AMS アドバンストアプリケーションデプロイオプション

AMS Advanced Application デベロッパーガイド

Version September 13, 2024

AMS Advanced Application デベロッパーガイド: AMS アドバンストアプリケーションデプロイオプション

Table of Contents

アプリケーションのオンボーディング

AWS Managed Services (AMS) AMS オペレーションプランへようこそ。このドキュメントの目的は、最初のネットワークとアクセス管理が設定されたらアプリケーションを AMS にオンボーディングするときに使用できるさまざまな方法と、それらの方法を選択するときに考慮すべき問題を説明することです。

このドキュメントは、新しい AMS のお客様向けのアプリケーションプロセスの決定と作成を支援するシステムインテグレーターとアプリケーション開発者を対象としています。

アプリケーションオンボーディングとは

AMS アプリケーションのオンボーディングとは、必要に応じてリソースとアプリケーションを AMS インフラストラクチャにデプロイすることです。AMS プラットフォームでのアプリケーションとインフラストラクチャの設計は、ネイティブでのアーキテクチャと非常によく似ています AWS。AMS が提供する機能を検討しながら、アプリケーションとインフラストラクチャの設計のベストプラクティスに従うこと AWS で、AMS 環境でホストされる有能で運用可能なアプリケーションが生成されます。

> **Note**
>
> - 米国東部 (バージニア)
> - 米国西部 (北カリフォルニア)
> - 米国西部 (オレゴン)
> - 米国東部 (オハイオ)
> - カナダ (中部)
> - 南米 (サンパウロ)
> - 欧州 (アイルランド)
> - 欧州 (フランクフルト)
> - 欧州 (ロンドン)
> - 欧州西部 (パリ)
> - アジアパシフィック (ムンバイ)
> - アジアパシフィック (ソウル)
> - アジアパシフィック (シンガポール)

- アジアパシフィック (シドニー)
- アジアパシフィック (東京)

新しいリージョンは頻繁に追加されます。詳細については、AWS リージョン 「」および「アベイラビリティーゾーン」を参照してください。

何をするか、何をしないか

AMS は、AWS インフラストラクチャをデプロイするための標準化されたアプローチを提供し、必要な継続的な運用管理を提供します。ロール、責任、サポートされているサービスの詳細については、「サービスの説明」を参照してください。

> **Note**
>
> AMS が追加の AWS サービスを提供するようにリクエストするには、サービスリクエストを提出します。詳細については、「サービスリクエストの作成」を参照してください。

- 当社の業務：

  オンボーディングが完了すると、AMS 環境は変更リクエスト (RFCs)、インシデント、サービスリクエストを受け取ることができます。AMS サービスとのやり取りは、アプリケーションスタックのライフサイクルを中心に展開されます。新しいスタックは、事前設定されたテンプレートのリストから順序付けられ、特定の Virtual Private Cloud (VPC) サブネットに起動され、変更リクエスト (RFCs) を通じて運用期間中に変更され、イベントとインシデントが 24 時間 365 日モニタリングされます。

  アクティブなアプリケーションスタックは、パッチ適用を含む AMS によってモニタリングおよび保守され、変更が必要な場合やスタックが廃止されない限り、スタックの存続期間中はそれ以上のアクションは必要ありません。スタックのヘルスと機能に影響する AMS によって検出されたインシデントは、通知を生成し、解決または検証するためのアクションが必要になる場合と必要でない場合があります。ハウツーに関する質問やその他の問い合わせは、サービスリクエストを送信することで行うことができます。

さらに、AMS では、AMS によって管理されていない互換性のある AWS サービスを有効にすることができます。AWS-AMS 互換サービスの詳細については、「セルフサービスプロビジョニングモード」を参照してください。

- 行わないこと：

  AMS は、手動オプションと自動オプションを多数提供することでアプリケーションのデプロイを簡素化しますが、アプリケーションの開発、テスト、更新、管理はお客様の責任となります。AMS は、アプリケーションに影響を与えるインフラストラクチャの問題のトラブルシューティングを支援しますが、AMS はアプリケーション設定にアクセスしたり検証したりすることはできません。

AMS Amazon マシンイメージ (AMIs)

AMS は、AMS がサポートするオペレーティングシステム用に毎月更新された Amazon マシンイメージ (AMIs) を生成します。さらに、AMS は、AMS でサポートされているオペレーティングシステムのサブセットの CIS レベル 1 ベンチマークに基づいてセキュリティ拡張イメージ (AMIs) も生成します。 https://docs.aws.amazon.com/managedservices/latest/userguide/supported-configs.htmlセキュリティ拡張イメージが利用可能なオペレーティングシステムについては、AMS セキュリティユーザーガイドを参照してください。このユーザーガイドは、AWS AWS Managed Services用にフィルタリングされた AWS Artifact -> レポートページ (左側のナビゲーションペインのレポートオプションを参照) から入手できます。AWS Artifact にアクセスするには、 から CSDM に連絡して手順を確認するか、「AWS Artifact の開始方法」を参照してください。

新しい AMS AMIs「AMS AMI」という Amazon Simple Notification Service (Amazon SNS) 通知トピックをサブスクライブできます。詳細については、「SNS を使用した AMS AMI 通知」を参照してください。

AMS AMI の命名規則は です (例: `customer-ams-rhel6-2018.11-3`) `customer-ams-<operating system>-<release date> - <version>`。

で始まる AMS AMIs のみを使用してください`customer`。

AMS では、常に最新の AMI を使用することをお勧めします。最新の AMIs は、次のいずれかで確認できます。

- AMS コンソールの AMIs ページを参照してください。

- CSDM から、またはこの ZIP ファイルを介して利用可能な最新の AMS AMI CSV ファイルを表示する: AMS 11.2024 AMI の内容と CSV ファイルを ZIP で表示します。

  過去の AMI ZIP ファイルについては、「Doc History」を参照してください。
- この AMS SKMS コマンドの実行 (AMS SKMS SDK が必要):

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

オペレーティングシステム (OS) によって AWS AMIsに追加された AMS AMI コンテンツ

- Linux AMIs:
  - AWS CLI ツール
  - NTP
  - Trend Micro Endpoint Protection Service エージェント
  - コードデプロイ
  - PBIS / Beyond Trust AD Bridge
  - SSM Agent
  - 重要なパッチの Yum アップグレード
  - AMS カスタムスクリプト/管理ソフトウェア (起動、AD 結合、モニタリング、セキュリティ、ログ記録の制御)
- Windows Server AMIs:
  - Microsoft .NET Framework 4.5
  - PowerShell 5.1
  - AWS Tools for Windows PowerShell
  - 起動、AD 結合、モニタリング、セキュリティ、ログ記録を制御する AMS PowerShell モジュール
  - Trend Micro Endpoint Protection Service エージェント
  - SSM Agent
  - CloudWatch エージェント
  - EC2Config サービス (Windows Server 2012 R2 経由)
  - EC2Launch (Windows Server 2016 および Windows Server 2019)
  - EC2LaunchV2 (Windows Server 2022 以降)

Linux ベースの AMIs :

- Amazon Linux 2023 (最新のマイナーリリース) (最小 AMI はサポートされていません)
- Amazon Linux 2 (最新のマイナーリリース)
- Amazon Linux 2 (ARM64)
- Red Hat Enterprise 7 (最新のマイナーリリース)
- Red Hat Enterprise 8 (最新のマイナーリリース)
- Red Hat Enterprise 9 (最新のマイナーリリース)
- SUSE Linux Enterprise Server 15 SP6
- Ubuntu Linux 18.04
- Ubuntu Linux 20.04
- Ubuntu Linux 22.04
- Ubuntu Linux 24.04
- Amazon Linux: 製品概要、料金情報、使用状況情報、およびサポート情報については、「Amazon Linux AMI (HVM / 64 ビット)」および「Amazon Linux 2」を参照してください。

  詳細については、「Amazon Linux 2 に関するよくある質問」を参照してください。
- RedHat Enterprise Linux (RHEL): 製品概要、料金情報、使用状況情報、サポート情報については、「Red Hat Enterprise Linux (RHEL) 7 (HVM)」を参照してください。
- Ubuntu Linux 18.04: 製品概要、料金情報、使用状況情報、サポート情報については、「Ubuntu 18.04 LTS - Bionic」を参照してください。
- SUSE Linux Enterprise Server for SAP アプリケーション 15 SP6:
  - アカウントごとに次の手順を 1 回実行します。
    1. AWS Marketplace に移動します。
    2. SUSE 15 SAP 製品を検索します。
    3. [サブスクリプションを続行する] を選択します。
    4. 次に [規約の受諾] を選択します。
  - SAP Applications 15 SP6 インスタンス用の新しい SUSE Linux Enterprise Server を起動する必要があるたびに、次の手順を実行します。
    1. サブスクライブしている SUSE Linux Enterprise Server for SAP Applications 15 AMI の AMI ID を書き留めます。

2. デプロイの作成 | 高度なスタックコンポーネント | EC2 スタック | 変更タイプの作成 ct-14027q0sjyt1h RFC。*InstanceAmiId* をサブスクライブした AWS Marketplace AMI ID に置き換えます。

Windows ベースの AMIs：

最新の Windows AMIs に基づく Microsoft Windows Server (2016、2019、2022)。

AMIs「AMI の作成」を参照してください。

AMS AMIsオフボーディング：

AMS は、非依存関係への影響を避けるため、オフボーディング中に AMIs の共有を解除しません。アカウントから AMS AMIs を削除する場合は、 `cancel-image-launch-permission` API を使用して特定の AMIs を非表示にできます。たとえば、以下のスクリプトを使用して、以前にアカウントと共有されたすべての AMS AMIs を非表示にできます。

```
for ami in $(aws ec2 describe-images --executable-users self --owners 027415890775 --
query 'Images[].ImageId' --output text) ;
    do
    aws ec2 cancel-image-launch-permission --image-id $ami ;
    done
```

スクリプトをエラーなしで実行するには、AWS CLI v2 がインストールされている必要があります。AWS CLI のインストール手順については、「AWS CLI の最新バージョンのインストールまたは更新」を参照してください。`cancel-image-launch-permission` コマンドの詳細については、「」を参照してください`cancel-image-launch-permission`。

セキュリティ強化 AMIs

AMS は、AMS でサポートされているオペレーティングシステムのサブセットの CIS レベル 1 ベンチマークに基づくセキュリティ拡張イメージ (AMIs) を提供します。セキュリティ拡張イメージが利用可能なオペレーティングシステムを確認するには、AWS Managed Services (AMS) カスタマーセキュリティガイドを参照してください。このガイドにアクセスするには、 を開き AWS Artifact、左側のナビゲーションペインでレポートを選択し、AWS Managed Services をフィルタリングします。アクセス方法については AWS Artifact、CSDM に問い合わせるか、「 の開始方法 AWS Artifact」を参照してください。

AMS の主要な用語

- AMS Advanced: AMS Advanced ドキュメントの「サービスの説明」セクションで説明されているサービス。「サービスの説明」を参照してください。
- AMS アドバンストアカウント: AMS アドバンストオンボーディング要件のすべての要件を満たしている AWS アカウント。AMS Advanced の利点、導入事例、販売担当者へのお問い合わせについては、AWS Managed Services」を参照してください。
- AMS Accelerate Accounts:AMS Accelerate オンボーディング要件のすべての要件を満たしている AWS アカウント。「AMS Accelerate の開始方法」を参照してください。
- AWS Managed Services: AMS および/または AMS Accelerate。
- AWS Managed Servicesアカウント: AMS アカウントまたは AMS Accelerate アカウント。
- 重要な推奨事項: リソースまたは に対する潜在的なリスクや中断から保護するためにアクションが必要であることを知らせるサービスリクエスト AWS を通じて によって発行された推奨事項 AWS のサービス。指定された日付までにクリティカルレコメンデーションに従わないことを決定した場合、その決定に起因する損害については、お客様が単独で責任を負います。
- お客様がリクエストした設定: 以下で特定されていないソフトウェア、サービス、またはその他の設定。
  - Accelerate: サポートされている設定または AMS Accelerate、サービスの説明。
  - AMS Advanced: サポートされている設定または AMS Advanced、サービスの説明。
- インシデントコミュニケーション: AMS は、AMS Accelerate 用サポートセンターおよび AMS 用 AMS コンソールで作成されたインシデントを介して、インシデントをユーザーに伝達するか、AMS でインシデントをリクエストします。AMS Accelerate コンソールには、ダッシュボードのインシデントとサービスリクエストの概要と、サポートセンターへのリンクが表示されます。
- マネージド環境: AMS Advanced アカウントまたは AMS Accelerate アカウント。

