Microsoft Windows Server (2012, 2012 R2, 2016, and 2019), based on latest Windows AMIs, (for example: Windows_Server-2012-R2_RTM-English-64Bit-Base-*).

Additional information: For product overview, pricing, usage, and support, see Microsoft Windows Server 2012 RTM.

For examples of creating AMIs, see Create AMI.

For details on the latest AMS AMIs, see the AMS Release Notes, Latest AMIs section.

Security enhanced AMIs

This section has been redacted because it contains sensitive AMS security-related information. This information is available through the AMS console **Documentation**. To access AWS Artifact, you can contact your CSDM for instructions or go to Getting Started with AWS Artifact.

The security settings spreadsheets are available through AWS Artifact, with an AMS account.

Key terms

- *AMS Advanced*: The services described in the "Service Description" section of the AMS Advanced Documentation. See Service Description.
- *AMS Advanced Accounts*: AWS accounts that at all times meet all requirements in the AMS Advanced Onboarding Requirements. For information on AMS Advanced benefits, case studies, and to contact a sales person, see AWS Managed Services.
- *AMS Accelerate Accounts*: AWS accounts that at all times meet all requirements in the AMS Accelerate Onboarding Requirements. See Getting Started with AMS Accelerate.
- *AWS Managed Services*: AMS and or AMS Accelerate.
- *AWS Managed Services Accounts*: the AMS Accounts and or AMS Accelerate Accounts.
- *Customer-Requested Configuration*: Any software, services or other configurations that are not identified in:
  - Accelerate: Supported Configurations or AMS Accelerate; Service Description.
  - AMS Advanced: Supported Configurations or AMS Advanced; Service Description.
- *Incident Communication*: AMS communicates an Incident to you or you request an Incident with AMS via an Incident created in Support Center for AMS Accelerate and in the AMS Console for AMS. The AMS Accelerate Console provides a summary of Incidents and Service Requests on the Dashboard and links to Support Center for details.
- *Managed Environment*: The AMS Advanced accounts and or the AMS Accelerate accounts operated by AMS.
- *Billing start date*: AWS Managed Services accounts are activated once you have granted access to AMS to a compatible account and AMS Activation notification occurs as defined in the AWS Managed

Services Documentation. If the activation of the AWS Managed Services accounts, Add-on Service Request, or Account tier Service Request is received by AWS on or prior to the 20th day of the month, then the change will be effective as of the first day of the calendar month following the AMS Activation notification or such Service Request. If the activation or Service Request is received by AWS after the 20th day of the month, then the change will be effective as of the first day of the second calendar month following AMS Activation notification or such Service Request.

AMS Activation Notification to the customer occurs when:

1. Customer grants access to a compatible AWS account and hands it over to AWS Managed Services.
2. AWS Managed Services designs and builds the AWS Managed Services Account.

- *Service Termination Date*: The last day of the calendar month in which the Customer provides the AMS Account Service Termination Request, or the last day of the calendar month following the end of the requisite notice period; provided that, if the Customer provides the AMS Account Service Termination Request after the 20th day of the calendar month, the Service Termination Date will be the last day of the calendar month following the calendar month that such AMS Account Service Termination Request was provided.
- *Provision of AWS Managed Services*: AWS will make available to Customer and Customer may access and use AWS Managed Services for each AWS Managed Services Account from the Service Commencement Date.
- *Termination for specified AWS Managed Services Accounts*: Customer may terminate the AWS Managed Services for a specified AWS Managed Services Account for any reason by providing AWS notice via a Service Request ("AMS Account Termination Request").
- *Effect of Termination of specified AWS Managed Services Accounts.*: On the Service Termination Date, AWS will (i) hand over the controls of all AMS Accounts or the specified AMS Account, as applicable, to Customer, or (ii) the parties will remove the AWS Identity and Access Management roles that give AWS access from all AMS Accelerate Accounts or the specified AMS Accelerate Account, as applicable.

Incident management terms:

- *Event*: A change in your AMS environment.
- *Alert*: Whenever an event from a supported AWS service exceeds a threshold and triggers an alarm, an alert is created and notice is sent to your contacts list. Additionally, an incident is created in your Incident list.
- *Incident*: An unplanned interruption or performance degradation of your AMS environment or AWS Managed Services that results in an impact as reported by AWS Managed Services or you.
- *Problem*: A shared underlying root cause of one or more incidents.
- *Incident Resolution* or *Resolve an Incident*:
  - AMS has restored all unavailable AMS services or resources pertaining to that incident to an available state, or
  - AMS has determined that unavailable stacks or resources cannot be restored to an available state, or
  - AMS has initiated an infrastructure restore authorized by you.
- *Incident Response Time*: The difference in time between when you create an incident, and when AMS provides an initial response by way of the console, email, service center, or telephone.
- *Incident Resolution Time*: The difference in time between when either AMS or you creates an incident, and when the incident is resolved.
- *Incident Priority*: How incidents are prioritized by AMS, or by you, as either Low, Medium, or High.
  - *Low*: A non-critical problem with your AMS service.
  - *Medium*: An AWS service within your managed environment is available but is not performing as intended (per the applicable service description).
  - *High*: Either (1) the AMS Console, or one or more AMS APIs within your managed environment are unavailable; or (2) one or more AMS stacks or resources within your managed environment are unavailable and the unavailability prevents your application from performing its function.

AMS may re-categorize incidents in accordance with the above guidelines.

- *Infrastructure Restore*: Re-deploying existing stacks, based on templates of impacted stacks, and initiating a data restore based on the last known restore point, unless otherwise specified by you, when incident resolution is not possible.

Infrastructure terms:

- *Managed production environment*: A customer account where the customer’s production applications reside.
- *Managed non-production environment*: A customer account that only contains non-production applications, such as applications for development and testing.
- *AMS stack*: A group of one or more AWS resources that are managed by AMS as a single unit.
- *Immutable infrastructure*: An infrastructure maintenance model typical for EC2 Auto Scaling groups (ASGs) where updated infrastructure components, (in AWS, the AMI) are replaced for every deployment, rather than being updated in-place. The advantages to immutable infrastructure is that all components stay in a synchronous state since they are always generated from the same base. Immutability is independent of any tool or workflow for building the AMI.
- *Mutable infrastructure*: An infrastructure maintenance model typical for stacks that are not EC2 Auto Scaling groups and contain a single instance or just a few instances. This model most closely represents traditional, hardware-based, system deployment where a system is deployed at the beginning of its life cycle and then updates are layered onto that system over time. Any updates to the system are applied to the instances individually, and may incur system downtime (depending on the stack configuration) due to application or system restarts.
- *Security groups*: Virtual firewalls for your instance to control inbound and outbound traffic. Security groups act at the instance level, not the subnet level. Therefore, each instance in a subnet in your VPC could have a different set of security groups assigned to it.
- *Service Level Agreements (SLAs)*: Part of AMS contracts with you that define the level of expected service.
- SLA *Unavailable* and *Unavailability*:
  - An API request submitted by you that results in an error .
  - A Console request submitted by you that results in a 5xx HTTP response (the server is incapable of performing the request).
  - Any of the AWS service offerings that constitute stacks or resources in your AMS-managed infrastructure are in a state of "Service Disruption" as shown in the Service Health Dashboard.
  - Unavailability resulting directly or indirectly from an AMS exclusion is not considered in determining eligibility for service credits. Services are considered available unless they meet the criteria for being unavailable.
- *Service Level Objectives (SLOs)*: Part of AMS contracts with you that define specific service goals for AMS services.

Patching terms:

- *Mandatory patches*: Critical security updates to address issues that could compromise the security state of your environment or account. A "Critical Security update" is a security update rated as "Critical" by the vendor of an AMS-supported operating system.
- *Patches announced versus released*: Patches are generally announced and released on a schedule. Emergent patches are announced when the need for the patch has been discovered and, usually soon after, the patch is released.
- *Patch add-on*: Tag-based patching for AMS instances that leverages AWS Systems Manager (SSM) functionality so you can tag instances and have those instances patched using a baseline and a window that you configure.

- *Patch methods*:
  - *In-place patching*: Patching that is done by changing existing instances.
  - *AMI replacement patching*: Patching that is done by changing the AMI reference parameter of an existing EC2 Auto Scaling group launch configuration.
- *Patch provider* (OS vendors, third party): Patches are provided by the vendor or governing body of the application.
- *Patch Types*:
  - *Critical Security Update (CSU)*: A security update rated as "Critical" by the vendor of a supported operating system.
  - *Important Update (IU)*: A security update rated as "Important" or a non-security update rated as "Critical" by the vendor of a supported operating system.
  - *Other Update (OU)*: An update by the vendor of a supported operating system that is not a CSU or an IU.
- *Supported patches*: AMS supports operating system level patches. Upgrades are released by the vendor to fix security vulnerabilities or other bugs or to improve performance. For a list of currently supported OSs, see Support Configurations.

Security terms:

- *Detective Controls*: A library of AMS-created or enabled monitors that provide ongoing oversight of customer managed environments and workloads for configurations that do not align with security, operational, or customer controls, and take action by notifying owners, proactively modifying, or terminating resources.

Service Request terms:

- *Service request*: A request by you for an action that you want AMS to take on your behalf.
- *Alert notification*: A notice posted by AMS to your **Service requests** list page when an AMS alert is triggered. The contact configured for your account is also notified by the configured method (for example, email). If you have contact tags on your instances/resources, and have provided consent to your cloud service delivery manager (CSDM) for tag-based notifications, the contact information (key value) in the tag is also notified for automated AMS alerts.
- *Service notification*: A notice from AMS that is posted to your **Service request** list page, usually to notify you of upcoming patching.

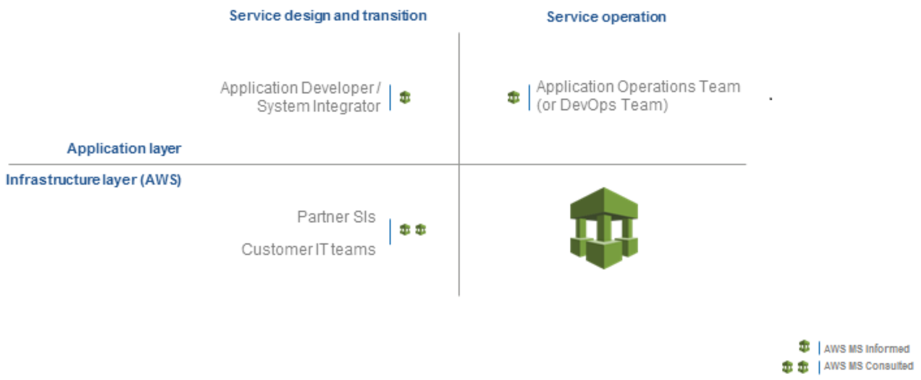
Miscellaneous terms:

- *AWS Managed Services Interface*: For AMS: The AWS Managed Services Advanced Console, AMS CM API, and AWS Support API. For AMS Accelerate: The AWS Support Console and AWS Support API.
- *Customer satisfaction (CSAT)*: AMS CSAT is informed with deep analytics including Case Correspondence Ratings on every case or correspondence when given, quarterly surveys, and so forth.
- *DevOps*: DevOps is a development methodology that strongly advocates automation and monitoring at all steps. DevOps aims at shorter development cycles, increased deployment frequency, and more dependable releases by bringing together the traditionally-separate functions of development and operations over a foundation of automation. When developers can manage operations, and operations informs development, issues and problems are more quickly discovered and solved, and business objectives are more readily achieved.
- *ITIL*: Information Technology Infrastructure Library (called ITIL) is an ITSM framework designed to standardize the lifecycle of IT services. ITIL is arranged in five stages that cover the IT service lifecycle: service strategy, service design, service transition, service operation, and service improvement.
- *IT service management (ITSM)*: A set of practices that align IT services with the needs of your business.

- *Managed Monitoring Services (MMS)*: AMS operates its own monitoring system, Managed Monitoring Service (MMS), that consumes AWS Health events and aggregates AWS CloudWatch data, and data from other AWS services, notifying AMS operators (online 24x7) of any alarms created through an Amazon Simple Notification Service (Amazon SNS) topic.
- *Namespace*: When you create IAM policies or work with Amazon Resource Names (ARNs), you identify an AWS service by using a namespace. You use namespaces when identifying actions and resources.

What is My Operating Model?

As an AMS customer your organization has decided to separate application and infrastructure operations and use AMS for infrastructure operations. AMS will work with your application design and development team along with your infrastructure design team to ensure that your infrastructure operations run smoothly. The following graphic illustrates this concept:


Service design and transition
Service operation
Application Developer / System Integrator
Application Operations Team (or DevOps Team)
Application layer
Infrastructure layer (AWS)
Partner SIs
Customer IT teams
AWS MS Informed
AWS MS Consulted

AMS takes responsibility for your AWS infrastructure operations while your teams are responsible for your application operations. As the application and infrastructure design teams, you must understand who will be operating the application once it has been deployed to production in the AMS infrastructure. This guide covers common approaches to infrastructure design as it relates to application deployment and maintenance.

AMS service management

Topics

How the AMS service works for you.

Account governance

This section covers AMS account governance.

You are designated a cloud service delivery manager (CSDM) who provides advisory assistance across AMS, and has a detailed understanding of your use case and technology architecture for the managed environment. CSDMs work with account managers, technical account managers, AWS Managed Services cloud architects (CAs), and AWS solution architects (SAs), as applicable, to help launch new projects and give best-practices recommendations throughout the software development and operations processes. The CSDM is the primary point of contact for AMS. Key responsibilities of your CSDM are:

- Organize and lead monthly service review meetings with customers.
- Provide details on security, software updates for environment and opportunities for optimization.
- Champion your requirements including feature requests for AMS.
- Respond to and resolve billing and service reporting requests.
- Provide insights for financial and capacity optimization recommendations.

Service commencement

Service Commencement: The *Service Commencement Date* for an AWS Managed Services account is the first day of the first calendar month after which AWS notifies you that the activities set out in the Onboarding Requirements for that AWS Managed Services account have been completed; provided that if AWS makes such notification after the 20th day of a calendar month, the Service Commencement Date is the first day of the second calendar month following the date of such notification.

Service Commencement

- **R** stands for responsible party that does the work to achieve the task.
- **I** stands for informed; a party which is informed on progress, often only on completion of the task or deliverable.

Service commencement

Step #	Step title	Description	Customer	AMS
1.	Customer AWS account handover	Customer creates a new AWS account and hands it over to AWS Managed Services	R	I
2.	AWS Managed Services Account - design	Finalize design of AWS Managed Services Account	I	R
3.	AWS Managed Services Account - build	An AWS Managed Services account is built per the design in Step 2	I	R

AMS customer relationship management (CRM)

The purpose of AMS's customer relationship management (CRM) process is to ensure that a well-defined relationship is established and maintained with you. The foundation of this relationship is based on AMS's insight into your business requirements. The CRM process facilitates accurate and comprehensive understanding of:

- Your business needs and how to fill those needs
- Your capabilities and constraints
- AMS and your different responsibilities and obligations

The CRM process allows AMS to use consistent methods to deliver services to you and provide governance for your relationship with AMS. The CRM process includes:

- Identifying your key stakeholders
- Establishing a governance team
- Conducting and documenting service review meetings with you
- Providing a formal service complaint procedure with an escalation procedure
- Implementing and monitoring your satisfaction and feedback process
- Managing your contract

CRM Process

The CRM process includes these activities:

- Identifying and understanding your business processes and needs. Your agreement with AMS identifies your stakeholders.
- Defining the services to be provided to meet your needs and requirements.
- Meeting with you in the service review meetings to discuss any changes in the AMS service scope, SLA, contract, and your business needs. Interim meetings may be held with you to discuss performance, achievements, issues, and action plans.

- Monitoring your satisfaction by using our customer satisfaction survey and feedback given at meetings.
- Reporting performance on monthly internally-measured performance reports.
- Reviewing the service with you to determine opportunities for improvements. This includes frequent communication with you regarding the level and quality of the AMS service provided.

CRM meetings

AMS cloud service delivery managers (CSDMs) conduct meetings with you regularly to discuss service tracks (operations, security, and product innovations) and executive tracks (SLA reports, satisfaction measures, and changes in your business needs).

Meeting	Purpose	Mode	Participants
Weekly status review (optional)	Outstanding issues or incidents, patching, security events, problem records 12-week operational trend (+/- 6) Application operator concerns Weekend schedule	On-site customer location/Telecom/Chime	AMS: CSDM and cloud architect (CA) Customer assigned team members (ex: Cloud/Infrastructure, Application Support, Architecture teams, etc.)
Monthly business review	Review service level performance (reports, analysis, and trends) Financial analysis Product roadmap CSAT	On-site customer location/Telecom/Chime	AMS: CSDM, cloud architect (CA), AMS account team, AMS technical product manager (TPM) (optional), AMS OPS manager (optional) You: Application Operator representative
Quarterly business review	Scorecard and service level agreement (SLA) performance and trends (6 months) Upcoming 3/6/9/12 months plans/migrations Risk and risk mitigations Key improvement initiatives Product roadmap items Future direction aligned opportunities	On-site customer location	AMS: CSDM, cloud architect, AMS account team, AMS service director, AMS operation manager You: Application operator representative, service representative, service director

Meeting	Purpose	Mode	Participants
	Financials Cost savings initiatives Business optimization		

CRM Meeting Arrangements

The AMS CSDM is responsible for documenting the meeting, including:

- Creating the agenda, including action items, issues, and list of attendees.
- Creating the list of action items reviewed at each meeting to ensure items are completed and resolved on schedule.
- Distributing meeting minutes and the action item list to meeting attendees by email within one business day after the meeting.
- Storing meeting minutes in the appropriate document repository.

In absence of the CSDM, the AMS representative leading the meeting creates and distributes minutes.

> **Note**
> Your CSDM works with you to establish your account governance.

CRM monthly reports

Your AMS CSDM prepares and sends out monthly service performance presentations. The presentations include information on the following:

- Report date
- Summary and Insights:
  - Key Call Outs: total and active stack count, stack patching status, account onboarding status (during onboarding only), customer-specific issues summaries
  - Performance: Stats on incident resolution, alerts, patching, requests for change (RFCs), service requests, and console and API availability
  - Issues, challenges, concerns, and risks: Customer-specific issues status
  - Upcoming items: Customer-specific onboarding or incident resolution plans
- Managed Resources: Graphs and pie charts of stacks
- AMS Metrics: Monitoring and event metrics, incident metrics, AMS SLA adherence metrics, service request metrics, change management metrics, storage metrics, continuity metrics, Trusted Advisor metrics, and cost summaries (presented several ways). Feature requests. Contact information.

> **Note**
> In addition to the described information, your CSDM also informs you of any material change in scope or terms, including use of subcontractors by AMS for operational activities.
> AMS generates reports about patching and backup that your CSDM includes in your monthly report. As part of the report generating system, AMS adds some infrastructure to your account that is not accessible to you:
>
> - An S3 Bucket, with the raw data reported
> - An Athena instance, with query definitions to query the data

- A Glue Crawler to read the raw data from the S3 bucket

Updates to shared services: Multi-Account Landing Zone

AMS uses the core OU to provide shared services such as access, networking, EPS, log storage, alert aggregation in your Multi-Account Landing Zone. AMS is responsible for addressing vulnerabilities, patching, and deployments of these shared services. AMS regularly updates the resources used for providing these shared services so that users have access to latest features, and security updates. The updates typically happen on a monthly basis. Resources that are part of these updates are:

- Accounts that are part of the core OU.

  The management account, shared services account, network account, security account, and log archive account have resources for RDP and SSH bastions, proxies, management hosts, and endpoint security (EPS), that are typically updated every month. AMS uses immutable EC2 deployments as part of the shared services infrastructure.
- New AMS AMIs incorporating the latest updates.

Note
AMS operators utilize an internal alarm suppression change type (CT) when executing data plane changes and the RFC for that CT appears in your RFC list. This is because, as the data plane release is deployed, various infrastructure may be shut down, rebooted, taken offline, or there may be CPU spikes or other effects of the deployment that trigger alarms that, during the data plane deployment, are extraneous. Once the deployment is complete, all infrastructure is verified to be running properly and alarms are re-enabled.

AMS planned event management

AWS Managed Services (AMS) planned event management (PEM) is an AMS service offering. PEM is used to engage, plan, and run customer events and projects using AMS Change Management Services and dedicated AMS resources. Change management delivers an individual request for change (RFC). The PEM delivers a set of related RFCs that align with the scope and timeline of the PEM event or project.

AMS PEM criteria

A planned event is defined as a scope-bound and time-bound project. For example, migrations, game days, disaster recovery tests, projects, or events that require dedicated on-site or off-site AMS resources such as Operation Engineers or Cloud Architects.

The AMS PEM process

The PEM process consists of the following phases:

- **Initiation —**: You engage with the Cloud Service Deliver Managers (CSDM), Technical Delivery Managers (TDM), and Cloud Architects (CA) to provide project information and the technical details to AMS. AMS works with you to ensure that the PEM plan information is correct and complete. For PEM acceptance, AMS Operations requires a lead time of 2 weeks to allow the AMS Operations appropriate time to ensure planning, technical review and resource assignment. Additional time may be required for delivery of pre-PEM tasks.

- **Technical Review —**: AMS Cloud Architects review the technical aspects of the PEM plan. They work with AMS Security and Operations to ensure compliance, provide execution optimization and automation, and define pre-PEM execution tasks and deliverables.
- **Planning —**: AMS ensures that the necessary AMS resources are assigned.
- **Readiness and Execution —**: AMS ensures pre-execution tasks are completed, and facilitates internal and customer communications. AMS also ensures execution of the PEM plan and provides execution status and progress reporting.

Getting help

You can reach out to AMS to identify the root cause of your failure. AMS business hours are 24 hours a day, 7 days a week, 365 days a year.

AMS provides several avenues for you to ask for help or make service requests.

- To ask for information or advice, or for access to an AMS-managed IT service, or to request an additional service from AMS, use the AMS console and submit a service request. For details, see Creating a Service Request. For general information about AMS service requests, see Service Request Management.
- To report an AWS or AMS service performance issue that impacts your managed environment, use the AMS console and submit an incident report. For details, see Reporting an incident. For general information about AMS incident management, see Incident response.
- For specific questions about how you or your resources or applications are working with AMS, or to escalate an incident, email one or more of the following:
  1. First, if you are unsatisfied with the service request or incident report response, email your CSDM: ams-csdm@amazon.com
  2. Next, if escalation is required, you can email the AMS Operations Manager (your CSDM will most likely do this): ams-opsmanager@amazon.com
  3. Further escalation would be to the AMS Director: ams-director@amazon.com
  4. Finally, you are always able to reach the AMS VP: ams-vp@amazon.com

Customer contacts with AMS that require escalation will follow the escalation path described next.

Service hours

Feature	AMS Accelerate		AMS Advanced	
Service request	Monday to Friday: 08:00–18:00, local business hours	24/7	Monday to Friday: 08:00–18:00, local business hours	24/7
Incident management (P1)	24/7			
Incident management (P2-P3)	Monday to Friday: 08:00–18:00, local business hours	24/7	Monday to Friday: 08:00–18:00, local business hours	24/7
Backup and recovery	24/7			

Feature	AMS Accelerate	AMS Advanced	
Patch management	24/7		
Monitoring and alerting	24/7		
Automated request for change (RFC)	Not Applicable	24/7	
Non-automated request for change (RFC)	Not Applicable	Monday to Friday: 08:00–18:00, local business hours	24/7
Cloud service delivery manager (CSDM)	Monday to Friday: 08:00–17:00, local business hours		

How do I get offboard assistance from AMS Single-Account Landing Zone accounts?

AMS offers off-boarding assistance within 30 days prior to termination of AMS.

You must request off-boarding assistance at least 7 days before such assistance can be provided. Off-boarding assistance can be offered in two forms:

- Control hand-over: AMS will transfer account control back to the Customer along with access credentials for all AMS Managed Applications, or
- Resource termination and data transfer: AMS backs-up all the data, deletes all the data in customer’s Managed Environment, de-provisions any active resources in the account, and hands over the data backup to the Customer. At customer’s request AMS can transfer customer data in the existing format using Snowball or any other media with which AWS can interface. In addition to data backups, the following customer data can be provided as part of off-boarding assistance:
  - Data stored in storage services including logs
  - Customer-specific Change type schemas
  - CloudFormation templates for Change type schemas.

If off-boarding activities are not completed upon the termination of AMS, we hand over the controls of the account(s) to enable you to complete any pending activity.

How do I offboard from AMS Multi-Account Landing Zone accounts?

Currently AMS supports 3 types of offboarding for multi-account landing zone accounts:

- Multi-Account Landing Zone environmental offboarding
- Application account offboarding
- Application account VPC offboarding.

How do I offboard a Multi-Account Landing Zone environment?

Scenario:

You want to leave AMS, and close (terminate) your AMS AWS account completely (including the primary account that you provided).

Process:

1. You communicate with your CSDMs or CAs and request offboarding via email or a service request.
2. AMS works on offboarding your accounts.
3. Your CSDM or CA sends you an outbound email containing further instructions; see How do I close my AWS account?

Prerequisites: Verify that you can access the account email used for account closures.

Offboarding Conclusions:

- All components are disassociated from the AMS services, but are not yet deleted in all accounts. Account closure will eventually delete all resources.
- Billing is *not stopped* until you request the account closures.
- After the account is closed, you can still sign in and file a support case or contact AWS Support for 90 days.
- After 90 days, any content remaining in the account is permanently deleted, and AWS services that aren't already terminated, are terminated.

How do I offboard a Multi-Account Landing Zone application account?

Some offboarding scenarios.

Scenario:

You want to offboard one, or more than one, application accounts from your multi-account landing zone environment, close (terminate) those accounts completely.

Process:

1. You communicate with your CSDMs or CAs to request offboarding via email or a service request.
2. AMS works on offboarding your accounts from AMS.
3. Your CSDM or CA sends you an outbound email containing further instructions; see How do I close my AWS account?

Prerequisites: Verify that you can access the account email used for account closures.

Offboarding Conclusions:

- All components are disassociated from AMS services, and most of the resources are deleted in the requested accounts. Your resources remain in the account until you request account closure.
- Core accounts and other application accounts function normally after the offboarding request.

- Billing is *not stopped* until you request account closures.
- After the account is closed, you can still sign in and file a support case or contact AWS Support for 90 days.
- After 90 days, any content remaining in the account is permanently deleted, and AWS services that aren't already terminated, are terminated.

How do I offboard a Multi-Account Landing Zone application account VPC?

Scenario:

You want to offboard one of your VPCs from an AMS-managed application account.

Process:

1. You communicate with your CSDMs or CAs via email or a service request, to request VPC offboarding.
2. AMS works on offboarding the VPC from AMS.
3. Your CSDM and CA notify you of the completion of the offboarding.

Prerequisites:

- Verify there are *no running instance stacks* associated with this offboarding VPC.

  If there are running instance stacks associated with the offboarding VPC, you are responsible for deleting those instance stacks prior to requesting a VPC offboarding.
- Your application account should always contain at least one VPC. If you request a VPC offboarding and that VPC is the only VPC in the account, follow up with an application VPC create RFC.
- Verification email to confirm the request. Your request should contain the VPC name of the VPC that you want to delete, and the account ID that the VPC is associated with.

Offboarding Conclusions:

Application accounts function normally after VPC offboarding request.

Application Development

This section reviews application development processes and practices that enable effective design and deployment of applications into an AWS Managed Services (AMS) environment. AMS guides you through the following high level process:

1. Envision and architect an application to be developed or integrated to your AMS-managed environment. Some considerations:
    a. How will you deploy your application? With automation using a deployment tool such Ansible, or manually by directly uploading needed files?
    b. How will you update your application? With a mutable approach updating each instance separately, or with an immutable approach updating each instance with a single, updated, AMI in an Auto Scaling group?
2. Plan and architect the infrastructure that will be used to host the application using AWS architecture libraries, AWS "Well-Architected" guidance, and AMS and other cloud architecture subject matter experts. The following sections of this guide provide information that can help with this.
3. Select an infrastructure deployment approach:
    a. Full Stack: All infrastructure components are deployed at once, together.
    b. Tier and Tie: Infrastructure deployments are deployed separately and, after, tied together with security group modifications. This type of deployment is also achieved by a serial configuration of stack components that builds upon each other; for example, specifying the load balancer that you previously created when you create an Auto Scaling group.
    c. What environments, such as Dev, Staging, and Prod, will you employ?
4. Choose AMS change types (CTs) that will provision the necessary stacks, or tiers, and prepare the necessary requests for change (RFCs).
5. Submit the RFCs to trigger the deployment of the infrastructure to the appropriate environment.
6. Deploy the application using the application deployment approach selected.
7. Re-work infrastructure and applications as needed.
8. Deploy infrastructure and applications to appropriate follow-on environments, assuming your first deployment is to a non-production environment.
9. Ongoing maintenance is handled by AMS operating the underlying infrastructure, and your operations teams operating the application(s) infrastructures.
10. To decommission an application, terminate the AMS infrastructure for it.

Being Well Architected

At AWS we believe that well-architected systems greatly increase the likelihood of business success. The AWS Architecture Center provides expert guidance on architecting in the AWS Cloud.

We recommend the following articles and white papers to help you understand the pros and cons of the decisions you must make while building systems on AWS.

Are You Well-Architected?: Introduces the AWS Well-Architected Framework, based around five pillars:

- **Security**: The security pillar focuses on protecting information & systems. Key topics include confidentiality and integrity of data, identifying and managing who can do what with privilege management, protecting systems, and establishing controls to detect security events.

- **Reliability**: The reliability pillar focuses on the ability to prevent, and quickly recover from failures to meet business and customer demand. Key topics include foundational elements around setup, cross project requirements, recovery planning, and how we handle change.
- **Performance Efficiency**: The performance efficiency pillar focuses on using IT and computing resources efficiently. Key topics include selecting the right resource types and sizes based on workload requirements, monitoring performance, and making informed decisions to maintain efficiency as business needs evolve.
- **Cost Optimization**: Cost Optimization focuses on avoiding un-needed costs. Key topics include understanding and controlling where money is being spent, selecting the most appropriate and right number of resource types, analyzing spend over time, and scaling to meet business needs without overspending.
- **Operational Excellence**: The operational excellence pillar focuses on running and monitoring systems to deliver business value, and continually improving processes and procedures. Key topics include managing and automating changes, responding to events, and defining standards to successfully manage daily operations.

AWS Well-Architected Framework: Describes how AWS enables customers to assess and improve their cloud-based architectures and better understand the business impact of their design decisions. It addresses general design principles as well as specific best practices and guidance in four conceptual areas that AWS defines as the pillars of the "Well-Architected Framework."

Architecting for the Cloud: Best Practices: This white paper is targeted towards cloud architects who are gearing up to move an enterprise-class application from a fixed physical environment to a virtualized cloud environment. The focus of this paper is to highlight concepts, principles and best practices in creating new cloud applications or migrating existing applications to the cloud.

Application Layer Responsibilities vs Infrastructure Layer Responsibilities

By using AMS, your infrastructure, and all it needs for maintenance and growth, is maintained by AMS. However, whatever you need for line-of-business applications or product applications, is developed, deployed, and maintained by you.

With the help of application deployment tools, such as CodeDeploy and AWS CloudFormation, or Chef, Puppet, Ansible, or Saltstack, your application deployment to your AMS-managed infrastructure can be fully automated.

For details about what AMS does and does not do, see What we do, what we do not do (p. 1).

EC2 Instance Mutability

You and AMS can maintain the Amazon Elastic Compute Cloud (EC2) instances in your infrastructure in one of two manners:

- Immutable: This model uses Amazon machine images (AMIs) "baked" (created) with the necessary features. When deploying an update, the existing instances are torn down and completely replaced with new ones created from an updated AMI. To minimize down-time, this rolling process leaves some instances un-updated and accessible while others are being updated until, eventually, the new change is completely deployed.
- Mutable: In this model, the infrastructure is updated with new code being deployed on existing systems in the Cloud. This model is a mix of manually pushing updates and using infrastructure-as-code to deploy updates and does not rely on new AMIs.

These maintenance models are discussed in more detail in later sections of this guide.

Using AWS Secrets Manager with AMS Resources

There are many cases where you may need to share secrets with AMS, for example:

- Master password reset for RDS instance
- Certificates for load balancers
- Obtaining long-lived credentials for IAM users from AMS

The safest way to share confidential information with AMS is through the AWS Secrets Manager; follow these steps:

1. Login to the AWS Console using your federated access and the Customer ReadOnly role.
2. Navigate to the AWS Secrets manager console and click **Store a new secret**.
3. Select "Other type of secrets".
4. Enter the secret value as a plain-text and click **Next**.
5. Enter the secret name and description. The name should always starts with "**customer-shared/***". For example "**customer-shared/license-2018**". Once you are done continue by clicking **Next**.
6. Use the default KMS encryption.
7. Leave automatic rotation disabled and click **Next**.
8. Review and click **Store**, to save the secret.
9. Reply to us in an AMS service request with the secret name and ARN, so we can identify and retrieve the secret. For information on creating service requests, see Service Request Examples.

Application Deployment

During onboarding, AWS Managed Services (AMS) works with you to determine the infrastructure that you need.

The basic infrastructure includes an AWS virtual private cloud (VPC), communication security via an ADFS forest trust, the basic subnets (DMZ, Shared Services, and Private) mirrored across two availability zones and configured with a managed NAT, bastions, public load balancers, AWS DirectConnect (DX), and required security. Your application resources will be deployed in your private, or customer-applications, subnet. You can learn more about a typical AMS architecture in the AWS Managed Services User Guide.

The infrastructure you deploy once the basics are done, should include all components for your applications and application development.

Application Deployment Capabilities

This section provides an overview of some of the ways you can deploy applications in AMS. Details on each method follow.

Application Deployment Capabilities Examples

Method Name	Infrastructure Deployment	AMI or Key Element(s)	Application Install
Mutable Applications, AMS AMI			
Manual application deployment	Full stack CT or Tier and Tie CTs	AMS-provided AMI	Submit Access management CT, install application manually.
UserData application deployment with application agent (i.e. Chef, Puppet, etc.)			Use provisioning CT with UserData scripting that installs an application agent, and that script/agent installs the application.
UserData agentless application deployment (i.e. Ansible, Salt SSH, etc.)			Submit Access management CT, install application agent. Deploy application with application deploy tooling.
Mutable Applications, Custom AMI			
Custom AMI application deployment (non-ASG)	Full stack CT or Tier and Tie CTs	Custom AMI. AMS AMI -> customize with application deploy tooling agent -> create EC2 instance (CT) -> create AMI (CT).	Application deploy tooling (i.e. Chef), leveraging agents, deploys application.

Method Name	Infrastructure Deployment	AMI or Key Element(s)	Application Install
AWS Database Migration Service (DMS) application deployment	AWS DMS sync to existing AMS Relational Database stack.	Custom AMI	Customer or partner employs AWS Database Migration Service; AMS verifies AMS components on launch
Workload Ingest application deployment	Partner-migrated instance/AMI and customer-initiated Workload Ingest CT.		Partner migrates instance, creates AMI in customer AMS-managed VPC; customer uses Workload Ingest CT to launch stack in AMS. For details, see AMS Workload Ingest (WIGS) (p. 23).
Immutable Applications			
Custom AMI application deployment (ASG)	Full stack CT or Tier and Tie CTs	AMS AMI -> customize -> create EC2 instance (CT) -> create AMI (CT) -> create Auto Scaling group.	Auto Scaling deploys application with the custom AMI For details, see AMS Tier and Tie App Deployments (p. 118).
Mutable or Immutable Applications			
Custom CloudFormation Template application deployment	CloudFormation template	AWS CloudFormation template -> customize/ prepare for AMS -> Deployment | Ingestion | Stack from CloudFormation Template | Create (ct-36cn2avfrrj9v).	AMS deploys your application to your account using your custom CloudFormation template, and validates the application deployment. For details, see AMS CloudFormation ingest (p. 42).
SQL Database Import	AMS operations (Other | Other CT)	On premise SQL database -> .bak file -> AMS RDS SQL database -> Management | Other | Other | Create (ct-1e1xtak34nx76) for the import.	AMS imports your on-premises database to your AMS-managed RDS database. For details, see Database (DB) Import to AMS SQL RDS (p. 116).

Method Name	Infrastructure Deployment	AMI or Key Element(s)	Application Install
Database Migration Service (DMS)	AMS operations (Multiple CTs)	On premise database -> DMS replication instance -> DMS replication subnet group -> DMS target endpoint -> DMS source endpoint -> DMS replication task.	AMS imports your on-premises database to your AMS-managed S3 or target RDS database. For details, see Database Migration Service (DMS) (p. 85).
CodeDeploy application deployment	CodeDeploy	Application -> CodeDeploy application -> CodeDeploy deployment group -> CodeDeploy deployment.	Depending on usage, In-place or Blue/ Green application deployment. For details, see CodeDeploy requests (p. 74).

Planning Your Application Deployment

For a recommended set of questions to be answered to enable application deployments, see Appendix: Application Onboarding Questionnaire (p. 140). Questions cover describing your:

- Deployment Summary (p. 140)
- Infrastructure Deployment Components (p. 140)
- Application Hosting Platform (p. 141)
- Application Deployment Model (p. 141)
- Application Dependencies (p. 141)
- SSL Certificates for Product Applications (p. 142)

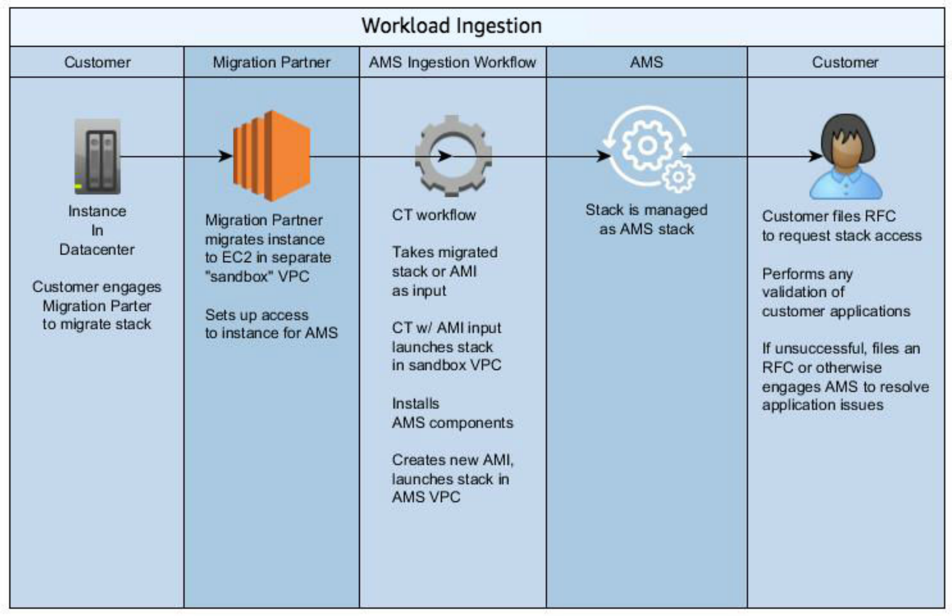
AMS Workload Ingest (WIGS)

Topics

The new AMS workload ingest CT allows you, along with an AMS cloud migration partner, to move your existing workloads into an AMS-managed VPC. Using AMS workload ingest, you can create a custom AMS AMI after moving migrated instances onto AMS. This section describes the process, pre-requisites, and steps that your migration partner and yourself take for AMS workload ingestion.

Important
The operating system must be supported by AMS workload ingest. For supported operating systems, see Migrating Workloads: Prerequisites for Linux and Windows (p. 24).
Every workload and account is different. AMS will work with you to prepare for a successful result.

The following diagram depicts the AMS workload ingestion process.


Workload Ingestion
Customer
Migration Partner
AMS Ingestion Workflow
AMS
Customer
Instance In Datacenter
Customer engages Migration Parter to migrate stack
Migration Partner migrates instance to EC2 in separate "sandbox" VPC
Sets up access to instance for AMS
CT workflow
Takes migrated stack or AMI as input
CT w/ AMI input launches stack in sandbox VPC
Installs AMS components
Creates new AMI, launches stack in AMS VPC
Stack is managed as AMS stack
Customer files RFC to request stack access
Performs any validation of customer applications
If unsuccessful, files an RFC or otherwise engages AMS to resolve application issues

Migrating Workloads: Prerequisites for Linux and Windows

Before ingesting a copy of an on-premises instance into AWS Managed Services (AMS), certain prerequisites must be met. These are the prerequisites, including those that differ between Windows and Linux operating systems.

Note
To simplify the process of determining if the instances are ready for ingestion, validation tools for both Windows and Linux have been created. These tools can be downloaded and run directly on your on-premises servers as well as EC2 instances in AWS. Linux Pre-WIGS Validation.zip, Windows Pre-WIGS Validation.zip.

BEFORE YOU BEGIN, for Linux and Windows:

- Perform a full virus scan.
- The instance must have the `customer-mc-ec2-instance-profile` instance profile.
- Install the Amazon EC2 Systems Manager (SSM) Agent and make sure that the SSM Agent is up and running.
- A minimum of 10GB of free disk space on the root volume is recommended to run AMS workload ingest (WIGS). Operationally, AMS recommends disk utilization less than 75% and alerts when disk utilization reaches 85%.
- Determine a time frame for the ingestion with your migration partner.
- The custom AMI exists as an EC2 instance in the target production AMS account (this is the migration partner's responsibility).

Important

The operating system must be supported by AMS workload ingest. The following operating systems are supported:

- Microsoft Windows Server: 2012, 2012 R2, and 2016, 2019
- Linux: Amazon Linux 2 and Amazon Linux, CentOS 7.x, CentOS 6.5-6.10, Oracle Linux 7.5 and later minor versions, RHEL 8.x, RHEL 7.x, RHEL 6.5-6.10, SUSE Linux Enterprise Server 15 SP0, SP1 and SAP specific versions, SUSE Linux Enterprise Server 12 SP4*, SP5.

Note

The AMS API/CLI (amscm and amsskms) endpoints are in the AWS N. Virginia Region, `us-east-1`. Depending on how your authentication is set, and what AWS Region your account and resources are in, you may need to add `--region us-east-1` when issuing commands. You may also need to add `--profile saml`, if that is your authentication method.

LINUX Prerequisites

Observe the requirements listed in Migrating Workloads: Prerequisites for Linux and Windows (p. 24) and ensure the following before submitting a WIGS RFC:

- The latest enhanced networking drivers are installed; see Enhanced Networking on Linux.
- Third-party software components that will conflict with AMS components have been removed:
  - Anti-virus Clients
  - Backup Clients
  - Virtualization software (such as VM Tools or Hyper-V Integration services)
  - Access Management Software (Such as SSSD, Centrify, or PBIS)
- Ensure SSH is properly configured - This temporarily enables private key authentication for SSH. AMS uses this with our configuration management tool. Use these commands:

```
sudo grep -q "^PubkeyAuthentication" /etc/ssh/sshd_config && sudo sed "s/
^PubkeyAuthentication=.*/PubkeyAuthentication yes/" -i /etc/ssh/sshd_config || sudo sed
 "$ a\PubkeyAuthentication yes" -i /etc/ssh/sshd_config
```

```
sudo grep -q "^AuthorizedKeysFile" /etc/ssh/sshd_config && sudo sed "s/
^AuthorizedKeysFile=.*/AuthorizedKeysFile %h\/.ssh\/authorized_keys/" -i /etc/ssh/
sshd_config || sudo sed "$ a\AuthorizedKeysFile %h/.ssh/authorized_keys" -i /etc/ssh/
sshd_config
```

- Ensure Yum is properly configured - RedHat requires licensing to use their Yum Repositories. The instance needs to be licensed via a Satellite Server or RedHat Cloud Server. Use one of these links if licensing is needed:
  - Red Hat Satellite
  - Red Hat Cloud Access
- If you use Red Hat Satellite, WIGS requires the addition of Red Hat Software Collections (RHSCL). The WIGS system uses RHSCL to add a Python3.6 interpreter alongside whatever is configured on the system. To support this solution, the following repositories must be available:
  - rhel-server-rhscl
  - rhel-server-releases-optional

Windows Prerequisites

Observe the requirements listed in Migrating Workloads: Prerequisites for Linux and Windows (p. 24) and ensure the following before submitting a WIGS RFC:

- Powershell version 3 or higher is installed.
- AWS EC2 Config is installed on the instance with the workload that you will migrate.
- Install the AWS drivers that support the latest generation instance types: PV, ENA, and NVMe. You can use the information in these links:
  - Upgrading PV Drivers on Your Windows Instances
  - Enhanced Networking on Windows
  - AWS NVMe Drivers for Windows Instances
  - Part 3: Upgrading AWS NVMe drivers
  - Part 5: Installing the Serial Port Driver for Bare Metal Instances
  - Part 6: Updating Power Management Settings
- (Optional but recommended) Disable critical Services – Set critical application services, such as databases, to disabled, but ensure that any changes are documented so they can be reverted to their original startup mode during the application verification stage.
- (Optional but recommended) Create a Failsafe AMI from the prepared instance:
  - Use the Deployment | Advanced stack components | AMI | Create
  - During creation, add a tag Key=Name, Value=APPLICATION-ID_IngestReady
  - Wait until AMI is created before proceeding
- Third-party software components that will conflict with AMS components have been removed:
  - Anti-virus Clients
  - Backup Clients
  - Virtualization software (such as VM Tools or Hyper-V Integration services)

Note
The End-of-Support Migration Program for Windows server (EMP) includes tooling to migrate your legacy applications from Windows Server 2003, 2008, and 2008 R2 to newer, supported versions on AWS, without any refactoring.

How Migration Changes Your Resource

The ingestion RFC described in this section takes the next step of adding configurations to the instance, once it is migrated to your AMS account, so that AMS can manage it.

The configurations added are AMS-specific as follows.

Changes made to ingested Linux instances:

- Software that is installed:
  - Cloud Init: Used to configure private keys for Jarvis Access.
  - Python 3 (scripting language) for all supported operating systems (Except for CentOS 6, RHEL 8, OracleLinux 7).
  - AWS CloudFormation Python Helper Scripts: AWS CloudFormation provides scripts used to install software and start services on an Amazon EC2 instances.
  - AWS CLI: The AWS CLI is an open source tool built on top of the AWS SDK for Python (Boto) that provides commands for interacting with AWS services.
  - AWS SSM Agent: The SSM Agent processes requests from the Systems Manager service configures the machine as specified in the request.

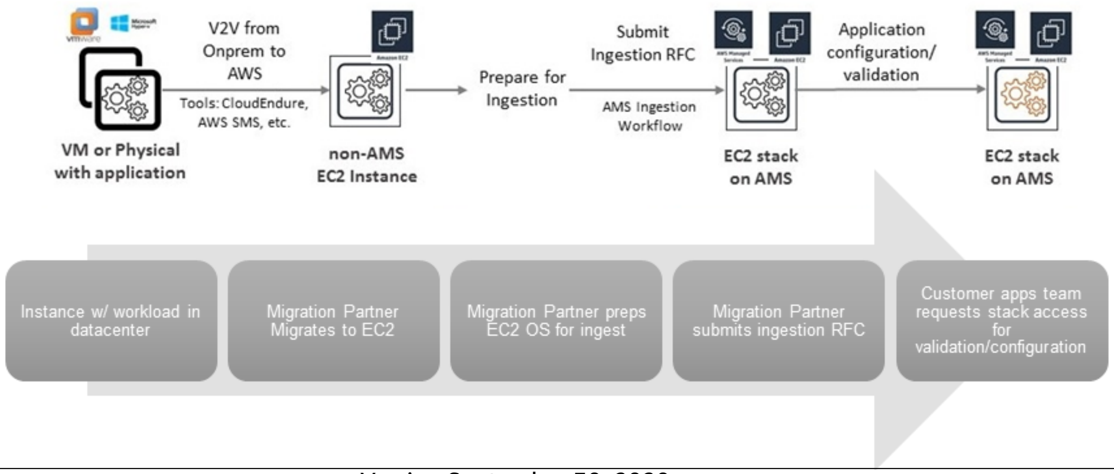
  - AWS CloudWatch Logs Agent: Sends logs to CloudWatch.
  - AWS CodeDeploy: A deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, or serverless Lambda functions.
  - Ruby: Required for CodeDeploy
  - System Performance Tools (sysstat): Sysstat contains various utilities to monitor system performance and usage activity.
  - AD Bridge (Formerly PowerBroker Identity Services): Joins non-Microsoft hosts to Active Directory domains.
  - Trend Micro Deep Security Agent: Anti-Virus software.
- Software that is changed:
  - The instances are configured to use the UTC timezone.

Changes made to ingested Windows instances:

- Software that is installed:
  - AWS Tools for Windows PowerShell: The AWS Tools for PowerShell let developers and administrators manage their AWS services and resources in the PowerShell scripting environment.
  - Trend Micro Deep Security Agent: Anti-Virus protection
  - AMS PowerShell Modules containing PowerShell code for controlling Boot, Active Directory Join, Monitoring, Security, and Logging.
- Software that is changed:
  - Server Message Block (SMB) version 1 is disabled.
  - Windows Remote Management (WinRM) is enabled and configured to listen on port 5986. A firewall rule allowing this inbound port is also created.
- Software that *might be* installed or changed:
  - Microsoft .Net Framework 4.5 (Developer platform), if a version lower then .Net Framework 4.5 is detected.
  - For Windows 2012, ad Windows 2012R2, we upgrade to PowerShell 5.1.

Migrating Workloads: Standard Process

Note
Because two parties are required for this process, this section describes the tasks for each: an AMS Cloud Migration Partner (migration partner), and an Application Owner (you).


V2V from Onprem to AWS
Tools: CloudEndure, AWS SMS, etc.
VM or Physical with application
non-AMS EC2 Instance
Prepare for Ingestion
Submit Ingestion RFC
AMS Ingestion Workflow
EC2 stack on AMS
Application configuration/ validation
EC2 stack on AMS
Instance w/ workload in datacenter
Migration Partner Migrates to EC2
Migration Partner preps EC2 OS for ingest
Migration Partner submits ingestion RFC
Customer apps team requests stack access for validation/configuration

1. Migration partner, Set Up:

   a. The migration partner submits a Service Request to AMS for an IAM role for the purpose of migrating your instance. For details on submitting service requests, see Service Request Examples.

   b. The migration partner submits a Admin Access Request. The AMS Operations team provides the migration partner access to your account via the requested IAM role.

2. Migration partner, Migrate Individual Workloads:

   a. The migration partner migrates your non-AWS instance to a subnet in your AMS account via native EC2 or other migration tooling, with the `customer-mc-ec2-instance-profile` IAM instance profile (must be in the account).

   b. The migration partner submits an RFC with the Deployment | Ingestion | Stack from migration partner migrated instance | Create CT (ct-257p9zjk14ija); for details on creating and submitting this RFC, see Workload Ingest Stack: Creating (p. 39).

      The execution output of the RFC returns an instance ID, IP address, and AMI ID.

      The migration partner provides you with the instance ID of the workload created in your account.

3. You, Access and Validate the Migration:

   a. Using the execution output provided you (AMI ID, instance ID, and IP address) by the migration partner, submit an access RFC and log into the newly-created AMS stack and verify that your application is working properly. For details, see Requesting Instance Access.

   b. If satisfied, you can continue to use the launched instance as a 1-tier stack and/or use the AMI to create additional stacks, including Auto Scaling groups.

   c. If not satisfied with the migration, file a service request and reference the stack and RFC IDs; AMS will work with you to address your concerns.

CloudEndure landing zone workload ingest process is described next.

Migrating Workloads: CloudEndure Landing Zone (SALZ)

This section provides information on setting up an intermediate migration single-account landing zone (SALZ) for CloudEndure (CE) cutover instances to be available for a workload ingest (WIGS) RFC.

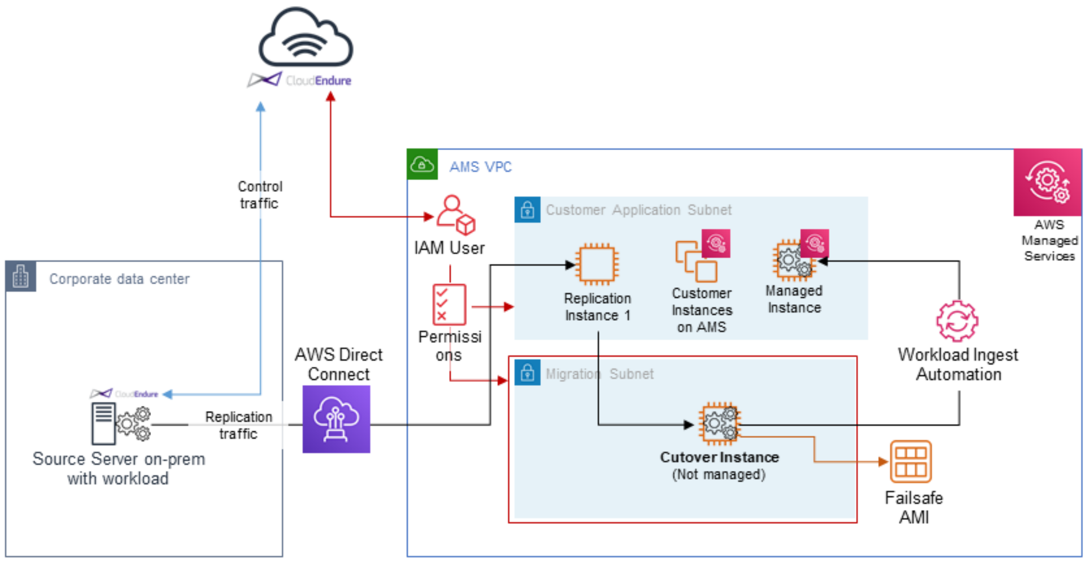
To learn more about CloudEndure, see CloudEndure Migration.

> **Note**
> This is a predefined, security hardened, migration LZ and pattern.

Prerequisites:

- A customer AMS account
- Network and access integration between AMS account and the customer on-premise
- A CloudEndure account
- A pre-approval workflow for an AMS Security review and signoff, run with your CA and/or CSDM, (for example, misuse of the IAM user permanent credentials provides the ability to create/delete instances and security groups)

Note
Specific preparation and migration processes are described in this section.



Preparation: You and AMS operator:

1. Prepare a Request for Change (RFC) with the Management | Other | Other | Update change type to AMS for the following resources and updates. You can submit separate Other | Other Update RFCs, or one. For details on that RFC/CT, see Other | Other Update with these requests:

   a. Assign a secondary CIDR block in your AMS VPC; a temporary CIDR block that will be removed after the migration is completed. Ensure that the block will not conflict with any existing routes back to your on-premise network. For example, if your AMS VPC CIDR is 10.0.0.0/16, and there is a route back to your on-premise netword of 10.1.0.0/16, then the temporary secondary CIDR could be 10.255.255.0/24. For information on AWS CIDR blocks, see VPC and Subnet Sizing.

   b. Create a new, private, subnet inside the initial-garden AMS VPC. Example name: `migration-temp-subnet`.

   c. Create a new route table for the subnet with only local VPC and NAT (Internet) routes, to avoid conflicts with the source server during instance cutover and possible outages. Ensure outbound traffic to the Internet is allowed for patch downloads, and so that AMS WIGS pre-requisites can be downloaded and installed.

   d. Update your Managed AD security group to allow inbound and outbound traffic to/from `migration-temp-subnet`. Also request that your EPS load balancer (ELB) security group (ex: `mc-eps-McEpsElbPrivateSecurityGroup-M79OXBZEEX74`) be updated to allow the new, private, subnet (i.e. `migration-temp-subnet`). If the traffic from the dedicated CloudEndure (CE) subnet is not allowed on all three TCP ports, WIGS ingestion will fail.

   e. Finally, request a new CloudEndure IAM policy and IAM user. The policy needs your correct account number, and the subnet IDs in the `RunInstances` statement should be: your <Customer Application Subnet(s) + Temp Migration Subnet>.

      To see an AMS pre-approved IAM CloudEndure policy: Unpack the WIGS Cloud Endure Landing Zone Example file and open the `customer_cloud_endure_policy.json`.

      Note
      If you want a more permissive policy, discuss what you need with your CloudArchitect/ CSDM and obtain, if needed, an AMS Security Review and Signoff before submitting an RFC implementing the policy.

2. Your preparation steps to use CloudEndure for AMS workload ingestion are done and, if your migration partner has completed their preparation steps, migration is ready to be performed. The WIGS RFC is submitted by your migration partner.

   Note
   IAM user keys won't be directly shared, but must be typed into the CloudEndure management console by the AMS operator in a screen-sharing session.

Preparation: Migration Partner and AMS Operator:

1. Create CloudEndure migration project.
   a. During project creation, have AMS type-in IAM user credentials in screen-sharing sessions.
   b. In **Replication Settings** -> **Choose the subnet where the Replication Servers will be launched**, select **customer-application-x** subnet.
   c. In **Replication Settings** -> **Choose the Security Groups to apply to the Replication Servers**, select both Sentinel security groups (Private Only and EgressAll).
2. Define cutover options for the machines (instances).
   a. Subnet: **migration-temp-subnet**.
   b. Security Group: Both "Sentinel" security groups (Private Only and EgressAll).

      Cutover instances must be able to communicate to the AMS Managed AD and to AWS public endpoints.
   c. Elastic IP: **None**
   d. Public IP: **no**
   e. IAM role: **customer-mc-ec2-instance-profile**

      The IAM role must allow for SSM communication. Better to use AMS default.
   f. Set tags as per convention.

Migration: Migration Partner:

1. Create a dummy stack on AMS. You use the stack ID to gain access to the bastions.
2. Install the CloudEndure (CE) agent on the source server. For details, see Installing the Agents.
3. Create local admin credentials on the source server.
4. Schedule a short cutover window and click **Cutover**, when ready. This finalizes the migration and redirects users to the target AWS Region.
5. Request stack Admin access to the dummy stack, see Admin Access Request.
6. Log into the bastion, then to the cutover instance using the local admin credentials you created.
7. Create a failsafe AMI. For details on creating AMIs, see AMI Create.
8. Prepare the instance for ingestion, see Migrating Workloads: Prerequisites for Linux and Windows (p. 24).
9. Run WIGS RFC against the instance, see Workload Ingest Stack: Creating (p. 39).

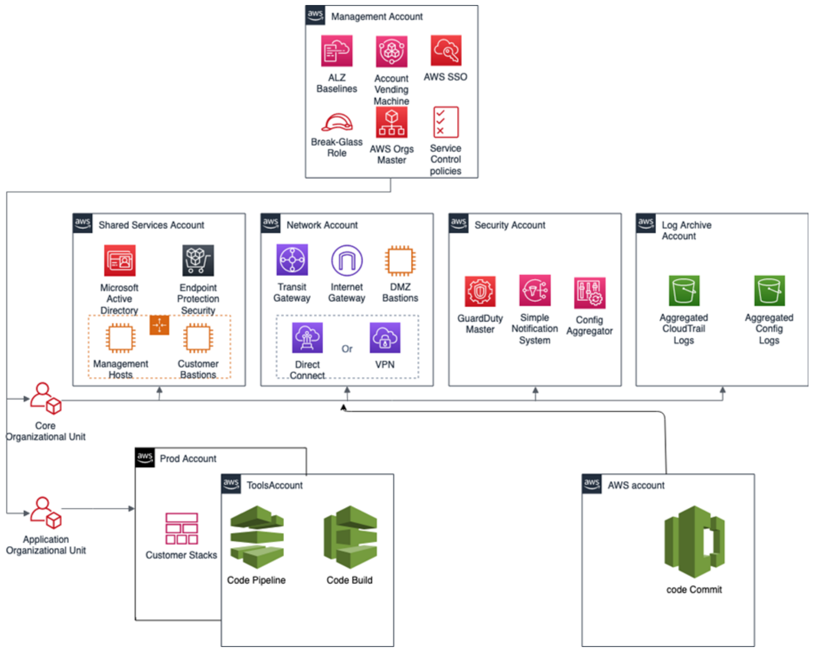
Tools account, Migrating Workloads: CloudEndure Landing Zone (MALZ)

Your Multi-Account Landing Zone tools account (with VPC) helps accelerate migration efforts, increases your security position, reduces cost and complexity, and standardizes your usage pattern.

A tools account provides the following:

- A well-defined boundary for access to replication instances for system integrators outside of your production workloads.
- Enables you to create an isolated chamber to check a workload for malware, or unknown network routes, before placing it into an account with other workloads.
- As a defined account setup, it provides faster time to onboard and get set up for migrating workloads.
- Isolated network routes to secure traffic from on-premise -> CloudEndure -> Tools account -> AMS ingested image. Once an image has been ingested, you can share the image to the destination account via an AMS Management | Advanced stack components | AMI | Share (ct-1eiczxw8ihc18) RFC.

High level architecture diagram:



Use the Deployment | Managed landing zone | Management account | Create tools account (with VPC) change type (ct-2j7q1hgf26x5c), to quickly deploy a tools account and instantiate a Workload Ingestion process within a Multi-Account Landing Zone environment. See Management account, Tools account: Creating (with VPC).

Note

We recommend having two availability zones (AZs), since this is a migration hub.
By default, AMS creates the following two security groups (SGs) in every account. Confirm the that the two SGs are present, and, if not, open a new Management | Other | Other | Create CT (ct-1e1xtak34nx76) to request them:

- SentinelDefaultSecurityGroupPrivateOnlyEgressAll
- InitialGarden-SentinelDefaultSecurityGroupPrivateOnly

Ensure that CloudEndure replication instances are created in the private subnet where there are routes back to on-premise. You can confirm that by ensuring that the route tables for the private subnet has a default route back to TGW. However, performing a CloudEndure machine cut over should go into the "isolated" private subnet where there is no route back to on-premise, only Internet outbound traffic is allowed. It is critical to ensure cutover occurs in the isolated subnet to avoid potential issues to the on-premise resources.

Prerequisites:

1. Either **Plus** or **Premium** support level.
2. The application account IDs for the KMS key where the AMIs are deployed.
3. The tools account, created as described previously.

AWS Application Migration Service (AWS MGN)

AWS Application Migration Service (AWS MGN) can be used in your MALZ Tools account through the CustomerMigrationAccessRole IAM role that is created automatically during Tools account provisioning. You can use AWS MGN to migrate applications and databases that run on supported versions of Windows and Linux operating systems.

For the most up-to-date information on AWS Region support, see the AWS Regional Services List.

If your preferred AWS Region is not currently supported by AWS MGN, or the operating system on which your applications run is not currently supported by AWS MGN, consider using the CloudEndure Migration in your Tools account instead.

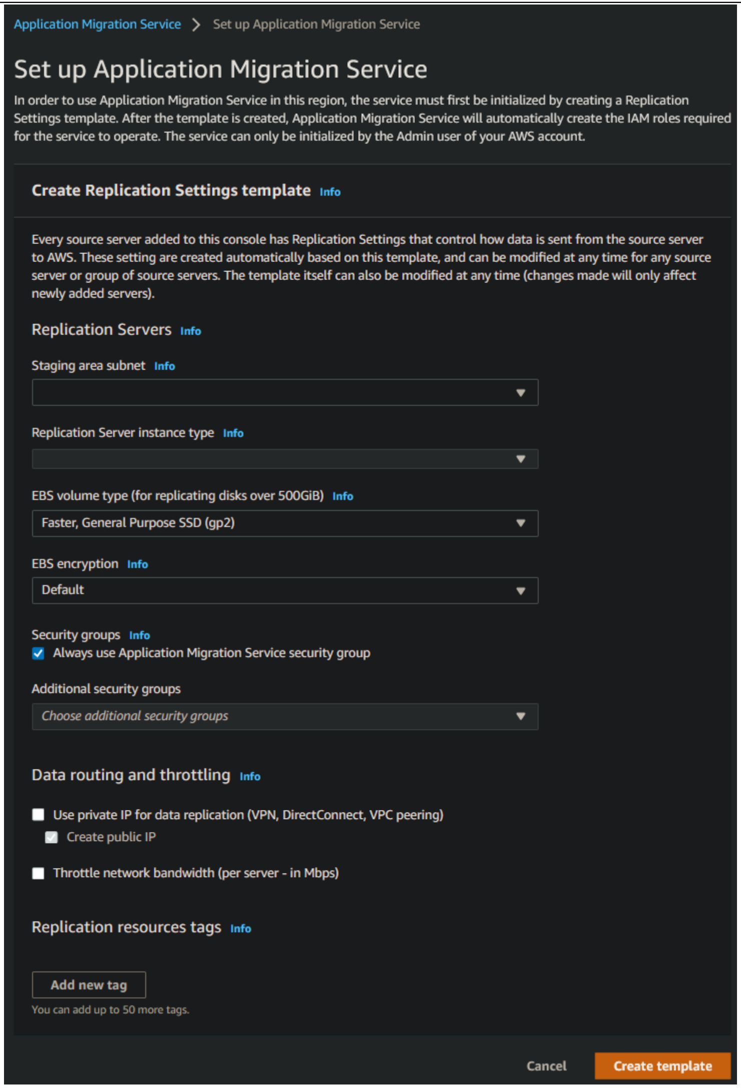
Requesting AWS MGN Initialization

AWS MGN must be initialized by AMS before first use. To request this for a new Tools account, submit a Management | Other | Other RFC from the Tools account with these details:

```
RFC Subject=Please initialize AWS MGN in this account
RFC Comment=Please click 'Get started' on the MGN welcome page here:

    https://console.aws.amazon.com/mgn/home?region=MALZ_PRIMARY_REGION#/welcome using all
 default values
    to 'Create template' and complete the initialization process.
```

Once AMS successfully completes the RFC and initializes AWS MGN in your Tools account, you can use CustomerMigrationAccessRole to edit the default template for your requirements.


Application Migration Service > Set up Application Migration Service

Set up Application Migration Service

In order to use Application Migration Service in this region, the service must first be initialized by creating a Replication Settings template. After the template is created, Application Migration Service will automatically create the IAM roles required for the service to operate. The service can only be initialized by the Admin user of your AWS account.

Create Replication Settings template Info

Every source server added to this console has Replication Settings that control how data is sent from the source server to AWS. These setting are created automatically based on this template, and can be modified at any time for any source server or group of source servers. The template itself can also be modified at any time (changes made will only affect newly added servers).

Replication Servers Info

Staging area subnet Info

Replication Server instance type Info

EBS volume type (for replicating disks over 500GiB) Info: Faster, General Purpose SSD (gp2)

EBS encryption Info: Default

Security groups Info
[✓] Always use Application Migration Service security group

Additional security groups: Choose additional security groups

Data routing and throttling Info

[ ] Use private IP for data replication (VPN, DirectConnect, VPC peering)
[✓] Create public IP

[ ] Throttle network bandwidth (per server - in Mbps)

Replication resources tags Info

You can add up to 50 more tags.

Enable access to the new Tools account

Once the tools account is created, AMS provides you with an account ID. Your next step is to configure access to the new account. Follow these steps.

1. Update the appropriate Active Directory groups to the appropriate account IDs.

   New AMS-created accounts are provisioned with the ReadOnly role policy as well as a role to allow users to file RFCs.

   The tools account also has these additional IAM roles available:
   - AMS Migration role
   - CloudEndure user role
2. Request policies and roles to allow service integration team members to set up the next level of tools.

   Navigate to the AMS console and file the following RFCs:

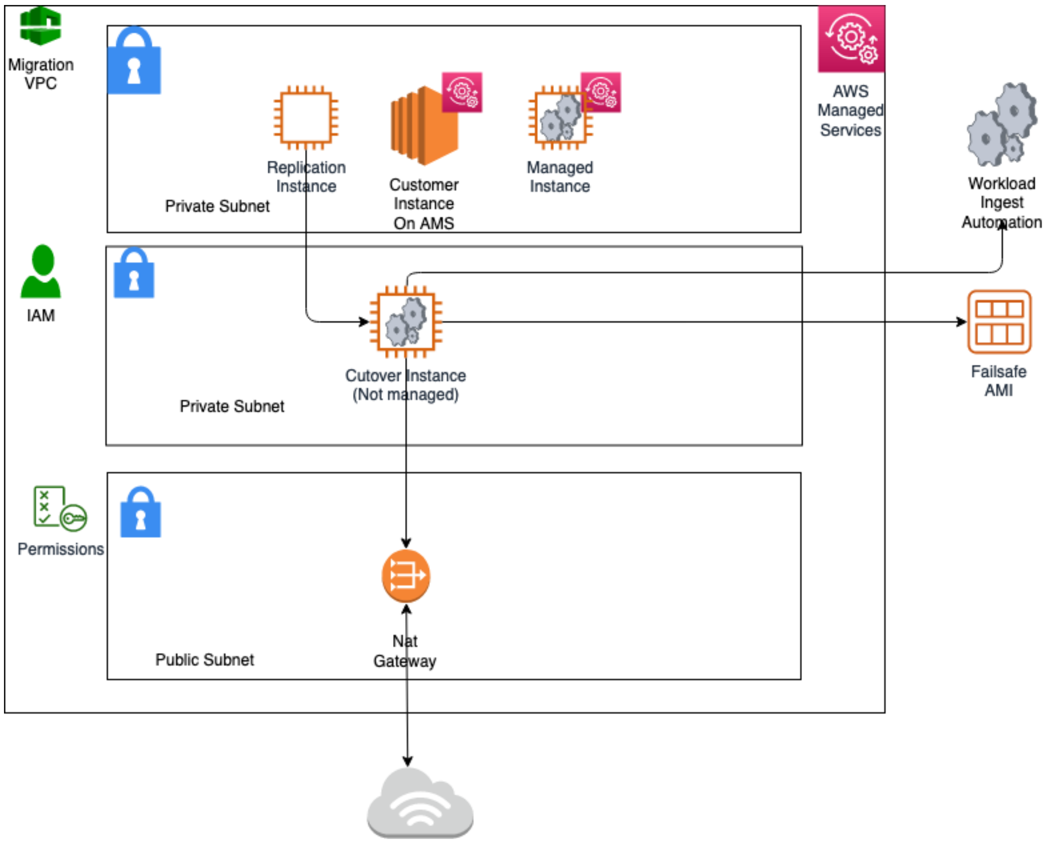
   a. Create KMS key. Use either Create KMS Key (auto) or Create KMS Key (review required).

As you use KMS to encrypt ingested resources, using a single KMS key that is shared with the rest of the Multi-Account Landing Zone application accounts, provides security for ingested images where they can be decrypted in the destination account.

b. Share the KMS key.

   Use the Management | Other | Other | Create (ct-1e1xtak34nx76) change type to request that the new KMS key be shared with your application accounts where ingested AMIs will reside.

Example graphic of a final account setup:



Example policy

To see an AMS pre-approved IAM CloudEndure policy: Unpack the WIGS Cloud Endure Landing Zone Example file and open the `customer_cloud_endure_policy.json`.

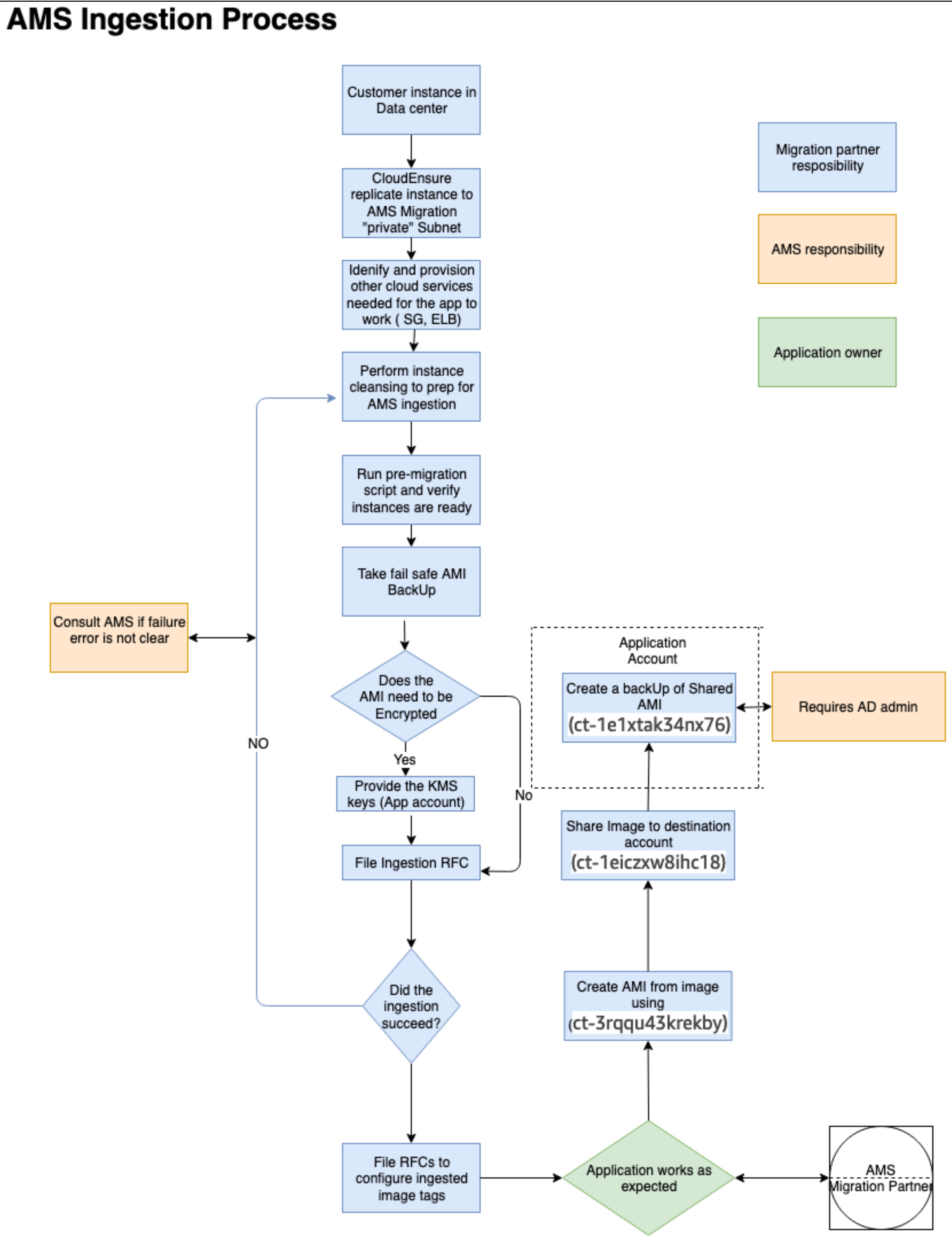
Testing connectivity and end-to-end setup

To test the tools account, follow these steps.

1. Start with configuring CloudEndure and installing the CloudEndure agent on a server that will replicate to AMS.
2. Create a project in CloudEndure.
3. Enter the AWS credentials shared when you performed the prerequisites, though secrets manager.
4. In **Replication settings**:
   a. Select both AMS "Sentinel" security groups (Private Only and EgressAll) for the **Choose the Security Groups to apply to the Replication Servers** option.

   b. Define cutover options for the machines (instances). For information, see Step 5. Cut over
   c. **Subnet**: Private subnet.
5. **Security Group**:
   a. Select both AMS "Sentinel" security groups (Private Only and EgressAll).
   b. Cutover instances have to communicate to the AMS-managed Active Directory (MAD) and to AWS public endpoints:
      i. **Elastic IP**: None
      ii. **Public IP**: no
      iii. **IAM role**: customer-mc-ec2-instance-profile
   c. Set tags as per your internal tagging convention.
6. Install the CloudEndure agent on the machine and look for the replication instance to come up in your AMS account in the EC2 console.