  AMS Advanced の場合、マルチアカウントランディングゾーン (MALZ) アカウントとシングルアカウントランディングゾーン (SALZ) アカウントが含まれます。
- 請求開始日: が AWS Managed Services オンボーディング E メールでリクエストされた情報 AWS を受信する翌営業日。AWS マネージドサービスのオンボーディング E メールは、アカウントで AWS Managed Servicesアクティブ化するために必要な情報を収集するために から AWS に送信される E メールを指します。

  後で登録されたアカウントの場合、請求開始日は、AWS Managed Services が登録済みアカウントの AWS Managed Services アクティベーション通知を送信した翌日です。AWS Managed Servicesアクティベーション通知は、次の場合に発生します。

1. 互換性のある AWS アカウントへのアクセスを許可し、AWS Managed Services に引き渡します。
2. AWS Managed Services は、AWS Managed Services アカウントを設計および構築します。

- サービスの終了: すべての AWS Managed Servicesアカウント、または指定された AWS Managed Servicesサービスアカウントの AWS Managed Servicesサービスは、サービスリクエストを通じて AWS 少なくとも 30 日前に通知することで、何らかの理由で終了できます。サービス終了日に、次のいずれかを実行します。
  1. AWS 必要に応じて、すべての AWS Managed Servicesアカウントまたは指定された AWS Managed Servicesアカウントのコントロールをユーザーに引き渡します。
  2. 関係者は、必要に応じて、すべての AWS Managed Servicesアカウントまたは指定された AWS Managed Servicesアカウントから AWS アクセスを許可する AWS Identity and Access Management ロールを削除します。
- サービス終了日: サービス終了日は、30 日間の必要な終了通知期間の終了後の暦月の最終日です。必要な終了通知期間の終了日が暦月の 20 日以降である場合、サービス終了日は翌月の最終日になります。終了日のシナリオの例を次に示します。
  - 終了通知が 4 月 12 日に提供された場合、30 日間の通知は 5 月 12 日に終了します。サービス終了日は 5 月 31 日です。
  - 終了通知が 4 月 29 日に提供された場合、30 日間の通知は 5 月 29 日に終了します。サービス終了日は 6 月 30 日です。
- AWS Managed Services:makes のプロビジョニングを利用でき、サービス開始日から AWS Managed Services アカウントごとに AWS Managed Services にアクセスして使用できます。AWS
- 指定された AWS Managed Servicesアカウントの終了: サービスリクエスト (AWS Managed Services 「AMS アカウント終了リクエスト」) を通じて AWS 通知を提供することで、何らかの理由で指定された AWS AWS Managed Services マネージドサービスを終了できます。

インシデント管理の条件：

- イベント: AMS 環境の変更。
- アラート: サポートされている からのイベントがしきい値 AWS のサービス を超え、アラームをトリガーするたびに、アラートが作成され、連絡先リストに通知が送信されます。さらに、インシデントリストにインシデントが作成されます。
- インシデント: AMS 環境または AWS Managed Services予期しない中断またはパフォーマンスの低下。AWS Managed ServicesまたはユーザーによってAWS 報告された影響をもたらします。

- 問題: 1 つ以上のインシデントの根本的な根本原因の共有。
- インシデントの解決またはインシデントの解決：
  - AMS が、そのインシデントに関連するすべての利用できない AMS サービスまたはリソースを利用可能な状態に復元した場合、または
  - AMS が、使用できないスタックまたはリソースを使用可能な状態に復元できないと判断した場合、または
  - AMS は、ユーザーによって承認されたインフラストラクチャの復元を開始しました。
- インシデント対応時間: インシデントを作成するときと、AMS がコンソール、E メール、データセンター、または電話を使用して初期応答を提供するときの時間の差。
- インシデント解決時間: AMS またはユーザーがインシデントを作成してからインシデントが解決されるまでの時間の差。
- Incident Priority: 低、中、高のいずれかで、AMS またはユーザーによってインシデントが優先される方法。
  - 低: AMS サービスの重大でない問題。
  - 中: マネージド環境内の AWS サービスを利用できますが、 (該当するサービスの説明に従って) 意図したとおりに動作しません。
  - 高: (1) AMS コンソール、またはマネージド環境内の 1 つ以上の AMS APIs が使用できない、または (2) マネージド環境内の 1 つ以上の AMS スタックまたはリソースが使用できないため、アプリケーションが関数を実行できない。

  AMS は、上記のガイドラインに従ってインシデントを再分類することがあります。
- インフラストラクチャの復元: 影響を受けたスタックのテンプレートに基づいて既存のスタックを再デプロイし、インシデント解決が不可能な場合に、ユーザーが特に指定しない限り、最後の既知の復元ポイントに基づいてデータ復元を開始します。

インフラストラクチャ用語：

- マネージド型本番稼働環境: 顧客の本番稼働用アプリケーションが存在するユーザーアカウント。
- マネージド非本番環境: 開発およびテスト用のアプリケーションなど、非本番アプリケーションのみを含むユーザーアカウント。
- AMS スタック: AMS によって単一のユニットとして管理される 1 つ以上の AWS リソースのグループ。
- イミュータブルインフラストラクチャ: Amazon EC2 Auto Scaling グループ (ASGs) に一般的なインフラストラクチャメンテナンスモデル。更新されたインフラストラクチャコンポーネント

(AMI 内) は AWS、インプレースで更新されるのではなく、デプロイごとに置き換えられます。イミュータブルインフラストラクチャの利点は、すべてのコンポーネントが常に同じベースから生成されるため、同期状態のままであることです。イミュータビリティは、AMI を構築するためのツールやワークフローとは無関係です。

- ミュータブルインフラストラクチャ: Amazon EC2 Auto Scaling グループではなく、単一のインスタンスまたは少数のインスタンスのみを含むスタックに一般的なインフラストラクチャメンテナンスモデル。このモデルは、ライフサイクルの開始時にシステムがデプロイされ、時間の経過とともに更新がシステムにレイヤー化される、従来のハードウェアベースのシステムデプロイに最も近いものです。システムの更新はインスタンスに個別に適用され、アプリケーションまたはシステムの再起動により (スタック設定に応じて) システムのダウンタイムが発生する可能性があります。
- セキュリティグループ: インバウンドトラフィックとアウトバウンドトラフィックを制御するためのインスタンスの仮想ファイアウォール。セキュリティグループは、サブネットレベルでなくインスタンスレベルで動作します。したがって、VPC 内のサブネット内の各インスタンスには、異なるセキュリティグループのセットを割り当てることができます。
- サービスレベルアグリーメント (SLAs): 期待されるサービスのレベルを定義する AMS 契約の一部。
- SLA が使用不可および使用不可：
  - ユーザーが送信した API リクエストがエラーになります。
  - 5xx HTTP レスポンスになるコンソールリクエスト (サーバーはリクエストを実行できません)。
  - AMS マネージドインフラストラクチャのスタックまたはリソースを構成する AWS のサービスサービスはすべて、Service Health Dashboard に示すように「サービス中断」の状態です。
  - AMS 除外から直接または間接的に生じる利用不可は、サービスクレジットの適格性を判断する際に考慮されません。サービスは、利用不可の基準を満たしていない限り、利用可能と見なされます。
- サービスレベル目標 (SLOs): AMS サービスの特定のサービス目標を定義する AMS 契約の一部。

パッチ適用用語：

- 必須パッチ: 環境またはアカウントのセキュリティ状態を損なう可能性のある問題に対処するための重要なセキュリティ更新プログラム。「Critical Security update」は、AMS がサポートするオペレーティングシステムのベンダーによって「Critical」と評価されたセキュリティ更新プログラムです。

- 発表されたパッチとリリースされたパッチ: パッチは通常、スケジュールに従って発表およびリリースされます。エマージェントパッチは、パッチの必要性が検出された時点で発表され、通常はその直後にパッチがリリースされます。
- パッチアドオン: AWS Systems Manager (SSM) 機能を利用する AMS インスタンスのタグベースのパッチ適用。これにより、ベースラインと設定したウィンドウを使用してインスタンスにタグを付け、それらのインスタンスにパッチを適用できます。
- パッチメソッド :
  - インプレースパッチ適用: 既存のインスタンスを変更することで実行されるパッチ適用。
  - AMI 置換パッチ適用: 既存の EC2 Auto Scaling グループ起動設定の AMI リファレンスパラメータを変更することで実行されるパッチ適用。
- パッチプロバイダー (OS ベンダー、サードパーティー): パッチは、アプリケーションのベンダーまたは管理機関によって提供されます。
- パッチタイプ :
  - 重要なセキュリティ更新プログラム (CSU): サポートされているオペレーティングシステムのベンダーによって「重要な」と評価されたセキュリティ更新プログラム。
  - 重要な更新 (IU): サポートされているオペレーティングシステムのベンダーによって「重要」と評価されたセキュリティ更新プログラム、または「重要」と評価されたセキュリティ以外の更新プログラム。
  - その他の更新 (OU): CSU または IU ではないサポートされているオペレーティングシステムのベンダーによる更新。
- サポートされているパッチ: AMS はオペレーティングシステムレベルのパッチをサポートしています。アップグレードは、セキュリティの脆弱性やその他のバグを修正するため、またはパフォーマンスを向上させるために、ベンダーによってリリースされます。現在サポートされている OSs「サポート設定」を参照してください。

セキュリティ条件 :

- Detective Controls: AMS が作成または有効化したモニターのライブラリ。セキュリティ、運用、またはカスタマーコントロールと一致しない設定について、カスタマーマネージド環境とワークロードを継続的に監視し、所有者に通知し、リソースをプロアクティブに変更または終了することでアクションを実行します。

サービスリクエストの条件 :

- サービスリクエスト: AMS がユーザーに代わって実行するアクションに対するユーザーによるリクエスト。
- アラート通知: AMS アラートがトリガーされたときに、AMS によってサービスリクエストリストページに投稿された通知。アカウントに設定された連絡先は、設定された方法 (E メールなど) からも通知されます。インスタンス/リソースに問い合わせタグがあり、タグベースの通知について Cloud Service Delivery Manager (CSDM) に同意している場合、タグの連絡先情報 (キー値) にも自動 AMS アラートが通知されます。
- サービス通知: サービスリクエストリストページに投稿される AMS からの通知。

その他の用語：

- AWS Managed Servicesインターフェイス: AMS の場合: AWS Managed Servicesアドバンストコンソール、AMS CM API、 サポート および API。AMS Accelerate の場合: サポート コンソールとサポート API。
- 顧客満足度 (CSAT): AMS CSAT には、提供されたすべてのケースやコレスポンデンスに関するケースコレスポンデンス評価、四半期ごとのアンケートなど、詳細な分析情報が提供されます。
- DevOps: DevOps は、すべてのステップで自動化とモニタリングを強く推奨する開発手法です。DevOps は、開発と運用の従来分離された機能をオートメーションの基盤の上に統合することで、開発サイクルの短縮、デプロイ頻度の増加、より信頼性の高いリリースを目指しています。デベロッパーがオペレーションを管理でき、オペレーションが開発を通知すると、問題や問題がより迅速に発見されて解決され、ビジネス目標がより簡単に達成されます。
- ITIL: Information Technology Infrastructure Library (ITIL) は、IT サービスのライフサイクルを標準化するために設計された ITSM フレームワークです。ITIL は、IT サービスのライフサイクルをカバーする 5 つの段階で構成されています。サービス戦略、サービス設計、サービス移行、サービス運用、サービス改善です。
- IT サービス管理 (ITSM): IT サービスをビジネスニーズに合わせる一連のプラクティス。
- Managed Monitoring Services (MMS): AMS は、独自のモニタリングシステムである Managed Monitoring Service (MMS) を運用します。これは、 AWS ヘルスイベントを消費し、Amazon CloudWatch データ、および他の からのデータを集約し AWS のサービス、Amazon Simple Notification Service (Amazon SNS) トピックを通じて作成されたアラームを AMS オペレーター (オンライン 24 時間 365 日) に通知します。
- 名前空間: IAM ポリシーを作成するとき、または Amazon リソースネーム (ARNs) を操作するときに、名前空間 AWS のサービス を使用して を識別します。アクションとリソースを識別するときに名前空間を使用します。