The AMS ingestion process:


AMS Ingestion Process
Customer instance in Data center
CloudEnsure replicate instance to AMS Migration "private" Subnet
Idenify and provision other cloud services needed for the app to work (SG, ELB)
Perform instance cleansing to prep for AMS ingestion
Run pre-migration script and verify instances are ready
Take fail safe AMI BackUp
Consult AMS if failure error is not clear
NO
Does the AMI need to be Encrypted
Yes
Provide the KMS keys (App account)
No
File Ingestion RFC
Did the ingestion succeed?
File RFCs to configure ingested image tags
Application works as expected
AMS Migration Partner
Create AMI from image using (ct-3rqqu43krekby)
Share Image to destination account (ct-1eiczxw8ihc18)
Application Account
Create a backUp of Shared AMI (ct-1e1xtak34nx76)
Requires AD admin
Migration partner resposibility
AMS responsibility
Application owner

Tools account hygiene

You'll want to clean up after you are done in the account have shared the AMI and no longer have a need for the replicated instances:

- Post instance WIGs ingestion:

- Cutover instance: At a minimum, stop or terminate this instance, after the work has been completed, via the AWS console
- Pre-Ingestion AMI backups: Remove once the instance has been ingested and the on-premise instance terminated
- AMS-ingested instances: Turn off the stack or terminate once the AMI has been shared
- AMS-ingested AMIs: Delete once sharing with the destination account is completed

- End of migration clean up: Document the resources deployed via DevMode to ensure clean-up happens on regular basis, for example:
  - Security groups
  - Resources created via Cloud-formation
  - Network ACK
  - Subnet
  - VPC
  - Route Table
  - Roles
  - User Accounts

Migration at scale - Migration Factory

See Introducing AWS CloudEndure Migration Factory Solution.

Migrating workloads: Linux pre-ingestion validation

You can validate that your instance is ready for ingestion into your AMS account. The workload ingest (WIGS) pre-ingestion validation performs checks such as operating system type, available disk space, the existence of conflicting third party software, etc. When executed, the WIGS pre-ingestion validation produces an on-screen table, with an optional log file. The results provide a pass/fail status for each validation check along with the reason for any failures. In addition, you can customize the validation tests to suit your needs.

Frequently asked questions:

- **How do I use Linux WIGS pre-ingestion validation?**

  Follow these steps to download and use the AMS Linux WIGS pre-ingestion validation scripts:

  1. Download a ZIP file with the validation scripts

     Linux WIGS Pre-ingestion Validation zip file.
  2. Unzip attached rules to a directory of your choice.
  3. Follow the instructions in the **readme.md** file.

- **What validations are performed by the Linux WIGS pre-ingestion validation?**

  The AMS Linux WIGS pre-ingestion validation solution validates the following:

  1. There is at least 5 Gigabytes free on the boot volume.
  2. The operating system is supported by AMS.
  3. The instance has a specific instance profile.
  4. The instance does not contain Antivirus software or Virtualization software.
  5. SSH is properly configured.
  6. The instance has access to Yum Repositories.
  7. Enhanced networking drivers are installed.

8. The instance has the SSM Agent and it is running.

- **Why is there support for a custom configuration file?**

  The scripts are designed to run on both on-premise physical servers and on AWS EC2 instances. However, as shown in the list above, some tests will fail when run on-premises. For example, a physical server in a datacenter would not have an instance profile. In cases like these, you can edit the configuration file to skip the instance profile test to avoid confusion.
- **How do I ensure I have the latest version of the script?**

  An up-to-date version of the Linux WIGS Pre-Ingestion Validation solution will be available under **AMS Helper Files** section on the main Documentation page.
- **Is the script read-only?**

  The script is designed to be read-only except for the log files it produces, but best practices should be followed to run the script in a non-production environment.
- **Is WIGS pre-ingestion validation available for Windows?**

  Yes. It is available under the **AMS Helper Files** section on the main Documentation page.

Migrating workloads: Windows pre-ingestion validation

You can validate that your instance is ready for ingestion into your AMS account. The workload ingest (WIGS) pre-ingestion validation performs checks such as operating system type, available disk space, the existence of conflicting third party software, etc. When executed, the WIGS pre-ingestion validation produces an on-screen table, with an optional log file. The results provide a pass/fail status for each validation check along with the reason for any failures. In addition, you can customize the validation tests to suit your needs.

Frequently asked questions:

- **How do I use Windows WIGS pre-ingestion validation?**

  Follow these steps to download and use the AMS Windows WIGS pre-ingestion validation scripts:
  1. Download a ZIP file with the validation scripts:

     Windows WIGS Pre-ingestion Validation zip file.
  2. Unzip attached rules to a directory of your choice.
  3. Follow the instructions in the **README.md** file.
- **What validations are performed by the Windows WIGS Pre-Ingestion Validation?**

  The AMS Windows WIGS pre-ingestion validation solution validates the following:
  1. There is at least 10 Gigabytes free on the boot volume.
  2. The operating system is supported by AMS.
  3. The instance has a specific instance profile.
  4. The instance does not contain antivirus software or virtualization software.
  5. DHCP is enabled on at least one network adapter.
  6. The instance is ready for Sysprep.
  7. The Windows management instrumentation (WMI) subsystem is healthy.
  8. Required drivers are installed.
  9. The SSM Agent and is installed and running.

- **Why is there support for a custom configuration file?**

  The scripts are designed to run on both on-premise physical servers and on AWS EC2 instances. However, as shown in the list above, some tests will fail when run on-premises. For example, a physical server in a datacenter would not have an instance profile. In cases like these, you can edit the configuration file to skip the instance profile test to avoid confusion.
- **How do I ensure I have the latest version of the script?**

  An up-to-date version of the Windows WIGS pre-ingestion validation solution will be available under the **AMS Helper Files** section on the main Documentation page.
- **Is the script read-only?**

  The script is designed to be read-only except for the log files it produces, but best practices should be followed to run the script in a non-production environment.
- **Is WIGS Pre-Ingestion Validation available for Linux?**

  Yes. The Linux version launched on 31 October, 2019. It is available under the **AMS Helper Files** section on the main Documentation page.

Workload Ingest Stack: Creating

You can use the AMS console or CLI to create an AMS instance from a non-AMS instance migrated to an AMS account.

Classification: Deployment | Ingestion | Stack from migration partner migrated instance | Create

Note
Be sure you have followed the prerequisites; see Migrating Workloads: Prerequisites for Linux and Windows (p. 24).

Required Data:

- **Subject**: A description of the RFC.
- **InstanceId**: The ID of the instance that your cloud migration partner created in your AMS account from your datacenter instance.
- **TargetVpcId**: The VPC in which the migrated instance should be added, in the form vpc-a1b2c3d4e5f67890e. You can find this out on the **VPCs** page of the AMS console.
- **TargetSubnetId**: The subnet that you want to launch the instance into, in the form subnet-a0b2c9d8. You can find this out on the **VPCs** -> VPC details (for the relevant VPC) page of the AMS console.
- **Description**: A reason for the request.
- **Name**: A name for the stack.

Optional Data:

- **TargetSecurityGroupIds**: IDs of the existing security groups to associate with the migrated stack. If nothing is specified, the default AMS security groups will be applied (SentinelDefaultSecurityGroupPrivateOnlyEgressAll and SentinelDefaultSecurityGroupPrivateOnly). You can find security group IDs in the EC2 console for your AMS account. For information about AMS security groups, see Default Security Groups.
- **TargetInstanceType**: The type of EC2 instance to deploy from the migrated instance. For a list, see Amazon EC2 Instance Types.
- **ApplyInstanceValidation**: Leave at default, **true**, to run AMS pre-migration validation checks on the instance. Set to **false** to not run the checks.

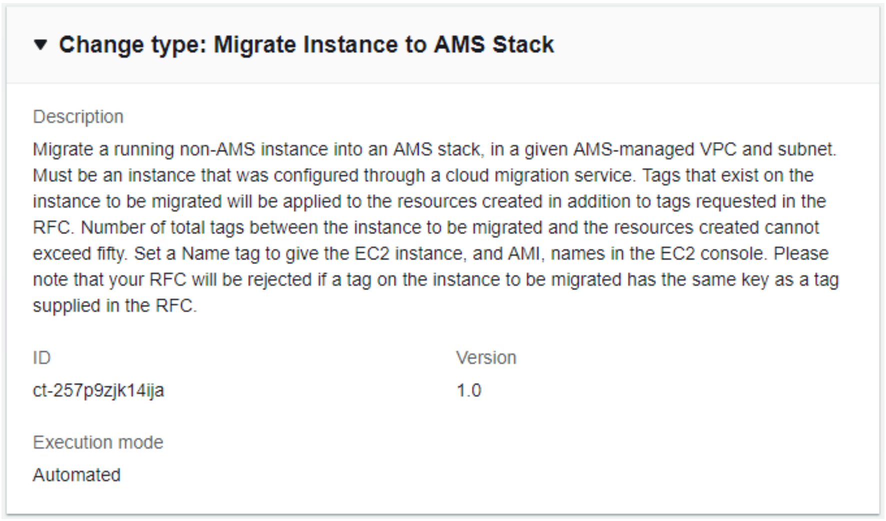
- **KmsKeyId**: KMS key to automatically encrypt the resulting AMI with. Use any format specified in the AWS EC2 copy-image API documentation.
- **Tags**: Up to fifty tags (key/value pairs) to categorize the resources created (AMI and EC2 instance). Set a Name tag to give the EC2 instance, and AMI, names in the EC2 console. Tags that exist on the instance being migrated are applied to the resources created, in addition to tags submitted in the RFC. The number of total tags between the instance being migrated and the resources created, cannot exceed fifty.

  Note
  If a tag on the instance being migrated has the same key as a tag supplied in the RFC, the RFC fails.

Note
You can specify up to four Target IDs, Ports, and Availability Zones.

Migrating an instance to an AMS stack with the Console

Screenshot of this change type in the AMS console:



▼ Change type: Migrate Instance to AMS Stack

Description

Migrate a running non-AMS instance into an AMS stack, in a given AMS-managed VPC and subnet. Must be an instance that was configured through a cloud migration service. Tags that exist on the instance to be migrated will be applied to the resources created in addition to tags requested in the RFC. Number of total tags between the instance to be migrated and the resources created cannot exceed fifty. Set a Name tag to give the EC2 instance, and AMI, names in the EC2 console. Please note that your RFC will be rejected if a tag on the instance to be migrated has the same key as a tag supplied in the RFC.

ID	Version
ct-257p9zjk14ija	1.0

Execution mode

Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Note
If the RFC is rejected, the execution output includes a link to Amazon CloudWatch logs. AMS Workload Ingest (WIGS) RFCs are rejected when requirements are not met; for example, if anti-virus software is detected on the instance. The CloudWatch logs will include information about the failed requirement and the actions to take for remediation.

Migrating an instance to an AMS stack with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

You can use the AMS CLI to create an AMS instance from a non-AMS instance migrated to an AMS account.

Note
Be sure you have followed the prerequisites; see Migrating Workloads: Prerequisites for Linux and Windows (p. 24).

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotes when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws amscm create-rfc --change-type-id "ct-257p9zjk14ija" --change-type-version "1.0" --
title "AMS-WIG-TEST-NO-ACTION" --execution-parameters "{\"InstanceId\":\"INSTANCE_ID\",
\"TargetVpcId\":\"VPC_ID\",\"TargetSubnetId\":\"SUBNET_ID\",\"TargetInstanceType\":
\"t2.large\",\"ApplyInstanceValidation\":true,\"Name\":\"WIG-TEST\",\"Description\":\"WIG-
TEST\"}"
```

TEMPLATE CREATE:

1. Output the execution parameters JSON schema for this change type to a file; example names it MigrateStackParams.json:

```
aws amscm get-change-type-version --change-type-id "ct-257p9zjk14ija" --query
 "ChangeTypeVersion.ExecutionInputSchema" --output text > MigrateStackParams.json
```

2. Modify and save the execution parameters JSON file. For example, you can replace the contents with something like this:

```
{
"InstanceId":             "MIGRATED_INSTANCE_ID",
"TargetVpcId":            "VPC_ID",
"TargetSubnetId":         "SUBNET_ID",
"Name":                   "Migrated-Stack",
"Description":            "Create-Migrated-Stack"
}
```

3. Output the RFC template JSON file; example names it MigrateStackRfc.json:

```
aws amscm create-rfc --generate-cli-skeleton > MigrateStackRfc.json
```

4. Modify and save the MigrateStackRfc.json file. For example, you can replace the contents with something like this:

```
{
"ChangeTypeId":           "ct-257p9zjk14ija",
"ChangeTypeVersion":      "1.0",
"Title":                  "Migrate-Stack-RFC"
}
```

5. Create the RFC, specifying the MigrateStackRfc file and the MigrateStackParams file:

```
aws amscm create-rfc --cli-input-json file://MigrateStackRfc.json  --execution-
parameters file://MigrateStackParams.json
```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

   The new instance appears in the Instance list for the application owner's account for the relevant VPC.

6. Once the RFC completes successfully, notify the application owner so he or she can log into the new instance and verify that the workload is operational.

Note
If the RFC is rejected, the execution output includes a link to Amazon CloudWatch logs. AMS Workload Ingest (WIGS) RFCs are rejected when requirements are not met; for example, if anti-virus software is detected on the instance. The CloudWatch logs will include information about the failed requirement and the actions to take for remediation.

If needed, see Workload ingestion (WIGS) failure.

AMS CloudFormation ingest

The AMS AWS CloudFormation ingest change type (CT) enables you to use your existing CloudFormation templates, with some modifications, to deploy custom stacks in an AMS-managed VPC.

Topics

The AMS AWS CloudFormation ingest process involves the following:

- Prepare and upload your custom CloudFormation template to an S3 bucket, or provide the template inline when creating the RFC. If you are using an S3 bucket with a presigned URL, see presign for more information.
- Submit the CloudFormation ingest change type to AMS in an RFC.
- Once your stack is created, you can update it, and remediate drift on it; additionally, should the update fail, you can explicitly approve and implement the update. All of these procedures are described in this section.

  For information on CFN drift detection, see New – CloudFormation Drift Detection.

  Note

  - This change type now has a version 2.0. Version 2.0 is automated; not manually executed. This enables the CT execution to go more quickly. Two new parameters are introduced with this version: **CloudFormationTemplate**, which enables you to paste a custom CloudFormation template into the RFC, and **VpcId**, which enables CloudFormation ingest to be used with AMS multi-account landing zone.
  - Version 1.0 is a manual change type. This means that an AMS operator must take some action before the change type can successfully conclude. At minimum, a review is required. This version also requires the **CloudFormationTemplateS3Endpoint** parameter value to be a pre-signed URL.

AWS CloudFormation Ingest Guidelines, Best Practices, and Limitations

For AMS to process your CloudFormation template, there are some guidelines and restrictions, which are described in this section.

Guidelines

To reduce AWS CloudFormation errors while performing AWS CloudFormation ingest, follow these guidelines:

- **Don't embed credentials or other sensitive information in the template** – The CloudFormation template is visible in the AWS CloudFormation console, so you don't want to embed credentials or sensitive data in the template. The template can't contain sensitive information. The following resources are allowed only if you use AWS Secrets Manager for the value:
  - `AWS::RDS::DBInstance` - [MasterUserPassword,TdeCredentialPassword]
  - `AWS::RDS::DBCluster` - [MasterUserPassword]
  - `AWS::ElastiCache::ReplicationGroup` - [AuthToken]

Note

For information about using an AWS Secrets Manager secret in a resource property, see How to create and retrieve secrets managed in AWS Secrets Manager using AWS CloudFormation templates and Using Dynamic References to Specify Template Values.

- **Use Amazon RDS snapshots to create RDS DB instances** – By doing this you avoid having to provide a MasterUserPassword.
- If the template you submit contains an IAM instance profile, it must be prefixed with 'customer'. For example, using an instance profile with the name 'example-instance-profile', causes failure. Instead, use an instance profile with the name 'customer-example-instance-profile'.
- **Don't include any sensitive data in `AWS::EC2::Instance`** - [UserData]. UserData should not contain passwords, API keys, or any other sensitive data. This type of data can be encrypted and stored in an S3 bucket and downloaded onto the instance using UserData.
- **IAM policy creation using CloudFormation templates isn't supported** – IAM policies have to be reviewed and approved by AMS SecOps. IAM policies can't be created using CloudFormation templates because that would override the AMS SecOps process.
- **SSH KeyPairs aren't supported** – EC2 instances must be accessed through the AMS access management system. The AMS RFC process authenticates you. You cannot include SSH keypairs in CloudFormation templates because you don't have the permissions to create SSH keypairs and override the AMS access management model.
- **Security Group ingress rules are restricted** – You can't have a source CIDR range from 0.0.0.0/0, or a publicly routable address space, with a TCP port that is anything other than 80 or 443.
- **Follow AWS CloudFormation guidelines when writing CloudFormation resource templates** – Ensure that you use the right data type/property name for the resource by referring to the *AWS CloudFormation User Guide* for that resource. For example, the data type of SecurityGroupIds property in an AWS::EC2::Instance resource is 'List of String values', so ["sg-aaaaaaaa"] is ok (with brackets), but "sg-aaaaaaaa" is not (without brackets).

  For more information, see AWS Resource and Property Types Reference.

- **Configure your custom CloudFormation templates to use parameters defined in the AMS CloudFormation ingest CT** – When you configure your CloudFormation template to use parameters defined in the AMS CloudFormation ingest CT, you can reuse the CloudFormation template to create similar stacks by submitting it with changed parameter values in the CT input with the Management | Custom stack | Stack from CloudFormation template | Update CT (ct-361tlo1k7339x). For an example, see AWS CloudFormation Ingest Examples: Defining Resources (p. 57).

- **S3 bucket endpoints with a presigned URL can't be expired** – If you are using an S3 bucket endpoint with a presigned URL, verify that the presigned S3 URL isn't expired. A CloudFormation ingest RFC submitted with an expired presigned S3 bucket URL will be rejected.

- **Wait Condition requires signal logic** – Wait Condition is used to coordinate stack resource creation with configuration actions that are external to the stack creation. If you use the Wait Condition resource in the template, AWS CloudFormation waits for a success signal, and it marks stack creation as a failure if the number of success signals aren't made. You need to have a logic for the signal if you use the Wait Condition resource. For more information, see Creating Wait Conditions in a Template.

Best Practices

Following are some best practices you can use to migrate resources using the AMS AWS CloudFormation ingest process:

- **Submit IAM and other policy-related resources in one CT**– AMS recommends that you gather all IAM or other policy-related resources and submit them in a single Management | Other | Other | Create change type (ct-1e1xtak34nx76). For example, combine needed all IAM roles, IAM EC2 instance

profiles, IAM policy updates for existing IAM roles, S3 Bucket policies, SNS/SQS policies, and so forth, and submit a ct-1e1xtak34nx76 RFC so that these pre-existing resources can simply be referenced inside the future CloudFormation ingest templates.
- **EC2 instances are bootstrapped and successfully joined to the domain** – This is done automatically as a best practice. To ensure that the EC2 instances launched via a CloudFormation ingest stack are bootstrapped and join the domain successfully, AMS includes a CreationPolicy and an UpdatePolicy for an Auto Scaling group resource (that is, if these policies don't already exist).
- **Amazon RDS DB instance parameter must be specified** – When creating an RDS database via AWS CloudFormation ingest, you must specify the `DBSnapshotIdentifier` parameter in order to restore from a previous DB snapshot. This is required because AWS CloudFormation ingest does not currently handle sensitive data.

For an example of how to use a CloudFormation template for AMS CloudFormation template ingest, see AWS CloudFormation Ingest: Examples (p. 56).

Template Validation

You can self-validate your CloudFormation template before submitting it to AMS.

Templates submitted to AMS AWS CloudFormation ingest are validated to ensure they are safe to deploy within an AMS Account. The validation process checks the following:

- **Supported resources** – Only AMS AWS CloudFormation ingest-supported resources are used. For more information, see Supported Resources (p. 49).
- **Supported AMIs** – The AMI in the template is an AMS-supported AMI. For information about AMS AMIs, see AMS Amazon Machine Images (AMIs) (p. 2).
- **AMS Shared Services subnet** – The template does not attempt to launch resources into the AMS Shared Services subnet.
- **Resource policies** – There are no overly permissive resource policies, such as a publicly readable or writeable S3 bucket policy. AMS doesn't allow publicly readable or writable S3 buckets in AWS accounts.

Validate with AWS CloudFormation Linter

You can self-validate your CloudFormation template before submitting it to AMS by using the AWS CloudFormation Linter tool.

The AWS CloudFormation Linter tool is the best way to validate your CloudFormation template as it provides validation for resource/property names, data types, and functions. For more information, see aws-cloudformation/cfn-python-lint.

The AWS CloudFormation Linter output of the template shown previously is as follows:

```
$ cfn-lint -t ./testtmpl.json
E3002 Invalid Property Resources/SNSTopic/Properties/Name
./testtmpl.json:6:9
```

To assist with offline validation of CloudFormation templates, AMS has developed a set of pluggable custom validation rules for the AWS CloudFormation Linter tool. They're located on the main Documentation page of the AMS console.

Follow these steps to use AWS CloudFormation pre-ingestion validation scripts:

1. Install the AWS CloudFormation Linter tool. For installation instructions, see aws-cloudformation / cfn-lint .

2. Download a .zip file with validation scripts:

   CFN Lint Custom Rules.
3. Unzip the attached rules to a directory of your choice.
4. Validate your CloudFormation template by running the following command:

```
cfn-lint --template {TEMPLATE_FILE} --append-rules {DIRECTORY_WITH_CUSTOM_RULES}
```

CloudFormation ingest stack: CFN validator examples

These examples can help you prepare your template for a successful ingest.

Format validation

Validate that the template contains a "Resources" section, and all resources defined under it have a "Type" value.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description" : "Create a SNS topic",
  "Resources": {
    "SnsTopic": {
      "Type": "AWS::SNS::Topic"
    }
  }
}
```

Validate that the root keys of the template are allowed. Allowed root keys are:

```
[
  "AWSTemplateFormatVersion",
  "Description",
  "Mappings",
  "Parameters",
  "Conditions",
  "Resources",
  "Rules",
  "Outputs",
  "Metadata"
]
```

Manual review required validation

If the template contains the following resources, automatic validation fails and you'll need a manual review.

The shown policies are high risk areas from a security standpoint. For example, an S3 bucket policy allowing anyone except for specific users or groups to create objects or write permissions, is extremely dangerous. So we validate the policies and approve or deny based on the contents, and those polices cannot be auto-created. We are investigating possible approaches to address this issue.

We currently don’t have automated validation around the following resources.

```
[
    "S3::BucketPolicy",
    "SNS::TopicPolicy",
    "SQS::QueuePolicy"
]
```

Parameter validation

Validate that if a template parameter doesn't have a value provided; it must have a default value.

Resource attribute validation

Required attribute check: Certain attributes must exist for certain resource types.

- "VPCOptions" must exist in `AWS::Elasticsearch::Domain`
- "CludsterSubnetGroupName" must exist in `AWS::Redshift::Cluster`

```
{
    "AWS::Elasticsearch::Domain": [
      "VPCOptions"
    ],
    "AWS::Redshift::Cluster": [
      "ClusterSubnetGroupName"
    ]
}
```

Disallowed attributes check: Certain attributes must *not* exist for certain resource types.

- "SecretString" must not exist in "AWS::SecretsManager::Secret"
- "MongoDbSettings" must not exist in "AWS::DMS::Endpoint"

```
{
  "AWS::SecretsManager::Secret": [
    "SecretString"
  ],
  "AWS::DMS::Endpoint": [
    "MongoDbSettings"
  ]
}
```

SSM parameter check: For attributes in the following list, values must be specified via Secrets Manager or Systems Manager Parameter Store (Secure String Parameter):

```
{
  "RDS::DBInstance": [
    "MasterUserPassword",
    "TdeCredentialPassword"
  ],
  "RDS::DBCluster": [
    "MasterUserPassword"
  ],
  "ElastiCache::ReplicationGroup": [
    "AuthToken"
  ],
  "DMS::Certificate": [
    "CertificatePem",
    "CertificateWallet"
  ],
  "DMS::Endpoint": [
    "Password"
  ],
  "CodePipeline::Webhook": {
    "AuthenticationConfiguration": [
        "SecretToken"
    ]
  },
  "DocDB::DBCluster": [
    "MasterUserPassword"
  ]
},
```

Some attributes must comply with certain patterns; for example, IAM instance profile names must not start with AMS reserved prefixes, and the attribute value must match the specific regex as shown:

```
{
    "AWS::EC2::Instance": {
      "IamInstanceProfile": [
        "^(?!arn:aws:iam|ams|Ams|AMS|AWSManagedServices|Managed_Services|mc|Mc|MC|sentinel|
Sentinel).+",
        "arn:aws:iam::(\\$\\{AWS::AccountId\\}|[0-9]+):instance-profile/(?!ams|Ams|AMS|
AWSManagedServices|Managed_Services|mc|Mc|MC|sentinel|Sentinel).+"
      ]
    },
    "AWS::AutoScaling::LaunchConfiguration": {
      "IamInstanceProfile": [
        "^(?!arn:aws:iam|ams|Ams|AMS|AWSManagedServices|Managed_Services|mc|Mc|MC|sentinel|
Sentinel).+",
        "arn:aws:iam::(\\$\\{AWS::AccountId\\}|[0-9]+):instance-profile/(?!ams|Ams|AMS|
AWSManagedServices|Managed_Services|mc|Mc|MC|sentinel|Sentinel).+"
      ]
    },
    "AWS::EC2::LaunchTemplate": {
      "LaunchTemplateData.IamInstanceProfile.Name": [
        "^(?!ams|Ams|AMS|AWSManagedServices|Managed_Services|mc|Mc|MC|sentinel|Sentinel).+"
      ],
      "LaunchTemplateData.IamInstanceProfile.Arn": [
        "arn:aws:iam::(\\$\\{AWS::AccountId\\}|[0-9]+):instance-profile\/(?!ams|Ams|AMS|
AWSManagedServices|Managed_Services|mc|Mc|MC|sentinel|Sentinel).+"
      ]
    }
}
```

Resource validation

Only allowlisted resources can be specified in the template; those resources are described in Supported Resources (p. 49).

EC2 stacks and Auto Scaling groups (ASGs) are not allowed in the same stack due to patching limitations.

Security group ingress rule validation

- For requests that come from the CFN Ingest Create or Stack Update CT change types:
  - If (`IpProtocol` is tcp or 6) AND (Port is 80 or 443) , there are no restrictions around the `CidrIP` value
  - Otherwise, the `CidrIP` cannot be 0.0.0.0/0
- For requests that come from Service Catalog (Service Catalog products):
  - In addition to the CFN Ingest Create or Stack Update CT change type validation, the port in `management_ports` with the protocol in `ip_protocols` can only be accessed via `allowed_cidrs`:

```
{
      "ip_protocols": ["tcp", "6", "udp", "17"],
      "management_ports": [22, 23, 389, 636, 1494, 1604, 2222, 3389, 5900, 5901, 5985,
 5986],
      "allowed_cidrs": ["10.0.0.0/8", "100.64.0.0/10", "172.16.0.0/12",
 "192.168.0.0/16"]
  }
```

Limitations

The following features and functionality currently aren't supported by the AMS AWS CloudFormation ingest process.

- **YAML** – Not supported. Only JSON-based CloudFormation templates are supported.
- **Nested stacks** – Instead, architect your application infrastructure to use a single template. Or, alternatively you can make use of cross-stack referencing to separate resources across multiple stacks where one resource has a dependency on another. For more information, see Walkthrough: Refer to Resource Outputs in Another AWS CloudFormation Stack.
- **CloudFormation stack sets** – Not supported, due to security implications.
- **IAM role creation using CloudFormation templates** – Not supported, due to security implications.
- **Sensitive data** – Not supported. Do not include sensitive data in the template or in the parameter values. If you need to reference sensitive data, use Secrets Manager to store and retrieve these values. For information about using AWS Secrets Managers secrets in a resource property, see How to create and retrieve secrets managed in AWS Secrets Manager using AWS CloudFormation templates and Using Dynamic References to Specify Template Values.

Supported Resources

The following AWS resources are supported in the AMS AWS CloudFormation ingest process.

CloudFormation Ingest Stack: Supported resources

The instance operating system must be supported by AMS workload ingestion. Only those AWS resources listed here are supported.

- Amazon API Gateway
  - AWS::ApiGateway::Account
  - AWS::ApiGateway::ApiKey
  - AWS::ApiGateway::Authorizer
  - AWS::ApiGateway::BasePathMapping
  - AWS::ApiGateway::ClientCertificate
  - AWS::ApiGateway::Deployment
  - AWS::ApiGateway::DocumentationPart
  - AWS::ApiGateway::DocumentationVersion
  - AWS::ApiGateway::DomainName
  - AWS::ApiGateway::GatewayResponse
  - AWS::ApiGateway::Method
  - AWS::ApiGateway::Model
  - AWS::ApiGateway::RequestValidator
  - AWS::ApiGateway::Resource
  - AWS::ApiGateway::RestApi
  - AWS::ApiGateway::Stage

- AWS::ApiGateway::UsagePlan
- AWS::ApiGateway::UsagePlanKey
- AWS::ApiGateway::VpcLink
- Amazon API Gateway V2
  - AWS::ApiGatewayV2::Api
  - AWS::ApiGatewayV2::ApiGatewayManagedOverrides
  - AWS::ApiGatewayV2::ApiMapping
  - AWS::ApiGatewayV2::Authorizer
  - AWS::ApiGatewayV2::Deployment
  - AWS::ApiGatewayV2::DomainName
  - AWS::ApiGatewayV2::Integration
  - AWS::ApiGatewayV2::IntegrationResponse
  - AWS::ApiGatewayV2::Model
  - AWS::ApiGatewayV2::Route
  - AWS::ApiGatewayV2::RouteResponse
  - AWS::ApiGatewayV2::Stage
  - AWS::ApiGatewayV2::VpcLink
- AWS AppSync
  - AWS::AppSync::ApiCache
  - AWS::AppSync::ApiKey
  - AWS::AppSync::DataSource
  - AWS::AppSync::FunctionConfiguration
  - AWS::AppSync::GraphQLApi
  - AWS::AppSync::GraphQLSchema
  - AWS::AppSync::Resolver
- Amazon Athena
  - AWS::Athena::NamedQuery
  - AWS::Athena::WorkGroup
- AWS Backup
  - AWS::Backup::BackupVault
- Amazon CloudFront
  - AWS::CloudFront::Distribution
  - AWS::CloudFront::CloudFrontOriginAccessIdentity
  - AWS::CloudFront::StreamingDistribution
- Amazon CloudWatch
  - AWS::CloudWatch::Alarm
  - AWS::CloudWatch::AnomalyDetector
  - AWS::CloudWatch::CompositeAlarm
  - AWS::CloudWatch::Dashboard
  - AWS::CloudWatch::InsightRule
- Amazon CloudWatch Logs
  - AWS::Logs::LogGroup
  - AWS::Logs::LogStream
  - AWS::Logs::MetricFilter
  - AWS::Logs::SubscriptionFilter

- Amazon Cognito
  - AWS::Cognito::IdentityPool
  - AWS::Cognito::IdentityPoolRoleAttachment
  - AWS::Cognito::UserPool
  - AWS::Cognito::UserPoolClient
  - AWS::Cognito::UserPoolDomain
  - AWS::Cognito::UserPoolGroup
  - AWS::Cognito::UserPoolIdentityProvider
  - AWS::Cognito::UserPoolResourceServer
  - AWS::Cognito::UserPoolRiskConfigurationAttachment
  - AWS::Cognito::UserPoolUICustomizationAttachment
  - AWS::Cognito::UserPoolUser
  - AWS::Cognito::UserPoolUserToGroupAttachment
- Amazon DocumentDB
  - AWS::DocDB::DBCluster
  - AWS::DocDB::DBClusterParameterGroup
  - AWS::DocDB::DBInstance
  - AWS::DocDB::DBSubnetGroup
- Amazon DynamoDB
  - AWS::DynamoDB::Table
- Amazon EC2
  - AWS::EC2::Volume
  - AWS::EC2::VolumeAttachment
  - AWS::EC2::Instance
  - AWS::EC2::EIP
  - AWS::EC2::EIPAssociation
  - AWS::EC2::NetworkInterface
  - AWS::EC2::NetworkInterfaceAttachment
  - AAWS::EC2::SecurityGroup
  - AWS::EC2::SecurityGroupIngress
  - AWS::EC2::SecurityGroupEgress
  - AWS::EC2::LaunchTemplate
- AWS Batch
  - AWS::Batch::ComputeEnvironment
  - AWS::Batch::JobDefinition
  - AWS::Batch::JobQueue
- Amazon Elastic Container Registry (ECR)
  - AWS::ECR::Repository
- Amazon Elastic Container Service (ECS) (Fargate)
  - AWS::ECS::CapacityProvider
  - AWS::ECS::Cluster
  - AWS::ECS::PrimaryTaskSet
  - AWS::ECS::Service
  - AWS::ECS::TaskDefinition
  - AWS::ECS::TaskSet
- Amazon Elastic File System (EFS)

  - AWS::EFS::FileSystem
  - AWS::EFS::MountTarget
- Amazon ElastiCache
  - AWS::ElastiCache::CacheCluster
  - AWS::ElastiCache::ParameterGroup
  - AWS::ElastiCache::ReplicationGroup
  - AWS::ElastiCache::SecurityGroup
  - AWS::ElastiCache::SecurityGroupIngress
  - AWS::ElastiCache::SubnetGroup
- Amazon EventBridge
  - AWS::Events::EventBus
  - AWS::Events::EventBusPolicy
  - AWS::Events::Rule
- Amazon FSx
  - AWS::FSx::FileSystem
- Amazon Inspector
  - AWS::Inspector::AssessmentTarget
  - AWS::Inspector::AssessmentTemplate
  - AWS::Inspector::ResourceGroup
- Amazon Kinesis Data Analytics
  - AWS::KinesisAnalytics::Application
  - AWS::KinesisAnalytics::ApplicationOutput
  - AWS::KinesisAnalytics::ApplicationReferenceDataSource
- Amazon Kinesis Data Firehose
  - AWS::KinesisFirehose::DeliveryStream
- Amazon Kinesis Data Streams
  - AWS::Kinesis::Stream
  - AWS::Kinesis::StreamConsumer
- Amazon MQ
  - AWS::AmazonMQ::Broker
  - AWS::AmazonMQ::Configuration
  - AWS::AmazonMQ::ConfigurationAssociation
- Amazon Relational Database Service (RDS)
  - AWS::RDS::DBCluster
  - AWS::RDS::DBClusterParameterGroup
  - AWS::RDS::DBInstance
  - AWS::RDS::DBParameterGroup
  - AWS::RDS::DBSubnetGroup
  - AWS::RDS::EventSubscription
  - AWS::RDS::OptionGroup
- Amazon Route 53
  - AWS::Route53::RecordSet
  - AWS::Route53::HostedZone
  - AWS::Route53::HealthCheck
  - AWS::Route53::RecordSetGroup

- Amazon S3
  - AWS::S3::Bucket
  - AWS::S3::BucketPolicy
- Amazon Sagemaker
  - AWS::SageMaker::CodeRepository
  - AWS::SageMaker::Endpoint
  - AWS::SageMaker::EndpointConfig
  - AWS::SageMaker::Model
  - AWS::SageMaker::NotebookInstance
  - AWS::SageMaker::NotebookInstanceLifecycleConfig
  - AWS::SageMaker::Workteam
- Amazon Simple Email Service (SES)
  - AWS::SES::ConfigurationSet
  - AWS::SES::ConfigurationSetEventDestination
  - AWS::SES::ReceiptFilter
  - AWS::SES::ReceiptRule
  - AWS::SES::ReceiptRuleSet
  - AWS::SES::Template
- Amazon SimpleDB
  - AWS::SDB::Domain
- Amazon SNS
  - AWS::SNS::Subscription
  - AWS::SNS::Topic
  - AWS::SNS::TopicPolicy (see note at end)
- Amazon SQS
  - AWS::SQS::Queue
  - AWS::SQS::QueuePolicy (see note at end)
- Amazon WorkSpaces
  - AWS::WorkSpaces::Workspace
- Application AutoScaling
  - AWS::ApplicationAutoScaling::ScalableTarget
  - AWS::ApplicationAutoScaling::ScalingPolicy
- Amazon EC2 AutoScaling
  - AWS::AutoScaling::AutoScalingGroup
  - AWS::AutoScaling::LaunchConfiguration
  - AWS::AutoScaling::LifecycleHook
  - AWS::AutoScaling::ScalingPolicy
  - AWS::AutoScaling::ScheduledAction
- AWS Certificate Manager
  - AWS::CertificateManager::Certificate
- AWS CloudFormation
  - AWS::CloudFormation::CustomResource
  - AWS::CloudFormation::Designer
  - AWS::CloudFormation::WaitCondition
  - AWS::CloudFormation::WaitConditionHandle

- AWS CodeBuild
  - AWS::CodeBuild::Project
  - AWS::CodeBuild::ReportGroup
  - AWS::CodeBuild::SourceCredential
- AWS CodeCommit
  - AWS::CodeCommit::Repository
- AWS CodeDeploy
  - AWS::CodeDeploy::Application
  - AWS::CodeDeploy::DeploymentConfig
  - AWS::CodeDeploy::DeploymentGroup
- AWS CodePipeline
  - AWS::CodePipeline::CustomActionType
  - AWS::CodePipeline::Pipeline
  - AWS::CodePipeline::Webhook
- AWS Database Migration Service (DMS)
  - AWS::DMS::Certificate
  - AWS::DMS::Endpoint
  - AWS::DMS::EventSubscription
  - AWS::DMS::ReplicationInstance
  - AWS::DMS::ReplicationSubnetGroup
  - AWS::DMS::ReplicationTask

  The MongoDbSettings property in AWS::DMS::Endpoint resource is not allowed.