運用モデルとは

AMS のお客様として、組織はアプリケーションとインフラストラクチャのオペレーションを分離し、インフラストラクチャのオペレーションに AMS を使用することを決定しました。AMS は、アプリケーション設計および開発チームとインフラストラクチャ設計チームと協力して、インフラストラクチャ運用がスムーズに実行されるようにします。次の図は、この概念を示しています。

AMS は AWS インフラストラクチャオペレーションを担当し、チームはアプリケーションオペレーションを担当します。アプリケーションとインフラストラクチャの設計チームとして、AMS インフラストラクチャの本番環境にデプロイされたら、誰がアプリケーションを運用するかを理解する必要があります。このガイドでは、アプリケーションのデプロイとメンテナンスに関連するインフラストラクチャ設計への一般的なアプローチについて説明します。

AWS Managed Services でのサービス管理

トピック

AMS サービスの仕組み。

AWS Managed Services でのアカウントガバナンス

このセクションでは、AMS アカウントガバナンスについて説明します。

お客様は、AMS 全体でアドバイザリ支援を提供するクラウドサービスデリバリーマネージャー (CSDM) として指定され、マネージド環境のユースケースとテクノロジーアーキテクチャを詳細に理解しています。CSDMs、必要に応じてアカウントマネージャー、テクニカルアカウントマネージャー、AWS Managed Services クラウドアーキテクト (CAs)、および AWS ソリューションアーキテクト (SAs) と連携して、新しいプロジェクトの起動を支援し、ソフトウェア開発および運用プロセス全体でベストプラクティスのレコメンデーションを提供します。CSDM は AMS の主な連絡先です。CSDM の主な責任は次のとおりです。

- 顧客との毎月のサービスレビュー会議を組織し、主導します。
- セキュリティ、環境のソフトウェア更新、最適化の機会に関する詳細を提供します。
- AMS の機能リクエストを含む要件を支持します。
- 請求およびサービスレポートリクエストに応答して解決します。
- 財務およびキャパシティ最適化のレコメンデーションに関するインサイトを提供します。

AWS Managed Servicesサービスのサービス開始

サービス開始日: AWS Managed Servicesアカウントのサービス開始日は、その AWS AWS Managed Servicesアカウントのオンボーディング要件に規定されたアクティビティが完了したことを AWS が

通知した最初の暦月の最初の日です。ただし、AWS が暦月の 20 日以降にそのような通知を行う場合、サービス開始日は、そのような通知の日付の翌日の 2 番目の暦月の最初の日です。

サービスの開始

- R は、タスクを達成するために作業を行う責任者を表します。
- 「情報」の略です。多くの場合、タスクまたは成果物の完了時にのみ、進捗状況が通知される当事者です。

サービス開始

ステップ #	ステップタイトル	説明	お客様	AMS
1.	お客様の AWS アカウントの引き渡し	顧客が新しい AWS アカウントを作成し、AWS Managed Servicesに引き渡す	R	I
2.	AWS Managed Servicesアカウント - 設計	AWS Managed Servicesアカウントの設計を確定する	I	R
3.	AWS Managed Servicesアカウント - ビルド	AWS Managed Services アカウントは、ステップ 2 の設計に従って構築されます。	I	R

顧客関係管理 (CRM)

AWS Managed Services (AMS) は、明確に定義された関係が確立され、維持されるように、顧客関係管理 (CRM) プロセスを提供します。この関係の基盤は、ビジネス要件に関する AMS のインサイトに基づいています。CRM プロセスは、以下を正確かつ包括的に理解するのに役立ちます。

- ビジネスニーズとそのニーズを満たす方法
- 機能と制約
- AMS とさまざまな責任と義務

CRM プロセスにより、AMS は一貫した方法を使用してサービスを提供し、AMS との関係のガバナンスを提供できます。CRM プロセスには以下が含まれます。

- 主要な利害関係者の特定
- ガバナンスチームの設立
- サービスレビューミーティングの実施と文書化
- エスカレーション手順を含む正式なサービス苦情手順の提供
- 満足度とフィードバックプロセスの実装とモニタリング
- 契約の管理

CRM プロセス

CRM プロセスには、次のアクティビティが含まれます。

- ビジネスプロセスとニーズを特定して理解します。AMS との契約により、ステークホルダーが特定されます。
- ニーズと要件を満たすために提供されるサービスを定義します。
- サービスレビュー会議で会議を開き、AMS サービスの範囲、SLA、契約、ビジネスニーズの変更について話し合います。暫定会議を開催して、パフォーマンス、成果、問題、アクションプランについて話し合うことができます。
- 顧客満足度調査と会議で提供されたフィードバックを使用して、満足度をモニタリングします。
- 内部で測定された毎月のパフォーマンスレポートのパフォーマンスを報告します。
- サービスを確認して、改善の機会を決定します。これには、提供される AMS サービスのレベルと品質に関する頻繁なコミュニケーションが含まれます。

CRM 会議

AMS クラウドサービスデリバリーマネージャー (CSDMs) は、サービストラック (運用、セキュリティ、製品イノベーション) とエグゼクティブトラック (SLA レポート、満足度評価、ビジネスニーズの変化) について、定期的に会議を開催します。

会議	目的	モード	参加者
週次ステータスレビュー (オプション)	未解決の問題またはインシデント、パッチ適用、セキュリティイベント、問題レコード 12 週間の運用傾向 (+/- 6) アプリケーションオペレーターの懸念 週末のスケジュール	オンサイトのお客様のlocation/Telecom/Chime	AMS: CSDM とクラウドアーキテクト (CA) 顧客に割り当てられたチームメンバー (クラウド/インフラストラクチャ、アプリケーションサポート、アーキテクチャチームなど)
毎月のビジネスレビュー	サービスレベルのパフォーマンス (レポート、分析、傾向) を確認する 財務分析 製品ロードマップ CSAT	オンサイトのお客様のlocation/Telecom/Chime	AMS: CSDM、クラウドアーキテクト (CA)、AMS アカウントチーム、AMS テクニカルプロダクトマネージャー (TPM) (オプション)、AMS OPS マネージャー (オプション) お客様: アプリケーションオペレーターの担当者
四半期ごとのビジネスレビュー	スコアカードとサービスレベルアグリーメント (SLA) のパフォーマンスと傾向 (6 か月) 今後の 3/6/9/12 か月の計画/移行	オンサイトのカスタマーロケーション	AMS: CSDM、クラウドアーキテクト、AMS アカウントチー

会議	目的	モード	参加者
	リスクとリスクの軽減 主要な改善イニシアチブ 製品ロードマップ項目 将来の方向性に沿った機会 財務 コスト削減の取り組み ビジネスの最適化		ム、AMS サービスディレクター、AMS オペレーションマネージャー お客様: アプリケーションオペレーター担当者、サービス担当者、サービスディレクター

CRM ミーティングの手配

AMS CSDM は、以下を含む会議を文書化する責任があります。

- アクション項目、問題、参加者リストを含むアジェンダの作成。
- 各会議でレビューされたアクション項目のリストを作成して、項目がスケジュールどおりに完了および解決されていることを確認します。
- 会議の後 1 営業日以内に、会議の参加者に会議議事録とアクション項目リストを E メールで配布します。
- 会議の議事録を適切なドキュメントリポジトリに保存します。

CSDM がない場合、会議を主導する AMS 担当者が議事録を作成して配布します。

> **Note**
>
> CSDM はお客様と協力してアカウントガバナンスを確立します。

CRM 月次レポート

AMS CSDM は、毎月のサービスパフォーマンスプレゼンテーションを準備して送信します。プレゼンテーションには、以下に関する情報が含まれています。

- レポート日
- 概要とインサイト：
  - 主なコールアウト: 合計スタック数とアクティブなスタック数、スタックパッチ適用ステータス、アカウントのオンボーディングステータス (オンボーディング中のみ) 、お客様固有の問題の概要
  - パフォーマンス: インシデント解決、アラート、パッチ適用、変更リクエスト (RFCs)、サービスリクエスト、コンソールと API の可用性に関する統計
  - 問題、課題、懸念、リスク: お客様固有の問題のステータス
  - 今後のアイテム: 顧客固有のオンボーディングプランまたはインシデント解決プラン
- マネージドリソース: スタックのグラフと円グラフ
- AMS メトリクス: モニタリングとイベントメトリクス、インシデントメトリクス、AMS SLA 準拠メトリクス、サービスリクエストメトリクス、変更管理メトリクス、ストレージメトリクス、継続性メトリクス、Trusted Advisor メトリクス、コスト概要 (いくつかの方法を提示) 。機能リクエスト。連絡先情報。

> **Note**
>
> 説明されている情報に加えて、CSDM は、AMS による運用活動のための下請業者の使用など、範囲または条件の重要な変更についても通知します。
> AMS は、CSDM が月次レポートに含めるパッチ適用とバックアップに関するレポートを生成します。レポート生成システムの一部として、AMS はアクセスできないインフラストラクチャをアカウントに追加します。
>
> - 未加工データが報告された S3 バケット
> - データをクエリするためのクエリ定義を含む Athena インスタンス
> - S3 バケットから raw データを読み取る Glue クローラ

AWS Managed Servicesコスト最適化

AWS Managed Services は、毎月のビジネスレビュー (MBRs) 中に、毎月詳細なコスト使用率と削減額レポートを提供します。

AMS は、標準のプロセスとメカニズムのセットに従って、マネージドアカウントのコスト削減手段を特定し、AWS 支出を最適化するための変更の計画とロールアウトを支援します。

> **Note**
>
> AMS は、コスト最適化に役立つ動画を開発しています。最初のステップでは、コスト最適化のベストプラクティスの PDF と Excel スプレッドシートを提供します。これらのリソースにアクセスするには、クイックガイドを開いてコスト最適化 ZIP ファイルにアクセスします。

コスト最適化フレームワーク

AMS は、AWS コストを最適化するために 3 段階のアプローチに従います。

1. マネージド環境のコスト最適化方法を特定する
2. コスト最適化計画を提示する
3. 測定可能な方法でコスト最適化の達成を支援する

マネージド環境のコスト最適化手段を特定する

AMS は Cost Explorer や Trusted Advisor などの AWS ネイティブツールを使用し、アーキテクチャの最適化、EC2 インスタンス、 AWS アカウントに焦点を当てた最適化全体で 20 を超えるコスト削減パターンを活用して、カスタマイズされたコスト削減レコメンデーションを構築します。

最適化に関する推奨事項には、次のようなものがあります。

アーキテクチャ最適化の推奨事項：

- 最適な S3 ストレージクラスの使用: Amazon S3 は、データアクセス、耐障害性、コストに基づいて、さまざまなワークロード要件を満たすさまざまなストレージクラスを提供します。ワークロードのニーズに基づく S3 Intelligent-Tiering と S3 ストレージクラス分析により、S3 のコストを効率的に管理できます。
- キャッシュアーキテクチャの使用: 必要に応じてキャッシュインスタンスを活用すると、IOPS 要件を満たしながら、一部のデータベースインスタンスを置き換えるのに役立ちます。
- EBS アップグレードの削減: EBS ボリュームを gp2 から gp3 に移行すると、最大 20% のコスト削減を実現し、ボリュームサイズに関係なく、予測可能な 3,000 IOPS ベースラインパフォーマンスと 125 MiB/秒を活用できます。

- 伸縮性の使用: AWS が提供する自動スケーリング機能により、効果的なリソース使用率とコスト最適化の手段が可能になります。必要に応じてインスタンススケーリングポリシーを定期的に確認および更新することで、さらにコスト削減につながります。

EC2 インスタンスに焦点を当てた推奨事項

- インスタンスの適正化: インスタンスのサイズ設定と使用状況に基づく最適な設定に焦点を当てた推奨事項。推奨事項には、Amazon EC2 Auto Scaling 機能の使用や、該当する場合は EC2 インスタンスを Amazon S3 AWS Lambda の静的ウェブコンテンツに置き換えることなどが含まれます。
- インスタンススケジューリング: AMS Resource Scheduler を使用して、時間スケジュールに基づいてインスタンスを自動的に開始および停止すると、特に営業時間外に使用されていない非本番稼働用インスタンスのコストを抑えるのに役立ちます。
- Savings Plans にサブスクライブする: Savings Plans は、 AWS 使用量を節約する最も簡単な方法です。EC2 Instance Savings Plans は、Amazon EC2 インスタンスの使用量に対するオンデマンド料金と比較して、最大 72% の割引を提供します。Amazon SageMaker AI Savings Plans は、Amazon SageMaker AI サービスの使用量を最大 64% 削減します。AMS は、 AWS リソースの使用状況に基づいて Savings プランに関する適切な推奨事項を提供します。
- リザーブドインスタンス (RI) の使用および消費ガイダンス: Amazon EC2 リザーブドインスタンス (RI) は、オンデマンド料金と比較して大幅な割引 (最大 75%) を提供し、特定のアベイラビリティーゾーンで使用するとキャパシティ予約を提供します。
- スポットインスタンスの使用: フォールトトレラントワークロードはスポットインスタンスを活用し、最大 90% の料金を削減できます。
- アイドルインスタンスの終了: アイドル状態のインスタンス、または終了できる使用率が低いインスタンスを特定してレポートします。

アカウントに焦点を当てたレコメンデーション

- アカウントのクリーンアップ: アカウントレベルでは、AMS は未使用の EBS ボリューム、重複する CloudTrail 証跡、未使用のリソースを持つ空のアカウントなども識別し、クリーンアップの推奨事項を提供します。
- SLA レコメンデーション: さらに、AMS は Plus アカウントと Premium アカウントを定期的に確認し、アカウントに適した SLA レベルを選択することをお勧めします。
- AMS オートメーションの最適化: AMS は、AMS サービスを提供するために使用される AMS オートメーションとインフラストラクチャを継続的に最適化します。

顧客に提示し、計画を支援する

AMS は、主要な顧客のステークホルダーと毎月のビジネスレビュー (MBRs) を実施し、特定されたコスト削減の方法、メカニズム、推奨事項と潜在的なコスト削減を提示します。さらに、お客様と協力して必要な変更を計画します。

レコメンデーションの実装を支援し、コストへの影響を測定する

AMS は、コストへの影響と最適化の変更の達成と測定を支援します。

推奨される変更のアプリケーションへの影響、リスク、成功基準を評価し、AMS コンソールを使用して適切な変更リクエスト (RFCs) を作成します。AMS はユーザーと協力して、マネージドアカウントのコスト最適化に関連する変更を実装します。AMS はコストへの影響を測定し、月次ビジネスレビュー (MBRs。

コスト最適化責任マトリックス

AMS コスト最適化の責任。

コスト最適化 RACI

アクティビティ	お客様	AMS
コスト削減レコメンデーションのコンパイルとレポートの準備	I	R
コスト削減レポートの表示	C	R
コスト削減に関連	R	C

アクティビティ	お客様	AMS
する変更の計画		
変更の影響とリスクの評価	R	C
変更を実装するための RFCs の引き上げ	R	C
RFCs の確認と変更の実装	C	R
アプリケーションのテストと変更実装の検証	R	C
変更後のコストへの影響を測定し、顧客に提示する	I	R

AWS Managed Services のサービス時間

機能	AMS アドバンスト
	プレミアム階層
サービスリクエスト	24 時間 365 日
インシデント管理 (P2-P3)	24 時間 365 日
バックアップとリカバリ	24 時間 365 日
パッチ管理	24 時間 365 日
モニタリングとアラート	24 時間 365 日
変更の自動リクエスト (RFC)	24 時間 365 日
自動変更リクエスト (RFC)	24 時間 365 日
クラウドサービスデリバリーマネージャー (CSDM)	月曜日 ~ 金曜日: 8:00 ~ 17:00、現地営業時間

AWS Managed Services でのヘルプの取得

AMS は、インシデント管理、サービスリクエスト管理、変更管理を 1 日 24 時間、週 7 日、年 365 日サポートします (アカウントに適用される AMS サービスレベルアグリーメントに準拠) 。

マネージド環境に影響する AWS または AMS サービスのパフォーマンスの問題を報告するには、AMS コンソールを使用してインシデントレポートを送信します。詳細については、「インシデントの報告」を参照してください。AMS インシデント管理の一般的な情報については、「インシデント対応」を参照してください。

情報やアドバイスを求めたり、AMS から追加のサービスをリクエストしたりするには、AMS コンソールを使用してサービスリクエストを送信します。詳細については、サービスリクエストの作成を参照してください。AMS サービスリクエストの一般的な情報については、「サービスリクエスト管理」を参照してください。

アプリケーション開発

AWS Managed Services (AMS) 環境へのアプリケーションの効果的な設計とデプロイを可能にするアプリケーション開発プロセスとプラクティス。AMS は、以下の大まかなプロセスをガイドします。

1. AMS マネージド環境に開発または統合するアプリケーションを構想し、設計します。いくつかの考慮事項：
    a. アプリケーションをどのようにデプロイしますか？ Ansible などのデプロイツールを使用して自動化する場合、または必要なファイルを直接アップロードして手動で自動化する場合
    b. アプリケーションを更新する方法 変更可能なアプローチで各インスタンスを個別に更新するか、変更不可能なアプローチで Auto Scaling グループの 1 つの更新された AMI で各インスタンスを更新しますか？
2. AWS アーキテクチャライブラリ、 AWS 「Well-Architected」ガイダンス、AMS およびその他のクラウドアーキテクチャ分野のエキスパートを使用して、アプリケーションをホストするために使用されるインフラストラクチャを計画および設計します。このガイドの以下のセクションでは、これに役立つ情報を提供します。
3. インフラストラクチャのデプロイ方法を選択します。
    a. フルスタック: すべてのインフラストラクチャコンポーネントが一度にデプロイされます。
    b. 階層と階層: インフラストラクチャのデプロイは個別にデプロイされ、その後、セキュリティグループの変更と結び付けられます。このタイプのデプロイは、Auto Scaling グループの作成時に以前に作成したロードバランサーを指定するなど、 が相互に構築するスタックコンポーネントのシリアル設定によっても実現されます。
    c. Dev、Staging、Prod など、どのような環境を採用しますか？
4. 必要なスタックまたは階層をプロビジョニングする AMS 変更タイプ (CTs) を選択し、必要な変更リクエスト (RFCs。
5. RFCs を送信して、適切な環境へのインフラストラクチャのデプロイをトリガーします。
6. 選択したアプリケーションデプロイアプローチを使用してアプリケーションをデプロイします。
7. 必要に応じてインフラストラクチャとアプリケーションを再処理します。
8. 最初のデプロイが非本番環境であると仮定して、インフラストラクチャとアプリケーションを適切なフォローアップ環境にデプロイします。
9. 継続的なメンテナンスは、基盤となるインフラストラクチャを運用する AMS と、アプリケーション (複数可) インフラストラクチャを運用する運用チームによって処理されます。

10. アプリケーションを廃止するには、その AMS インフラストラクチャを終了します。

優れた設計

適切に設計されたシステムは、ビジネスの成功の可能性を大幅に高めると AWS 確信しています。AWS アーキテクチャセンターは、 でのアーキテクトに関する専門的なガイダンスを提供します AWS クラウド。

システムの構築時に行う必要がある決定の長所と短所を理解するため、以下の記事とホワイトペーパーをお勧めします AWS。

Well-Architected とは？: AWS Well-Architected フレームワークを、次の 6 つの柱に基づいて紹介します。

- 運用上の優秀性: 運用上の優秀性の柱は、ビジネス価値を提供し、プロセスと手順を継続的に改善するためのシステムの実行とモニタリングに焦点を当てています。主なトピックには、変更の管理と自動化、イベントへの対応、日常業務を正常に管理するための標準の定義が含まれます。
- セキュリティ: セキュリティの柱は、情報とシステムの保護に焦点を当てています。主なトピックには、データの機密性と完全性、アクセス許可管理で誰が何をできるかの特定と管理、システムの保護、セキュリティイベントを検出するためのコントロールの確立などがあります。
- 信頼性: 信頼性の柱は、ビジネスと顧客の需要を満たすための障害を防止し、迅速に復旧する能力に焦点を当てています。主なトピックには、セットアップ、クロスプロジェクト要件、復旧計画、変更の処理方法に関する基本的な要素が含まれます。
- パフォーマンス効率: パフォーマンス効率の柱は、IT とコンピューティングリソースの効率的な使用に焦点を当てています。主なトピックには、ワークロード要件に基づいた適切なリソースの種類とサイズの選択、パフォーマンスの監視、ビジネスニーズの変化に応じて効率を維持するための情報に基づいた意思決定が含まれます。
- コスト最適化: コスト最適化の柱は、不要なコストを回避することに重点を置いています。主なトピックには、支出先の理解と制御、最も適切で適切なリソースタイプの選択、時間の経過に伴う支出の分析、過剰支出のないビジネスニーズを満たすためのスケーリングなどがあります。
- 持続可能性: 持続可能性の柱は、プロビジョニングされたリソースの利点を最大化し、必要なリソースの合計を最小限に抑えることで、エネルギー消費を削減し、ワークロードのすべてのコンポーネントにわたって効率を高めることで、持続可能性への影響を継続的に改善する能力に焦点を当てています。

AWS Well-Architected フレームワーク: がクラウドベースのアーキテクチャを評価して改善し、設計上の意思決定によるビジネスへの影響をよりよく理解 AWS できるようにする方法について説明します。Well-Architected フレームワークの柱として AWS を定義する 6 つの概念領域における一般的な設計原則と、特定のベストプラクティスとガイダンスについて説明します。

AMS でのアプリケーションレイヤーの責任とインフラストラクチャレイヤーの責任

AMS、インフラストラクチャ、およびメンテナンスと成長に必要なものはすべて、AMS によって維持されます。ただし、line-of-businessアプリケーションや製品アプリケーションに必要なものは、お客様が開発、デプロイ、保守します。

CodeDeploy や CloudFormation Chef、Puppet、Ansible、Saltstack などのアプリケーションデプロイツールを使用すると、AMS マネージドインフラストラクチャへのアプリケーションのデプロイを完全に自動化できます。

AMS が行うこととしないことの詳細については、「」を参照してください何をするか、何をしないか。

AMS での Amazon EC2 インスタンスのミュータビリティ

ユーザーと AMS は、次の 2 つの方法のいずれかでインフラストラクチャに Amazon Elastic Compute Cloud (Amazon EC2) インスタンスを維持できます。

- イミュータブル: このモデルは、必要な機能でベイク (作成された) された Amazon マシンイメージ (AMIs) を使用します。更新をデプロイすると、既存のインスタンスは破棄され、更新された AMI から作成された新しいインスタンスに完全に置き換えられます。ダウンタイムを最小限に抑えるために、このローリングプロセスでは、一部のインスタンスは更新されず、アクセス可能になりますが、他のインスタンスは、最終的に新しい変更が完全にデプロイされるまで更新されます。
- ミュータブル: このモデルでは、インフラストラクチャが更新され、 クラウド内の既存のシステムに新しいコードがデプロイされます。このモデルは、手動で更新をプッシュし、infrastructure-as-codeを使用して更新をデプロイする組み合わせであり、新しい AMIs に依存しません。