  The following properties are only allowed if they are resolved by AWS Secrets Manager: CertificatePem and CertificateWallet properties in the AWS::DMS::Certificate resource, and the Password property in the AWS::DMS::Endpoint resource.
- AWS Elastic Load Balancing - Application Load Balancer / Network Load Balancer
  - AWS::ElasticLoadBalancingV2::Listener
  - AWS::ElasticLoadBalancingV2::ListenerCertificate
  - AWS::ElasticLoadBalancingV2::ListenerRule
  - AWS::ElasticLoadBalancingV2::LoadBalancer
  - AWS::ElasticLoadBalancingV2::TargetGroup
- AWS Elastic Load Balancing - Classic Load Balancer
  - AWS::ElasticLoadBalancing::LoadBalancer
- AWS Elemental MediaConvert
  - AWS::MediaConvert::JobTemplate
  - AWS::MediaConvert::Preset
  - AWS::MediaConvert::Queue
- AWS Elemental MediaStore
  - AWS::MediaStore::Container
- AWS Managed Streaming for Apache Kafka (MSK)
  - AWS::MSK::Cluster
- AWS Glue
  - AWS::Glue::Classifier
  - AWS::Glue::Connection
  - AWS::Glue::Crawler

  - AWS::Glue::Database
  - AWS::Glue::DataCatalogEncryptionSettings
  - AWS::Glue::DevEndpoint
  - AWS::Glue::Job
  - AWS::Glue::MLTransform
  - AWS::Glue::Partition
  - AWS::Glue::SecurityConfiguration
  - AWS::Glue::Table
  - AWS::Glue::Trigger
  - AWS::Glue::Workflow
- AWS Key Management Service (KMS)
  - AWS::KMS::Key
  - AWS::KMS::Alias
- AWS Lake Formation
  - AWS::LakeFormation::DataLakeSettings
  - AWS::LakeFormation::Permissions
  - AWS::LakeFormation::Resource
- AWS Lambda
  - AWS::Lambda::Alias
  - AWS::Lambda::EventInvokeConfig
  - AWS::Lambda::EventSourceMapping
  - AWS::Lambda::Function
  - AWS::Lambda::LayerVersion
  - AWS::Lambda::LayerVersionPermission
  - AWS::Lambda::Permission
  - AWS::Lambda::Version
- Amazon Redshift
  - AWS::Redshift::Cluster
  - AWS::Redshift::ClusterParameterGroup
  - AWS::Redshift::ClusterSubnetGroup
- AWS Secrets Manager
  - AWS::SecretsManager::ResourcePolicy
  - AWS::SecretsManager::RotationSchedule
  - AWS::SecretsManager::Secret
  - AWS::SecretsManager::SecretTargetAttachment
- AWS Security Hub
  - AWS::SecurityHub::Hub
- AWS Step Functions
  - AWS::StepFunctions::Activity
  - AWS::StepFunctions::StateMachine
- AWS Systems Manager (SSM)
  - AWS::SSM::Parameter
- Amazon CloudWatch Synthetics
  - AWS::Synthetics::Canary
- AWS Transfer Family

  - AWS::Transfer::Server
  - AWS::Transfer::User
- AWS WAF
  - AWS::WAF::ByteMatchSet
  - AWS::WAF::IPSet
  - AWS::WAF::Rule
  - AWS::WAF::SizeConstraintSet
  - AWS::WAF::SqlInjectionMatchSet
  - AWS::WAF::WebACL
  - AWS::WAF::XssMatchSet
- AWS WAF Regional
  - AWS::WAFRegional::ByteMatchSet
  - AWS::WAFRegional::GeoMatchSet
  - AWS::WAFRegional::IPSet
  - AWS::WAFRegional::RateBasedRule
  - AWS::WAFRegional::RegexPatternSet
  - AWS::WAFRegional::Rule
  - AWS::WAFRegional::SizeConstraintSet
  - AWS::WAFRegional::SqlInjectionMatchSet
  - AWS::WAFRegional::WebACL
  - AWS::WAFRegional::WebACLAssociation
  - AWS::WAFRegional::XssMatchSet
- AWS WAFv2
  - AWS::WAFv2::IPSet
  - AWS::WAFv2::RegexPatternSet
  - AWS::WAFv2::RuleGroup
  - AWS::WAFv2::WebACL
  - AWS::WAFv2::WebACLAssociation

Note
AWS::S3::BucketPolicy, AWS::SQS::QueuePolicy, and AWS::SNS::TopicPolicy resources require manual validation by an AMS Operations engineer. Including one of those resources in your CloudFormation template increases validation time.

AWS CloudFormation Ingest: Examples

This section provides some detailed examples of how to use the Create stack with CloudFormation template change type.

To download a set of sample CloudFormation templates per AWS Region, see Sample Templates.

For reference information on AWS CloudFormation resources, see AWS Resource and Property Types Reference. However, AMS supports a smaller set of resources, which are described in AMS CloudFormation ingest (p. 42).

Note
AMS advises you to gather all IAM or other policy-related resources and submit them in a single Management | Other | Other | Create change type (ct-1e1xtak34nx76). For example, combine all needed IAM roles, IAM EC2 instance profiles, IAM policy updates for existing IAM roles, S3 bucket

policies, SNS/SQS policies, and so forth, and then submit a ct-1e1xtak34nx76 RFC so that these pre-existing resources can be referenced inside the future CFN Ingest templates.

Topics

AWS CloudFormation Ingest Examples: Defining Resources

When using AMS AWS CloudFormation ingest, you customize a CloudFormation template and submit it to AMS in an RFC with the CloudFormation ingest change type (ct-36cn2avfrrj9v). To create a CloudFormation template that can be reused multiple times, you add the stack configuration parameters to the CloudFormation ingest change type execution input rather than hard coding them in the CloudFormation template. The biggest benefit is that you can reuse the template.

The AMS CloudFormation ingest change type input schema enables you to choose up to sixty parameters in a CloudFormation template and provide custom values.

This example shows how to define a resource property, which can be used in a variety of CloudFormation templates, as a parameter in the AMS CloudFormation ingest CT. The examples in this section specifically show SNS topic usage.

Topics

Example 1: Hard code the AWS CloudFormation SNSTopic resource `TopicName` property

In this example, you hard code the AWS CloudFormation SNSTopic resource `TopicName` property in the CloudFormation template. Note that the `Parameters` section is empty.

To have a CloudFormation template that allows you to change the value for the SNSTopic name for a new stack without having to create a new CloudFormation template, you can use the AMS `Parameters` section of the CloudFormation ingest change type to make that configuration. By doing this, you use the same CloudFormation template later to create a new stack with a different `SNSTopic` name.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "My SNS Topic",
  "Parameters" : {
  },
  "Resources" : {
    "SNSTopic" : {
      "Type" : "AWS::SNS::Topic",
      "Properties" : {
        "TopicName" : "MyTopicName"
      }
    }
  }
}
```

Example 2: Use an SNSTopic resource to reference a parameter in the AMS change type

In this example, you use an `SNSTopic` resource `TopicName` property defined in the CloudFormation template to reference a `Parameter` in the AMS change type.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",
  "Description" : "My SNS Topic",
  "Parameters" : {
    "TopicName" : {
      "Type" : "String",
      "Description" : "Topic ID",
      "Default" : "MyTopicName"
    }
  },
  "Resources" : {
    "SNSTopic" : {
      "Type" : "AWS::SNS::Topic",
      "Properties" : {
        "TopicName" : { "Ref" : "TopicName"}
      }
    }
  }
}
```

Example 3: Create an SNS topic by submitting a JSON execution parameters file with the AMS ingest change type

In this example, you submit a JSON execution parameters file with the AMS ingest CT that creates the SNS topic `TopicName`. The SNS topic must be defined in the CloudFormation template in the modifiable way shown in this example.

```
{
  "Name": "cfn-ingest",
  "Description": "CFNIngest Web Application Stack",
  "CloudFormationTemplateS3Endpoint": "$S3_PRESIGNED_URL",
  "VpcId": "VPC_ID",
  "Tags": [
    {"Key": "Enviroment Type", "Value": "Dev"}
  ],
  "Parameters": [
    {"Name": "TopicName", "Value": "MyTopic1"}
  ],
  "TimeoutInMinutes": 60
}
```

Example 4: Submit a new change type that references the same CloudFormation template

This JSON example changes the SNS `TopicName` value without making a change to the CloudFormation template. Instead, you submit a new Deployment | Ingestion | Stack from CloudFormation Template | Create change type that references the same CFN template.

```
{
  "Name": "cfn-ingest",
  "Description": "CFNIngest Web Application Stack",
  "CloudFormationTemplateS3Endpoint": "$S3_PRESIGNED_URL",
  "VpcId": "VPC_ID",
  "Tags": [
    {"Key": "Enviroment Type", "Value": "Dev"}
  ],
  "Parameters": [
    {"Name": "TopicName", "Value": "MyTopic2"}
  ],
  "TimeoutInMinutes": 60
}
```

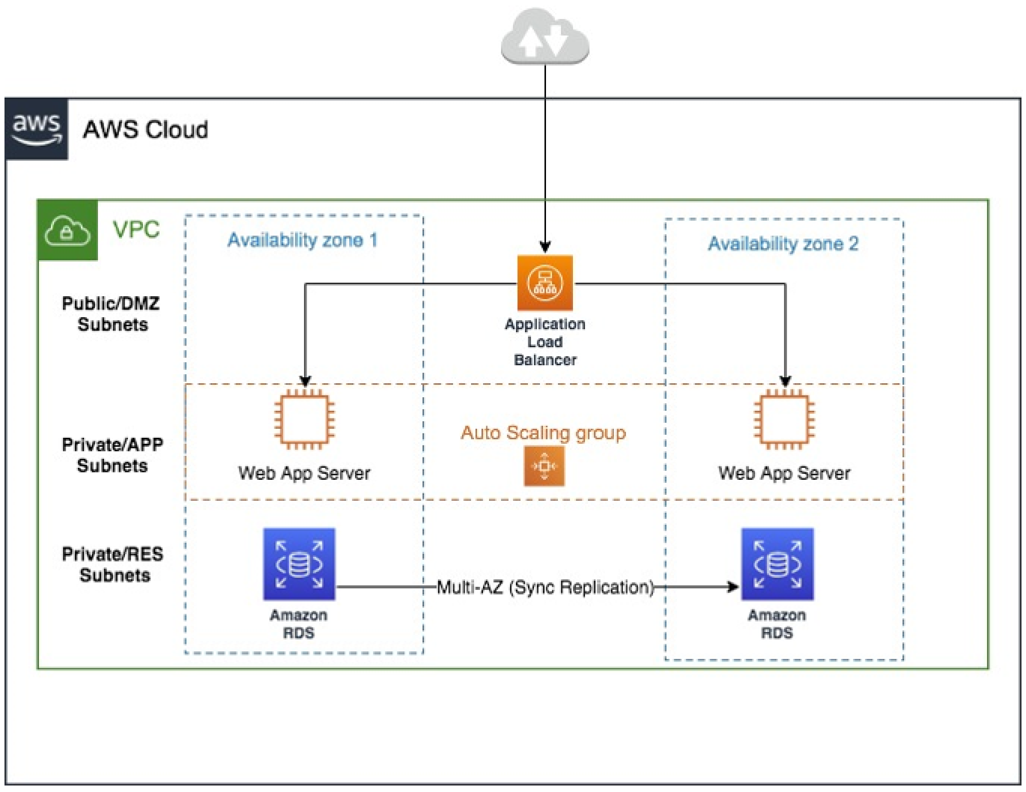
Example 5: Use the default parameter values in the CloudFormation template

In this example, the SNS `TopicName` = 'MyTopicName' is created because no `TopicName` value was provided in the `Parameters` execution parameter. If you don't provide `Parameters` definitions, the default parameter values in the CloudFormation template are used.

```
{
  "Name": "cfn-ingest",
  "Description": "CFNIngest Web Application Stack",
  "CloudFormationTemplateS3Endpoint": "$S3_PRESIGNED_URL",
  "VpcId": "VPC_ID",
  "Tags": [
    {"Key": "Enviroment Type", "Value": "Dev"}
  ],
  "TimeoutInMinutes": 60
}
```

CloudFormation Ingest Examples: 3-tier Web App

Ingest a CloudFormation template for a standard 3-Tier Web Application.



This includes an Application Load Balancer, Application Load Balancer target group, Auto Scaling group, Auto Scaling group launch configuration, Amazon Relational Database Service (Amazon RDS) with a MySQL database, AWS SSM Parameter store and AWS Secrets Manager. Allow 30-60 minutes to walk through this example.

Prerequisites

- Create a secret containing a username and password with corresponding values using the AWS Secrets Manager. You can refer to this sample JSON template that contains the secret name `ams-shared/myapp/dev/dbsecrets`, and replace it with your secret name. For information about using AWS Secrets Manager with AMS, see Using AWS Secrets Manager with AMS Resources (p. 20).
- Set up required parameters in the AWS SSM Parameter Store (PS). In this example, the `VPCId` and `Subnet-Id` of the Private and Public subnets are stored in the SSM PS in paths like `/app/DemoApp/PublicSubnet1a`, `PublicSubnet1c`, `PrivateSubnet1a`, `PrivateSubnet1c` and `VPCCidr`. Update the paths and parameter names and values for your needs.
- Create an IAM EC2 instance role with read permissions to the AWS Secrets Manager and SSM Parameter Store paths (the IAM role created and used in these examples is `customer-ec2_secrets_manager_instance_profile`). If you create an IAM-standard policies like Instance profile role, the role name must start with `customer-`. To create a new IAM role, (you can name it `customer-ec2_secrets_manager_instance_profile`, or something else) use the AMS change type Management | Applications | IAM instance profile | Create (ct-0ixp4ch2tiu04) CT, and attach the required policies. You can review the AMS IAM standard policies, `customer_secrets_manager_policy` and `customer_systemsmanager_parameterstore_policy`, in the AWS IAM Console to be used as-is or as a reference.

Ingest a CloudFormation template for a standard 3-Tier Web Application

1. Upload the attached sample CloudFormation JSON template, 3-tier-cfn-ingest.zip to an S3 bucket and generate a signed S3 URL to use in the CFN Ingest RFC. For more information, see presign. The CFN template can also be copy/pasted into the CFN Ingest RFC when you submit the RFC via the AMS console.
2. Create a CloudFormation Ingest RFC (Deployment | Ingestion | Stack from CloudFormation template | Create (ct-36cn2avfrrj9v)), either via the AMS console or the AMS CLI. The CloudFormation ingest automation process validates the CFN template to ensure that the template has valid AMS-supported resources, and adheres to security standards.

   - Using the console - For the change type, select **Deployment** -> **Ingestion** -> **Stack from CloudFormation Template** -> **Create**, and then add the following parameters as an example (note that the default for `MultiAZDatabase` is false):

   ```
   CloudFormationTemplateS3Endpoint: "https://s3-ap-
   southeast-2.amazonaws.com/my-bucket/3-tier-cfn-ingest.json?
   AWSAccessKeyId=#{S3_ACCESS_KEY_ID}&Expires=#{EXPIRE_DATE}&Signature=#{SIGNATURE}"
   VpcId: "VPC_ID"
   TimeoutInMinutes: 120
   IAMEC2InstanceProfile: "customer_ec2_secrets_manager_instance_profile"
   MultiAZDatabase: "true"
   WebServerCapacity: "2"
   ```

   - Using the AWS CLI - For details about creating RFCs using the AWS CLI, see Creating a Request for Change (RFC). For example, run the following command:

   ```
   aws --profile=saml amscm create-rfc  --change-type-id ct-36cn2avfrrj9v
    --change-type-version "2.0" --title "TEST_CFN_INGEST" --execution-
   parameters "{\"CloudFormationTemplateS3Endpoint\":\"https://s3-
   ap-southeast-2.amazonaws.com/my-bucket/3-tier-cfn-ingest.json?
   ```

```
AWSAccessKeyId=#{S3_ACCESS_KEY_ID}&Expires=#{EXPIRE_DATE}&Signature=#{SIGNATURE}\",
\"TimeoutInMinutes\":120,\"Description\":\"TEST\",\"VpcId”\":\”VPC_ID\”,
\"Name\":\"MY_TEST\",\"Tags\":[{\"Key\":\"env\",\"Value\":\"test\"}],
\"Parameters\":[{\"Name\":\"IAMEC2InstanceProfile\",\"Value\":
\"customer_ec2_secrets_manager_instance_profile\"},{\"Name\":\"MultiAZDatabase\",
\"Value\":\"true\"},{\"Name\":\"VpcId\",\"Value\":\"VPC_ID\"},{\"Name\":
\"WebServerCapacity\",\"Value\":\"2\"}]}" --endpoint-url https://amscm.us-
east-1.amazonaws.com/operational/ --no-verify-ssl
```

Find the Application Load Balancer URL in the AWS CloudFormation RFC execution output to access the website. For information about accessing resources, see Accessing Instances.

Create CloudFormation ingest stack

Create an AMS stack from a custom AWS CloudFormation template using the AWS and AMS CLI.

Classification: Deployment | Ingestion | Stack from CloudFormation template | Create

Change type ID: ct-36cn2avfrrj9v

Version: 2.0

Note
This change type is at version 2.0 and is automated (not manually executed). This allows the CT execution to go more quickly, and, a new parameter, **CloudFormationTemplate**, allows you to paste into the RFC a custom CloudFormation template. Additionally, In this version, we do not attach the default AMS security groups if the you specify your own security groups. If you do not specify your own security groups in the request, AMS will attach the AMS default security groups. In CFN Ingest v1.0, we always appended the AMS default security groups whether or not you provided your own security groups.
AMS has enabled 17 AMS Self-Provisioned services for use in this change type. For information about supported resources, see CloudFormation Ingest Stack: Supported Resources.

Note
Version 2.0 accepts an S3 endpoint that is not a presigned URL.
If you use the previous version of this CT, the **CloudFormationTemplateS3Endpoint** parameter value must be a presigned URL.
Example command for generating a presigned S3 bucket URL (Mac/Linux):

```
export S3_PRESIGNED_URL=$(aws s3 presign DASHDASHexpires-in 86400
 s3://BUCKET_NAME/CFN_TEMPLATE.json)
```

Example command for generating a presigned S3 bucket URL (Windows):

```
for /f %i in ('aws s3 presign DASHDASHexpires-in 86400
 s3://BUCKET_NAME/CFN_TEMPLATE.json') do set S3_PRESIGNED_URL=%i
```

See also Creating Pre-Signed URLs for Amazon S3 Buckets.

To learn more about AWS CloudFormation, see AWS CloudFormation. To see CloudFormation templates, open the AWS CloudFormation Template Reference.

Required Data:

- **Description**: A reason for the request.
- **Name**: A name for the stack or stack component. This becomes the stack name.

- **TimeoutInMinutes**: The maximum amount of time, in minutes, to allow for execution of the change. This will not prolong execution, but the RFC fails if the change is not completed in the specified time. The default timeout is 60 minutes, and the maximum timeout is 360 minutes.
- **VpcId**: ID of the VPC to use, in the form vpc-0123abcd or vpc-01234567890abcdef.

Optional Data - Either **CloudFormationTemplate** or **CloudFormationTemplateS3Endpoint** are required:

- **CloudFormationTemplate** - Your customized CloudFormation template, copied directly into this input parameter. Use this parameter or the **CloudFormationTemplateS3Endpoint** parameter. Don't use both.
- **CloudFormationTemplateS3Endpoint** - The S3 bucket endpoint where you uploaded the custom CloudFormation template file. Note: If your template is not in the same AMS account as the stack you are launching into, you also need to generate and provide a presigned URL for the S3 bucket endpoint. For more information about how to do this, see presign.

  To upload the template file after it's created (Windows/Mac/Linux), run the following command:

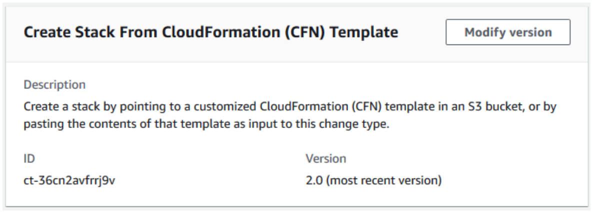
  ```
  aws s3 cp CFN_TEMPLATE.json s3://BUCKET_NAME/CFN_TEMPLATE.json
  ```

  Note
  If the S3 bucket exists in an AMS account, you must use your AMS credentials for this command. For example, you may need to append `--profile saml` after obtaining your AMS AWS Security Token Service (AWS STS) credentials.

  The S3 bucket must contain a valid AMS-ready CloudFormation template. Validate the template using the AWS CloudFormation API/CLI. A sample JSON validation file is included at the end of this procedure.

  ```
  aws cloudformation validate-template --template-body file://./ TEMPLATE_FILE.json
  ```
- **Parameters**: Add up to sixty parameters (parameter name/value pairs) to supply alternate values for parameters in your customized CloudFormation template. By providing the parameters this way, you can reuse your CloudFormation template with different parameter values when needed and can update any parameter value with the CFN Ingest Stack Update change type.
- **Tags**: Up to fifty tags (key/value pairs) to categorize the resource.

Creating a CloudFormation ingest stack using the console


Create Stack From CloudFormation (CFN) Template

Description
Create a stack by pointing to a customized CloudFormation (CFN) template in an S3 bucket, or by pasting the contents of that template as input to this change type.
ID
ct-36cn2avfrrj9v
Version
2.0 (most recent version)

To create a CloudFormation ingest stack using the console

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.

4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Creating a CloudFormation ingest stack using the CLI

To create a CloudFormation ingest stack using the CLI

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

1. Prepare the CloudFormation template that you will use to create the stack, and upload it to your S3 bucket. For important details, see AWS CloudFormation Ingest Guidelines, Best Practices, and Limitations (p. 43).
2. Create and submit the RFC to AMS:

   - Create and save the execution parameters JSON file, include the CloudFormation template parameters that you want. The following example names it CreateCfnParams.json.

     Example Web application stack CreateCfnParams.json file:

```
{
  "Name": "cfn-ingest",
  "Description": "CFNIngest Web Application Stack",
  "VpcId": "VPC_ID",
  "CloudFormationTemplateS3Endpoint": "$S3_URL",
  "TimeoutInMinutes": 120,
  "Tags": [
   {
    "Key":    "Enviroment Type"
    "Value": "Dev",
   },
   {
    "Key":    "Application"
    "Value": "PCS",
   }
  ],
  "Parameters": [
   {
    "Name": "Parameter-for-S3Bucket-Name",
    "Value":  "BUCKET-NAME"
   },
   {
    "Name": "Parameter-for-Image-Id",
    "Value":  "AMI-ID"
   }
  ],
}
```

Example SNS topic CreateCfnParams.json file:

```
{
  "Name": "cfn-ingest",
  "Description": "CFNIngest Web Application Stack",
  "CloudFormationTemplateS3Endpoint": "$S3_URL",
  "Tags": [
    {"Key": "Enviroment Type", "Value": "Dev"}
  ],
  "Parameters": [
    {"Name": "TopicName", "Value": "MyTopic1"}
  ]
}
```

3. Create and save the RFC parameters JSON file with the following content. The following example names it CreateCfnRfc.json file:

```
{
   "ChangeTypeId": "ct-36cn2avfrrj9v",
   "ChangeTypeVersion": "2.0",
   "Title": "cfn-ingest"
}
```

4. Create the RFC, specifying the CreateCfnRfc file and the CreateCfnParams file:

```
aws amscm create-rfc --cli-input-json file://CreateCfnRfc.json  --execution-parameters
 file://CreateCfnParams.json
```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

Validating a AWS CloudFormation ingest

The template is validated to ensure that it can be created in an AMS account. If it passes validation, it's updated to include any resources or configurations required for it to conform with AMS. This includes adding resources such as Amazon CloudWatch alarms in order to allow AMS Operations to monitor the stack.

The RFC is rejected if any of the following are true:

- RFC JSON Syntax is incorrect or does not follow the given format.
- The provided S3 bucket presigned URL is not valid.

- The template is not valid AWS CloudFormation syntax.
- The template does not have defaults set for all parameter values.
- The template fails AMS validation. For AMS validation steps, see the information later in this topic.

The RFC fails if the CloudFormation stack fails to create due to a resource creation issue.

To learn more about CFN validation and validator, see Template Validation (p. 45) and CloudFormation ingest stack: CFN validator examples (p. 46).

Update AWS CloudFormation ingest stack

Update an AMS stack created from ingestion of a custom AWS CloudFormation template using the AWS and AMS CLI.

Classification: Management | Custom stack | Stack from CloudFormation template | Update

Change type ID: ct-361tlo1k7339x

Version: 1.0

> **Note**
> This change type is now at version 2.0. Changes include removing the **AutoApproveUpdateForResources** parameter, which was used in version 1.0 of this CT, and adding two new parameters: **AutoApproveRiskyUpdates** and **BypassDriftCheck**.

For a list of which self-provisioned services you can add using AWS CloudFormation Ingest, see CloudFormation Ingest Stack: Supported Resources.

To learn more about AWS CloudFormation, see AWS CloudFormation.

Required Data - The following data are required:

- **VpcId**: The VPC to use. For help finding VPC IDs, see Finding VPC IDs.
- **StackId**: The stack identifier. For help finding stack IDs, see Finding Stack IDs.

  Only stacks matching the pattern `stack-[a-z0-9]{17}$.` are supported by this Update change type.
- **TimeoutInMinutes**: The maximum amount of time, in minutes, to allow for execution of the change. This will not prolong execution, but the RFC fails if the change is not completed in the specified time. The default is 360 minutes, and the maximum timeout is 1080 minutes.

Optional Data - The following data are optional, but at least one parameter is required:

- **CloudFormationTemplateS3Endpoint**: The S3 bucket endpoint where you uploaded the custom CloudFormation template file. You also need to generate and provide a presigned URL for the S3 bucket endpoint.

  To generate a presigned S3 bucket URL (Mac/Linux), run the following command:

```
export S3_PRESIGNED_URL=$(aws s3 presign --expires-in 86400
 s3://BUCKET_NAME/CFN_TEMPLATE.json)
```

  To generate a presigned S3 bucket URL (Windows), run the following command:

```
for /f %i in ('aws s3 presign --expires-in 86400 s3://BUCKET_NAME/CFN_TEMPLATE.json') do
 set S3_PRESIGNED_URL=%i
```

For more details, see Creating Pre-Signed URLs for Amazon S3 Buckets.

To upload the template file after it's created (Windows/Mac/Linux):

```
aws s3 cp CFN_TEMPLATE.json s3://BUCKET_NAME/CFN_TEMPLATE.json
```

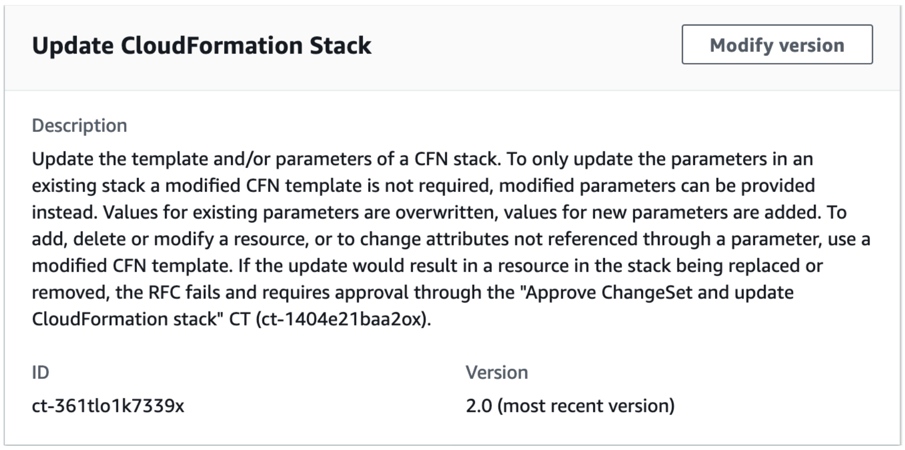
Note
If the S3 bucket exists in an AMS account, you must use your AMS credentials for this command. For example, you may need to append `--profile saml` after obtaining your AMS AWS Security Token Service (AWS STS) credentials.

- **CloudFormationTemplate**: A valid AMS-ready CFN template. Validate the template using the AWS CloudFormation API/CLI. A sample JSON validation file is included at the end of this procedure.

```
aws cloudformation validate-template --template-body file://./ TEMPLATE_FILE.json
```

- **TemplateParameters**: Up to sixty parameters (key/value pairs) from the CloudFormation template to configure the stack. By providing the parameters this way, you can reuse your CloudFormation template with different parameters when needed. Use any parameter that is in your custom, or a standard, CloudFormation template. To see CloudFormation templates, open the AWS CloudFormation Template Reference.
- **AutoApproveRiskyUpdates**: This parameter accepts a list of resources in a stack that are safe to be deleted or replaced if a requested update might cause deletion or replacement. Use this parameter if you want to update one or more resources with a replacement, a conditional replacement, or if you want to delete one or more resources.
- **BypassDriftCheck**: Accepts a comma-separated list of strings, where each string is the logical ID of your stack resource. This parameter can be used to bypass drift and update a drifted or drift-detection-unsupported resource in an AWS CloudFormation stack by providing the logical IDs specified in a CloudFormation template.
- **AutoApproveUpdateForResources**: *This parameter is only valid for version 1.0 of this CT*. Logical IDs in your template that represent resources for which a high-risk update should be automatically approved, without requiring your approval of a change set. High-risk is defined as an update that could cause resource deletion or replacement, or an update on a CloudFormation resource that we cannot guarantee has not drifted in state from the underlying resource. If the stack update includes high-risk changes that are not included in this list, you will be required to approve a change set to execute the change.

Updating a CloudFormation ingest stack using the console



Update CloudFormation Stack

Description

Update the template and/or parameters of a CFN stack. To only update the parameters in an existing stack a modified CFN template is not required, modified parameters can be provided instead. Values for existing parameters are overwritten, values for new parameters are added. To add, delete or modify a resource, or to change attributes not referenced through a parameter, use a modified CFN template. If the update would result in a resource in the stack being replaced or removed, the RFC fails and requires approval through the "Approve ChangeSet and update CloudFormation stack" CT (ct-1404e21baa2ox).

ID	Version
ct-361tlo1k7339x	2.0 (most recent version)

To update a CloudFormation Ingest Stack using the console

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Updating a CloudFormation ingest stack using the CLI

To update a CloudFormation ingest stack using the CLI

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

> **Note**
> You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

1. Prepare the AWS CloudFormation template that you want to use to update the stack, and upload it to your S3 bucket. For important details, see AWS CloudFormation Ingest Guidelines, Best Practices, and Limitations (p. 43).
2. Create and submit the RFC to AMS:

   - Create and save the execution parameters JSON file, include the CloudFormation template parameters that you want. This example names it UpdateCfnParams.json.

     Example UpdateCfnParams.json file with inline parameter updates:

     ```
     {
       "StackId": "stack-yjjoo9aicjyqw4ro2",
     ```

```
  "VpcId": "VPC_ID",
  "CloudFormationTemplate": "{\"AWSTemplateFormatVersion\":\"2010-09-09\",
\"Description\":\"Create a SNS topic\",\"Parameters\":{\"TopicName\":{\"Type\":
\"String\"},\"DisplayName\":{\"Type\":\"String\"}},\"Resources\":{\"SnsTopic\":
{\"Type\":\"AWS::SNS::Topic\",\"Properties\":{\"TopicName\":{\"Ref\":\"TopicName
\"},\"DisplayName\":{\"Ref\":\"DisplayName\"}}}}}",
  "TemplateParameters": [
    {
      "Key": "TopicName",
      "Value": "TopicNameCLI"
    },
    {
      "Key": "DisplayName",
      "Value": "DisplayNameCLI"
    }
  ],
  "TimeoutInMinutes": 1440
}
```

Example UpdateCfnParams.json file with S3 bucket endpoint containing an updated CloudFormation template:

```
{
  "StackId": "stack-yjjoo9aicjyqw4ro2",
  "VpcId": "VPC_ID",
  "CloudFormationTemplateS3Endpoint": "s3_url",
  "TemplateParameters": [
    {
      "Key": "TopicName",
      "Value": "TopicNameCLI"
    },
    {
      "Key": "DisplayName",
      "Value": "DisplayNameCLI"
    }
  ],
  "TimeoutInMinutes": 1080
}
```

3. Create and save the RFC parameters JSON file with the following content. This example names it UpdateCfnRfc.json file.

```
{
   "ChangeTypeId": "ct-361tlo1k7339x",
   "ChangeTypeVersion": "1.0",
   "Title": "cfn-ingest-template-update"
}
```

4. Create the RFC, specifying the UpdateCfnRfc file and the UpdateCfnParams file:

```
aws amscm create-rfc --cli-input-json file://UpdateCfnRfc.json  --execution-parameters
 file://UpdateCfnParams.json
```

You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

Validating a AWS CloudFormation ingest

The template is validated to ensure that it can be created in an AMS account. If it passes validation, it's updated to include any resources or configurations required for it to conform with AMS. This includes adding resources such as Amazon CloudWatch alarms in order to allow AMS Operations to monitor the stack.

The RFC is rejected if any of the following are true:

- RFC JSON Syntax is incorrect or does not follow the given format.
- The provided S3 bucket presigned URL is not valid.
- The template is not valid AWS CloudFormation syntax.
- The template does not have defaults set for all parameter values.
- The template fails AMS validation. For AMS validation steps, see the information later in this topic.

The RFC fails if the CloudFormation stack fails to create due to a resource creation issue.

To learn more about CFN validation and validator, see Template Validation (p. 45) and CloudFormation ingest stack: CFN validator examples (p. 46).

Approve a CloudFormation ingest stack changeset

Update an AMS stack that was created by ingesting a custom AWS CloudFormation template, but was rejected. The rejection error message contains a changeset identifier that you can use with this change type to approve the requested update.

> **Note**
> If there are multiple resources in a stack, and you want to delete only a subset of the stack resources, use the CloudFormation Update CT; see CloudFormation Ingest Stack: Updating. You can also submit a Management | Other | Other | Update change type and AMS engineers can help you craft the changeset, if needed.

Classification: Management | Custom stack | Stack from CloudFormation Template | Approve Changeset and Update

Change type ID: ct-1404e21baa2ox

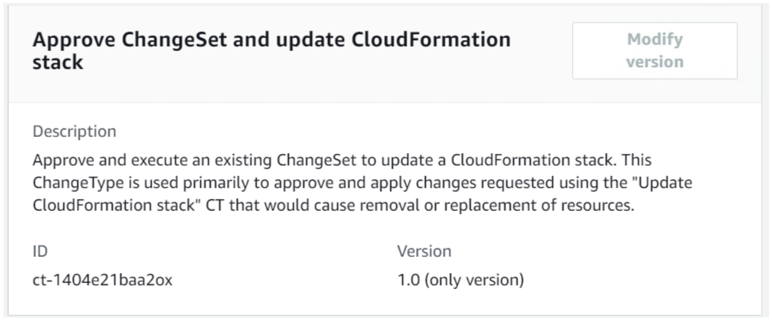
Version: 1.0

To learn more about AWS CloudFormation, see AWS CloudFormation.

Required Data:

- **VpcId**: The VPC to use. For help finding VPC IDs, see Finding VPC IDs.
- **StackId**: The stack identifier. For help finding stack IDs, see Finding Stack IDs.
- **ChangeSetName**: Name of the ChangeSet to execute against the stack. If the stack update was requested using the Update CloudFormation stack CT, the ChangeSet name can be found in the failure reason of that RFC.
- **TimeoutInMinutes**: The maximum amount of time, in minutes, to allow for execution of the change. This will not prolong execution, but the RFC fails if the change is not completed in the specified time. The default timeout is 360 minutes, and the maximum timeout is 1080 minutes.

Approving and updating a CloudFormation ingest stack using the console



Approve ChangeSet and update CloudFormation stack

Description

Approve and execute an existing ChangeSet to update a CloudFormation stack. This ChangeType is used primarily to approve and apply changes requested using the "Update CloudFormation stack" CT that would cause removal or replacement of resources.

ID	Version
ct-1404e21baa2ox	1.0 (only version)

To approve and update a CloudFormation ingest stack using the console

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Approving and updating a CloudFormation ingest stack using the CLI

To approve and update a CloudFormation ingest stack using the CLI

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution

parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

1. Output the execution parameters JSON schema for this change type to a file in your current folder. This example names it CreateAsgParams.json:

```
aws amscm create-rfc --change-type-id "ct-1404e21baa2ox" --change-type-version "1.0"
 --title "Approve Update" --execution-parameters file://PATH_TO_EXECUTION_PARAMETERS --
profile saml
```

2. Modify and save the schema as follows:

```
{
  "StackId": "STACK_ID",
  "VpcId": "VPC_ID",
  "ChangeSetName": "UPDATE-ef81e2bc-03f6-4b17-a3c7-feb700e78faa",
  "TimeoutInMinutes": 1080
}
```

Update AWS CloudFormation stacks termination protection

Update the existing termination protection configuration for an AWS CloudFormation stack, using the AMS console or the AMS API/CLI.

Classification: Management | Standard stacks | Stack | Update termination protection

Change type ID: ct-2uzbqr7x7mekd

Version: 1.0

To learn more about termination protection, see Protecting a stack from being deleted.

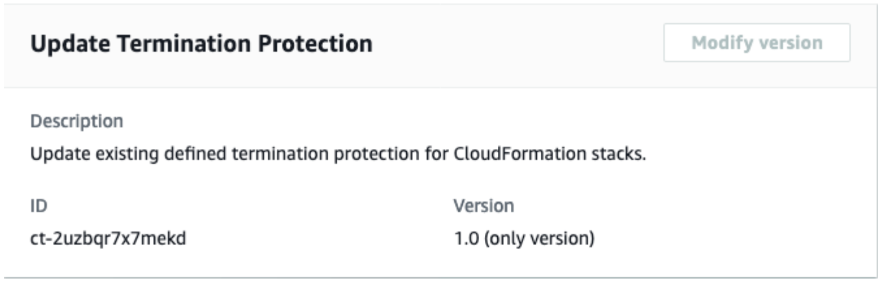
> **Note**
> There is a related CT for Amazon EC2, EC2 stack: Updating termination protection.

Required Data:

- **DocumentName**: Must be AWSManagedServices-ManageResourceTerminationProtection.
- **Region**: The AWS Region in which the AWS CloudFormation stack is located, in the form us-east-1.
- **Parameters**:
  - **ResourceId**: AWS CloudFormation stack ID.
  - **TerminationProtectionDesiredState**: Enabled to protect your stack against elimination. Disabled to allow your stack to be eliminated.

Updating an AWS CloudFormation termination protection stack with the console

The following shows this change type in the AMS console.



Description

Update existing defined termination protection for CloudFormation stacks.

ID

ct-2uzbqr7x7mekd

Version

1.0 (only version)

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Updating an AWS CloudFormation stack termination protection with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

Only specify the parameters you want to change. Absent parameters retain the existing values.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotation marks when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws amscm create-rfc \
--change-type-id "ct-2uzbqr7x7mekd" \
--change-type-version "1.0" \
--title "Enable termination protection on CFN stack" \
--execution-parameters "{\"DocumentName\":\"AWSManagedServices-
ManageResourceTerminationProtection\",\"Region\":\"us-east-1\",\"Parameters\":{\"ResourceId
\":[\"stack-psvnq6cupymio3enl\"],\"TerminationProtectionDesiredState\":[\"enabled\"]}}"
```

TEMPLATE CREATE:

1. Output the execution parameters for this change type to a JSON file; this example names it EnableTermProCFNParams.json:

   ```
   aws amscm get-change-type-version --change-type-id "ct-2uzbqr7x7mekd" --query
    "ChangeTypeVersion.ExecutionInputSchema" --output text > EnableTermProCFNParams.json
   ```

2. Modify and save the EnableTermProCFNParams file, retaining only the parameters that you want to change. For example, you can replace the contents with something like this:

   ```
   {
     "DocumentName": "AWSManagedServices-ManageResourceTerminationProtection",
     "Region": "us-east-1",
     "Parameters": {
       "ResourceId": ["stack-psvnq6cupymio3enl"],
       "TerminationProtectionDesiredState": ["enabled"]
     }
   }
   ```

3. Output the RFC template to a file in your current folder; this example names it EnableTermProCFNRfc.json:

   ```
   aws amscm create-rfc --generate-cli-skeleton > EnableTermProCFNRfc.json
   ```

4. Modify and save the EnableTermProCFNRfc.json file. For example, you can replace the contents with something like this:

   ```
   {
       "ChangeTypeId": "ct-2uzbqr7x7mekd",
       "ChangeTypeVersion": "1.0",
       "Title": "Enable termination protection on CFN instance"
   }
   ```

5. Create the RFC, specifying the EnableTermProCFNRfc file and the EnableTermProCFNParams file:

   ```
   aws amscm create-rfc --cli-input-json file://EnableTermProCFNRfc.json  --execution-
   parameters file://EnableTermProCFNParams.json
   ```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

CodeDeploy requests

You can use AWS CodeDeploy to create application containers that you can then deploy through a CodeDeploy application group. For more information about CodeDeploy, see AWS CodeDeploy Documentation.

Working with AWS CodeDeploy involves the following process:

1. Create a CodeDeploy application. The CodeDeploy application is a name or container used by CodeDeploy to ensure that the correct revision, deployment configuration, and deployment group are referenced during a deployment.
2. Create a CodeDeploy deployment group. A CodeDeploy deployment group defines a set of individual instances targeted for a deployment. AMS has a separate change type for CodeDeploy deployment groups for EC2.
3. Deploy the CodeDeploy application through the CodeDeploy deployment group.

Create CodeDeploy application

Create a CodeDeploy application using the AMS console or the AMS API/CLI.

Classification: Deployment | Applications | CodeDeploy application | Create

Change type ID: ct-0ah3gwb9seqk2

Version: 1.0

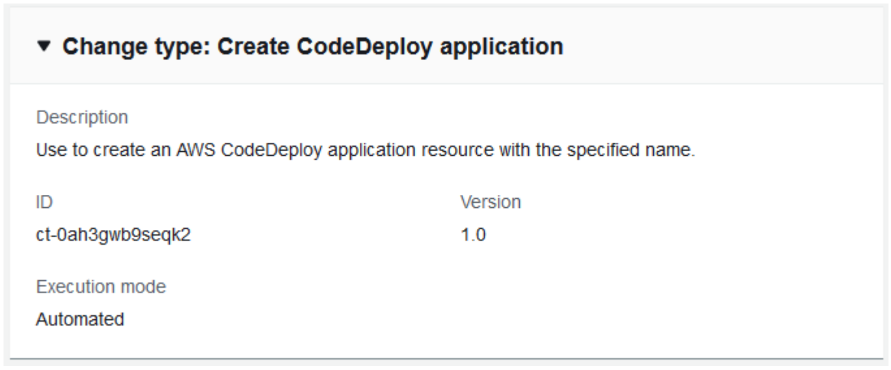
The CodeDeploy application is a name or container used by AWS CodeDeploy to ensure that the correct revision, deployment configuration, and deployment group are referenced during a deployment.

For more information about AWS CodeDeploy, see Create an Application with AWS CodeDeploy.

Required data:

- **VpcId**: The VPC that you are using.
- **CodeDeployApplicationName**: The CodeDeploy application name must be unique in the account. For existing application names, check the CodeDeploy console.

Creating a CodeDeploy application with the console



▼ **Change type: Create CodeDeploy application**

Description
Use to create an AWS CodeDeploy application resource with the specified name.

ID
ct-0ah3gwb9seqk2

Version
1.0

Execution mode
Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.

2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Creating a CodeDeploy application with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotation marks when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws amscm create-rfc --change-type-id "ct-0ah3gwb9seqk2" --change-type-version "1.0"
 --title "Stack-Create-CD-App" --execution-parameters "{\"Description\":\"TestCdApp\",
\"VpcId\":\"VPC_ID\",\"StackTemplateId\":\"stm-sft6rv00000000000\",\"Name\":\"Test\",
\"TimeoutInMinutes\":60,\"Parameters\":{\"CodeDeployApplicationName\":\"Test\"}}"
```

TEMPLATE CREATE:

1. Output the execution parameters JSON schema for the CodeDeploy application CT to a file in your current folder; this example names it CreateCDAppParams.json:

```
aws amscm get-change-type-version --change-type-id "ct-0ah3gwb9seqk2" --query
 "ChangeTypeVersion.ExecutionInputSchema" --output text > CreateCDAppParams.json
```

2. Modify and save the JSON file as follows. For example, you can replace the contents with something like this:

```
{
"Description":                          "Create WP CodeDeploy App",
"VpcId":                                "VPC_ID",
"StackTemplateId":                      "stm-sft6rv00000000000",
"Name":                                 "WpCDApp",
"TimeoutInMinutes":                     60,
"Parameters":   {
    "CodeDeployApplicationName":        "WordPressCDApp"
    }
}
```

3. Output the JSON template for CreateRfc to a file in your current folder; this example names it CreateCDAppRfc.json:

```
aws amscm create-rfc --generate-cli-skeleton > CreateCDAppRfc.json
```

4. Modify and save the JSON file as follows. For example, you can replace the contents with something like this:

```
{
"ChangeTypeVersion":    "1.0",
"ChangeTypeId":         "ct-0ah3gwb9seqk2",
"Title":                "CD-App-Stack-RFC"
}
```

5. Create the RFC, specifying the CreateCDAppRfc file and the execution parameters file:

```
aws amscm create-rfc --cli-input-json file://CreateCDAppRfc.json --execution-parameters
 file://CreateCDAppParams.json
```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

Create CodeDeploy deployment group

Create a CodeDeploy deployment group using the AMS console or the AMS API/CLI.

Classification: Deployment | Applications | CodeDeploy deployment group | Create

Change type ID: ct-2gd0u847qd9d2

Version: 1.0

A CodeDeploy deployment group defines a set of individual instances targeted for a deployment.

For more information about AWS CodeDeploy deployment groups, see Create a Deployment Group with AWS CodeDeploy.

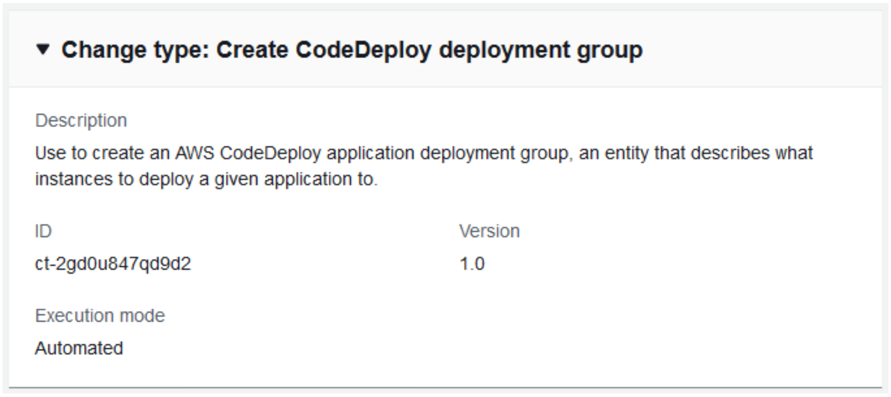
Required data:

- **VpcId**: The VPC that you are using.

- **CodeDeployApplicationName**: An existing CodeDeploy application associated with the applicable IAM user or AMS account.
- **CodeDeployAutoScalingGroups**: An existing Auto Scaling group that you created previously.
- **CodeDeployDeploymentGroupName**: A name for the deployment group. This name must be unique for each application associated with the deployment group.
- **CodeDeployServiceRoleArn**: Use this formula: `arn:aws:iam::ACCOUNT_ID:role/aws-codedeploy-role`.

Optional Data: **CodeDeployDeploymentConfigName**, The configuration for deployment operations: as many instances as possible at once, half of the instances at a time, or only one instance at a time. The default is **CodeDeployDefault.OneAtATime**.

Creating a CodeDeploy deployment group with the console



▼ Change type: Create CodeDeploy deployment group

Description
Use to create an AWS CodeDeploy application deployment group, an entity that describes what instances to deploy a given application to.

ID	Version
ct-2gd0u847qd9d2	1.0

Execution mode
Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Creating a CodeDeploy deployment group with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.

2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

> **Note**
> You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotation marks when providing execution parameters inline) and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws amscm create-rfc --change-type-id "ct-2gd0u847qd9d2" --change-type-version
 "1.0" --title "Stack-Create-CD-Dep-Group" --execution-parameters "{\"Description
\":\"TestCdDepGroupRfc\",\"VpcId\":\"VPC_ID\",\"StackTemplateId\":\"stm-
sp9lrk00000000000\",\"Name\":\"MyTestCDDepGroup\",\"TimeoutInMinutes\":60,\"Parameters
\":{\"CodeDeployApplicationName\":\"TestCDApp\",\"CodeDeployAutoScalingGroups\":
[\"TestASG\"],\"CodeDeployDeploymentConfigName\":\"CodeDeployDefault.OneAtATime\",
\"CodeDeployDeploymentGroupName\":\"Test\",\"CodeDeployServiceRoleArn\":
\"arn:aws:iam::000000000:role/aws-codedeploy-role\"}}"
```

TEMPLATE CREATE:

1. Output the execution parameters JSON schema to a file in your current folder; this example names it CreateCDDepGroupParams.json:

   ```
   aws amscm get-change-type-version --change-type-id "ct-2gd0u847qd9d2" --query
    "ChangeTypeVersion.ExecutionInputSchema" --output text > CreateCDDepGroupParams.json
   ```

2. Modify and save the JSON file. For example, you can replace the contents with something like this:

   ```
   {
   "Description":                        "CreateCDDeploymentGroup",
   "VpcId":                              "VPC_ID",
   "StackTemplateId":                    "stm-sp9lrk00000000000",
   "Name":                               "WordPressCDAppGroup",
   "TimeoutInMinutes":                   60,
   "Parameters":   {
       "CodeDeployApplicationName":          "WordPressCDApp",
       "CodeDeployAutoScalingGroups":        ["ASG_NAME"],
       "CodeDeployDeploymentConfigName":     "CodeDeployDefault.HalfAtATime",
       "CodeDeployDeploymentGroupName":      "UNIQUE_CDDepGroupNAME",
       "CodeDeployServiceRoleArn":           "arn:aws:iam::ACCOUNT_ID:role/aws-codedeploy-
   role"
       }
   }
   ```

3. Output the JSON template for CreateRfc to a file in your current folder; this example names it CreateCDDepGroupRfc.json:

```
aws amscm create-rfc --generate-cli-skeleton > CreateCDDepGroupRfc.json
```

4. Modify and save the JSON file. For example, you can replace the contents with something like this:

```
{
"ChangeTypeVersion":    "1.0",
"ChangeTypeId":         "ct-2gd0u847qd9d2",
"Title":                "CD-Dep-Group-RFC"
}
```

5. Create the RFC, specifying the CreateCDDepGroupRfc file and the execution parameters file:

```
aws amscm create-rfc --cli-input-json file://CreateCDDepGroupRfc.json --execution-
parameters file://CreateCDDepGroupParams.json
```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

Create CodeDeploy deployment group for EC2

Create a CodeDeploy deployment group for EC2 using the AMS console or the AMS API/CLI.

Classification: Deployment | Applications | CodeDeploy deployment group | Create (for EC2 instance)

Change type ID: ct-00tlkda4242x7

Version: 1.0

A CodeDeploy deployment group defines a set of individual instances targeted for a deployment.

For more information about AWS CodeDeploy deployment groups, see Create a Deployment Group with AWS CodeDeploy.

Required data:

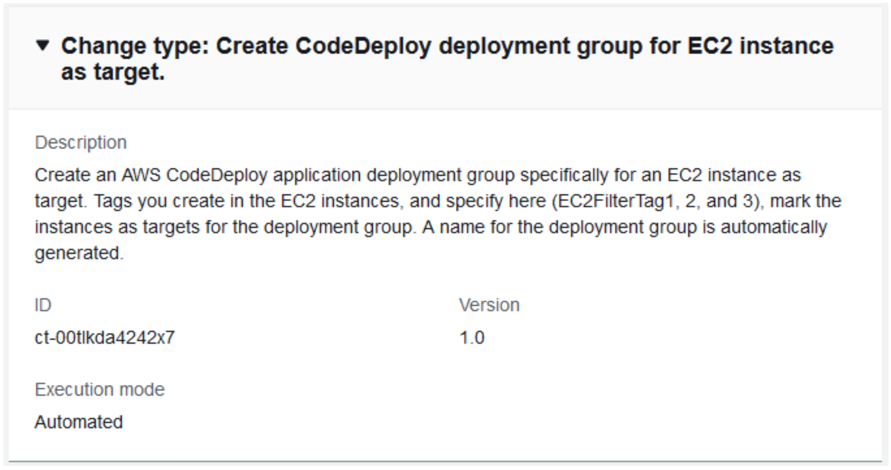
- **VpcId**: The VPC that you are using.
- **ApplicationName**: The name of an existing AWS CodeDeploy application associated with the applicable IAM user or AMS account.
- **EC2FilterTag (0-4)**: Key=Value pair tag for CodeDeploy to filter EC2 instances; for example Name=Application01. The specified tag is used to identify instances as targets for the deployment group.

Optional data:

- **DeploymentConfigName**: The configuration for deployment operations. To deploy as many instances as possible at once, use CodeDeployDefault.OneAtATime. To deploy half of the instances at a time, use CodeDeployDefaultHalfAtATime. To deploy only one instance at a time, use CodeDeployDefault.OneAtATime.
- **AutoRollbackEnabled**: True to enable an automatic rollback of a deployment if it fails; if that happens, CodeDeploy redeploys the last known good revision as a new deployment. False to not enable the automatic rollback.
- **AvailabilityZone**: The Availability Zone (AZ) to create the volume in. Must match the Availability Zone (AZ) of the instance ID in order to attach successfully.
- **InstanceId**: Identifier of an instance that the EBS volumes are attached to.

- **ServiceRoleArn**: The Amazon Resource Name (ARN) of an IAM role that grants AWS CodeDeploy permission to make calls to AWS services on your behalf.
- **Tags**: Up to fifty tags (key/value pairs) to categorize the CodeDeploy deployment group.

Creating a CodeDeploy deployment group for EC2 with the console


▼ Change type: Create CodeDeploy deployment group for EC2 instance as target.

Description

Create an AWS CodeDeploy application deployment group specifically for an EC2 instance as target. Tags you create in the EC2 instances, and specify here (EC2FilterTag1, 2, and 3), mark the instances as targets for the deployment group. A name for the deployment group is automatically generated.

ID

ct-00tlkda4242x7

Version

1.0

Execution mode

Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Creating a CodeDeploy deployment group for EC2 with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

> **Note**
> You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotation marks when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws amscm create-rfc --change-type-id "ct-00tlkda4242x7" --change-type-
version "1.0" --title "Stack-Create-CD-Ec2-Dep-Group" --execution-parameters
 "{\"Description\":\"MyTestCdDepEc2DepGroup\",\"VpcId\":\"VPC_ID\",\"Name\":
\"TestCDDepEc2Group\",\"StackTemplateId\":\"stm-n3hsoirgqeqqdbpk2\",\"TimeoutInMinutes
\":60,\"Parameters\":{\"ApplicationName\":\"TestCDApp\",\"DeploymentConfigName\":
\"CodeDeployDefault.OneAtATime\",\"AutoRollbackEnabled\":\"False\",\"EC2FilterTag\":
\"Name=Test\",\"EC2FilterTag2\":\"\",\"EC2FilterTag3\":\"\",\"ServiceRoleArn\":\"\"}}"
```

TEMPLATE CREATE:

1. Output the execution parameters JSON schema to a file; this example names it CreateCDDepGroupEc2Params.json:

   ```
   aws amscm get-change-type-version --change-type-id "ct-00tlkda4242x7"
    --query "ChangeTypeVersion.ExecutionInputSchema" --output text >
    CreateCDDepGroupEc2Params.json
   ```

2. Modify and save the JSON file. For example, you can replace the contents with something like this:

   ```
   {
   "Description":                        "CreateCDDepGroupEc2",
   "VpcId":                              "VPC_ID",
   "StackTemplateId":                    "stm-n3hsoirgqeqqdbpk2",
   "Name":                               "CDAppGroupEc2",
   "TimeoutInMinutes":                   60,
   "Parameters":   {
       "ApplicationName":          "CDAppEc2",
       "DeploymentConfigName":     "CodeDeployDefault.OneAtATime",
       "CodeDeployDeploymentGroupName":     "UNIQUE_CDDepGroupNAME",
       "CodeDeployServiceRoleArn":          "arn:aws:iam::ACCOUNT_ID:role/aws-codedeploy-
   role"
       }
   }
   ```

3. Output the JSON template for CreateRfc to a file in your current folder; this example names it CreateCDDepGroupEc2Rfc.json:

   ```
   aws amscm create-rfc --generate-cli-skeleton > CreateCDDepGroupEc2Rfc.json
   ```

4. Modify and save the JSON file. For example, you can replace the contents with something like this:

   ```
   {
   "ChangeTypeVersion":    "1.0",
   ```

```
"ChangeTypeId":          "ct-00tlkda4242x7",
"Title":                 "CD-Dep-Group-For-Ec2-Stack-RFC"
}
```

5. Create the RFC, specifying the CreateCDDepGroupEc2Rfc file and the execution parameters file:

   ```
   aws amscm create-rfc --cli-input-json file://CreateCDDepGroupEc2Rfc.json --execution-
   parameters file://CreateCDDepGroupEc2Params.json
   ```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

Deploy CodeDeploy application

Once you have your CodeDeploy application bundle and deployment group, use this RFC to deploy the application.

Classification: Deployment | Applications | CodeDeploy application | Deploy

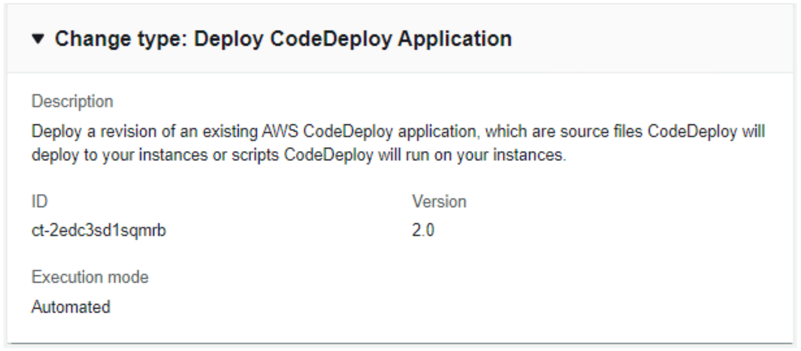
Change type ID: ct-2edc3sd1sqmrb

Version: 2.0

Required data:

- **VPC-ID**: The VPC you are using.
- **CodeDeployApplicationName**: Use the name for the CodeDeploy application that you previously created.
- **CodeDeployDeploymentGroupName**: Use the name of the CodeDeploy deployment group that you created previously.
- **CodeDeployIgnoreApplicationStopFailures**: True to ignore the failure of an ApplicationStop lifecycle event and continue to the BeforeInstall event; false to stop the deployment if the ApplicationStop event fails. Default is false.
- **CodeDeployRevision**: The type and location of the revision to deploy.
- **S3Location** (where you uploaded the application bundle):
  - **S3Bucket**: The name of the Amazon S3 bucket where the application revision is stored.
  - **S3BundleType**: The file type of the application revision.
  - **S3ETag**: The ETag of the Amazon S3 object that represents the bundled artifacts for the application revision.
  - **S3Key**: The name of the Amazon S3 object that represents the bundled artifacts for the application revision (for example, my_app.zip or path/to/my_app.zip).
  - **S3Version**: A specific version of the Amazon S3 object that represents the bundled artifacts for the application revision.

Deploying a CodeDeploy application with the console



▼ Change type: Deploy CodeDeploy Application

Description

Deploy a revision of an existing AWS CodeDeploy application, which are source files CodeDeploy will deploy to your instances or scripts CodeDeploy will run on your instances.

ID

ct-2edc3sd1sqmrb

Version

2.0

Execution mode

Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Deploying a CodeDeploy application with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id` *`ID`* command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id` *`ID`* command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" :`

`[\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotation marks when providing execution parameters inline) and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws amscm create-rfc --change-type-id "ct-2edc3sd1sqmrb" --change-type-version "2.0" --
title "Stack-Deploy-CD-App" --execution-parameters "{\"Description\":\"MyCDAppDeployTest\",
\"VpcId\":\"VPC_ID\",\"Name\":\"Test\",\"TimeoutInMinutes\":60,\"Parameters\":
{\"CodeDeployApplicationName\":\"TestCDApp\",\"CodeDeployDeploymentConfigName\":
\"CodeDeployDefault.OneAtATime\",\"CodeDeployDeploymentGroupName\":\"TestCDDepGroup\",
\"CodeDeployIgnoreApplicationStopFailures\":false,\"CodeDeployRevision\":{\"RevisionType
\":\"S3\",\"S3Location\":{\"S3Bucket\":\"TestBucket\",\"S3BundleType\":\"tar\",\"S3Key\":
\"TestKey\"}}}}"Test\"}}"
```

TEMPLATE CREATE:

1. Output the execution parameters JSON schema for the CodeDeploy application deployment CT; this example names it DeployCDAppParams.json:

```
aws amscm get-change-type-version --change-type-id "ct-2edc3sd1sqmrb" --query
 "ChangeTypeVersion.ExecutionInputSchema" --output text > DeployCDAppParams.json
```

2. Modify the JSON file as follows. For example, you can replace the contents with something like this:

```
{
"Description":                         "Deploy WordPress CodeDeploy Application",
"VpcId":                               "VPC_ID",
"Name":                                "WP CodeDeploy Deployment Group",
"TimeoutInMinutes":                    360,
"Parameters":   {
    "CodeDeployApplicationName":           "WordPressCDApp",
    "CodeDeployDeploymentGroupName":       "WordPressCDDepGroup",
    "CodeDeployIgnoreApplicationStopFailures": false,
    "CodeDeployRevision": {
      "RevisionType": "S3",
      "S3Location": {
        "S3Bucket": "ACCOUNT_ID.BUCKET_NAME",
        "S3BundleType": "zip",
        "S3Key": "wordpress.zip" }
        }
    }
}
```

3. Output the JSON template for CreateRfc to a file in your current folder; this example names it DeployCDAppRfc.json:

```
aws amscm create-rfc --generate-cli-skeleton > DeployCDAppRfc.json
```

4. Modify and save the DeployCDAppRfc.json file. For example, you can replace the contents with something like this:

```
{
"ChangeTypeVersion":      "2.0",
"ChangeTypeId":           "ct-2edc3sd1sqmrb",
"Title":                  "CD-Deploy-For-CD-APP-Stack-RFC"
}
```

5. Create the RFC, specifying the execution parameters file and the DeployCDAppRfc file:

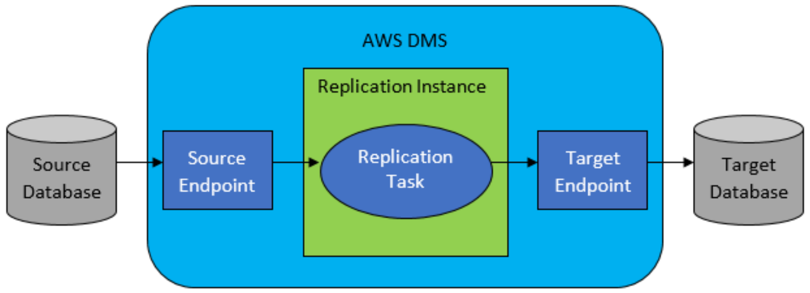
```
aws amscm create-rfc --cli-input-json file://DeployCDAppRfc.json  --execution-
parameters file://DeployCDAppParams.json
```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

Database Migration Service (DMS)

AWS Database Migration Service (DMS) helps you migrate databases to AMS easily and securely. You can migrate your data to and from most widely used commercial and open-source databases, such as Oracle, MySQL, and PostgreSQL. The service supports homogeneous migrations such as Oracle to Oracle, and also heterogeneous migrations between different database platforms, such as Oracle to PostgreSQL or MySQL to Oracle. DMS is an AWS service; the AMS CTs help you create AWS DMS resources in your AMS-managed account

The following graphic depicts the workflow of an database migration.



For a database migration, you must do the following:

- Plan your database migration, this includes setting up a replication subnet group.
- Allocate a replication instance that performs all the processes for the migration.
- Specify a source and a target database endpoint.
- Create a task or set of tasks to define what tables and replication processes you want to use.

These walkthroughs provide an example of using the AMS Console or AMS CLI to create a Database Migration Service (DMS). CLI commands for creating the DMS replication instance, subnet group, and task as well as a DMS source endpoint and target endpoint are provided.

To learn more about AMS DMS, see AWS Database Migration Service for general information and AWS Database Migration Service FAQs for answers to common questions.

Database Migration Service (DMS), Planning

When planning a database migration using the AMS Database Migration Service, consider the following:

- Source and Target Endpoints: You need to know what information and tables in the source database need to be migrated to the target database. AMS DMS supports basic schema migration, including the creation of tables and primary keys. However, AMS DMS doesn't automatically create secondary indexes, foreign keys, user accounts, and so on in the target database. See Sources for Data Migration and Targets for Data Migration for more information.

- Schema/Code Migration: AMS DMS doesn't perform schema or code conversion. You can use tools such as Oracle SQL Developer, MySQL Workbench, or pgAdmin III to convert your schema. If you want to convert an existing schema to a different database engine, you can use the AWS Schema Conversion Tool. It can create a target schema and also can generate and create an entire schema: tables, indexes, views, and so on. You can also use the tool to convert PL/SQL or TSQL to PgSQL and other formats.
- Unsupported Data Types: Some source data types need to be converted into the equivalent data types for the target database.

DMS Scenarios to Consider

The following, documented, scenarios might help you craft your own database migration path.

- Migrate data from an on-prem MySQL server to Amazon RDS MySQL: See AWS blog post Migrate On-Premises MySQL Data to Amazon RDS (and back)
- Migrate data from an Oracle database to Amazon RDS Aurora PostgreSQL database: See AWS blog post A quick introduction to migrating from an Oracle database to an Amazon Aurora PostgreSQL database
- Migrate data from RDS MySQL to S3: See AWS blog post How to archive data from relational databases to Amazon Glacier using AWS DMS

DMS Setup, All Tasks

For each of the following DMS walkthroughs, some data in common is needed.

REQUIRED DATA, All DMS CTs:

- `Description`: Meaningful information about the resource, this is separate from other parameter `Description` options.
- `VpcId`: The VPC to use. You can find this out by running the ListVpcSummaries operation of the SKMS API (`list-vpc-summaries` in the CLI) or on the **VPCs** page in the AMS Console.
- `Name`: A name for the stack or stack component; this becomes the Stack Name.
- `TimeoutInMinutes`: How many minutes are allowed for the creation of the stack before the RFC is failed. This setting will not delay the RFC execution, but you must give enough time (for example, don't specify `"5"`).
- `ChangeTypeId`, `ChangeTypeVersion`, and `StackTemplateId`: These are required but vary per CT and their values are provided in each relevant section, following.

Create DMS replication subnet group

As part of the network to use for database migration, you need to specify what subnets in your AMS VPC you plan to use. A subnet is a range of IP addresses in your VPC in a given Availability Zone (AZ). These subnets can be distributed among the AZs for the AMS Region where your VPC is located.

You create a replication instance in a subnet that you select, and you can manage what subnet a source or target endpoint uses by using the AWS DMS console.

Classification: Deployment | Advanced stack components | Database Migration Service (DMS) | Create replication subnet group

For more information about DMS replication instances and subnet groups, see Setting Up a Network for a Replication Instance.

Note
This CT fails if the 'dms-vpc-role' IAM role doesn't exist in the account.

Required Data:

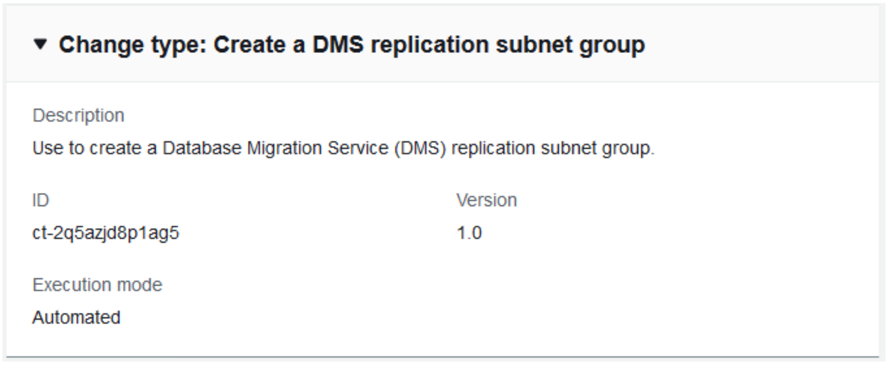
- **Description**: Meaningful information about the resource.
- **Parameters**: **SubnetIds**: Two or more subnet IDs for the replication subnet group. For help finding subnet IDs, see Find Subnet.

Optional Data:

- **Identifier**: A name for the replication subnet group that contains from 8 to 16 printable ASCII characters (excluding /,", and @), cannot exceed 255 alphanumeric characters, periods, spaces, underscores, or hyphens. Must not be "default". Given a unique ID if none is provided.
- **Description**: The description for the replication subnet group.

Note
You can add up to 50 tags, but to do so you must enable the **Additional configuration** view.

Creating a DMS Replication Subnet Group with the Console



▼ Change type: Create a DMS replication subnet group

Description
Use to create a Database Migration Service (DMS) replication subnet group.

ID
ct-2q5azjd8p1ag5

Version
1.0

Execution mode
Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Creating a DMS Replication Subnet Group with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotes when providing execution parameters inline) and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws --profile saml --region us-east-1 amscm create-rfc --change-type-id "ct-2q5azjd8p1ag5"
 --change-type-version "1.0" --title "TestDMSRepSG" --execution-parameters "{\"Description
\":\"DMSTestRepSG\",\"VpcId\":\"VPC-ID\",\"Name\":\"Test Stack\",\"Parameters\":
{\"Description\":\"DESCRIPTION\",\"SubnetIds\":[\"SUBNET-ID\", \"SUBNET-ID\"]},
\"TimeoutInMinutes\":60,\"StackTemplateId\":\"stm-j637f961s1h4oy5fj\"}"
```

TEMPLATE CREATE:

1. Output the execution parameters for this change type to a JSON file; this example names it CreateDmsRsgParams.json:

```
aws amscm get-change-type-version --change-type-id "ct-2q5azjd8p1ag5" --query
 "ChangeTypeVersion.ExecutionInputSchema" --output text > CreateDmsRsgParams.json
```

2. Modify and save the execution parameters CreateDmsRsgParams.json file. For example, you can replace the contents with something like this:

```
{
"Description":           "DMSTestRepSG",
"VpcId":                 "VPC_ID",
"TimeoutInMinutes":      60,
"StackTemplateId":       "stm-j637f961s1h4oy5fj",
"Name":                  "Test RSG",
"Parameters":   {
    "Description":             "DESCRIPTION",
    "SubnetIds":               ["SUBNET_ID", "SUBNET_ID"]
    }
}
```

3. Output the JSON template to a file in your current folder; this example names it CreateDmsRsgRfc.json:

```
aws amscm create-rfc --generate-cli-skeleton > CreateDmsRsgRfc.json
```

4. Modify and save the CreateDmsRsgRfc.json file. For example, you can replace the contents with something like this:

```
{
"ChangeTypeVersion":     "1.0",
"ChangeTypeId":          "ct-2q5azjd8p1ag5",
"Title":                 "DMS-RSG-Create-RFC"
}
```

5. Create the RFC, specifying the execution parameters file and the CreateDmsRsgRfc file:

```
aws amscm create-rfc --cli-input-json file://CreateDmsRsgRfc.json --execution-
parameters file://CreateDmsRsgParams.json
```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

2: Create DMS replication instance

You must create a replication instance on an EC2 instance in your AMS VPC that has sufficient storage and processing power to perform the tasks you assign and migrate data from your source database to the target database. The required size of this instance varies depending on the amount of data you need to migrate and the tasks that you need the instance to perform. The replication instance provides high availability and failover support using a Multi-AZ deployment when you select the `MultiAZ` option. For more information about replication instances, see Working with an AWS DMS Replication Instance.

Classification: Deployment | Advanced stack components | Database Migration Service (DMS) | Create replication instance

Required Data, DMS Replication Instance:

- **Parameters**:
  - **InstanceClass**: The Amazon EC2 instance class for the replication instance to use to perform your database migration, in the form **dms.t2.micro**. AMS DMS currently supports the T2, C4, and R4 Amazon EC2 instance classes for replication instances. For information on how to determine which instance class is best for your migration, see Working with an AWS DMS Replication Instance.
  - **ReplicationSubnetGroupIdentifier**: The replication subnet group identifier to associate with the replication instance. You create this in Create DMS replication subnet group (p. 86).
  - **SecurityGroupIds**: The identifiers of the security groups to control traffic to and from the replication instance. If your source database is in a VPC, select the VPC security group that provides access to the DB instance where the database resides. Use the EC2 or VPC console to view all security groups for the selected VPC.

    If needed, you can create a security group first; for details see Create Security Group.

Optional Data, DMS Replication Instance:

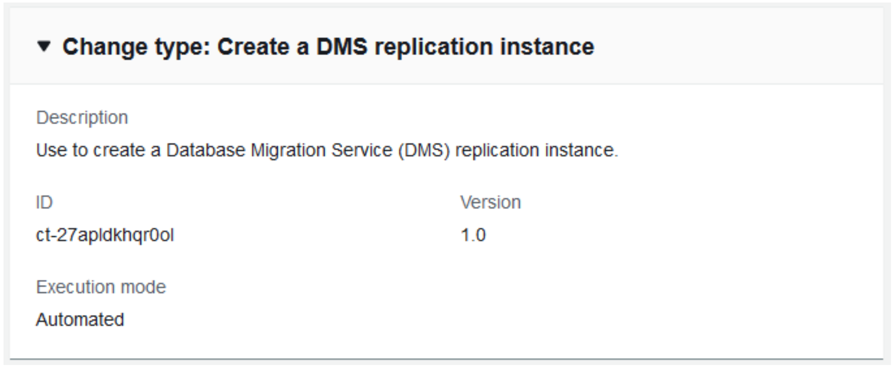
- **AllocatedStorage**: The amount of storage, in gigabytes, to be initially allocated for the replication instance.

- **AutoMinorVersionUpgrade**: **True** if the replication instance should have automatic minor engine upgrade during the maintenance window. **False** if it should not.
- **AvailabilityZone**: The availability zone where your source database is located. Only applicable if **MultiAZ** = **false**.
- **EngineVersion**: The engine version number of the replication instance, in the form 2.4.3. By default, the replication instance runs the latest version of the AWS DMS replication engine software; AMS recommends accepting the default by leaving this option blank.
- **KmsKeyId**: The KMS key identifier to use to encrypt replication storage and connection information. For information on using the encryption key, see Setting an Encryption Key and Specifying KMS Permissions.
- **MultiAZ**: **True** if the replication instance is a Multi-AZ deployment. **False** if it is not. Use this optional parameter to create a standby replica of your replication instance in another Availability Zone for failover support. If you intend to use change data capture (CDC) or ongoing replication, for your replication task, you should enable this option.
- **PreferredMaintenanceWindow**: The weekly time range during which system maintenance can occur, in Universal Coordinated Time (UTC). Must be in the format ddd:hh24:mi-ddd:hh24:mi, and must be at least 30 minutes.
- **Identifier**: The identifier for the replication instance. Given a unique ID if none is provided.