これらのメンテナンスモデルについては、このガイドの後のセクションで詳しく説明します。

AMS リソースでの AWS Secrets Manager の使用

シークレットを AMS と共有する必要がある場合があります。次に例を示します。

- RDS インスタンスのマスターパスワードのリセット
- ロードバランサーの証明書
- AMS から IAM ユーザーの存続期間の長い認証情報を取得する

AMS と機密情報を共有する最も安全な方法は、 AWS Secrets Manager を使用することです。以下の手順に従ってください。

1. シングルアカウントランディングゾーン (SALZ) のフェデレーティッドアクセスと CustomerReadOnly ロールを使用して AWS コンソールにログインします。マルチアカウントランディングゾーン (MALZ) のロール、AWSManagedServicesSecurityOpsRole、AWSManagedServicesAdminRole、および AWSManagedServicesChangeManagementRole のいずれかを使用します。
2. AWS Secrets Manager コンソールに移動し、新しいシークレットを保存するをクリックします。
3. 「その他のタイプのシークレット」を選択します。
4. シークレット値をプレーンテキストとして入力し、次へをクリックします。
5. シークレットの名前と説明を入力します。名前は常に「customer-shared/*」で始まる必要があります。たとえば、「customer-shared/license-2018」です。完了したら、次へをクリックして続行します。
6. デフォルトの KMS 暗号化を使用します。
7. 自動ローテーションを無効にしたまま、次へをクリックします。
8. Store を確認してクリックし、シークレットを保存します。
9. シークレットを識別して取得できるように、シークレット名と ARN を含む AMS サービスリクエストで返信してください。サービスリクエストの作成については、「サービスリクエストの例」を参照してください。

AMS でのアプリケーションのデプロイ

オンボーディング中、AWS Managed Services (AMS) はお客様と協力して必要なインフラストラクチャを決定します。

基本インフラストラクチャには、 AWS 仮想プライベートクラウド (VPC)、ADFS フォレスト信頼を介した通信セキュリティ、2 つのアベイラビリティーゾーンにミラーリングされた基本サブネット (DMZ、共有サービス、プライベート) 、マネージド NAT、踏み台、パブリックロードバランサー、Direct Connect (DX)、および必要なセキュリティが含まれます。アプリケーションリソースは、プライベートサブネットまたはカスタマーアプリケーションサブネットにデプロイされます。一般的な AMS アーキテクチャの詳細については、AWS Managed Services ユーザーガイドを参照してください。

基本が完了したら、デプロイするインフラストラクチャには、アプリケーションとアプリケーション開発のすべてのコンポーネントを含める必要があります。

AMS でのアプリケーションのデプロイ機能

AMS でアプリケーションをデプロイする方法の一部。各方法の詳細については、以下を参照してください。

アプリケーションデプロイ機能の例

メソッド名	インフラストラクチャのデプロイ	AMI またはキー要素 (複数可)	アプリケーションのインストール
ミュータブルアプリケーション、AMS AMI			
手動アプリケーションデプロイ	フルスタック CT または階層 CT と絞り込み CTs	AMS 提供の AMI	アクセス管理 CT を送信し、アプリケーションを手動でインストールします。
アプリケーションエージェント (Chef、Puppet など) を使用した UserData			アプリケーションエージェントをインストールする UserData スクリプトでプロビジョニング CT を使用

メソッド名	インフラストラクチャのデプロイ	AMI またはキー要素 (複数可)	アプリケーションのインストール
アプリケーションのデプロイ			し、そのスクリプト/エージェントがアプリケーションをインストールします。
UserData エージェントレスアプリケーションのデプロイ (Ansible、Salt SSH など)			アクセス管理 CT を送信し、アプリケーションエージェントをインストールします。アプリケーションデプロイツールを使用してアプリケーションをデプロイします。
ミュータブルアプリケーション、カスタム AMI			
カスタム AMI アプリケーションのデプロイ (ASG 以外)	フルスタック CT または階層 CT と絞り込み CTs	カスタム AMI。AMS AMI -> アプリケーションデプロイツールエージェントでカスタマイズ -> EC2 インスタンスの作成 (CT) -> AMI の作成 (CT)。	アプリケーションデプロイツール (Chef など) は、エージェントを活用してアプリケーションをデプロイします。
AWS Database Migration Service (DMS) アプリケーションのデプロイ	既存の AMS リレーショナルデータベーススタックへの AWS DMS 同期。	カスタム AMI	お客様またはパートナーが AWS Database Migration Service を採用している。AMS は起動時に AMS コンポーネントを検証する

メソッド名	インフラストラクチャのデプロイ	AMI またはキー要素 (複数可)	アプリケーションのインストール
ワークロード取り込みアプリケーションのデプロイ	パートナーが移行したインスタンス/AMI とお客様が開始したワークロード取り込み CT。		パートナーはインスタンスを移行し、お客様の AMS マネージド VPC に AMI を作成します。お客様はワークロード取り込み CT を使用して AMS でスタックを起動します。 詳細については、「AMS ワークロード取り込み (WIGS)」を参照してください。
イミュータブルなアプリケーション			
カスタム AMI アプリケーションデプロイ (ASG)	フルスタック CT または階層 CT と絞り込み CTs	AMS AMI -> カスタマイズ -> EC2 インスタンスの作成 (CT) -> AMI の作成 (CT) -> Auto Scaling グループの作成。	Auto Scaling がカスタム AMI を使用してアプリケーションをデプロイする 詳細については、「AMS でのアプリデプロイの階層化と階層化」を参照してください。
ミュータブルアプリケーションまたはイミュータブルアプリケーション			

メソッド名	インフラストラクチャのデプロイ	AMI またはキー要素 (複数可)	アプリケーションのインストール
カスタム CloudFormation テンプレートアプリケーションのデプロイ	CloudFormation テンプレート	AWS CloudFormation テンプレート -> AMS のカスタマイズ/準備 -> デプロイ | 取り込み | CloudFormation テンプレートからのスタック | 作成 (ct-36cn2avfrrj9v)。	AMS は、カスタム CloudFormation テンプレートを使用してアプリケーションをアカウントにデプロイし、アプリケーションのデプロイを検証します。 詳細については、「AMS CloudFormation の取り込み」を参照してください。
SQL データベースのインポート	AMS オペレーション (その他 | その他 CT)	オンプレミス SQL データベース -> .bak ファイル -> AMS RDS SQL データベース -> 管理 | その他 | その他 | インポート用に (ct-1e1xtak34nx76) を作成します。	AMS は、オンプレミスデータベースを AMS 管理の RDS データベースにインポートします。詳細については、「AMS RDS for Microsoft SQL Server へのデータベース (DB) のインポート」を参照してください。

メソッド名	インフラストラクチャのデプロイ	AMI またはキー要素 (複数可)	アプリケーションのインストール
データベース移行サービス (DMS)	AMS オペレーション (複数 CTs)	オンプレミスデータベース -> DMS レプリケーションインスタンス -> DMS レプリケーションサブネットグループ -> DMS ターゲットエンドポイント -> DMS ソースエンドポイント -> DMS レプリケーションタスク。	AMS は、オンプレミスデータベースを AMS 管理の S3 またはターゲット RDS データベースにインポートします。詳細については、「AWS Database Migration Service (AWS DMS)」を参照してください。
CodeDeploy アプリケーションのデプロイ	CodeDeploy	アプリケーション -> CodeDeploy アプリケーション -> CodeDeploy デプロイグループ -> CodeDeploy デプロイ。	使用状況、インプレースまたはブルー/グリーンアプリケーションのデプロイによって異なります。詳細については、CodeDeploy リクエスト を参照してください

AMS でのアプリケーションデプロイの計画

アプリケーションのデプロイを有効にするために回答すべき一連の推奨質問については、「」を参照してください付録: アプリケーションオンボーディングアンケート。質問では、以下について説明します。

- デプロイの概要
- インフラストラクチャデプロイコンポーネント
- アプリケーションホスティングプラットフォーム
- アプリケーションデプロイモデル
- アプリケーション依存関係

- 製品アプリケーションの SSL 証明書

AMS ワークロード取り込み (WIGS)

トピック

- ワークロードの移行: Linux と Windows の前提条件
- 移行でリソースを変更する方法
- ワークロードの移行 : 標準プロセス
- ワークロードの移行: CloudEndure ランディングゾーン (SALZ)
- AMS Tools アカウント (ワークロードの移行)
- ワークロードの移行: Linux の取り込み前検証
- ワークロードの移行: Windows の取り込み前検証
- ワークロード取り込みスタック: の作成

AMS クラウド移行パートナーで AMS ワークロード取り込み変更タイプ (CT) を使用して、既存のワークロードを AMS マネージド VPC に移動します。AMS ワークロード取り込みを使用して、移行されたインスタンスを AMS に移行した後にカスタム AMS AMI を作成できます。このセクションでは、移行パートナーと自分自身が AMS ワークロードの取り込みのために実行するプロセス、前提条件、および手順について説明します。

> **Important**
>
> オペレーティングシステムは、AMS ワークロードの取り込みでサポートされている必要があります。サポートされているオペレーティングシステムについては、「」を参照してくださいワークロードの移行: Linux と Windows の前提条件。
> ワークロードとアカウントはそれぞれ異なります。AMS はお客様と協力して、成功した結果に備えます。

次の図は、AMS ワークロードの取り込みプロセスを示しています。

ワークロードの移行: Linux と Windows の前提条件

オンプレミスインスタンスのコピーを AWS Managed Services (AMS) に取り込む前に、特定の前提条件を満たす必要があります。これらは、Windows オペレーティングシステムと Linux オペレーティングシステムで異なるものを含む前提条件です。

> **Note**
>
> インスタンスが取り込む準備ができているかどうかを判断するプロセスを簡素化するために、Windows と Linux の両方の検証ツールが作成されました。これらのツールは、オンプレミスサーバーと AWS の EC2 インスタンスに直接ダウンロードして実行できます。Linux Pre-WIGS Validation.zip、Windows Pre-WIGS Validation.zip。

開始する前に、Linux および Windows の場合：

- フルウイルススキャンを実行します。
- インスタンスにはインスタンス`customer-mc-ec2-instance-profile`プロファイルが必要です。
- Amazon EC2 Systems Manager (SSM) エージェントをインストールし、SSM Agent が稼働していることを確認します。
- AMS ワークロード取り込み (WIGS) を実行するには、ルートボリュームで最低 10GB の空きディスク容量が推奨されます。運用上、AMS はディスク使用率を 75% 未満にすることを推奨し、ディスク使用率が 85% に達したときに警告します。
- 移行パートナーとの取り込みの時間枠を決定します。
- カスタム AMI は、ターゲットの本番稼働用 AMS アカウントに EC2 インスタンスとして存在します (これは移行パートナーの責任です) 。

> **Important**
>
> オペレーティングシステムは、AMS ワークロードの取り込みでサポートされている必要があります。
>
> - 以下のオペレーティングシステムがサポートされています。
>   - Microsoft Windows Server: 2008 R2、2012、2012 R2、2016、2019、2022

- Linux: Amazon Linux 2023、Amazon Linux 2、Amazon Linux、CentOS 7.x、CentOS 6.5-6.10、Oracle Linux 7: マイナーバージョン 7.5 以降、Oracle Linux 8: 8.3 までのマイナーバージョン、RHEL 8.x、RHEL 7.x、RHEL 6.5-6.10、SUSE Linux Enterprise Server 15 SP3, SP4、SAP 固有のバージョン、SUSE Linux Enterprise Server 12 SP5、Ubuntu 18.04

- 次の AMIsはサポートされていません。
  - Amazon Linux 2023 最小 AMI。

> **Note**
>
> AMS API/CLI (amscm および amsskms) エンドポイントは、AWS N. Virginia リージョン にありますus-east-1。認証の設定方法、およびアカウントとリソースがどの AWS リージョンにあるかによっては、コマンドの発行--region us-east-1時に追加する必要がある場合があります。認証方法である場合は--profile saml、 を追加する必要がある場合もあります。

LINUX の前提条件

WIGS RFC を送信する前に、 に記載されている要件を確認し、以下ワークロードの移行: Linux と Windows の前提条件を確認してください。

- 最新の拡張ネットワーキングドライバーがインストールされています。「Linux での拡張ネットワーキング」を参照してください。
- AMS コンポーネントと競合するサードパーティーのソフトウェアコンポーネントは削除されました。
  - ウイルス対策クライアント
  - バックアップクライアント
  - 仮想化ソフトウェア (VM Tools や Hyper-V 統合サービスなど)
  - アクセス管理ソフトウェア (SSSD、Centrify、PBIS など)
- SSH が正しく設定されていることを確認します。これにより、SSH のプライベートキー認証が一時的に有効になります。AMS は、設定管理ツールでこれを使用します。次のコマンドを使用します。

```
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
```