Note
You can add up to 50 tags, but to do so you must enable the **Additional configuration** view.

Creating a DMS Replication Instance with the Console

Screenshot of this change type in the AMS console:


▼ Change type: Create a DMS replication instance

Description
Use to create a Database Migration Service (DMS) replication instance.

ID
ct-27apldkhqr0ol

Version
1.0

Execution mode
Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.

6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Creating a DMS Replication Instance with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotes when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws --profile saml --region us-east-1 amscm create-rfc --change-type-id
 "ct-27apldkhqr0ol" --change-type-version "1.0" --title "TestDMSRepInstance" --
execution-parameters "{\"Description\":\"DMSTestRepInstance\",\"VpcId\":\"VPC-ID\",
\"Name\":\"REP-INSTANCE-NAME\",\"Parameters\":{\"InstanceClass\":\"dms.t2.micro\",
\"ReplicationSubnetGroupIdentifier\":\"TEST-REP-SG\",\"SecurityGroupIds\":\"SG-ID, SG-
ID\"},\"TimeoutInMinutes\":60,\"StackTemplateId\":\"stm-3n1j5hdrmiiiuqk6v\"}"
```

While your replication instance is being created, you can specify the source and target data stores. The source and target data stores can be on an Amazon Elastic Compute Cloud (Amazon EC2) instance, an AWS S3 Bucket, an Amazon Relational Database Service (Amazon RDS) DB instance, or an on-premises database.

TEMPLATE CREATE:

1. Output the execution parameters for this change type to a JSON file; this example names it CreateDmsRiParams.json:

   ```
   aws amscm get-change-type-version --change-type-id "ct-27apldkhqr0ol" --query
    "ChangeTypeVersion.ExecutionInputSchema" --output text > CreateDmsRiParams.json
   ```

2. Modify and save the execution parameters CreateDmsRiParams.json file. For example, you can replace the contents with something like this:

```
{
"Description":           "DMSTestRepInstance",
"VpcId":                 "VPC_ID",
"Name":                  "Test RI",
"StackTemplateId":       "stm-3n1j5hdrmiiiuqk6v",
"TimeoutInMinutes":      60,
"Parameters":   {
    "Description":                          "DESCRIPTION",
    "InstanceClass":                        "dms.t2.micro",
    "ReplicationSubnetGroupIdentifier":  "TEST-REP-SG",
    "SecurityGroupIds":                     ["SG-ID, SG-ID"}
    }
}
```

3. Output the JSON template to a file in your current folder; this example names it CreateDmsRiRfc.json:

```
aws amscm create-rfc --generate-cli-skeleton > CreateDmsRiRfc.json
```

4. Modify and save the CreateDmsRiRfc.json file. For example, you can replace the contents with something like this:

```
{
"ChangeTypeVersion":     "1.0",
"ChangeTypeId":          "ct-27apldkhqr0ol",
"Title":                 "DMS-RI-Create-RFC"
}
```

5. Create the RFC, specifying the execution parameters file and the CreateDmsRiRfc file:

```
aws amscm create-rfc --cli-input-json file://CreateDmsRiRfc.json --execution-parameters
 file://CreateDmsRiParams.json
```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

3: DMS Source Endpoint: Create, Create for Mongo DB, Create for S3

You can use the AMS console or API/CLI to create an AMS DMS source endpoint for various databases, we provide three examples.

DMS source endpoint: creating

AMS Database Migration Service (DMS) can use many of the most popular data engines as a source for data replication. The database source can be a self-managed engine running on an Amazon EC2 instance or an on-premises database. It can be a data source such as Amazon S3 or Amazon RDS with Aurora, AzureDB, IBM Db2, MariaDB, MySQL, Oracle, Postgressql, SQL Servier, or SAP ASE (formerly SAP Sybase ASE).

Classification: Deployment | Advanced stack components | Database Migration Service (DMS) | Create source endpoint

Change type ID: ct-0attesnjqy2cx

Version:

To learn more, see Sources for Data Migration.

For an S3 source endpoint, see DMS source endpoint for S3: creating (p. 99).

For a Mongo DB source endpoint, see DMS source endpoint for MongoDB: Creating (p. 95).

Required Data, DMS Source Endpoint:

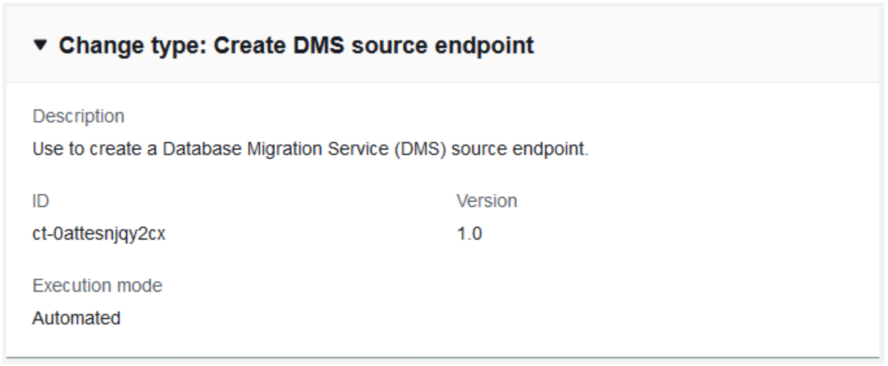
- **Parameters**:
  - **EngineName**: The type of engine this source endpoint is connected to. Some parameters become required depending on the specified **EngineName**.
  - **ServerName**: The name of the server where the source database resides.
  - **Port**: The port used by the source database.
  - **Username**: The user name to be used to log in to the source database.
  - **Password**: The password to be used to log in to the source database.

Optional Data, DMS Source Endpoint:

- **EndpointIdentifier**: A meaningful identifier for the source database endpoint. Must be unique for all endpoints owned by your AWS account in the current region. Must begin with a letter, must contain only ASCII letters, digits and hyphens and must not end with a hyphen or contain two consecutive hyphens.
- **DatabaseName**: The name of the source database. Must not be blank if **EngineName** = **azuredb**, **db2**, **oracle**, **postgres**, **sqlserver**, or **sybase**.
- **SslMode**: The SSL mode to use for the SSL connection. For details, see Using SSL With AWS Database Migration Service.
- **CertificateArn**: The Amazon Resource Name (ARN) for the certificate to use with the source. This is required if **SslMode** = **verify-ca** or **verify-full**.
- **KmsKeyId**: The AWS Key Management Service (AWS KMS) customer master key (CMK) ID to use for encrypting volumes associated with the replication instance. If not specified, the default CMK for Amazon DMS is used.
- **ExtraConnectionAttributes**: Additional attributes associated with the connection. For more information on the supported extra connection attributes for the **EngineName** you have selected, see Sources for Data Migration.

Creating a DMS Source Endpoint with the Console

Screenshot of this change type in the AMS console:



▼ **Change type: Create DMS source endpoint**

Description
Use to create a Database Migration Service (DMS) source endpoint.

ID	Version
ct-0attesnjqy2cx	1.0

Execution mode
Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.

2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Creating a DMS Source Endpoint with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotes when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws --profile saml --region us-east-1 amscm create-rfc --title "MariaDB-DMS-Source-
Endpoint" --aws-account-id ACCOUNT-ID --change-type-id ct-0attesnjqy2cx --change-type-
version 1.0 --execution-parameters "{\"Description\":\"DESCRIPTION.\",\"VpcId\":\"VPC-
ID\",\"Name\":\"MariaDB-DMS-SE\",\"Parameters\":{\"EngineName\":\"mariadb\",\"ServerName\":
\"mariadb.db.example.com\",\"Port\":3306,\"Username\":\"DB-USER\",\"Password\":\"DB-PW\"},
\"TimeoutInMinutes\":60,\"StackTemplateId\":\"stm-pud4ghhkp7395n9bc\"}"
```

TEMPLATE CREATE:

1. Output the execution parameters for this change type to a JSON file named CreateDmsSeParams.json.

```
aws amscm get-change-type-version --change-type-id "ct-0attesnjqy2cx" --query
 "ChangeTypeVersion.ExecutionInputSchema" --output text > CreateDmsSeParams.json
```

2. Modify and save the execution parameters JSON file. For example, you can replace the contents with something like this:

```
{
"Description":            "MariaDB-DMS-SE",
"VpcId":                  "VPC_ID",
"Name":                   "Test SE",
"StackTemplateId":        "stm-pud4ghhkp7395n9bc",
"TimeoutInMinutes":       60,
"Parameters":    {
    "Description":        "DESCRIPTION",
    "EngineName":         "mariadb",
    "ServerName":         "mariadb.db.example.com",
    "Port":               "3306",
    "Username":           "DB-USER",
    "Password":           "DB-PW",}
    }
}
```

3. Output the JSON template to a file in your current folder; this example names it CreateDmsSeRfc.json:

```
aws amscm create-rfc --generate-cli-skeleton > CreateDmsSeRfc.json
```

4. Modify and save the CreateDmsSeRfc.json file. For example, you can replace the contents with something like this:

```
{
"ChangeTypeVersion":      "1.0",
"ChangeTypeId":           "ct-0attesnjqy2cx",
"Title":                  "MariaDB-DMS-Source-Endpoint"
}
```

5. Create the RFC, specifying the execution parameters file and the CreateDmsSeRfc file:

```
aws amscm create-rfc --cli-input-json file://CreateDmsSeRfc.json --execution-parameters
 file://CreateDmsSeParams.json
```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

DMS source endpoint for MongoDB: Creating

AMS DMS can use Mongo or any Relational Database Service (RDS) as a source endpoint. For an S3 source endpoint, see DMS source endpoint for S3: creating (p. 99).

Classification: Deployment | Advanced stack components | Database Migration Service (DMS) | Create source endpoint (MongoDB)

Required Data:

- **Parameters**:

- **EngineName**: Must be **mongodb**.
- **ServerName**: The name of the server where the source database resides.
- **Port**: The port used by the source database.
- **DatabaseName**: The name of the source database.

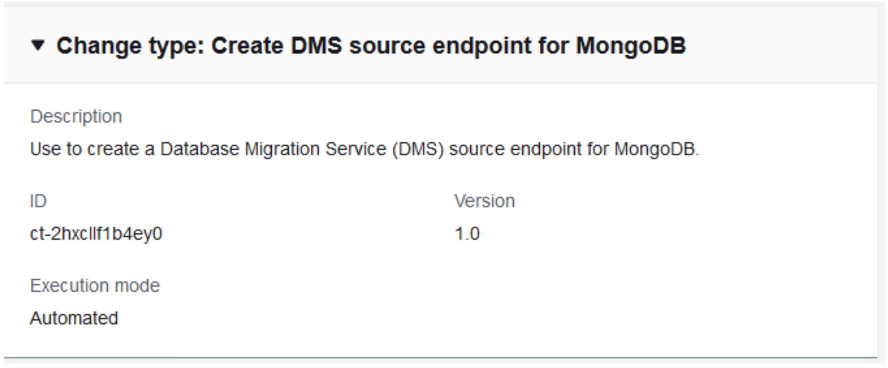
Optional Data:

- **EndpointIdentifier**: A meaningful identifier for the source database endpoint. Must be unique for all endpoints owned by your AWS account in the current region. Must begin with a letter, must contain only ASCII letters, digits and hyphens and must not end with a hyphen or contain two consecutive hyphens.
- **Username**: The user name to be used to log in to the source database.
- **Password**: The password to be used to log in to the source database.
- **EndpointIdentifier**: A meaningful identifier for the source database endpoint. Must be unique for all endpoints owned by your AWS account in the current region. Must begin with a letter, must contain only ASCII letters, digits and hyphens and must not end with a hyphen or contain two consecutive hyphens.
- **ExtraConnectionAttributes**: Additional attributes associated with the connection. For more information, see Using MongoDB as a Source for AWS DMS.
- **SslMode**: The SSL mode to use for the SSL connection. For details, see Using SSL With AWS Database Migration Service.
- **CertificateArn**: The Amazon Resource Name (ARN) for the certificate to use with the source. This is required if **SslMode** = **verify-ca** or **verify-full**.
- **ExtraConnectionAttributes**: Additional attributes associated with the connection. For more information on the supported extra connection attributes for the **EngineName** you have selected, see Sources for Data Migration.
- **MongoDbAuthType**: The authentication type or mode used to access the MongoDB source endpoint.
- **MongoDbAuthMechanism**: The authentication mechanism used to access the MongoDB source endpoint. Do not use if **MongoDbAuthType** = no.
- **MongoDbAuthSource**: The MongoDB database name. Do not use if **MongoDbAuthType** = no.
- **MongoDbMetadataMode**: The mode used for MongoDB metadata. For document mode use **none**, for table mode use **one**.
- **MongoDbDocsToInvestigate**: The number of documents to preview to determine the document organization. Use if **MongoDbMetadataMode** = one. Must be a positive value greater than 0.
- **MongoDbExtractDocId**: True to extract the MongoDB document ID as a separate column; false to not. Use if **MongoDbMetadataMode** = none.

Note
You can add up to 50 tags, but to do so you must enable the **Additional configuration** view.

Creating a DMS Mongo DB Source Endpoint with the Console

Screenshot of this change type in the AMS console:


▼ Change type: Create DMS source endpoint for MongoDB

Description
Use to create a Database Migration Service (DMS) source endpoint for MongoDB.

ID
ct-2hxcllf1b4ey0

Version
1.0

Execution mode
Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Creating a DMS Mongo DB Source Endpoint with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution

parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotes when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws amscm  --profile saml --region us-east-1 create-rfc --change-type-id "ct-2hxcllf1b4ey0"
 --change-type-version "1.0" --title 'DMS_Source_MongoDB' --description "DESCRIPTION"
  --execution-parameters "{\"Description\":\"DMS_MongoDB_Source_Endpoint\",\"VpcId\":
\"VPC_ID\",\"Name\":\"DMS-Mongo-SE\",\"StackTemplateId\":\"stm-pud4ghhkp7395n9bc\",
\"TimeoutInMinutes\":60,\"Parameters\":{\"DatabaseName\":\"mytestdb\",\"EngineName\":
\"mongodb\",\"Port\":27017,\"ServerName\":\"test.example.com\"}}"
```

TEMPLATE CREATE:

1. Output the execution parameters for this change type to a JSON file named CreateDmsSeMongoParams.json.

```
aws amscm get-change-type-version --change-type-id "ct-2hxcllf1b4ey0" --query
 "ChangeTypeVersion.ExecutionInputSchema" --output text > CreateDmsSeMongoParams.json
```

2. Modify and save the execution parameters JSON file. For example, you can replace the contents with something like this:

```
{
"Description":            "MongoDB-DMS-SE",
"VpcId":                  "VPC_ID",
"StackTemplateId":        "stm-pud4ghhkp7395n9bc",
"Name":                   "Test Mongo SE",
"TimeoutInMinutes":       60,
"Parameters":    {
    "Description":        "DESCRIPTION",
    "DatabaseName":         "mytestdb",
    "EngineName":         "mongodb",
    "ServerName":         "test.example.com",
    "Port":               "27017"
    }
}
```

3. Output the JSON template to a file in your current folder; this example names it CreateDmsSeMongoRfc.json:

```
aws amscm create-rfc --generate-cli-skeleton > CreateDmsSeMongoRfc.json
```

4. Modify and save the CreateDmsSeMongoRfc.json file. For example, you can replace the contents with something like this:

```
{
"ChangeTypeVersion":    "1.0",
"ChangeTypeId":         "ct-2hxcllf1b4ey0",
"Title":                "DMS_Source_MongoDB"
}
```

5. Create the RFC, specifying the execution parameters file and the CreateDmsSeMongoRfc file:

```
aws amscm create-rfc --cli-input-json file://CreateDmsSeMongoRfc.json --execution-
parameters file://CreateDmsSeMongoParams.json
```

You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

DMS source endpoint for S3: creating

AMS DMS can use S3 or any Relational Database Service (RDS) source endpoint. For a Mongo DB source endpoint, see DMS source endpoint for MongoDB: Creating (p. 95).

Classification: Deployment | Advanced stack components | Database Migration Service (DMS) | Create source endpoint (S3)

Required Data, DMS Source Endpoint (S3):

- **Parameters**:
  - **EngineName**: Must be **s3**.
  - **S3BucketName**: The name of the S3 bucket where the source database resides. You can check bucket names in the S3 Console.
  - **S3ExternalTableDefinition**: The definition of the external table. A JSON document describing the structure of the tables and columns in the CSV files. For more information, see Defining External Tables for S3 as a Source for AWS DMS.
  - **S3ServiceAccessRoleArn**: The Amazon Resource Name (ARN) of the service access IAM role.

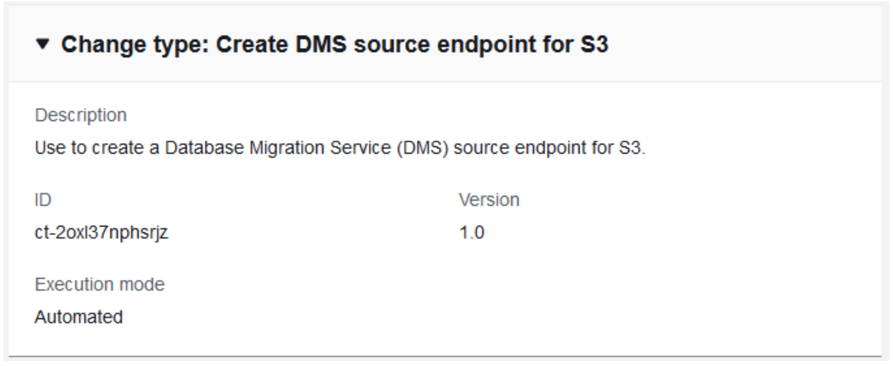
Optional Data, DMS Source Endpoint (S3):

- **EndpointIdentifier**: A meaningful identifier for the source database endpoint. Must be unique for all endpoints owned by your AWS account in the current region. Must begin with a letter, must contain only ASCII letters, digits and hyphens and must not end with a hyphen or contain two consecutive hyphens.
- **S3BucketFolder**: The folder name in the S3 bucket. This is the Amazon S3 bucket path where the CSV files can be found. If specified, source data files and CDC files are read from the path `BUCKET-FOLDER/SCHEMA-NAME/TABLE-NAME/`. If not specified, then the path used is `SCHEMA-NAME/TABLE-NAME/`.
- **S3CompressionType**: Type of compression to use. **GZIP** or **NONE**.
- **S3CsvDelimiter**: The delimiter used to separate columns in the source files. The default (used if unspecified) is a comma.
- **S3CsvRowDelimiter**: The delimiter used to separate rows in the source files. The default (used if unspecified) is a carriage return (\\n).
- **ExtraConnectionAttributes**: Additional attributes associated with the connection. For more information on the supported extra connection attributes for an S3 DMS source endpoint, see Extra Connection Attributes for S3 as a Source for AWS DMS.

Note
You can add up to 50 tags, but to do so you must enable the **Additional configuration** view.

Creating a DMS S3 Source Endpoint with the Console

Screenshot of this change type in the AMS console:



▼ Change type: Create DMS source endpoint for S3

Description
Use to create a Database Migration Service (DMS) source endpoint for S3.

ID
ct-2oxl37nphsrjz

Version
1.0

Execution mode
Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Creating a DMS S3 Source Endpoint with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution

parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotes when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws --profile saml --region us-east-1 amscm create-rfc --title "S3DMSSourceEndpoint"
 --aws-account-id ACCOUNT-ID --change-type-id ct-2oxl37nphsrjz --change-type-version
 1.0 --execution-parameters "{\"Description\":\"TestS3DMS-SE\",\"VpcId\":\"VPC-ID\",
\"Name\":\"S3-DMS-SE\",\"Parameters\":{\"EngineName\":\"s3\",\"S3BucketName\":\"BUCKET-
NAME\",\"S3ExternalTableDefinition\":\"{\\\"TableCount\\\":\\\"1\\\",\\\"Tables\\\":[{\
\\"TableName\\\":\\\"employee\\\",\\\"TablePath\\\":\\\"hr/employee/\\\",\\\"TableOwner
\\\":\\\"hr\\\",\\\"TableColumns\\\":[{\\\"ColumnName\\\":\\\"Id\\\",\\\"ColumnType\\
\":\\\"INT8\\\",\\\"ColumnNullable\\\":\\\"false\\\",\\\"ColumnIsPk\\\":\\\"true\\\"},
{\\\"ColumnName\\\":\\\"LastName\\\",\\\"ColumnType\\\":\\\"STRING\\\",\\\"ColumnLength
\\\":\\\"20\\\"},{\\\"ColumnName\\\":\\\"FirstName\\\",\\\"ColumnType\\\":\\\"STRING\\
\",\\\"ColumnLength\\\":\\\"30\\\"},{\\\"ColumnName\\\":\\\"HireDate\\\",\\\"ColumnType
\\\":\\\"DATETIME\\\"},{\\\"ColumnName\\\":\\\"OfficeLocation\\\",\\\"ColumnType\\\":\\
\"STRING\\\",\\\"ColumnLength\\\":\\\"20\\\"}],\\\"TableColumnsTotal\\\":\\\"5\\\"}]}\",
\"S3ServiceAccessRoleArn\":\"arn:aws:iam::945533541580:role/ams-ops-ct-authors-dms-s3-test-
role\"},\"TimeoutInMinutes\":60,\"StackTemplateId\":\"stm-pud4ghhkp7395n9bc\"}"
```

TEMPLATE CREATE:

1. Output the execution parameters for this change type to a JSON file named CreateDmsSeS3Params.json.

   ```
   aws amscm get-change-type-version --change-type-id "ct-2oxl37nphsrjz" --query
    "ChangeTypeVersion.ExecutionInputSchema" --output text > CreateDmsSeS3Params.json
   ```

2. Modify and save the execution parameters JSON file. For example, you can replace the contents with something like this:

   ```
   {
   "Description":           "TestS3DMS-SE",
   "VpcId":                 "VPC_ID",
   "Name":                  "S3-DMS-SE",
   "StackTemplateId":       "stm-pud4ghhkp7395n9bc",
   "TimeoutInMinutes":      60,
   "Parameters":   {
       "EngineName":                "s3",
       "S3BucketName":               "BUCKET-NAME",
       "S3ExternalTableDefinition":  "BUCKET-NAME",
       {"TableCount":                "1",
         "Tables":[{"TableName":"employee","TablePath":"hr/
   employee/","TableOwner":"hr","TableColumns":
   [{"ColumnName":"Id","ColumnType":"INT8","ColumnNullable":"false","ColumnIsPk":"true"},
   {"ColumnName":"LastName","ColumnType":"STRING","ColumnLength":"20"},
   {"ColumnName":"FirstName","ColumnType":"STRING","ColumnLength":"30"},
   {"ColumnName":"HireDate","ColumnType":"DATETIME"},
   {"ColumnName":"OfficeLocation","ColumnType":"STRING","ColumnLength":"20"}],"TableColumnsTotal":"5"}
       "S3ServiceAccessRoleArn":      "arn:aws:iam::945533541580:role/ams-ops-ct-authors-
   dms-s3-test-role",
         }
   }
   ```

3. Output the JSON template to a file in your current folder; this example names it CreateDmsSeS3Rfc.json:

```
aws amscm create-rfc --generate-cli-skeleton > CreateDmsSeS3Rfc.json
```

4. Modify and save the CreateDmsSeS3Rfc.json file. For example, you can replace the contents with something like this:

```
{
"ChangeTypeVersion":     "1.0",
"ChangeTypeId":          "ct-2oxl37nphsrjz",
"Title":                 "DMS_Source_S3"
}
```

5. Create the RFC, specifying the execution parameters file and the CreateDmsSeS3Rfc file:

```
aws amscm create-rfc --cli-input-json file://CreateDmsSeS3Rfc.json --execution-
parameters file://CreateDmsSeS3Params.json
```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

4: DMS Target Endpoint: Create, Create for S3

You can use the AMS console or API/CLI to create an AMS DMS target endpoint for various databases, we provide two examples.

DMS target endpoint: creating

AMS DMS can use S3 or any Relational Database Service (RDS) with MySQL, MariaDB, Oracle, Postgresql, or Microsoft SQL as a target endpoint. For an S3 target endpoint, see DMS target endpoint for S3: creating (p. 105).

Classification: Deployment | Advanced stack components | Database Migration Service (DMS) | Create target endpoint

For more information, see Targets for Data Migration.

Required Data:

- **Parameters**:
  - **EngineName**: The type of engine this target endpoint is connected to. Amazon RDS-supported MySQL, MariaDB, PostgreSQL, Oracle and Microsoft SQL are the options.
  - **ServerName**: The name of the server where the target database resides. Use the endpoint of your target RDS database.
  - **Port**: The port used by the endpoint database.
  - **Username**: The user name to log in to the target database.
  - **Password**: The password to log in to the endpoint database.

Optional Data:

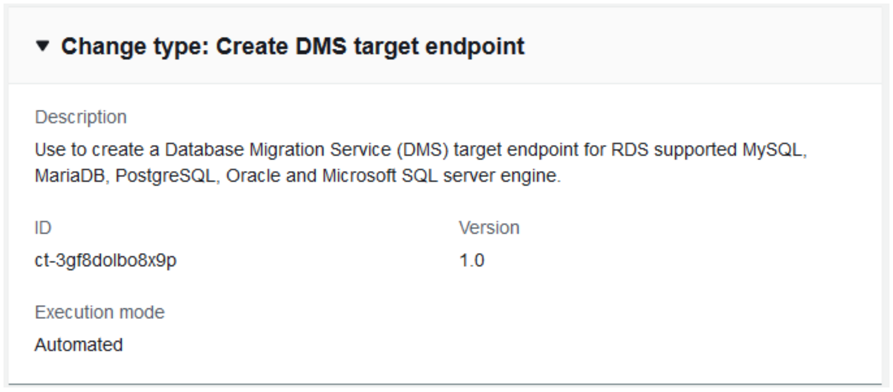
- **EndpointIdentifier**: The identifier to be used for the target endpoint. This is a label for the endpoint to help you identify it. It must be unique for all endpoints owned by your AWS account in the current region. It must begin with a letter, must contain only ASCII letters, digits and hyphens and must not end with a hyphen or contain two consecutive hyphens.
- **DatabaseName**: The name of the target database. Must not be blank if **EngineName** = **oracle**, **postgres**, or **sqlserver**.

- **SslMode**: The SSL mode to encrypt connections for target endpoint. Not all SSL modes work with all database endpoints. For more information, see Using SSL With AWS Database Migration Service.
- **CertificateArn**: The Amazon Resource Name (ARN) for the certificate to use with the target. This is required if **SslMode** = **verify-ca** or **verify-full**. To find ARNs, look in the relevant Console.
- **KmsKeyId**: This is the customer master key (CMK) that is used to encrypt connection parameters. If not specified the default CMK for AWS DMS is used.
- **ExtraConnectionAttributes**: Additional attributes associated with the connection. For example, to disable foreign key checks in MySQL compatible database as targets add **initstmt=SET FOREIGN_KEY_CHECKS=0**.

Note
You can add up to 50 tags, but to do so you must enable the **Additional configuration** view.

Creating a DMS Target Endpoint with the Console

Screenshot of this change type in the AMS console:



▼ Change type: Create DMS target endpoint

Description
Use to create a Database Migration Service (DMS) target endpoint for RDS supported MySQL, MariaDB, PostgreSQL, Oracle and Microsoft SQL server engine.

ID
ct-3gf8dolbo8x9p

Version
1.0

Execution mode
Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Creating a DMS Target Endpoint with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and

one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.

2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotes when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws --profile saml --region us-east-1 amscm create-rfc --change-type-id "ct-3gf8dolbo8x9p"
 --change-type-version "1.0" --title "TestDMSTargetEndpointSql" --execution-parameters
 "{\"Description\":\"TestSQLTE\",\"VpcId\":\"VPC-ID\",\"Name\":\"SQLTE-NAME\",
\"StackTemplateId\":\"stm-knghtmmgefafdq89u\",\"TimeoutInMinutes\":60,\"Parameters\":
{\"EngineName\":\"mysql\",\"Password\":\"testpw123\",\"Port\":\"3306\",\"ServerName\":
\"mytestdb.d5fga0rf2wpi.ap-southeast-2.rds.amazonaws.com\",\"Username\":\"USERNAME\"}}"
```

TEMPLATE CREATE:

1. Output the execution parameters for this change type to a JSON file named CreateDmsTeParams.json.

```
aws amscm get-change-type-version --change-type-id "ct-3gf8dolbo8x9p" --query
 "ChangeTypeVersion.ExecutionInputSchema" --output text > CreateDmsTeParams.json
```

2. Modify and save the execution parameters JSON file. For example, you can replace the contents with something like this:

```
{
"Description":           "TestSQLTE",
"VpcId":                 "VPC_ID",
"StackTemplateId":       "stm-knghtmmgefafdq89u",
"Name":                  "SQLTE-NAME",
"TimeoutInMinutes":      60,
"Parameters":   {
    "EngineName":        "mysql",
    "ServerName":        "sql.db.example.com",
    "Port":              "3306",
    "Username":          "DB-USER",
    "Password":          "DB-PW",}
    }
}
```

3. Output the JSON template to a file in your current folder; this example names it CreateDmsTeRfc.json:

```
aws amscm create-rfc --generate-cli-skeleton > CreateDmsTeRfc.json
```

4. Modify and save the CreateDmsTeRfc.json file. For example, you can replace the contents with something like this:

```
{
"ChangeTypeVersion":    "1.0",
"ChangeTypeId":         "ct-3gf8dolbo8x9p",
"Title":                "SQLDB-DMS-Target-Endpoint"
}
```

5. Create the RFC, specifying the execution parameters file and the CreateDmsTeRfc file:

```
aws amscm create-rfc --cli-input-json file://CreateDmsTeRfc.json --execution-parameters
 file://CreateDmsTeParams.json
```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

DMS target endpoint for S3: creating

AMS provides a separate change type for creating a target endpoint for S3. For more information, see Using Amazon S3 as a Target for AWS Database Migration Service and Extra Connection Attributes When Using Amazon S3 as a Target for AWS DMS.

Classification: Deployment | Advanced stack components | Database Migration Service (DMS) | Create target endpoint (S3)

Required Data, DMS Target Endpoint (S3):

- **Parameters**:
  - **EngineName**: Must be **s3**.
  - **S3BucketName**: The name of the S3 bucket for the target endpoint. Must be in the same region as the DMS replication instance you are using to migrate data. To check bucket names, see the S3 Console.
  - **S3ServiceAccessRoleArn**: The Amazon Resource Name (ARN) of the service access IAM role. To find ARNs, look in the relevant Console.

Optional Data, DMS Target Endpoint (S3):

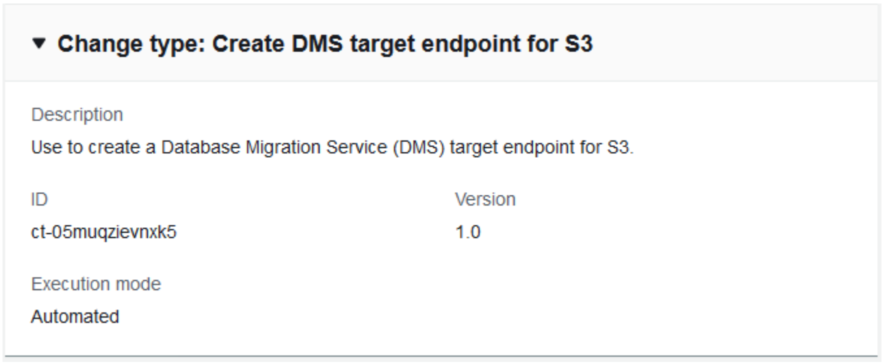
- **EndpointIdentifier**: A meaningful identifier for the source database endpoint. Must be unique for all endpoints owned by your AMS account in the current region. Must begin with a letter, must contain only ASCII letters, digits and hyphens and must not end with a hyphen or contain two consecutive hyphens.
- **S3BucketFolder**: The folder name in the S3 bucket. If provided, tables are created in the path `BUCKET-FOLDER/SCHEMA-NAME/TABLE-NAME/`. If not specified, then the path used is `SCHEMA-NAME/TABLE-NAME/` within the bucket.
- **S3CompressionType**: Type of compression to use. GZIP or NONE.
- **S3CsvDelimiter**: The delimiter used to separate columns in the source files. The default is a comma.
- **S3CsvRowDelimiter**: The delimiter used to separate rows in the source files. The default is a carriage return (\\n).

- **ExtraConnectionAttributes**: Additional attributes associated with the connection. For example, to specify a maximum file size of 512 KB of any CSV file created while migrating to S3 specify `maxFileSize=512`.

Note
You can add up to 50 tags, but to do so you must enable the **Additional configuration** view.

Creating a DMS S3 Target Endpoint with the Console

Screenshot of this change type in the AMS console:



▼ **Change type: Create DMS target endpoint for S3**

Description
Use to create a Database Migration Service (DMS) target endpoint for S3.

ID
ct-05muqzievnxk5

Version
1.0

Execution mode
Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Creating a DMS S3 Target Endpoint with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id` *`ID`* command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id` *`ID`* command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

> **Note**
> You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotes when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws --profile saml --region us-east-1 amscm create-rfc --change-type-id "ct-05muqzievnxk5"
 --change-type-version "1.0" --title "TestDMSTargetEndpointS3" --execution-parameters
 "{\"Description\":\"TestS3TE\",\"VpcId\":\"VPC-ID\",\"Name\":\"S3TE-NAME\",
\"StackTemplateId\":\"stm-knghtmmgefafdq89u\",\"TimeoutInMinutes\":60,\"Parameters\":
{\"EngineName\":\"s3\",\"S3BucketName\":\"mybucket.in.s3\",\"S3ServiceAccessRoleArn\":
\"arn:aws:iam::123456789123:role/my-s3-role\"}}"
```

TEMPLATE CREATE:

1. Output the execution parameters for this change type to a JSON file; this example names it CreateDmsTeS3Params.json:

   ```
   aws amscm get-change-type-version --change-type-id "ct-05muqzievnxk5" --query
    "ChangeTypeVersion.ExecutionInputSchema" --output text > CreateDmsTeS3Params.json
   ```

2. Modify and save the execution parameters CreateDmsTeS3Params.json file. For example, you can replace the contents with something like this:

   ```
   {
   "Description":            "TestS3DMS-TE",
   "VpcId":                  "VPC_ID",
   "StackTemplateId":        "stm-knghtmmgefafdq89u",
   "Name":                   "DMS-S3-TE",
   "TimeoutInMinutes":       60,
   "Parameters":    {
       "EngineName":         "s3",
       "S3BucketName":        "BUCKET-NAME",
       "S3ServiceAccessRoleArn":        "arn:aws:iam::945533541580:role/ams-ops-ct-authors-
   dms-s3-test-role"
         }
   }
   ```

3. Output the JSON template to a file in your current folder; this example names it CreateDmsTeS3Rfc.json:

   ```
   aws amscm create-rfc --generate-cli-skeleton > CreateDmsTeS3Rfc.json
   ```

4. Modify and save the CreateDmsTeS3Rfc.json file. For example, you can replace the contents with something like this:

   ```
   {
   ```

```
"ChangeTypeVersion":    "1.0",
"ChangeTypeId":         "ct-05muqzievnxk5",
"Title":                "DMS_Target_S3"
}
```

5. Create the RFC, specifying the execution parameters file and the CreateDmsTeS3Rfc file:

   ```
   aws amscm create-rfc --cli-input-json file://CreateDmsTeS3Rfc.json --execution-
   parameters file://CreateDmsTeS3Params.json
   ```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

5: Create DMS replication task

You can create a DMS task that captures three different types of changes or data. For more information, see Working with AWS DMS Tasks, Creating a Task, and Creating Tasks for Ongoing Replication Using AWS DMS.

Classification: Deployment | Advanced stack components | Database Migration Service (DMS) | Create replication task

Change type ID: ct-1d2fml15b9eth

Version:

Required Data, DMS Replication Task:

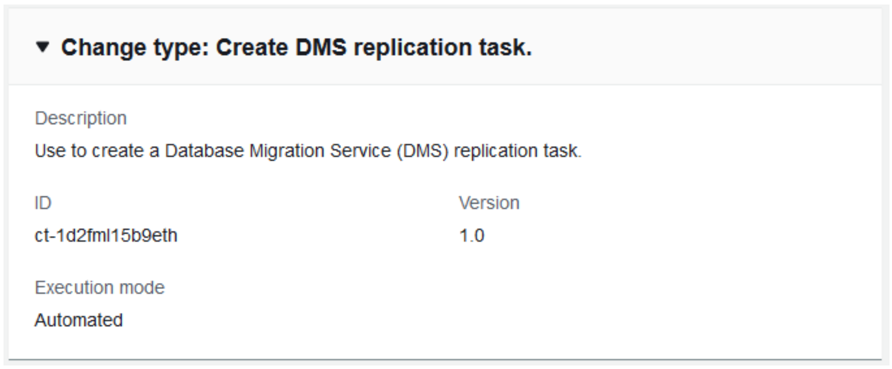
- **Parameters**:
  - **MigrationType**: The migration type or method. To migrate existing data use **full-load**, to migrate existing data and replicate ongoing changes use **full-load-and-cdc**, to replicate data changes only use **cdc**.
  - **ReplicationInstanceArn**: The Amazon Resource Name (ARN) of the DMS replication instance. To find ARNs, look in the relevant Console.
  - **SourceEndpointArn**: The Amazon Resource Name (ARN) of the DMS source endpoint for the task to use.
  - **TableMappings**: A JSON document to set rules for schema mapping, the mapping method, transformation and filters. For more information, see Using Table Mapping to Specify Task Settings.
  - **TargetEndpointArn**: The Amazon Resource Name (ARN) of the DMS target endpoint for the task to use.