- Yum が正しく設定されていることを確認します。RedHat では、Yum リポジトリを使用するためのライセンスが必要です。インスタンスは Satellite Server または RedHat Cloud Server を介してライセンスされる必要があります。ライセンスが必要な場合は、次のいずれかのリンクを使用します。
  - Red Hat 衛星
  - Red Hat クラウドアクセス
- Red Hat Satellite を使用する場合、WIGS には Red Hat Software Collections (RHSCL) の追加が必要です。WIGS システムは RHSCL を使用して、システムで設定されているものとともに Python3.6 インタープリタを追加します。このソリューションをサポートするには、次のリポジトリが使用可能である必要があります。
  - rhel-server-rhscl
  - rhel-server-releases-optional

Windows の前提条件

WIGS RFC を送信する前に、 に記載されている要件を確認し、以下ワークロードの移行: Linux と Windows の前提条件を確認してください。

- Powershell バージョン 3 以降がインストールされている。
- AWS EC2 Config は、移行するワークロードを持つインスタンスにインストールされます。
- PV、ENA、NVMe などの最新世代のインスタンスタイプをサポートする AWS ドライバーをインストールします。これらのリンクで情報を使用できます。
  - Windows インスタンスでの PV ドライバーのアップグレード
  - Windows での拡張ネットワーキング
  - Windows インスタンス用の AWS NVMe ドライバー
  - パート 3: AWS NVMe ドライバーのアップグレード

  - パート 5: ベアメタルインスタンス用のシリアルポートドライバーのインストール
  - パート 6: 電源管理設定の更新
- （オプションですが推奨) 重要なサービスを無効にする – データベースなどの重要なアプリケーションサービスを無効にしますが、アプリケーション検証段階で元の起動モードに戻すことができるように、変更が文書化されていることを確認します。
- （オプションですが推奨) 準備済みインスタンスからフェイルセーフ AMI を作成します。
  - デプロイの使用 | 高度なスタックコンポーネント | AMI | 作成
  - 作成時に、タグ Key=Name、Value=APPLICATION-ID_IngestReady を追加します。
  - AMI が作成されるまで待ってから次に進みます。
- AMS コンポーネントと競合するサードパーティーのソフトウェアコンポーネントは削除されました。
  - ウイルス対策クライアント
  - バックアップクライアント
  - 仮想化ソフトウェア (VM Tools や Hyper-V 統合サービスなど）

> **Note**
>
> Windows Server End-of-Support移行プログラム (EMP) には、Windows Server 2003、2008、2008 R2 から AWS でサポートされている新しいバージョンにレガシーアプリケーションをリファクタリングなしで移行するためのツールが含まれています。

移行でリソースを変更する方法

このセクションで説明する取り込み RFC は、インスタンスが AMS アカウントに移行されたら、AMS がインスタンスを管理できるように、インスタンスに設定を追加する次のステップを実行します。

追加された設定は、次のように AMS 固有です。

取り込まれた Linux インスタンスに加えられた変更：

- インストールされているソフトウェア：
  - Cloud Init: Jarvis Access のプライベートキーを設定するために使用されます。

  - サポートされているすべてのオペレーティングシステムの Python 3 (スクリプト言語) (CentOS 6、RHEL 8、OracleLinux 7 を除く) 。
  - AWS CloudFormation Python ヘルパースクリプト: AWS CloudFormation は、Amazon EC2 インスタンスにソフトウェアをインストールしてサービスを開始するために使用されるスクリプトを提供します。
  - AWS CLI: AWS CLI は、AWS サービスとやり取りするためのコマンドを提供する AWS SDK for Python (Boto) 上に構築されたオープンソースツールです。
  - AWS SSM エージェント: SSM エージェントは Systems Manager サービスからのリクエストを処理し、リクエストで指定されたとおりにマシンを設定します。
  - AWS CloudWatch Logs エージェント: CloudWatch にログを送信します。
  - AWS CodeDeploy: Amazon EC2 インスタンス、オンプレミスインスタンス、またはサーバーレス Lambda 関数へのアプリケーションのデプロイを自動化するデプロイサービス。
  - Ruby: CodeDeploy に必須
  - システムパフォーマンスツール (sysstat): Sysstat には、システムパフォーマンスと使用状況を監視するためのさまざまなユーティリティが含まれています。
  - AD Bridge (旧 PowerBroker Identity Services): Microsoft 以外のホストを Active Directory ドメインに結合します。
  - Trend Micro Deep Security Agent: ウイルス対策ソフトウェア。
- 変更されるソフトウェア :
  - インスタンスは UTC タイムゾーンを使用するように設定されています。

取り込まれた Windows インスタンスに加えられた変更 :

- インストールされているソフトウェア :
  - AWS Tools for Windows PowerShell: AWS Tools for PowerShell を使用すると、開発者と管理者は PowerShell スクリプト環境で AWS のサービスとリソースを管理できます。
  - Trend Micro Deep Security Agent: ウイルス対策保護
  - Boot、Active Directory Join、Monitoring、PowerShell、Logging を制御する PowerShell コードを含む AMS PowerShell モジュール。
- 変更されるソフトウェア :
  - サーバーメッセージブロック (SMB) バージョン 1 は無効になっています。
  - Windows リモート管理 (WinRM) が有効で、ポート 5986 でリッスンするように設定されています。このインバウンドポートを許可するファイアウォールルールも作成されます。

- インストールまたは変更される可能性のあるソフトウェア：
  - Microsoft .Net Framework 4.5 (開発者プラットフォーム) 。.Net Framework 4.5 より前のバージョンが検出された場合。
  - Windows 2012、広告 Windows 2012R2 の場合、PowerShell 5.1 にアップグレードします。

ワークロードの移行：標準プロセス

> **Note**
>
> このプロセスには 2 つの関係者が必要なため、このセクションでは、AMS クラウド移行パートナー (移行パートナー) とアプリケーション所有者 (ユーザー) の各タスクについて説明します。

1. 移行パートナー、セットアップ：
   a. 移行パートナーは、インスタンスを移行する目的で IAM ロールのサービスリクエストを AMS に送信します。サービスリクエストの送信の詳細については、「サービスリクエストの例」を参照してください。
   b. 移行パートナーは管理者アクセスリクエストを送信します。AMS オペレーションチームは、リクエストされた IAM ロールを通じて移行パートナーにアカウントへのアクセスを提供します。
2. 移行パートナー、個々のワークロードの移行：
   a. 移行パートナーは、IAM AWS インスタンスプロファイル (アカウントに存在する必要があります) を使用して、ネイティブ Amazon EC2 またはその他の移行ツールを通じて、非インスタンスを AMS `customer-mc-ec2-instance-profile` アカウントのサブネットに移行します。
   b. 移行パートナーは、移行パートナー移行インスタンスからのデプロイ | 取り込み | スタック | CT の作成 (ct-257p9zjk14ija) を使用して RFC を送信します。この RFC の作成と送信の詳細については、「」を参照してくださいワークロード取り込みスタック: の作成。

      RFC の実行出力は、インスタンス ID、IP アドレス、AMI ID を返します。

移行パートナーは、アカウントで作成されたワークロードのインスタンス ID を提供します。

3. 移行にアクセスして検証します。

   a. 移行パートナーから提供された実行出力 (AMI ID、インスタンス ID、IP アドレス) を使用して、アクセス RFC を送信し、新しく作成した AMS スタックにログインして、アプリケーションが正しく動作していることを確認します。詳細については、「インスタンスアクセスのリクエスト」を参照してください。

   b. 問題がなければ、起動したインスタンスを 1 層スタックとして引き続き使用したり、AMI を使用して Auto Scaling グループを含む追加のスタックを作成したりできます。

   c. 移行に満足しない場合は、サービスリクエストを提出し、スタックと RFC IDs を参照します。AMS はお客様と協力して問題に対処します。

CloudEndure ランディングゾーンのワークロード取り込みプロセスを次に示します。

ワークロードの移行: CloudEndure ランディングゾーン (SALZ)

このセクションでは、ワークロード取り込み (WIGS) RFC で使用できるように CloudEndure (CE) カットオーバーインスタンスの中間移行シングルアカウントランディングゾーン (SALZ) を設定する方法について説明します。

CloudEndure の詳細については、CloudEndure Migration」を参照してください。

> **Note**
>
> これは、事前定義されたセキュリティ強化移行 LZ とパターンです。

前提条件:

- 顧客 AMS アカウント
- AMS アカウントとお客様のオンプレミス間のネットワークとアクセスの統合
- CloudEndure アカウント

- AMS セキュリティのレビューとサインオフの事前承認ワークフローは、CA や CSDM で実行されます (IAM ユーザーの永続的な認証情報を悪用すると、インスタンスとセキュリティグループを作成/削除できます) 。

> **Note**
>
> 具体的な準備と移行プロセスについては、このセクションで説明します。

準備: ユーザーと AMS 演算子 :

1. Management | Other | Other | Update change type to AMS for the following resources and updates を使用して、変更リクエスト (RFC) を準備します。別のその他 | その他の更新 RFCsまたは 1 つを送信できます。その RFC/CT の詳細については、「その他 | これらのリクエストに関するその他の更新」を参照してください。

   a. AMS VPC にセカンダリ CIDR ブロックを割り当てます。一時的な CIDR ブロックは、移行の完了後に削除されます。ブロックがオンプレミスネットワークに戻る既存のルートと競合しないことを確認します。たとえば、AMS VPC CIDR が 10.0.0.0/16 で、オンプレミスのネットワード 10.1.0.0/16 に戻るルートがある場合、一時セカンダリ CIDR は 10.255.255.0/24 になります。AWS CIDR ブロックの詳細については、「VPC とサブネットのサイズ設定」を参照してください。

   b. 初期ガーデン AMS VPC 内に新しいプライベートサブネットを作成します。名前の例: `migration-temp-subnet`。

   c. ローカル VPC および NAT (インターネット) ルートのみを使用してサブネットの新しいルートテーブルを作成し、インスタンスのカットオーバー時や停止時のソースサーバーとの競合を回避します。インターネットへのアウトバウンドトラフィックがパッチダウンロードを許可され、AMS WIGS の前提条件をダウンロードしてインストールできることを確認します。

   d. Managed AD セキュリティグループを更新して、 との間のインバウンドトラフィックとアウトバウンドトラフィックを許可します`migration-temp-subnet`。また、EPS ロードバランサー (ELB) セキュリティグループ (例: `mc-eps-McEpsElbPrivateSecurityGroup-M790XBZEEX74`) を更新して、新しいプライベートサブネット (例: `migration-temp-subnet`) を許可するようにリクエストします。専用 CloudEndure (CE) サブネットからのト

ラフィックが 3 つの TCP ポートすべてで許可されていない場合、WIGS の取り込みは失敗します。

e. 最後に、新しい CloudEndure IAM ポリシーと IAM ユーザーをリクエストします。ポリシーには正しいアカウント番号が必要であり、`RunInstances`ステートメント内のサブネット IDs は <Customer Application Subnet (s) + Temp Migration Subnet> である必要があります。

AMS の事前承認された IAM CloudEndure ポリシーを表示するには: WIGS Cloud Endure ランディングゾーンの例ファイルを解凍し、 を開きます`customer_cloud_endure_policy.json`。

> **Note**
>
> より寛容なポリシーが必要な場合は、CloudArchitect/CSDM で必要なものについて話し合い、必要に応じて、ポリシーを実装する RFC を送信する前に AMS セキュリティレビューとサインオフを取得します。

2. AMS ワークロード取り込みに CloudEndure を使用する準備手順が完了し、移行パートナーが準備手順を完了すると、移行を実行する準備が整います。WIGS RFC は移行パートナーによって送信されます。

> **Note**
>
> IAM ユーザーキーは直接共有されませんが、画面共有セッションで AMS オペレーターが CloudEndure マネジメントコンソールに入力する必要があります。

準備: 移行パートナーと AMS オペレーター :

1. CloudEndure 移行プロジェクトを作成します。

   a. プロジェクトの作成中に、画面共有セッションで AMS タイプイン IAM ユーザー認証情報を取得します。

   b. レプリケーション設定 -> レプリケーションサーバーを起動するサブネットを選択し、customer-application-x サブネットを選択します。

c. レプリケーション設定 -> レプリケーションサーバーに適用するセキュリティグループを選択し、Sentinel セキュリティグループ (プライベートのみと EgressAll) の両方を選択します。

2. マシン (インスタンス) のカットオーバーオプションを定義します。

a. サブネット: migration-temp-subnet。

b. セキュリティグループ: 「Sentinel」セキュリティグループ (プライベートのみと EgressAll) の両方。

カットオーバーインスタンスは、AMS Managed AD および AWS パブリックエンドポイントと通信できる必要があります。

c. Elastic IP: なし

d. パブリック IP: なし

e. IAM ロール: customer-mc-ec2-instance-profile

IAM ロールは SSM 通信を許可する必要があります。AMS のデフォルトを使用することをお勧めします。

f. 規則に従ってタグを設定します。

移行: 移行パートナー :

1. AMS でダミースタックを作成します。スタック ID を使用して踏み台にアクセスします。
2. ソースサーバーに CloudEndure (CE) エージェントをインストールします。詳細については、「 エージェントのインストール」を参照してください。
3. ソースサーバーにローカル管理者認証情報を作成します。
4. 短いカットオーバーウィンドウをスケジュールし、準備ができたらカットオーバーをクリックします。これにより、移行が確定され、ユーザーはターゲット AWS リージョンにリダイレクトされます。
5. リクエストスタック ダミースタックへの管理者アクセス。「管理者アクセスリクエスト」を参照してください。
6. 踏み台にログインし、作成したローカル管理者認証情報を使用してカットオーバーインスタンスにログインします。
7. failafe AMI を作成します。AMIs「AMI の作成」を参照してください。
8. インスタンスを取り込み用に準備します。「」を参照してくださいワークロードの移行: Linux と Windows の前提条件。

9. インスタンスに対して WIGS RFC を実行します。「」を参照してくださいワークロード取り込みスタック: の作成。

AMS Tools アカウント (ワークロードの移行)

マルチアカウントランディングゾーンツールアカウント (VPC を使用) は、移行作業の迅速化、セキュリティポジションの向上、コストと複雑さの軽減、使用パターンの標準化に役立ちます。

ツールアカウントには以下が用意されています。

- 本番ワークロード外のシステムインテグレーターのレプリケーションインスタンスにアクセスするための明確に定義された境界。
- 分離されたチェンバーを作成して、他のワークロードのアカウントに配置する前に、ワークロードにマルウェアや不明なネットワークルートがないか確認します。
- 定義されたアカウント設定として、ワークロードの移行のためのオンボーディングとセットアップにかかる時間を短縮します。
- 隔離されたネットワークルートは、オンプレミス -> CloudEndure -> Tools account -> AMS 取り込みイメージからのトラフィックを保護します。イメージが取り込まれたら、AMS 管理 | 高度なスタックコンポーネント | AMI | 共有 (ct-1eiczxw8ihc18) RFC を介してイメージを送信先アカウントと共有できます。

高レベルのアーキテクチャ図：

Deployment | Managed Landing Zone | Management Account | Create tools account (with VPC) change type (ct-2j7q1hgf26x5c) を使用して、ツールアカウントをすばやくデプロイし、マルチアカウント Landing Zone 環境内でワークロード取り込みプロセスをインスタンス化します。「管理アカウント」、「ツールアカウント: 作成 (VPC を使用)」を参照してください。

> **Note**
>
> これは移行ハブであるため、2 つのアベイラビリティーゾーン (AZs) を使用することをお勧めします。
> デフォルトでは、AMS はすべてのアカウントに次の 2 つのセキュリティグループ (SGs) を作成します。これら 2 つの SGs が存在することを確認します。存在しない場合は、AMS チームで新しいサービスリクエストを開いてリクエストしてください。

- SentinelDefaultSecurityGroupPrivateOnlyEgressAll
- InitialGarden-SentinelDefaultSecurityGroupPrivateOnly

CloudEndure レプリケーションインスタンスが、オンプレミスに戻るルートがあるプライベートサブネットに作成されていることを確認します。プライベートサブネットのルートテーブルに TGW に戻るデフォルトルートがあることを確認することができます。ただし、CloudEndure マシンのカットオーバーを実行すると、オンプレミスに戻るルートがなく、インターネットアウトバウンドトラフィックのみが許可される「分離された」プライベートサブネットに入る必要があります。オンプレミスリソースへの潜在的な問題を回避するために、分離されたサブネットでカットオーバーが発生するようにすることが重要です。

前提条件:

1. Plus または Premium のサポートレベル。
2. AMIs がデプロイされる KMS キーのアプリケーションアカウント IDs。
3. 前述のように作成されたツールアカウント。

AWS アプリケーション移行サービス (AWS MGN)

AWS Application Migration Service (AWS MGN) は、ツールアカウントのプロビジョニング中に自動的に作成される IAM `AWSManagedServicesMigrationRole` ロールを介して MALZ Tools アカウントで使用できます。 AWS MGN を使用して、サポートされているバージョンの Windows および Linux オペレーティングシステムで実行されるアプリケーションとデータベースを移行できます。

AWS リージョン サポートに関するup-to-dateについては、AWS 「 リージョンサービスリスト」を参照してください。

希望する AWS リージョン が現在 MGN AWS でサポートされていない場合、またはアプリケーションが実行されているオペレーティングシステムが現在 MGN AWS でサポートされていない場合は、代わりにツールアカウントで CloudEndure Migration を使用することを検討してください。

MGN AWS 初期化のリクエスト

AWS MGN は、初めて使用する前に AMS によって初期化する必要があります。新しい Tools アカウントに対してこれをリクエストするには、ツールアカウントから 管理 | その他 | その他の RFC を以下の詳細とともに送信します。

```
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.
```

AMS が RFC を正常に完了し、ツールアカウントで AWS MGN を初期化したら、AWSManagedServicesMigrationRoleを使用して要件のデフォルトテンプレートを編集できます。

新しい AMS Tools アカウントへのアクセスを有効にする

ツールアカウントが作成されると、AMS はアカウント ID を提供します。次のステップでは、新しいアカウントへのアクセスを設定します。以下の手順に従ってください。

1. 適切な Active Directory グループを適切なアカウント IDs。

   新しい AMS 作成アカウントは、ReadOnly ロールポリシーと、ユーザーが RFCs。

   Tools アカウントには、追加の IAM ロールとユーザーが利用できます。

   - IAM ロール: AWSManagedServicesMigrationRole
   - IAM ユーザー: customer_cloud_endure_user

2. サービス統合チームのメンバーが次のレベルのツールを設定できるように、ポリシーとロールをリクエストします。

   AMS コンソールに移動し、次の RFCs。

   a. KMS キーを作成します。KMS キーの作成 (自動) または KMS キーの作成 (レビューが必要) を使用します。

      KMS を使用して取り込まれたリソースを暗号化する場合、他のマルチアカウントランディングゾーンアプリケーションアカウントと共有されている単一の KMS キーを使用すると、 は取り込まれたイメージを宛先アカウントで復号化できる場所にセキュリティを提供します。

   b. KMS キーを共有します。

      管理 | 高度なスタックコンポーネント | KMS キー | 共有 (レビューが必要) 変更タイプ (ct-05yb337abq3x5) を使用して、新しい KMS キーが取り込まれた AMIs が存在するアプリケーションアカウントと共有されるようにリクエストします。

最終的なアカウント設定の例：

AMS の事前承認された IAM CloudEndure ポリシーの例

AMS の事前承認された IAM CloudEndure ポリシーを表示するには: WIGS Cloud Endure ランディングゾーンの例ファイルを解凍し、 を開きますcustomer_cloud_endure_policy.json。

AMS Tools アカウント接続とend-to-endテスト

1. まず、CloudEndure を設定し、AMS にレプリケートするサーバーに CloudEndure エージェントをインストールします。
2. CloudEndure でプロジェクトを作成します。
3. シークレットマネージャーを使用して、前提条件を実行したときに共有された AWS 認証情報を入力します。
4. レプリケーション設定：
   a. レプリケーションサーバーに適用するセキュリティグループを選択するオプションで、両方の AMS "Sentinel" セキュリティグループ (プライベートのみと EgressAll) を選択します。
   b. マシン (インスタンス) のカットオーバーオプションを定義します。詳細については、「ステップ 5」を参照してください。カットオーバー
   c. サブネット: プライベートサブネット。
5. セキュリティグループ：
   a. AMS "Sentinel" セキュリティグループ (プライベートのみと EgressAll) の両方を選択します。
   b. カットオーバーインスタンスは、AMS 管理の Active Directory (MAD) と AWS パブリックエンドポイントと通信する必要があります。
      i. Elastic IP: なし
      ii. パブリック IP: なし
      iii. IAM ロール: customer-mc-ec2-instance-profile
   c. 内部タグ付け規則に従ってタグを設定します。
6. CloudEndure エージェントをマシンにインストールし、EC2 コンソールの AMS アカウントに表示されるレプリケーションインスタンスを探します。

AMS 取り込みプロセス：

AMS Tools アカウントのハイジーン

アカウントで AMI を共有し、レプリケートされたインスタンスが不要になったら、クリーンアップする必要があります。

- インスタンス後の WIGs取り込み :
  - カットオーバーインスタンス: 少なくとも、作業が完了したら、AWS コンソールを介してこのインスタンスを停止または終了します。
  - 取り込み前 AMI バックアップ: インスタンスが取り込まれ、オンプレミスインスタンスが終了したら削除する
  - AMS で取り込まれたインスタンス: スタックをオフにするか、AMI を共有したら終了します。
  - AMS で取り込まれた AMIs: 送信先アカウントとの共有が完了したら削除する
- 移行のクリーンアップの終了: デベロッパーモードでデプロイされたリソースを文書化して、クリーンアップが定期的に行われるようにします。次に例を示します。
  - セキュリティグループ
  - クラウド形式を使用して作成されたリソース
  - ネットワーク ACK
  - サブネット
  - VPC
  - ルートテーブル
  - ロール
  - ユーザーとアカウント

大規模な移行 - Migration Factory

「AWS CloudEndure Migration Factory ソリューションの紹介」を参照してください。

ワークロードの移行: Linux の取り込み前検証

インスタンスが AMS アカウントに取り込む準備ができていることを検証できます。ワークロード取り込み (WIGS) の取り込み前検証では、オペレーティングシステムのタイプ、使用可能なディスク容量、競合するサードパーティーソフトウェアの存在などのチェックを実行します。WIGS 取り込み前検証を実行すると、オプションのログファイルを含む画面上のテーブルが生成されます。結果には、各検証チェックの合格/不合格ステータスと、不合格の理由が表示されます。さらに、ニーズに合わせて検証テストをカスタマイズできます。

よくある質問：

- Linux WIGS の取り込み前検証を使用する方法

  AMS Linux WIGS 取り込み前検証スクリプトをダウンロードして使用するには、次の手順に従います。

  1. 検証スクリプトを含む ZIP ファイルをダウンロードする

     Linux WIGS 取り込み前検証 zip ファイル。

  2. アタッチされたルールを任意のディレクトリに解凍します。
  3. readme.md ファイルの指示に従います。

- Linux WIGS の取り込み前検証では、どのような検証が行われますか？

  AMS Linux WIGS 取り込み前検証ソリューションは、以下を検証します。

  1. ブートボリュームには少なくとも 5 GB の空きがあります。
  2. オペレーティングシステムは AMS でサポートされています。
  3. インスタンスには特定のインスタンスプロファイルがあります。
  4. インスタンスには、ウイルス対策ソフトウェアまたは仮想化ソフトウェアは含まれていません。
  5. SSH が正しく設定されています。
  6. インスタンスは Yum リポジトリにアクセスできます。
  7. 拡張ネットワーキングドライバーがインストールされています。
  8. インスタンスには SSM エージェントがあり、実行中です。

- カスタム設定ファイルがサポートされるのはなぜですか？

  スクリプトは、オンプレミスの物理サーバーと AWS EC2 インスタンスの両方で実行されるように設計されています。ただし、上記のリストに示すように、一部のテストはオンプレミスで実行されると失敗します。たとえば、データセンターの物理サーバーにはインスタンスプロファイルがありません。このような場合は、混乱を避けるために、設定ファイルを編集してインスタンスプロファイルテストをスキップできます。