Optional Data, DMS Replication Task:

- **CdcStartTime**: When the DMS starts change data capture (CDC), in epoch time (milliseconds). For example, for CDC to start on Thursday August 9, 2018 1:02:49 AM (UTC), enter **1533776569**. Must not be a future time and not all source endpoints support CDC start time.
- **ReplicationTaskSettings**: A JSON document defining settings for the task. For example, task metadata settings, logging settings etc. For large inputs, we recommend removing extra whitespaces. For details, see Specifying Task Settings for AWS Database Migration Service Tasks.
- **ReplicationTaskIdentifier**: An identifier for the task. Use to give the task a name or label.

Creating a DMS Replication Task with the Console

Screenshot of this change type in the AMS console:


▼ Change type: Create DMS replication task.

Description
Use to create a Database Migration Service (DMS) replication task.

ID
ct-1d2fml15b9eth

Version
1.0

Execution mode
Automated

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Creating a DMS Replication Task with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution

parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotes when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws --profile saml --region us-east-1 amscm create-rfc --change-type-id
 "ct-1d2fml15b9eth" --change-type-version "1.0" --title "TestDMSRepTask" --
execution-parameters "{\"Description\":\"TestRepTask\",\"VpcId\":\"VPC-ID\",\"Name
\":\"DMSRepTask\",\"Parameters\":{\"CdcStartTime\":\1533776569\"MigrationType\":
\"full-load\",\"ReplicationInstanceArn\":\"REP_INSTANCE_ARN\",\"SourceEndpointArn\":
\"SOURCE_ENDPOINT_ARN\",\"TableMappings\":\"{\\\"rules\\\": [{\\\"rule-type\\\": \\
\"selection\\\",\\\"rule-id\\\": \\\"1\\\",\\\"rule-name\\\": \\\"1\\\",\\\"object-locator
\\\": {\\\"schema-name\\\": \\\"Test\\\",\\\"table-name\\\": \\\"%\\\"}, \\\"rule-action\\
\": \\\"include\\\"}] }\",\"TargetEndpointArn\":\"TARGET_ENDPOINT_ARN\"},\"StackTemplateId
\":\"stm-eos7uq0usnmeggdet\",\"TimeoutInMinutes\":60}"
```

TEMPLATE CREATE:

1. Output the execution parameters for this change type to a JSON file; this example names it CreateDmsRtParams.json:

```
aws amscm get-change-type-version --change-type-id "ct-1d2fml15b9eth" --query
 "ChangeTypeVersion.ExecutionInputSchema" --output text > CreateDmsRtParams.json
```

2. Modify and save the execution parameters JSON file. For example, you can replace the contents with something like this:

```
{
"Description":          "DMSTestRepTask",
"VpcId":                "VPC_ID",
"StackTemplateId":      "stm-eos7uq0usnmeggdet",
"Name":                 "Test DMS RT",
"TimeoutInMinutes":     60,
"Parameters":   {
    "CdcStartTime":             "1533776569",
    "MigrationType":            "full-load",
    "ReplicationInstanceArn":   "REP_INSTANCE_ARN",
    "SourceEndpointArn":        "SOURCE_ENDPOINT_ARN",
    "TargetEndpointArn":        "TARGET_ENDPOINT_ARN"
    "TableMappings":            {"rules": [{"rule-type": "selection","rule-id":
 "1","rule-name": "1","object-locator": {"schema-name": "Test","table-name": "%"},
 "rule-action": "include"}] }",
    }
}
```

3. Output the JSON template to a file in your current folder; this example names it CreateDmsRtRfc.json:

```
aws amscm create-rfc --generate-cli-skeleton > CreateDmsRtRfc.json
```

4. Modify and save the CreateDmsRtRfc.json file. For example, you can replace the contents with something like this:

```
{
"ChangeTypeVersion":    "1.0",
"ChangeTypeId":         "ct-1d2fml15b9eth",
"Title":                "DMS-RI-Create-RFC"
}
```

5. Create the RFC, specifying the execution parameters file and the CreateDmsRtRfc file:

```
aws amscm create-rfc --cli-input-json file://CreateDmsRtRfc.json --execution-parameters
 file://CreateDmsRtParams.json
```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

Start DMS replication task

You can start a DMS replication task, using the AMS console or the AMS API/CLI. For more information, see Working with AWS DMS Tasks.

Classification: Deployment | Advanced stack components | Database Migration Service (DMS) | Start replication task

Required Data, DMS Replication Task:

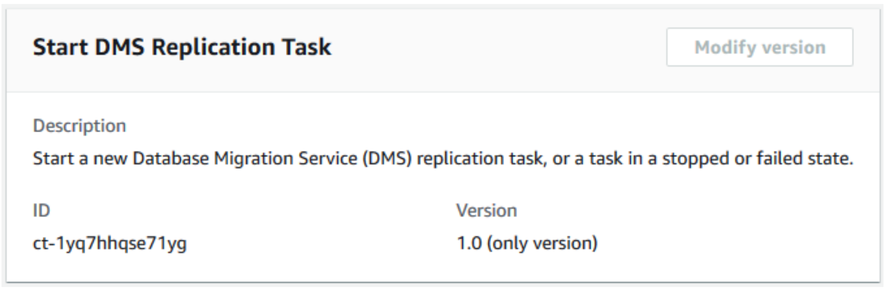
- **DocumentName**: Must be AWSManagedServices-StartDmsTask.
- **Region**: The AWS Region where the DMS replication task was created, in the form us-east-1.
- **Parameters**:
  - **ReplicationTaskArn**: The DMS replication task Amazon resource name (ARN). To find ARNs, look in the relevant Console.
  - **StartReplicationTaskType**: The type of DMS replication task. To start a new task, use start-replication. To restart a stopped task or failed task from the CDC position where the task stopped, use resume-processing. To restart a stopped or failed task of type full-load or full-load-and-cdc, use reload-target.

Optional Data, DMS Replication Task:

- **CdcStartPosition**: When to start the change data capture (CDC) operation. Use a timestamp in the format (yyyy-mm-ddThh:mm:ss), a log sequence number, or a checkpoint (either source database-engine specific, or AWS DMS-specific).
- **CdcStopPosition**: The timestamp in the format (server_time:yyyy-mm-ddThh:mm:ss) to stop the change data capture (CDC) operation.

Starting a DMS Replication Task with the Console

Screenshot of this change type in the AMS console:



Start DMS Replication Task

Description

Start a new Database Migration Service (DMS) replication task, or a task in a stopped or failed state.

ID: ct-1yq7hhqse71yg

Version: 1.0 (only version)

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Starting a DMS Replication Task with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id ID` command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id ID` command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotes when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws amscm create-rfc --change-type-id "ct-1yq7hhqse71yg" --change-type-version
 "1.0" --title "Start DMS Replication Task" --execution-parameters "{\"DocumentName
\":\"AWSManagedServices-StartDmsTask\",\"Region\":\"us-east-1\",\"Parameters\":
{\"ReplicationTaskArn\":[\"TASK_ARN\"],\"StartReplicationTaskType\":[\"start-
replication\"],\"CdcStartPosition\":[\"\"],\"CdcStopPosition\":[\"\"]}}"
```

TEMPLATE CREATE:

1. Output the execution parameters for this change type to a JSON file; this example names it StartDmsRtParams.json:

   ```
   aws amscm get-change-type-version --change-type-id "ct-1yq7hhqse71yg" --query
    "ChangeTypeVersion.ExecutionInputSchema" --output text > StartDmsRtParams.json
   ```

2. Modify and save the execution parameters JSON file. For example, you can replace the contents with something like this:

   ```
   {
     "DocumentName": "AWSManagedServices-StartDmsTask",
     "Region": "us-east-1",
     "Parameters": {
         "ReplicationTaskArn": [
           "TASK_ARN"
         ],
         "StartReplicationTaskType": [
           "start-replication"
         ],
         "CdcStartPosition": [
           ""
         ],
         "CdcStopPosition": [
           ""
         ]
     }
   }
   ```

3. Output the JSON template to a file in your current folder; this example names it StartDmsRtRfc.json:

   ```
   aws amscm create-rfc --generate-cli-skeleton > StartDmsRtRfc.json
   ```

4. Modify and save the StartDmsRtRfc.json file. For example, you can replace the contents with something like this:

   ```
   {
     "ChangeTypeId": "ct-1yq7hhqse71yg",
     "ChangeTypeVersion": "1.0",
     "Title": "Start DMS Replication Task"
   }
   ```

5. Create the RFC, specifying the execution parameters file and the StartDmsRtRfc file:

   ```
   aws amscm create-rfc --cli-input-json file://StartDmsRtRfc.json --execution-parameters
    file://StartDmsRtParams.json
   ```

   You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

Stop DMS replication task

You can stop a DMS replication task, using the AMS console or the AMS API/CLI. For more information, see Working with AWS DMS Tasks.

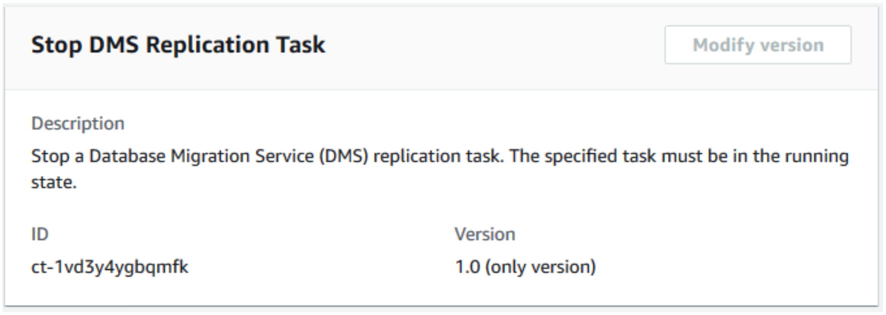
Classification: Deployment | Advanced stack components | Database Migration Service (DMS) | Stop replication task

Required Data:

- **DocumentName**: Must be AWSManagedServices-StartDmsTask.
- **Region**: The AWS Region where the DMS replication task was created, in the form us-east-1.
- **Parameters**:
  - **ReplicationTaskArn**: The DMS replication task Amazon resource name (ARN). To find ARNs, look in the relevant Console.

Stopping a DMS Replication Task with the Console

Screenshot of this change type in the AMS console:



Stop DMS Replication Task

Description

Stop a Database Migration Service (DMS) replication task. The specified task must be in the running state.

ID

ct-1vd3y4ygbqmfk

Version

1.0 (only version)

How it works:

1. Navigate to the **Choose change type** page: **RFCs** -> **Create RFC**.
2. Choose a change type from the drop-down lists. Optionally, open the **Additional configuration** area to select a change type version. After your selections are complete, a **Change type:** details area opens. Choose **Next**.
3. Configure the request for change. A **Subject** is required. Optionally, open the **Additional configuration** area to add information about the RFC. Choose **Next**.
4. Choose the execution parameters. At the top, in the **RFC configuration** area, enter values for the change type required parameters. These vary by change type. Open the **Additional configuration** area to add Tags or additional settings. Some change types also provide a **Parameters** area where only the required settings are visible. In that case, open the **Additional configuration** area to view optional parameters.
5. When finished, choose **Create**. If there are no errors, the **RFC successfully created** page displays with the submitted RFC details, and the initial **Execution output**.
6. Open the **Execution parameters** area to see the configurations you submitted. Refresh the page to update the RFC execution status. Optionally, cancel the RFC or create a copy of it with the options at the top of the page.

Stopping a DMS Replication Task with the CLI

How it works:

1. Use either the Inline Create (you issue a `create-rfc` command with all RFC and execution parameters included), or Template Create (you create two JSON files, one for the RFC parameters and one for the execution parameters) and issue the `create-rfc` command with the two files as input. Both methods are described here.
2. Submit the RFC: `aws amscm submit-rfc --rfc-id` *`ID`* command with the returned RFC ID.

   Monitor the RFC: `aws amscm get-rfc --rfc-id` *`ID`* command.

To check the change type version, use this command:

```
aws amscm list-change-type-version-summaries --filter Attribute=ChangeTypeId,Value=CT_ID
```

Note
You can use any `CreateRfc` parameters with any RFC whether or not they are part of the schema for the change type. For example, to get notifications when the RFC status changes, add this line, `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"` to the RFC parameters part of the request (not the execution parameters). For a list of all CreateRfc parameters, see the AMS Change Management API Reference.

INLINE CREATE:

Issue the create RFC command with execution parameters provided inline (escape quotes when providing execution parameters inline), and then submit the returned RFC ID. For example, you can replace the contents with something like this:

```
aws amscm create-rfc --change-type-id "ct-1vd3y4ygbqmfk" --change-type-version
 "1.0" --title "Stop DMS Replication Task" --execution-parameters "{\"DocumentName
\":\"AWSManagedServices-StopDmsTask\",\"Region\":\"us-east-1\",\"Parameters\":
{\"ReplicationTaskArn\":[\"TASK_ARN\"]}}"
```

TEMPLATE CREATE:

1. Output the execution parameters for this change type to a JSON file; this example names it StopDmsRtParams.json:

   ```
   aws amscm get-change-type-version --change-type-id "ct-1vd3y4ygbqmfk" --query
    "ChangeTypeVersion.ExecutionInputSchema" --output text > StopDmsRtParams.json
   ```

2. Modify and save the execution parameters JSON file. For example, you can replace the contents with something like this:

   ```
   {
     "DocumentName": "AWSManagedServices-StopDmsTask",
     "Region": "us-east-1",
     "Parameters": {
         "ReplicationTaskArn": [
           "TASK_ARN"
         ]
     }
   }
   ```

3. Output the JSON template to a file in your current folder; this example names it StopDmsRtRfc.json:

   ```
   aws amscm create-rfc --generate-cli-skeleton > StopDmsRtRfc.json
   ```

4. Modify and save the StopDmsRtRfc.json file. For example, you can replace the contents with something like this:

   ```
   {
     "ChangeTypeId": "ct-1vd3y4ygbqmfk",
     "ChangeTypeVersion": "1.0",
     "Title": "Stop DMS Replication Task"
   }
   ```

5. Create the RFC, specifying the execution parameters file and the StopDmsRtRfc file:

```
aws amscm create-rfc --cli-input-json file://StopDmsRtRfc.json --execution-parameters
 file://StopDmsRtParams.json
```

You receive the ID of the new RFC in the response and can use it to submit and monitor the RFC. Until you submit it, the RFC remains in the editing state and does not start.

Database (DB) Import to AMS SQL RDS

You can import your on-premises MS SQL database into a new database on your AMS-managed RDS SQL instance.

Note
The AMS API/CLI (amscm and amsskms) endpoints are in the AWS N. Virginia Region, `us-east-1`. Depending on how your authentication is set, and what AWS Region your account and resources are in, you may need to add `--region us-east-1` when issuing commands. You may also need to add `--profile saml`, if that is your authentication method.

This process relies on the AMS change types/RFCs and uses AWS RDS API parameters input. To learn more, see also: Amazon Relational Database Service (RDS) and rds.

Note
Make sure each RFC completes successfully before moving on to the next step.

High level import steps:

1. Back up your source on-premises MS SQL database into a .bak file
2. Copy the .bak file into the Transit S3 bucket
3. Import the .bak into a new DB on your target RDS MS SQL instance

Requirements:

- MS SQL RDS stack in AMS
- RDS stack with restore option (`SQLSERVER_BACKUP_RESTORE`)
- Transit S3 bucket
- IAM role with bucket access allowing Amazon Relational Database Service (RDS) to assume the role
- An EC2 instance with MS SQL Management Studio installed to manage the RDS (can be a workstation on-premises)

Setting Up

Complete these tasks to begin the import process.

1. Submit an RFC to create an RDS stack from AMS console using Deployment | Advanced stack components | RDS database stack | Create (ct-2z60dyvto9g6c). *Do not use the target DB name* (`RDSDBName` parameter) in the creation request, the target DB will be created during the import. Make sure to allow enough space (`RDSAllocatedStorage` parameter). For details on doing this, see the AMS User Guide Creating an RDS Database.
2. Submit an RFC to create the Transit S3 bucket (if does not exist already) from the AMS console using Deployment | Advanced stack components | S3 storage | Create (ct-1a68ck03fn98r). For details on doing this, see the AMS User Guide Creating an S3 Bucket Stack.
3. Submit a Management | Other | Other | Update (ct-1e1xtak34nx76) RFC to implement the `customer_rds_s3_role` with these details:

In the console:

- Subject: "To support MS SQL Server Database Import, implement `customer_rds_s3_role` on this account.
- Transit S3 bucket name: *`BUCKET_NAME`*.
- Contact information: *`EMAIL`*.

With an ImportDbParams.json file for the CLI:

```
{
       "Comment": "{"Transit S3 bucket name":"BUCKET_NAME"}",
       "Priority": "High"
     }
}
```

4. Submit a Management | Other | Other | Update RFC requesting AMS to set the `SQLSERVER_BACKUP_RESTORE` option to the RDS created in step 1 (use the stack ID from the step 1 output, and the `customer_rds_s3_role` IAM role in this request, in this request).
5. Submit an RFC to create an EC2 instance (you can use any existing EC2 or on-premise workstation/ instances), and install Microsoft SQL Management Studio on the instance.

Importing the Database

Follow these steps.

1. Back up your source on-premises database using MS SQL Native backup and restore. As the result, you should have a .bak file.
2. Upload the .bak file to the Transit S3 bucket using the AWS S3 CLI or AWS S3 console.
3. Import the .bak file into a new DB on your target RDS MS SQL instance.

   a. Log into the EC2 instance (on-premises workstation) and open MS SQL Management Studio
   b. Connect to the target RDS instance created as prerequisite in step #1. Follow this procedure to connect: Connecting to a DB Instance Running the Microsoft SQL Server Database Engine
   c. Start the import (restore) job via a new SQL query. Target database name must be new (do not use the same name as the database that you previously created). Example without encryption:

   ```
   exec msdb.dbo.rds_restore_database
           @restore_db_name=TARGET_DB_NAME,
           @s3_arn_to_restore_from=arn:aws:s3:::BUCKET_NAME/FILENAME.bak';
   ```

   d. Periodically check the status of the import job by running this query in a separate window:

   ```
   exec msdb.dbo.rds_task_status;
   ```

   If the status changes to Failed, look for the failure details in the message.

Cleanup

Once you have imported the database, you might want to remove unnecessary resources, follow these steps.

1. Delete the backup file (.bak) from the S3 bucket. You can use the S3 console to do this. For the CLI command to delete an object from an S3 bucket, see rm in the AWS CLI Command Reference.

2. Delete the S3 bucket if you're not planning to use it. For steps on doing that, see Delete Stack.
3. If you're not planning to do MS SQL imports, submit a Management | Other | Other | Update (ct-0xdawir96cy7k) RFC and request that AMS delete the IAM role `customer_rds_s3_role`.

AMS Tier and Tie App Deployments

A Tier and Tie deployment is where you create, configure, and deploy the resources of a stack independently using separate RFCs, and use the IDs of the stack components as you progress to associate them with each other. For example, if you were looking to deploy a "high availability" (redundant) website behind a load balancer, and a database, using a Tier and Tie approach, you would submit RFCs for a database, and a load balancer, and two EC2 instances or an Auto Scaling group, and configure the EC2 instances or Auto Scaling group with the ID of the ELB that you created. Once the resources were deployed, you could submit a security group create change to allow the resources to talk to the database. For details creating security groups, see Create Security Group.

AMS Full Stack App Deployments

A Full Stack deployment is where you submit an RFC with a CT that creates and configures everything you need at once. For example, to deploy the high availability website just described (EC2 instances, load balancer, and database) you would use a CT that, all together, created and configured an Auto Scaling group, a load balancer, a database, and the security group settings required for all instances to function as a stack. Examples of two AMS CTs that do this are described next.

- High Availability Two-Tier Stack (ct-06mjngx5flwto): This change type allows you to create a stack and configure an Auto Scaling Group, RDS-backed database, Load Balancer, and CodeDeploy application and configuration. Note that the load balancer isn't considered a tier as it is shared across multiple applications as a network appliance and the CodeDeploy functions are also considered an appliance. Additionally, it creates a CodeDeploy deployment group (with the name you give the CodeDeploy application) that can be used to deploy your applications. Security group settings to allow the resources to function together are automatically created.
- High Availability One-Tier Stack (ct-09t6q7j9v5hrn): This change type allows you to create a stack and configure an Auto Scaling Group, and an Application Load Balancer. Security group settings that allow the resources to function together are automatically created.

Working with Provisioning Change Types (CTs)

AMS has responsibility for your managed infrastructure, to make changes you must submit an RFC with the correct CT classification (category, subcategory, item, and operation). This section describes how to find CTs, determine if any are right for your needs, and request a new CT if none are.

See If an Existing CT Meets Your Requirements

Once you've determined what you want to deploy with AMS, the next step is to study the existing CTs and CloudFormation templates to see if a solution already exists.

When creating an RFC, you must specify the CT. You can use the Console or the AMS API/CLI. Examples of using both are described next.

You can use the console or the API/CLI to find a change type ID (CT) or version. There are two methods, either a search or choosing the classification. For both selection types, You can sort the search by choosing either **Most frequently used**, **Most recently used**, or **Alphabetical**.

YouTube Video: How do I create an RFC using the AWS Managed Services CLI and where can I find the CT Schema?

In the AMS console, on the **Create RFC** -> **Choose change type** page:

- Use **Browse by change type** selected (the default), and start typing a change type name. You don't have to have the exact or full change type name. As you type, the number of possible matches displays next to the **Change types** label.
- You can also search for a CT by change type ID, classification, or execution mode (automated or manual).
- Alternatively, and to explore change type choices, choose **Select by category** to open a series of drop-down option boxes.
- Choose **Category**, a **Subcategory**, an **Item**, and an **Operation**. The information box for that change type appears a panel appears at the bottom of the page.
- When you're ready, press **Enter**, and a list of matching change types appears.
- Choose a change type from the list. The information box for that change type appears at the bottom of the page.
- After you have the correct change type, choose **Next** to proceed to step two of the Create RFC Wizard.

Note
The AMS CLI must be installed for these commands to work. To install the AMS API or CLI, go to the AMS console **Developers Resources** page. For reference material on the AMS CM API or AMS SKMS API, see the AMS Information Resources section in the User Guide.

To search for a change type using the AMS CM API (see ListChangeTypeClassificationSummaries) or CLI:

You can use a filter or query to search. The ListChangeTypeClassificationSummaries operation has Filters options for `Category`, `Subcategory`, `Item`, and `Operation`, but the values must match the existing values exactly. For more flexible results when using the CLI, you can use the `--query` option.

Change type filtering with the API/CLI

Attribute	Valid values	Valid/Default condition	Notes
ChangeTypeId	Any string representing a ChangeTypeId (For ex: ct-abc123xyz7890)	Equals	For change type IDs, see Finding a Change Type or CSIO. For change type IDs, see Finding a Change Type or CSIO.
Category	Any free-form text	Contains	Regular expressions in each individual field are not supported. Case insensitive search
Subcategory			
Item			
Operation			

1. Here are some examples of listing change type classifications:

   The following command lists all change type categories.

   ```
   aws amscm list-change-type-categories
   ```

The following command lists the subcategories belonging to a specified category.

```
aws amscm list-change-type-subcategories --category CATEGORY
```

The following command lists the items belonging to a specified category and subcategory.

```
aws amscm list-change-type-items --category CATEGORY --subcategory SUBCATEGORY
```

2. Here are some examples of searching for change types with CLI queries:

   The following command searches CT classification summaries for those that contain "S3" in the Item name and creates output of the category, subcategory, item, operation, and change type ID in table form.

```
aws amscm list-change-type-classification-summaries --query
 "ChangeTypeClassificationSummaries [?contains(Item, 'S3')].
[Category,Subcategory,Item,Operation,ChangeTypeId]" --output table
```

```
+---------------------------------------------------------------+
|               ListChangeTypeClassificationSummaries           |
+----------+-------------------------+--+------+----------------+
|Deployment|Advanced Stack Components|S3|Create|ct-1a68ck03fn98r|
+----------+-------------------------+--+------+----------------+
```

3. You can then use the change type ID to get the CT schema and examine the parameters. The following command outputs the schema to a JSON file named CreateS3Params.schema.json.

```
aws amscm get-change-type-version --change-type-id "ct-1a68ck03fn98r" --query
 "ChangeTypeVersion.ExecutionInputSchema" --output text > CreateS3Params.schema.json
```

   For information about using CLI queries, see How to Filter the Output with the --query Option and the query language reference, JMESPath Specification.

4. After you have the change type ID, we recommend verifying the version for the change type to make sure it's the latest version. Use this command to find the version for a specified change type:

```
aws amscm list-change-type-version-summaries --filter
 Attribute=ChangeTypeId,Value=CHANGE_TYPE_ID
```

   To find the `AutomationStatus` for a specific change type, run this command:

```
aws amscm --profile saml get-change-type-version --change-type-id CHANGE_TYPE_ID --
query "ChangeTypeVersion.{AutomationStatus:AutomationStatus.Name}"
```

   To find the `ExpectedExecutionDurationInMinutes` for a specific change type, run this command:

```
aws amscm --profile saml get-change-type-version --change-type-id ct-14027q0sjyt1h --
query "ChangeTypeVersion.{ExpectedDuration:ExpectedExecutionDurationInMinutes}"
```

Once you have found a CT that you think is appropriate, look at the execution parameters JSON schema associated with it to learn if it addresses your use case.

Use this command to output a CT schema to a JSON file named after the CT; this example outputs the Create S3 storage schema:

```
aws amscm get-change-type-version --change-type-id "ct-1a68ck03fn98r"
--query "ChangeTypeVersion.ExecutionInputSchema" --output text >
CreateBucketParams.json
```

Let’s take a close look at what this schema offers.

S3 Bucket Create Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
"name": "Create S3 Storage
"description": "Use to create an Amazon Simple Storage
 Service stack.",
  "type": "object",
  "properties": {
    "Description": {
      "description": "The description of the stack.",
      "type": "string",
      "minLength": 1,
      "maxLength": 500
    },
    "VpcId": {
      "description": "ID of the VPC to create the S3 Bucket
 in, in the form vpc-a1b2c3d4e5f67890e.",
      "type": "string",
      "pattern": "^vpc-[a-z0-9]{17}$"
    },
    "StackTemplateId": {
      "description": "Required value: stm-
s2b72beb000000000.",
      "type": "string",
      "enum": ["stm-s2b72beb000000000"]
    },
    "Name":{
      "description": "The name of the stack to create.",
      "type": "string",
      "minLength": 1,
      "maxLength": 255
    },
    "Tags": {
      "description": "Up to seven tags (key/value pairs)
 for the stack.",
      "type": "array",
      "items": {
        "type": "object",
        "properties": {
          "Key": {
            "type": "string",
            "minLength": 1,
            "maxLength": 127
          },
          "Value": {
            "type": "string",
            "minLength": 1,
            "maxLength": 255
          }
        },
        "additionalProperties": false,
        "required": [
          "Key",
```

The schema begins with the CT (“description”), which tells you what the schema is for. In this case, to create an S3 storage stack.

Next, you have required and optional properties that you can specify. Default property values are given. The properties that are required are listed at the end of the schema.

In the StackTemplateId area, you see that there is one specific stack template for this CT and schema, and its ID is a required property value.

The schema allows you to tag the stack you are creating, for internal bookkeeping purposes. Additionally, some options, like backup, require a tag of Key:backup and Value:true. For in-depth information, read Tagging Your Amazon EC2 Resources.

```
          "Value"
        ]
      },
      "minItems": 1,
      "maxItems": 7
    },
    "TimeoutInMinutes": {
      "description": "The amount of time, in minutes, to
 allow for creation of the stack.",
      "type": "number",
      "minimum": 0,
      "maximum": 60
    },
    "Parameters": {
      "description": "Specifications for the stack.",
      "type": "object",
      "properties": {
        "AccessControl": {
          "description": "The canned (predefined) access
 control list (ACL) to assign to the bucket.",
          "type": "string",
          "enum": [
            "Private",
            "PublicRead",
            "AuthenticatedRead",
            "BucketOwnerRead"
          ]
        },
        "BucketName": {
          "description": "A name for the bucket. The bucket
 name must contain only lowercase letters, numbers, periods
 (.), and hyphens (-).",
          "type": "string",
          "pattern": "^[a-z0-9]([-.a-z0-9]+)[a-z0-9]$",
          "minLength": 3,
          "maxLength": 63
        }
      },
      "additionalProperties": false,
      "required": [
        "AccessControl",
        "BucketName"
      ]
    }
  },
  "additionalProperties": false,
  "required": [
    "Description",
    "VpcId",
    "StackTemplateId",
    "Name",
    "TimeoutInMinutes",
    "Parameters"
  ]
}
```

The Parameters section of the CT JSON schema is where you provide the execution parameters.

For this schema, only the ACL and BucketName are required execution parameters.

Request a New CT

After examining the schema, you may decide that it does not provide enough parameters to create the deployment that you want. If that is the case, examine existing CloudFormation templates to find one that is closer to what you want. Once you know what additional parameters you need, submit a Management | Other | Other | Create CT.

Note
All Other | Other Create and Update CTs receive the attention of an AMS operator, who will contact you to discuss the new CT.

To submit a request for a new CT, access the AMS console through the regular AWS Management Console and then follow these steps.

1. From the left navigation, click **RFCs**.

   The RFCs dashboard page opens.
2. Click **Create**.

   The Create a request for change page opens.
3. Select Management in the **Category** drop-down list, and Other for the **Subcategory** and **Item**. For the **Operation**, choose Create. The RFC will need approval before it can be implemented.
4. Enter information for why you want the CT, for example: Requesting a modified Create S3 storage CT that allows custom ACLs, based on the existing Create S3 storage CT. This should result in a new CT: Deployment | Advanced Stack Components | S3 storage | Create S3 custom ACL. This new CT could be public.
5. Click **Submit**.

   Your RFC displays on the RFC dashboard.

Test the New CT

Once AWS Managed Services has created that new CT, you test it by submitting an RFC with it. If you worked with AMS to make the new CT pre-approved, then you can simply follow a standard RFC submission, and watch for the result (for details on submitting RFCs, see Creating and Submitting an RFC). If the new CT is not pre-approved (you want to be sure that it is never run without explicit approval), then you will need to discuss its implementation with AMS each time you want to run it.

Application Maintenance

Once infrastructure is deployed, updating it in a consistent way across all your AMS environments, from QA to staging to production, is the challenge.

This section provides an overview of the AMS workload ingestion process and some examples of different methods you can use to keep your cloud infrastructure layer up to date.

Application Maintenance Strategies

How you deploy your applications impacts how you maintain them. This section provides some strategies for application maintenance.

Environment updates can involve any of these changes:

- Security updates
- New versions of your applications
- Application configuration changes
- Updates to dependencies

Note
For any application deployment, no matter the method, always file a service request beforehand to let AMS know that you are going to deploy an application.

Immutable vs Mutable Application Installation Examples

Compute Instance Mutability	App Install Method	AMI
Mutable	With CodeDeploy	AMS-provided
	Manually	
	With a Chef or Puppet, Pull-Based	
	With Ansible or Salt, Push-Based	
Immutable	With a Golden AMI	Custom (based on AMS-provided)

Mutable Deployment with a CodeDeploy-Enabled AMI

AWS CodeDeploy is a service that automates code deployments to any instance, including Amazon EC2 instances and instances running on-premises. You can use CodeDeploy with AMS to create and deploy a CodeDeploy application. Note that AMS provides a default instance profile for CodeDeploy applications.

- Amazon Linux (version 1)
- Amazon Linux 2
- RedHat 7
- CentOS 7

Before you use CodeDeploy for the first time, you must complete a number of setup steps:

1. Install or upgrade the AWS CLI
2. Create a Service Role for AWS CodeDeploy, you use the Service Role ARN in the deployment

IDs for all CT options can be found in the Change Types Overview.

> **Note**
> Currently, you must use Amazon S3 storage with this solution.

The basic steps are outlined here and the procedure is detailed in the AMS User Guide.

1. Create an Amazon S3 storage bucket. CT: ct-1a68ck03fn98r. The S3 bucket must have versioning enabled (for information on doing this, see Enabling Bucket Versioning).
2. Put your bundled CodeDeploy artifacts on it. You can do this with the Amazon S3 console without requesting access through AMS. Or using a variation of this command:

```
aws s3 cp ZIP_FILEPATH_AND_NAME s3://S3BUCKET_NAME/
```

3. Find an AMS `customer-` AMI; use either:
   - AMS Console: The VPC details page for the relevant VPC
   - AMS API ListAmis or CLI: `aws amsskms list-amis`
4. Create an Autoscaling group (ASG). CT: ct-2tylseo8rxfsc. Specify the AMS AMI, set the load balancer to have open ports, specify `customer-mc-ec2-instance-profile` for the `ASGIAMInstanceProfile`.
5. Create your CodeDeploy application. CT: ct-0ah3gwb9seqk2. Parameters include an application name; for example `WordpressProd`.
6. Create your CodeDeploy deployment group. CT: ct-2gd0u847qd9d2. Parameters include your CodeDeploy application name, ASG name, the configuration type name, and the service role ARN.
7. Deploy the CodeDeploy application. CT: ct-2edc3sd1sqmrb. Parameters include your CodeDeploy application name, configuration type name, deployment group name, revision type, and the S3 bucket location where the CodeDeploy artifacts are.

Mutable Deployment, Manually-Configured and Updated Application Instances

This application deployment strategy is a simple and manual update of application instances. These are the basic steps.

IDs for all CT options can be found in the Change Types Overview.

> **Note**
> Currently, you must use Amazon S3 storage with this solution.

The basic steps are outlined here; the various procedures are detailed in the AMS User Guide.

1. Create an Amazon S3 storage bucket. CT: ct-1a68ck03fn98r. The S3 bucket must have versioning enabled (for information on doing this, see Enabling Bucket Versioning).
2. Put your bundled application artifacts on it (everything your application needs to start on boot and work). You can do this with the Amazon S3 console without requesting access through AMS. Or using a variation of this command:

   ```
   aws s3 cp ZIP_FILEPATH_AND_NAME s3://S3BUCKET_NAME/
   ```
3. Find an AMS AMI, all will have CodeDeploy on them. To find a "customer-" AMI use either:
   - AMS Console: The VPC details page for the relevant VPC
   - AMS API ListAmis or CLI: `aws amsskms list-amis`
4. Create an EC2 instance with that AMI. CT: ct-14027q0sjyt1h. Specify the AMS AMI, set a tag `Key=backup, Value=true` and specify the `customer-mc-ec2-instance-profile` for the `InstanceProfile` parameter. Note the instance ID that is returned.
5. Request admin access to the instance. CT: ct-1dmlg9g1l91h6. You'll need the FQDN for your account. If you're unsure what your FQDN is, you can find it by:
   - Using the AWS Management Console for Directory Services (under Security and Identity) Directory Name tab.
   - Running one of these commands (return directory classes; DC+DC+DC=FQDN): Windows: `whoami /fqdn` or Linux: `hostname --fqdn`.
6. Log into the instance, see Accessing Instances via Bastions in the AMS User Guide.
7. Download your bundled application files from your S3 bucket to the instance.
8. Request an immediate backup with a service request to AMS, you will need to know the instance ID.
9. When you need to update your application, load new files to your S3 bucket and then follow steps 3 through 8.

Mutable Deployment with a Pull-Based Deployment Tool-Configured AMI

This strategy relies on the `InstanceUserData` parameter in the Managed Services Create EC2 CT. For more information on using this parameter, see Configuring Instances with User Data. This example assumes a pull-based application deployment tool like Chef or Puppet.

The CodeDeploy agent is supported on all AMS AMIs. Here is the list of supported AMIs:

- Amazon Linux (version 1)
- Amazon Linux 2
- RedHat 7
- CentOS 7

IDs for all CT options can be found in the Change Types Overview.

> **Note**
> Currently, you must use Amazon S3 storage with this solution.

The basic steps are outlined here and the procedure is detailed in the AMS User Guide.

1. Create an Amazon S3 storage bucket. CT: ct-1a68ck03fn98r. The S3 bucket must have versioning enabled (for information on doing this, see Enabling Bucket Versioning).

2. Put your bundled CodeDeploy artifacts on it. You can do this with the Amazon S3 console without requesting access through AMS. Or using a variation of this command:

```
aws s3 cp ZIP_FILEPATH_AND_NAME s3://S3BUCKET_NAME/
```

3. Find an AMS `customer-` AMI; use either:
   - AMS Console: The VPC details page for the relevant VPC
   - AMS API ListAmis or CLI: `aws amsskms list-amis`
4. Create an EC2 instance. CT: ct-14027q0sjyt1h; set a tag `Key=backup, Value=true`, and use the `InstanceUserData` parameter to specify a bootstrap and other scripts (download Chef/Puppet agent, etc.), and include the necessary authorization keys. You can find an example of doing this in the AMS User Guide, Change Mangement section examples of creating an HA Two Tier Deployment. Alternatively, request access to, and log into, the instance and configure it with the necessary deployment artifacts. Remember that pull-based deployment commands go from the agents on your instances to your corporate master server and may need authorization to go through bastions. You may need a service request to AMS to request security group/AD group access without bastions.
5. Repeat step 4 to create another EC2 instance and configure it with the deployment tool master server.
6. When you need to update your application, use the deployment tool to rollout the updates to your instances.

Mutable deployment with a Push-Based Deployment Tool-Configured AMI

This strategy relies on the `InstanceUserData` parameter in the Managed Services Create EC2 CT. For more information on using this parameter, see Configuring Instances with User Data. This example assumes a pull-based application deployment tool like Chef or Puppet.

IDs for all CT options can be found in the Change Types Overview.

Note
Currently, you must use Amazon S3 storage with this solution.

The basic steps are outlined here and the procedure is detailed in the AMS User Guide.

1. Create an Amazon S3 storage bucket. CT: ct-1a68ck03fn98r. The S3 bucket must have versioning enabled (for information on doing this, see Enabling Bucket Versioning).
2. Put your bundled CodeDeploy artifacts on it. You can do this with the Amazon S3 console without requesting access through AMS. Or using a variation of this command:

```
aws s3 cp ZIP_FILEPATH_AND_NAME s3://S3BUCKET_NAME/
```

3. Find an AMS AMI, all will have CodeDeploy on them. To find a "customer-" AMI use either:
   - AMS Console: The VPC details page for the relevant VPC
   - AMS API ListAmis or CLI: `aws amsskms list-amis`
4. Create an EC2 instance. CT: ct-14027q0sjyt1h; set a tag `Key=backup, Value=true`, and use the `InstanceUserData` parameter to run a bootstrap and other scripts including authorization keys, SALT stack (bootstrap a minion—for more information see Bootstrapping Salt on Linux EC2 with Cloud-Init) or Ansible (install a key pair—for more information see Getting Started with Ansible and Dynamic Amazon EC2 Inventory Management). Alternately, request access to, and log in to, the instance and configure it with the necessary deployment artifacts. Remember that push-based commands come from your corporate subnet to your instances and you may need to configure

authorization for them to go thru bastions. You may need a service request to AMS to request security group/AD group access without bastions.

5. Repeat step 4 to create another EC2 instance and configure it with the deployment tool master server.
6. When you need to update your application, use the deployment tool to rollout the updates to your instances.

Immutable Deployment with a Golden AMI

This strategy employs a “golden” AMI that you have configured to behave as you want all of your application instances to. For example, the instances created with this golden AMI would self-join the correct domain and DNS, self-configure, reboot and launch all necessary systems. When you want to update your application instances, you re-create the golden AMI and rollout all-new application instances with it.

The CodeDeploy agent is supported on all AMS AMIs. Here is the list of supported AMIs:

- Amazon Linux (version 1)
- Amazon Linux 2
- RedHat 7
- CentOS 7

IDs for all CT options can be found in the Change Types Overview.

Note
Currently, you must use Amazon S3 storage with this solution.

1. Create an Amazon S3 storage bucket. CT: ct-1a68ck03fn98r. The S3 bucket must have versioning enabled (for information on doing this, see Enabling Bucket Versioning).
2. Put your bundled application artifacts on it (everything your application needs to start on boot and work). You can do this with the Amazon S3 console without requesting access through AMS. Or using a variation of this command:

```
aws s3 cp ZIP_FILEPATH_AND_NAME s3://S3BUCKET_NAME/
```

3. Find an AMS `customer-` AMI; use either:
   - AMS Console: The VPC details page for the relevant VPC
   - AMS API ListAmis or CLI: `aws amsskms list-amis`
4. Create an EC2 instance with that AMI. CT: ct-14027q0sjyt1h. Specify the AMS AMI, set a tag `Key=backup, Value=true` and specify `customer-mc-ec2-instance-profile` for the `InstanceProfile`. Note the instance ID that is returned.
5. Request admin access to the instance. CT: ct-1dmlg9g1l91h6. You'll need the FQDN for your account. If you’re unsure what your FQDN is, you can find it by:
   - Using the AWS Management Console for Directory Services (under Security and Identity) Directory Name tab.
   - Running one of these commands (return directory classes; DC+DC+DC=FQDN): Windows: `whoami /fqdn` or Linux: `hostname --fqdn`.
6. Log into the instance, see Accessing Instances in the AMS User Guide.
7. Download to the instance your bundled application files from your S3 bucket. Configure the instance so that it self-deploys the fully-functioning application on boot.
8. Create the golden AMI on the instance. CT: ct-3rqqu43krekby. For details, see Create AMI.

9. Configure an Auto Scaling group to create new instances using that AMI. CT: ct-2tylseo8rxfsc. When you need to update your application, follow this procedure and request AMS to update the ASG to use the new golden AMI; use a Management | Other | Other | Update CT for this.

Update Strategies

There are a few different strategies you can employ to update your applications or instances in your AMS-managed environment.

- Scheduled Downtime: This simple strategy involves scheduling time for your application to be offline and manually updated. To do this, submit a Management | Other | Other | Update CT (ct-0xdawir96cy7k) request to stop the required instances. Make the necessary updates, and then submit another Management | Other | Other | Update CT (ct-0xdawir96cy7k) request to start the instances.

- Blue/Green: This strategy requires that you have a redundant environment (two completely functional environments) and take one environment offline using domain name system (DNS) or web firewall (WAF) updates to redirect traffic. Update one environment and then redirect again to update the other environment.

  To learn more, see AWS CodeDeploy Introduces Blue/Green Deployments.
- Rolling Update with new AMI: This is where you have a new AMI that you customize (see Create AMI) and then request that AMS deploy it to your Auto Scaling group. Use a Management | Other | Other | Update CT (ct-0xdawir96cy7k) to do this.

How the AMS Resource Scheduler works

Use AMS Resource Scheduler to schedule the automatic start and stop of AutoScaling groups, Amazon EC2 instances, and RDS instances in your account. This helps reduce infrastructure costs where the resources are not meant to be running 24/7. The solution is built on top of AWS Instance Scheduler but contains additional features and customizations specific to AMS needs.

AMS Resource Scheduler uses periods and schedules. Periods define the times the resource should run, such as start time, end time, and days of the month. Schedules contain your defined periods, along with additional configurations—SSM maintenance window, timezone, hibernate, etc—and specify when resources should run. You can configure these periods and schedules using AMS Resource Scheduler's automated change types (CTs).

AMS Resource Scheduler includes automated change types (CTs) that you use to deploy and configure it.

Resource Scheduler tips for multi-account landing zone

To deploy AMS Resource Scheduler, use the Management | Other | Other | Create change type (CT) to raise an RFC for AMS to deploy the solution in your account. Minimum details required in the RFC include Tag Name, Default TimeZone, Services to schedule. The current release of the automated CTs to deploy, enable, and disable AMS Resource Scheduler is for single-account landing zone accounts only; the other AMS Resource Scheduler CTs can be used for multi-account landing zone accounts.

To configure AMS Resource Scheduler; that is, create schedules or periods after the solution is deployed, use the automated AMS Resource Scheduler CTs to create, delete, update, and describe (get details on) required AMS Resource Scheduler schedules and periods. For details on schedule and period refer to AWS Instance Scheduler Solution Components

To select resources to be managed by AMS Resource Scheduler, following deployment and schedule creation, you use the AMS Tag Create CTs to tag AutoScaling groups, RDS stacks, and Amazon EC2 resources with that tag key you provided during deployment, and the defined schedule as the tag value. After the resources are tagged, the resources are scheduled for start or stop per your defined Resource Scheduler schedule.

AMS Resource Scheduler supports Amazon EC2 instances, RDS instances and clusters, and Auto Scaling groups.

There is no additional cost to using AMS Resource Scheduler. However the solution makes use of several AWS services and you're charged for these resources as they are used. For more details, see AWS Instance Scheduler Overview.

To opt out temporarily or completely of AMS Resource Scheduler, submit a Management | Other | Other | Update RFC requesting to disable or remove the solution from our release automation system.

Resource Scheduler tips for single-account landing zone

To deploy AMS Resource Scheduler, use the automated change type (CT) to raise an RFC that then deploys the solution in your account. As part of the request, we pre-provision a base CloudFormation stack containing only the required IAM roles for Resource Scheduler, and an empty CloudWatch log group. For more on Resource Scheduler change types, see AMS Resource Scheduler.

To configure AMS Resource Scheduler; that is, create schedules or periods, after the solution is deployed, use the automated Resource Scheduler CTs to create, delete, update, and describe (get details on) required AMS Resource Scheduler schedules and periods. See AMS Resource Scheduler.

To select resources to be managed by AMS Resource Scheduler, during deployment, choose a tag key. After you create a schedule, you use the AMS Tag Create CTs to tag AutoScaling groups, RDS stacks, and Amazon EC2 resources with that tag key, and the defined schedule as the tag value. After the resources are tagged, the resources are scheduled for start or stop per your defined Resource Scheduler schedule.

AMS Resource Scheduler supports Amazon EC2 instances, RDS instances and clusters, and Auto Scaling groups.

There is no additional cost to using AMS Resource Scheduler. However the solution makes use of several AWS services and you're charged for these resources as they are used. For more details, see AWS Instance Scheduler Overview.

To opt out of AMS Resource Scheduler, you can use the automated Resource Scheduler CTs to disable and enable the AMS Resource Scheduler on your account. However, if you want to completely opt out, you must submit a Management | Other | Other | Update RFC requesting to remove the solution from our release automation system.

AMS Resource Scheduler cost estimator

In order to track cost savings, AMS Resource Scheduler features a component that hourly calculates the estimated cost savings for Amazon EC2 and RDS resources that are managed by scheduler. This cost savings data is then published as a CloudWatch metric (`AMS/ResourceScheduler`) to help you track it. The cost savings estimator only estimates savings on instance running hours. It does not account any other cost, such as data transfer costs associated with a resource.

The cost savings estimator is enabled with Resource Scheduler. It runs hourly and retrieves cost and usage data from AWS Cost Explorer. From that data it calculates the average cost per hour for each instance type and then projects the cost for a full day if it was running without being scheduled. The cost savings is the difference between the projected cost and the actual reported cost from Cost Explorer for a given day.

For example, if instance A is configured with Resource Scheduler to run from 9 a.m. to 5 p.m., that is eight hours on a given day. Cost Explorer reports the cost as $1 and usage as 8. The average cost per hour is therefore $0.125. If the instance was not scheduled with Resource Scheduler, then the instance would run 24 hours on that day. In that case, the cost would have been 24x0.125 = $3. Resource Scheduler helped you achieve a cost savings of $2.

In order for the cost savings estimator to retrieve cost and usage only for resources managed by Resource Scheduler from Cost Explorer, the tag key that Resource Scheduler uses to target resources needs to be activated as the **Cost allocation** tag in Cost Explorer. For information on doing this, see Activating User-Defined Cost Allocation Tags and User-Defined Cost Allocation Tags

After the tag key is activated as Cost Allocation Tag, AWS billing starts tracking cost and usage for resources managed by Resource Scheduler, and after that data is available, the cost savings estimator starts to calculate the cost savings and publish the data under the `AMS/ResourceScheduler` metric namespace in CloudWatch.

> **Note**
> The Cost Savings Estimator is enabled where Resource Scheduler is deployed. For accounts with single-account landing zone we have released an update so Cost Savings Estimator is enabled where Resource Scheduler is deployed and enabled. In accounts where Resource Scheduler is not present, deploying the Resource Scheduler also deploys the feature.
> For accounts with multi-account landing zone submit a Management | Other | Other RFC to update the Resource Scheduler with the latest version in order to leverage Cost Savings Estimator.
> The cost savings estimator is available with latest v3.0 scheduler. Any prior version such as v1.x or v2.x, are not supported.

Cost estimator tips

Cost Savings Estimator does not accepts discounts such as reserved instances, savings plans, and so forth, into consideration in its calculation. The Estimator takes usage costs from Cost Explorer and calculates the average cost per hour for the resources. For more details, see Understanding your AWS Cost Datasets: A Cheat Sheet

In order for the cost savings estimator to retrieve cost and usage only for resources managed by Resource Scheduler from Cost Explorer, the tag key that Resource Scheduler uses to target resources needs to be activated as the Cost Allocation Tag in Cost Explorer. For information on doing this, see User-Defined Cost Allocation Tags. If the cost allocation tag is not activated, the estimator is not able to calculate the savings and publish the metric, even if it is enabled.

AMS Resource Scheduler best practices

Scheduling Amazon EC2 Instances

- Instance shut down behavior must be set to `stop` and not to `terminate`. This is pre-set to `stop` for instances that are created with the AMS Amazon EC2 Create automated change type (ct-14027q0sjyt1h) and can be set for Amazon EC2 instances created with AWS CloudFormation ingestion, by setting the `InstanceInitiatedShutdownBehavior` property to `stop`. If instances have shut down behavior set to `terminate`, then the instances will end when the Resource Scheduler stops them and the scheduler won't be able to start them back up.
- Amazon EC2 instances that are part of an Auto Scaling group aren't processed individually by AMS Resource Scheduler, even if they are tagged.
- If the target instance root volume is encrypted with a KMS customer master key (CMK) an additional `kms:CreateGrant` permission needs to be added to your Resource Scheduler IAM role, for the scheduler to be able to start or stop such instances. This permission is not added to the role by default for improved security. If you require this permission, submit an RFC with the Management | Advanced stack components | Identity and Access Management (IAM) | Update Entity or Policy change type.

Scheduling Auto Scaling groups

- AMS Resource Scheduler starts or stops the auto scaling of Auto Scaling groups, not individual instances in the group. That is, the scheduler restores the size of the Auto Scaling group (start) or sets the size to 0 (stop).
- Tag AutoScaling group with the specified tag and not the instances within the group.
- During stop, AMS Resource Scheduler stores the Auto Scaling group's Minimum, Desired, and Maximum capacity values and sets the Minimum and Desired Capacity to 0. During start, the scheduler restores the Auto Scaling group size as it was during the stop. Therefore, Auto Scaling group instances must use an appropriate capacity configuration so that the instances' termination and relaunch don't affect any application running in the Auto Scaling group.
- If the Auto Scaling group is modified (the minimum or maximum capacity) during a running period, the scheduler stores the new Auto Scaling group size and uses it when restoring the group at the end of a stop schedule.

Scheduling Amazon RDS instances

- The scheduler can take a snapshot before stopping the RDS instances (does not apply to Aurora DB cluster). This feature is turned on by default with the **Create RDS Instance Snapshot** AWS CloudFormation template parameter set to **true**. The snapshot is kept until the next time the Amazon RDS instance is stopped and a new snapshot is created.

  Scheduler can start/stop Amazon RDS instance that are part of a cluster or Amazon RDS Aurora database or in a multi availability zone (Multi-AZ) configuration. However, check Amazon RDS limitation when the scheduler won't be able to stop the Amazon RDS instance, especially Multi-AZ instances. To schedule Aurora Cluster for start or stop use the **Schedule Aurora Clusters** template parameter (default is **true**). The Aurora cluster (not the individual instances within the cluster) must be tagged with the tag key defined during initial configuration and the schedule name as the tag value to schedule that cluster.

  Every Amazon RDS instance has a weekly maintenance window during which any system changes are applied. During the maintenance window, Amazon RDS will automatically start instances that have been stopped for more than seven days to apply maintenance. Note that Amazon RDS will not stop the instance once the maintenance event is complete.

  The scheduler allows specifying whether to add the preferred maintenance window of an Amazon RDS instance as a running period to its schedule. The solution will start the instance at the beginning of the maintenance window and stop the instance at the end of the maintenance window if no other running period specifies that the instance should run, and if the maintenance event is completed.

  If the maintenance event is not completed by the end of the maintenance window, the instance will run until the scheduling interval after the maintenance event is completed.

  Note
  The Scheduler doesn't validate that a resource is started or stopped. It makes the API call and moves on. If the API call fails, it logs the error for investigation.

Application Security Considerations

Application security includes considering what permissions the application will need to run, what firewall rules, what IAM roles should be enabled for access to the application.

To better understand general AWS security, see Best Practices for Security, Identity, & Compliance.

Access for Configuration Management

AWS Managed Services (AMS) seeks to provide you with a headache-free infrastructure so you don't have to worry about security issues, patching issues, backup issues, etc. To do that, AMS recommends minimal IAM roles allowing only a specific group or a master server, if using an application deployment tool, access to the instances running your application.

Application Access Firewall Rules

Just like the operating system (OS), all application access should be governed using Active Directory (AD) groups. Using Amazon Relational Database Service (RDS) as an example, you must break the mirror (replication) to add a new user. The best approach is to create a group in AD and add it at database creation time. Having the groups in your AMS AD means that you can create CTs for application access. For information on the official grouping strategy for AD, see Using Group Nesting Strategy – AD Best Practices for Group Strategy.

To learn more about domain trees and parent/child domains, see How Domains and Forests Work.

The following rules illustrate a solution appropriate for a multi-domain forest trust with users located in child domains.

Windows Instances

These are the rules to configure for your Windows parent and child domain controllers.

Parent Domain Controller, Windows

FROM: Parent domain controllers TO: Windows stack and shared services subnets

Source Port	Destination Port	Protocol
88	49152 - 65535	TCP
389	49152 - 65535	UDP

FROM: Stack subnets, including shared services TO: Windows forest root domain controllers

Source Port	Destination Port	Protocol
49152 - 65535	88	TCP

Source Port	Destination Port	Protocol
49152 - 65535	389	UDP

Child Domain Controller, Windows

FROM: Child domain controllers TO: Windows AWS domain controllers

Source Port	Destination Port	Protocol
49152 - 65535	53	TCP
49152 - 65535	88	TCP
49152 - 65535	389	UDP

FROM: Child domain controllers TO: Windows stack and shared services subnets

Source Port	Destination Port	Protocol
88	49152 - 65535	TCP
135	49152 - 65535	TCP
389	49152 - 65535	TCP
389	49152 - 65535	UDP
445	49152 - 65535	TCP
49152 - 65535	49152 - 65535	TCP

FROM: Stack subnets, including shared services TO: Windows child domain controllers

Source Port	Destination Port	Protocol
49152 - 65535	88	TCP
49152 - 65535	135	TCP
49152 - 65535	389	TCP
49152 - 65535	389	UDP
49152 - 65535	445	TCP
49152 - 65535	49152 - 65535	TCP

Linux Instances

These are the rules to configure for your Linux parent and child domain controllers.

All testing was performed using Amazon Linux. While the dynamic port range for Windows is 49152 to 65535, many Linux kernels use the port range 32768 to 61000. Run the below command to view the IP port range.

```
cat /proc/sys/net/ipv4/ip_local_port_range
```

Parent Domain Controller, Linux

FROM: Parent domain controllers TO: Linux stack and shared services subnets

Source Port	Destination Port	Protocol
389	32768 - 61000	UDP
88	32768 - 61000	TCP

FROM: Stack subnets, including shared services TO: Linux forest root domain controllers

Source Port	Destination Port	Protocol
32768 - 61000	88	TCP
32768 - 61000	389	UDP

Child Domain Controller, Linux

FROM: Child domain controllers TO: Linux AWS domain controllers

Source Port	Destination Port	Protocol
49152 - 65535	53	TCP
49152 - 65535	88	TCP
389	49152 - 65535	UDP
49152 - 65535	389	UDP

FROM: Child domain controllers TO: Linux stack and shared services subnets

Source Port	Destination Port	Protocol
88	32768 - 61000	TCP
389	32768 - 61000	UDP

FROM: Stack subnets, including shared services TO: Linux child domain controller

Source Port	Destination Port	Protocol
32768 - 61000	88	TCP
32768 - 61000	389	UDP

AMS egress traffic management

By default, the route with a destination CIDR of 0.0.0.0/0 for AMS private and customer-applications subnets has a network address translation (NAT) gateway as the target. AMS services, TrendMicro and patching, are components that must have egress access to the Internet so that AMS is able to provide its service, and TrendMicro and operating systems can obtain updates.

AMS supports diverting the egress traffic to the internet through a customer-managed egress device as long as:

- It acts as an implicit (for example, transparent) proxy.

  and
- It allows AMS HTTP and HTTPS dependencies (listed in this section) in order to allow ongoing patching and maintenance of AMS managed infrastructure.

Some examples are:

- The transit gateway (TGW) has a default route pointing to the customer-managed, on-premises firewall over the AWS Direct Connect connection in the Multi-Account Landing Zone Networking account.
- The TGW has a default route pointing to an AWS endpoint in the Multi-Account Landing Zone egress VPC leveraging AWS PrivateLink, pointing to a customer-managed proxy in another AWS account.
- The TGW has a default route pointing to a customer-managed firewall in another AWS account, with site-to-site VPN connection as an attachment to the Multi-Account Landing Zone TGW.

AMS has identified the corresponding AMS HTTP and HTTPS dependencies, and develops and refines these dependencies on an ongoing basis. See Egress Management ZIP. Along with the JSON file, the ZIP contains a README.

Note

- This information isn't comprehensive--some required external sites aren't listed here.
- Do not use this list under a deny list or blocking strategy.
- This list is meant as a starting point for an egress filtering rule set, with the expectation that reporting tools will be used to determine precisely where the actual traffic diverges from the list.

To ask for information about filtering egress traffic, email your CSDM: ams-csdm@amazon.com.

Security groups

In AWS VPCs, AWS Security Groups act as virtual firewalls, controlling the traffic for one or more stacks (an instance or a set of instances). When a stack is launched, it's associated with one or more security groups, which determine what traffic is allowed to reach it:

- For stacks in your public subnets, the default security groups accept traffic from HTTP (80) and HTTPS (443) from all locations (the internet). The stacks also accept internal SSH and RDP traffic from your corporate network, and AWS bastions. Those stacks can then egress through any port to the Internet. They can also egress to your private subnets and other stacks in your public subnet.
- Stacks in your private subnets can egress to any other stack in your private subnet, and instances within a stack can fully communicate over any protocol with each other.

Important
The default security group for stacks on private subnets allows all stacks in your private subnet to communicate with other stacks in that private subnet. If you want to restrict communications between stacks within a private subnet, you must create new security groups that describe the

restriction. For example, if you want to restrict communications to a database server so that the stacks in that private subnet can only communicate from a specific application server over a specific port, request a special security group. How to do so is described in this section.

Default Security Groups

MALZ

The following table describes the default inbound security group (SG) settings for your stacks. The SG is named "SentinelDefaultSecurityGroupPrivateOnly-vpc-ID" where *ID* is a VPC ID in your AMS multi-account landing zone account. All traffic is allowed outbound to "mc-initial-garden-SentinelDefaultSecurityGroupPrivateOnly" via this security group (all local traffic within stack subnets is allowed).

All traffic is allowed outbound to 0.0.0.0/0 by a second security group "SentinelDefaultSecurityGroupPrivateOnly".

Tip
If you're choosing a security group for an AMS change type, such as EC2 create, or OpenSearch create domain, you would use one of the default security groups described here, or a security group that you created. You can find the list of security groups, per VPC, in either the AWS EC2 console or VPC console.

There are additional default security groups that are used for internal AMS purposes.

AMS default security groups (inbound traffic)

Type	Protocol	Port range	Source
All traffic	All	All	SentinelDefaultSecurityGroupPrivateOnly (restricts outbound traffic to members of the same security group)
All traffic	All	All	SentinelDefaultSecurityGroupPrivateOnlyEgressAll (does not restrict outbound traffic)
HTTP, HTTPS, SSH, RDP	TCP	80 / 443 (Source 0.0.0.0/0) SSH and RDP access is allowed from bastions	SentinelDefaultSecurityGroupPublic (does not restrict outbound traffic)
MALZ bastions:			
SSH	TCP	22	SharedServices VPC CIDR and DMZ VPC CIDR, plus Customer-provided on-prem CIDRs
SSH	TCP	22	
RDP	TCP	3389	
RDP	TCP	3389	
SALZ bastions:			
SSH	TCP	22	mc-initial-garden-LinuxBastionSG
SSH	TCP	22	mc-initial-garden-LinuxBastionDMZSG
RDP	TCP	3389	mc-initial-garden-WindowsBastionSG

Type	Protocol	Port range	Source
RDP	TCP	3389	mc-initial-garden-WindowsBastionDMZSG

SALZ

The following table describes the default inbound security group (SG) settings for your stacks. The SG is named "mc-initial-garden-SentinelDefaultSecurityGroupPrivateOnly-*ID*" where *ID* is a unique identifier. All traffic is allowed outbound to "mc-initial-garden-SentinelDefaultSecurityGroupPrivateOnly" via this security group (all local traffic within stack subnets is allowed).

All traffic is allowed outbound to 0.0.0.0/0 by a second security group "mc-initial-garden-SentinelDefaultSecurityGroupPrivateOnlyEgressAll-*ID*".

Tip
If you're choosing a security group for an AMS change type, such as EC2 create, or OpenSearch create domain, you would use one of the default security groups described here, or a security group that you created. You can find the list of security groups, per VPC, in either the AWS EC2 console or VPC console.

There are additional default security groups that are used for internal AMS purposes.

AMS default security groups (inbound traffic)

Type	Protocol	Port range	Source
All traffic	All	All	SentinelDefaultSecurityGroupPrivateOnly (restricts outbound traffic to members of the same security group)
All traffic	All	All	SentinelDefaultSecurityGroupPrivateOnlyEgressAll (does not restrict outbound traffic)
HTTP, HTTPS, SSH, RDP	TCP	80 / 443 (Source 0.0.0.0/0) SSH and RDP access is allowed from bastions	SentinelDefaultSecurityGroupPublic (does not restrict outbound traffic)
MALZ bastions:			
SSH	TCP	22	SharedServices VPC CIDR and DMZ VPC CIDR, plus Customer-provided on-prem CIDRs
SSH	TCP	22	
RDP	TCP	3389	
RDP	TCP	3389	
SALZ bastions:			
SSH	TCP	22	mc-initial-garden-LinuxBastionSG
SSH	TCP	22	mc-initial-garden-LinuxBastionDMZSG
RDP	TCP	3389	mc-initial-garden-WindowsBastionSG
RDP	TCP	3389	mc-initial-garden-WindowsBastionDMZSG

Create, Change, or Delete Security Groups

You can request custom security groups. In cases where the default security groups do not meet the needs of your applications or your organization, you can modify or create new security groups. Such a request would be considered approval-required and would be reviewed by the AMS operations team.

To create a security group outside of stacks and VPCs, submit an RFC using the `Management | Other | Other | Create` CT (ct-1e1xtak34nx76).

To add or remove a user from an Active Directory (AD) security group, submit a request for change (RFC) using the `Management | Other | Other | Update` CT (ct-0xdawir96cy7k).

Note
When using manual (approval required) CTs, AMS recommends that you use the ASAP option (choose **ASAP** in the console, leave start and end time blank in the API/CLI) as these CTs require an AMS operator to examine the RFC, and possibly communicate with you before it can be approved and run. If you schedule these RFCs, be sure to allow at least 24 hours. If approval does not happen before the scheduled start time, the RFC is rejected automatically.

Find Security Groups

To find the security groups attached to a stack or instance, use the EC2 console. After finding the stack or instance, you can see all security groups attached to it.

For ways to find security groups at the command line and filter the output, see `describe-security-groups`.

Appendix: Application Onboarding Questionnaire

Use this questionnaire to describe your deployment elements and structure so AMS can determine what infrastructure components are needed. The onboarding requirements for Line-of-Business applications are significantly different than Product applications, so this questionnaire is designed to address both.

Deployment Summary

A description of the deployment. For example:

- This account is for a Line-of-Business application deployment (as opposed to a Product application deployment).
- The deployment involves an auto-scaled ARP (authenticated reverse proxy) within the account's public/DMZ subnet.
- Web and application servers will be deployed within the account's private subnet.
- An RDS (AWS Relational Database Service) instance will also be deployed within the account's private Subnet.
- The servers (ARP, web, application, database, load balancer, etc.) are separated into distinct security groups.
- The account requires an HA (high availability) design spread across availability zones (AZs) i.e. "Multi-AZ".

Infrastructure Deployment Components

What are all the different components that will need configuring to support your application?

- Region: What AWS region or regions are needed?
- High Availability (HA): What availability zones will be used?
- Virtual Private Cloud (VPC): What is the CIDR block for the VPC?
- What server instances are needed?
  - Authenticated Reverse Proxy (ARP): OS, AMI, instance type, subnet ID, sec group, ingress port?
  - Application Deployment Tool server: OS, AMI, instance type, subnet ID, sec group, ingress port (Chef, Puppet) or egress port (Ansible, Saltstack) port?
  - AWS RDS with MySQL: DB version, Usage Type, instance class, subnet ID, security group, DB instance ID, storage size, Multi-AZ, Auth type, encryption?
  - Storage: Is your app stateless? Do you require S3 buckets? Do you require persistent storage? Do you require data at rest encryption on your EBS volumes? Do you require DB encryption?
  - External (to the Managed Services VPC) server endpoints: SMTP? LDAP?
  - Network requirements: Network filtering (based on sec groups?)? Web traffic inspection (inbound? outbound?)?
- Tagging: What tags should be used to group resources into logical collections? For example, all resources for an application stack. Select tags for your use case; for example, backup=true to enable

backups. Additionally, you must use the tag “name=value” in order for any EC2 instances you create to display a name in the console.

- Security groups:
  - What security groups are needed?
  - Security group ingress rules?
  - Security group egress rules?

Application Hosting Platform

For your application hosting platform, consider the following possible requirements:

- Databases encrypted?
- Encryption keys managed by whom?
- All data in-transit and at-rest encrypted?
- All user access to the system via HTTPS?
- All system-to-system interactions approved by your security Operations team?

Application Deployment Model

Considerations of how you plan your application deployments. See What is My Operating Model? (p. 8)

- Automated or manual? No deployment automation means no Auto Scale. If you request access and log in and manually update your application, and your update fails. AMS would expect you to rollback your update or alert us through a service request so we can assist you.
- If automated, what is the framework? Scripts? Agent-based (puppet/chef)? Agentless (SALT/Ansible)? CodeDeploy? Agent-based and agentless deployment tooling require a separate instance be created and deployed as the master server for the tooling. AMS expects you to be aware of all of the elements necessary for successful application deployment tooling; however, we are happy to help with related infrastructure questions.
- Do your Line-of-Business applications (those applications that you use to create and manage your applications) require patching?

Application Dependencies

Do you need instances for Line-of-Business (LoB) applications? For Product applications?

What do your Product applications need to function properly?

- Network level dependencies: For example, AWS DirectConnect
- Package dependencies: For example, pip
- Applications that this application depends on: For example, MySql
- Firewall dependencies?

What do your LoB applications need to function properly?

- Network level dependencies: For example, AWS DirectConnect

- Package dependencies: For example, firefox saucy
- Applications that this application depends on: For example, MySql
- Firewall dependencies?

SSL Certificates for Product Applications

What SSL certificates will your servers need so your applications (LoB and product) can reach everything they need to run and be accessible?

- Auto Scaling Group?
- Database (RDS)?
- Load Balancer?
- Deployment tool server?
- Web application firewall (WAF)?
- Other instances?

As an example, for each of the instances listed above you might need the following certificates:

WAF (cert 1) - > ELB-Ext (cert 2) - > ARP (cert 3) - > ELB-Int (cert 4) -> Website (cert 5) - > ELB-Int (cert 6) -> Web service (cert 7).

Document history

The following table describes the documentation for this release of AMS.

- **API version:** 2019-05-21
- **Latest documentation update:**September 30, 2021

Change	Description	Date
CFN ingest	Fixed example custom IAM policy name to match requirement that custom names start with `customer-`, previously the example used an underscore (_). CloudFormation Ingest Examples: 3-tier Web App (p. 59).	September 30, 2021
Database Migration Service	Deployment | Advanced stack components | Database Migration Service (DMS) | Create replication subnet group (ct-2q5azjd8p1ag5). Added note to warn that this CT will fail if the 'dms-vpc-role' IAM role doesn't exist in the account. Create DMS replication subnet group (p. 86).	September 16, 2021
CFN ingest	Update CFN ingest documentation for attribute value validation examples with reference to AMS reserved prefixes. Also added the list of AMS reserved prefixes to the Change Management guide and linked to it here. See CloudFormation ingest stack: CFN validator examples (p. 46).	August 12, 2021
WIGS Supported OSes	Windows 2008 R2 is not supported in workload Ingest. See Windows Prerequisites (p. 26).	July 29, 2021
CFN Ingest Supported Services	Updated the list of supported resource types in CFN Ingest. See Supported Resources (p. 49).	June 17, 2021
Moved the 'Security enhanced AMIs' section to the new private security guide, which is available on AWS Artifact.	To access AWS Artifact, you can contact your CSDM for instructions or go to Getting Started with AWS Artifact.	June 17, 2021
Updated Migrating Workloads: Prerequisites for Linux and Windows. Removed references to Windows 2008 R2.	Migrating Workloads: Prerequisites for Linux and Windows (p. 24).	April 15, 2021
Updated section: Workload Ingest, clarified what works for Single-Account Landing Zone.	See Migrating Workloads: CloudEndure Landing Zone (SALZ) (p. 28).	August 06, 2020
Updated section: Workload Ingest, clarified what works for Multi-Account Landing Zone.	See Tools account, Migrating Workloads: CloudEndure Landing Zone (MALZ) (p. 30).	August 06, 2020

Change	Description	Date
Updated Migrating Workloads: Prerequisites for Linux and Windows. AMS has added "perform a full virus scan" to the list of prerequisites for ingesting a copy of an on-premises instance into AMS.	Migrating Workloads: Prerequisites for Linux and Windows (p. 24).	February 11, 2021
New CT: "AWS CloudFormation stacks: Updating termination protection" allows you to update your existing termination protection configuration for an AWS CloudFormation stack.	Update AWS CloudFormation stacks termination protection (p. 71).	February 11, 2021
Updated section: The "AWS CloudFormation ingest stack: Updating" change type has two new parameters (AutoApproveRiskyUpdates and BypassDriftCheck). The parameter AutoApproveUpdateForResources was removed.	Update AWS CloudFormation ingest stack (p. 65).	January 14, 2021
Updated section: Python OS information was updated.	See How Migration Changes Your Resource (p. 26).	December 17, 2020
Updated sections: The FAQs appendix was added to the Service management chapter. Additionally, the AMS Service Description document was added to that same chapter and will no longer be a separate document.	See AMS service management (p. 9).	December 17, 2020
Updated section: Support for EC2 Launch Template in CFN Ingest and Stack Update CT.	See Supported Resources (p. 49).	November 12, 2020
Updated sections: The disk size for Windows and Linux pre-validation is updated.	See Migrating workloads: Windows pre-ingestion validation (p. 38) and Migrating workloads: Linux pre-ingestion validation (p. 38).	November 12, 2020
Updated section: RHEL 8 was added to the list of OSs.	See How Migration Changes Your Resource (p. 26).	October 15, 2020
Updated change type: A parameter, AutoApproveUpdateForResources, has been added to pre-approve update on resources in CFN stack based on logical IDs specified in the CFN template. The change type version remains v1.0.	See Update AWS CloudFormation ingest stack (p. 65).	October 15, 2020

Change	Description	Date
Additional self-provisioned services are available in CloudFormation ingest: AWS WAFv2	See Supported Resources (p. 49).	October 01, 2020
New section: CloudFormation Ingest validator examples.	See AMS CloudFormation ingest (p. 42).	September 03, 2020
Additional self-provisioned services are available in CloudFormation ingest: • Application Auto Scaling • CloudFormation • CloudWatch Logs • ECR • ECS (Fargate)	See Supported Resources (p. 49).	September 03, 2020
Updated section: Notes were made that IAM instance profiles must be prefixed with the word 'Customer'.	See AMS CloudFormation ingest (p. 42).	August 20, 2020
Additional self-provisioned services are available in CloudFormation ingest: • CodePipeline • Amazon DocumentDB	See Supported Resources (p. 49).	August 20, 2020
Updated section: Workload Ingest, added new, clarifying, information on using CloudEndure.	See Migrating Workloads: CloudEndure Landing Zone (SALZ) (p. 28).	August 06, 2020
Updated section: You can use contact tags or named lists to get notifications of AMS-generated incidents. Note, you must inform your CSDM of the tag key name or the named list.	Key terms (p. 4), Service Request terms.	July 23, 2020

Change	Description	Date
Additional self-provisioned services are available in CloudFormation ingest: • Certificate Manager • CloudWatch • Elasticache • EventBridge • Inspector • Kinesis Analytics • MediaConvert • MediaStore • SageMaker • Workspaces	See Supported Resources (p. 49).	July 23, 2020
Additional self-provisioned services are available in CloudFormation ingest: • CodeBuild • FSx • Amazon Kinesis • SES • WAF	See Supported Resources (p. 49).	July 09, 2020
Updated CFN Ingest sections	AWS Managed Services (AMS) has enabled 17 AMS Self Provisioned Services for use in the CloudFormation Stack Create (ct-36cn2avfrrj9v, version 2.0) and the CloudFormation Stack Update (ct-361tlo1k7339x, version 1.0) change types. See Create CloudFormation ingest stack (p. 61), Update AWS CloudFormation ingest stack (p. 65), and Supported Resources (p. 49).	June 18, 2020
Updated section	Updates for SUSE 15 SP1 SAP WIGS Support: Migrating Workloads: Prerequisites for Linux and Windows (p. 24).	June 04, 2020
New and/or Updated CT Walkthroughs:	Updated CTs with extended timeout: Approve a CloudFormation ingest stack changeset (p. 69) and Update AWS CloudFormation ingest stack (p. 65).	April 23, 2020
A note about a failed WIGS RFC was modified with new information about WIGS improved error messages.	See Workload Ingest Stack: Creating (p. 39).	March 19, 2020

Change	Description	Date
New walkthrough note about a change in how default security groups are applied, for Deployment | Ingestion | Stack from CloudFormation Template | Create (ct-36cn2avfrrj9v).	See Create CloudFormation ingest stack (p. 61).	March 05, 2020
New Database Migration System change type walkthroughs.	See Start DMS replication task (p. 111) and Stop DMS replication task (p. 113).	February 20, 2020
Notes and updates were made to the Workload Ingest walkthroughs.	See How Migration Changes Your Resource (p. 26) and Workload Ingest Stack: Creating (p. 39).	February 20, 2020

AWS glossary

For the latest AWS terminology, see the AWS glossary in the *AWS General Reference*.