- スクリプトの最新バージョンがあることを確認するにはどうすればよいですか？

  Linux WIGS 取り込み前検証ソリューションのup-to-dateは、メインドキュメントページの AMS ヘルパーファイルセクションで入手できます。

- スクリプトは読み取り専用ですか？

スクリプトは、生成するログファイルを除き読み取り専用であるように設計されていますが、スクリプトを非本番環境で実行するにはベストプラクティスに従う必要があります。

- WIGS の取り込み前検証は Windows で利用できますか？

  はい。これは、メインのドキュメントページの AMS ヘルパーファイルセクションで利用できます。

ワークロードの移行: Windows の取り込み前検証

WIGs 前検証スクリプトを使用して、インスタンスが AMS アカウントに取り込む準備ができていることを検証できます。ワークロード取り込み (WIGS) の取り込み前検証は、オペレーティングシステムのタイプ、使用可能なディスク容量、競合するサードパーティーソフトウェアの存在などのチェックを実行します。実行すると、WIGS 取り込み前検証によって画面上のテーブルとオプションのログファイルが生成されます。結果には、各検証チェックの合格/不合格ステータスと失敗の理由が表示されます。さらに、検証テストをカスタマイズすることもできます。

よくある質問：

- Windows WIGS の取り込み前検証を使用する方法

  GUI とウェブブラウザから検証を実行することも、Windows PowerShell、SSM Run Command、または SSM Session Manager を使用することもできます。

  オプション 1: GUI とウェブブラウザから を実行する

  GUI とウェブブラウザから Windows WIGs 前検証を実行するには、次の手順を実行します。

  1. 検証スクリプトを含む ZIP ファイルをダウンロードします。

     Windows WIGS 取り込み前検証 ZIP ファイル。

  2. アタッチされたルールを任意のディレクトリに解凍します。
  3. README.md ファイルの指示に従います。

  オプション 2: Windows PowerShell、SSM Run Command、または SSM Session Manager から を実行する

  Windows 2016 以降

  1. 検証スクリプトを含む ZIP ファイルをダウンロードします。

```
$DestinationFile = "$env:TEMP\WIGValidation.zip"

$Bucket = 'https://docs.aws.amazon.com/managedservices/latest/appguide/samples/
windows-prewigs-validation.zip'
$DestinationFile = "$env:TEMP\WIGValidation.zip"
$ScriptFolder = "$env:TEMP\AWSManagedServices.PreWigs.Validation"
```

2. から既存のファイルを削除しますC:\Users\AppData\Local\Temp \AWSManagedServices.PreWigs.Validation。

```
Remove-Item $scriptFolder -Recurse -Force -ErrorAction Ignore
```

3. スクリプトを呼び出します。

```
Invoke-WebRequest -Uri $bucket -OutFile $DestinationFile
Add-Type -Assembly "system.io.compression.filesystem"
```

4. アタッチされたファイルを任意のディレクトリに解凍します。

```
[io.compression.zipfile]::ExtractToDirectory($DestinationFile, $env:TEMP)
```

5. 検証スクリプトをインタラクティブに実行し、結果を表示します。

```
Import-Module .\AWSManagedServices.PreWigs.Validation.psm1 -force
Invoke-PreWIGsValidation -RunWithoutExitCodes
```

6. (オプション) Exit Codes セクションにリストされているエラーコードをキャプチャするには、RunWithoutExitCodesオプションを指定せずにスクリプトを実行します。このコマンドはアクティブな PowerShell セッションを終了することに注意してください。

```
Import-Module .\AWSManagedServices.PreWigs.Validation.psm1 -force
Invoke-PreWIGsValidation
```

Windows 2012 R2 以前

Windows Server 2012R2 以下を実行している場合は、zip ファイルをダウンロードする前に TLS を設定する必要があります。TLS を設定するには、次の手順を実行します。

1. 検証スクリプトを含む ZIP ファイルをダウンロードします。

```
$DestinationFile = "$env:TEMP\WIGValidation.zip"
```

```
$Bucket = 'https://docs.aws.amazon.com/managedservices/latest/appguide/samples/
windows-prewigs-validation.zip'
$DestinationFile = "$env:TEMP\WIGValidation.zip"
$ScriptFolder = "$env:TEMP\AWSManagedServices.PreWigs.Validation"
```

2. 既存の検証ファイルがある場合は、削除します。

```
Remove-Item $scriptFolder -Recurse -Force -ErrorAction Ignore
```

3. TLS バージョンを設定します。

```
[System.Net.ServicePointManager]::SecurityProtocol = 'TLS12'
```

4. WIG 検証をダウンロードします。

```
Invoke-WebRequest -Uri $bucket -OutFile $DestinationFile
Add-Type -Assembly "system.io.compression.filesystem"
```

5. アタッチされたルールを任意のディレクトリに解凍します。

```
[io.compression.zipfile]::ExtractToDirectory($DestinationFile, $env:TEMP)
```

6. 検証スクリプトをインタラクティブに実行し、結果を表示します。

```
Import-Module .\AWSManagedServices.PreWigs.Validation.psm1 -force
Invoke-PreWIGsValidation -RunWithoutExitCodes
```

7. (オプション) Exit Codes セクションにリストされているエラーコードをキャプチャするには、RunWithoutExitCodes オプションを使用せずにスクリプトを実行します。このコマンドはアクティブな PowerShell セッションを終了することに注意してください。

```
Import-Module .\AWSManagedServices.PreWigs.Validation.psm1 -force
Invoke-PreWIGsValidation
```

> **Note**
>
> PowerShell スクリプトをダウンロードして実行できます。これを行うには、pre-wigs-validation-powershell-scripts.zip をダウンロードします。

- Windows WIGS 取り込み前検証ではどのような検証が実行されますか？

AMS Windows WIGS 取り込み前検証ソリューションは、以下を検証します。

1. ブートボリュームには少なくとも 10 GB の空きがあります。
2. オペレーティングシステムは AMS でサポートされています。
3. インスタンスには特定のインスタンスプロファイルがあります。
4. インスタンスには、ウイルス対策ソフトウェアや仮想化ソフトウェアは含まれていません。
5. DHCP は、少なくとも 1 つのネットワークアダプタで有効になっています。
6. インスタンスは Sysprep の準備ができました。
   - 2008 R2 および 2012 Base および R2 の場合、Sysprep は以下を検証します。
     - unattend.xml ファイルがある
     - sppnp.dll ファイル (存在する場合) が破損していない
     - オペレーティングシステムがアップグレードされていません
     - Sysprep が Microsoft ガイドラインの最大回数を超えて実行されていない
   - 2016 年以降では、上記のチェックはすべてスキップされ、その OS に問題は発生しません。
7. Windows 管理計測 (WMI) サブシステムは正常です。
8. 必要なドライバーがインストールされています。
9. SSM エージェント と がインストールされ、実行されています。
10. RDS ライセンス設定が原因でマシンが猶予期間にあるかどうかを確認する警告が表示されます。
11. 必要なレジストリキーが正しく設定されている。詳細については、取り込み前検証 zip ファイルの README を参照してください。

- カスタム設定ファイルがサポートされるのはなぜですか？

  スクリプトは、オンプレミスの物理サーバーと AWS EC2 インスタンスの両方で実行されるように設計されています。ただし、上記のリストに示すように、一部のテストはオンプレミスで実行したときに失敗します。たとえば、データセンターの物理サーバーにはインスタンスプロファイルがありません。このような場合は、混乱を避けるために、設定ファイルを編集してインスタンスプロファイルテストをスキップできます。

- スクリプトの最新バージョンがあることを確認するにはどうすればよいですか？

  Windows WIGS の取り込み前検証ソリューションのup-to-dateは、メインドキュメントページの AMS ヘルパーファイルセクションで入手できます。

- スクリプトは読み取り専用ですか？

スクリプトは、生成するログファイルを除き読み取り専用であるように設計されていますが、スクリプトを非本番環境で実行するにはベストプラクティスに従う必要があります。

- WIGS 取り込み前検証は Linux で利用できますか？

  はい。2019 年 10 月 31 日にリリースされた Linux バージョン。これは、メインのドキュメントページの AMS ヘルパーファイルセクションで利用できます。

ワークロード取り込みスタック: の作成

コンソールを使用したインスタンスの AMS スタックへの移行

AMS コンソールでのこの変更タイプのスクリーンショット：

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプエリアを使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。
   - カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。
3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。

4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters 領域を開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

> **Note**
>
> RFC が拒否された場合、実行出力には Amazon CloudWatch logsへのリンクが含まれます。AMS ワークロード取り込み (WIGS) RFCs は、要件が満たされていない場合、例えば、インスタンスでウイルス対策ソフトウェアが検出された場合などに拒否されます。CloudWatch ログには、失敗した要件と修復のために実行するアクションに関する情報が含まれます。

CLI を使用してインスタンスを AMS スタックに移行する

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成します。1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id` *`ID`* コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id` *`ID`* コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたとき

に通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

AMS CLI を使用して、AMS アカウントに移行された非 AMS インスタンスから AMS インスタンスを作成できます。

> **Note**
>
> 前提条件に従っていることを確認してください。「ワークロードの移行: Linux と Windows の前提条件」を参照してください。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符を エスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
aws amscm create-rfc --change-type-id "ct-257p9zjk14ija" --change-type-version "2.0" --
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\",\"EnforceIMDSV2\":\"false\"}"
```

テンプレートの作成：

1. 0この変更タイプの実行パラメータ JSON スキーマをファイルに入力します。例の名前は MigrateStackParams.json:

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

2. 実行パラメータ JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
"InstanceId":          "MIGRATED_INSTANCE_ID",
"TargetVpcId":         "VPC_ID",
"TargetSubnetId":      "SUBNET_ID",
"Name":                "Migrated-Stack",
"Description":         "Create-Migrated-Stack",
"EnforceIMDSV2":       "false"
}
```

3. RFC テンプレート JSON ファイルを出力します。MigrateStackRfc.json:

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

4. MigrateStackRfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
"ChangeTypeId":         "ct-257p9zjk14ija",
"ChangeTypeVersion":    "2.0",
"Title":                "Migrate-Stack-RFC"
}
```

5. MigrateStackRfc ファイルと MigrateStackParams ファイルを指定して、RFC を作成します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

新しいインスタンスは、関連する VPC のアプリケーション所有者のアカウントのインスタンスリストに表示されます。

6. RFC が正常に完了したら、アプリケーション所有者に通知して、新しいインスタンスにログインし、ワークロードが動作していることを確認します。

> **Note**
>
> RFC が拒否された場合、実行出力には Amazon CloudWatch logsへのリンクが含まれます。AMS ワークロード取り込み (WIGS) RFCs は、要件が満たされていない場合、例えば、インスタンスでウイルス対策ソフトウェアが検出された場合などに拒否されます。CloudWatch ログには、失敗した要件と修復のために実行するアクションに関する情報が含まれます。

ヒント

> **Note**
>
> 前提条件に従っていることを確認してください。「ワークロードの移行: Linux と Windows の前提条件」を参照してください。

> **Note**
>
> 移行するインスタンスのタグに RFC で指定されたタグと同じキーがある場合、RFC は失敗します。

> **Note**
>
> 最大 4 つのターゲット IDs、ポート、アベイラビリティーゾーンを指定できます。

> **Note**
>
> RFC が拒否された場合、実行出力には Amazon CloudWatch logsへのリンクが含まれます。AMS ワークロード取り込み (WIGS) RFCs は、要件が満たされていない場合、例えば、インスタンスでウイルス対策ソフトウェアが検出された場合などに拒否されます。CloudWatch ログには、失敗した要件と修復のために実行するアクションに関する情報が含まれます。

> **Note**
>
> RFC が拒否された場合、実行出力には Amazon CloudWatch logsへのリンクが含まれます。AMS ワークロード取り込み (WIGS) RFCs は、要件が満たされていない場合、例えば、インスタンスでウイルス対策ソフトウェアが検出された場合などに拒否されます。CloudWatch ログには、失敗した要件と修復のために実行するアクションに関する情報が含まれます。

必要に応じて、「ワークロード取り込み (WIGS) 失敗」を参照してください。

AMS CloudFormation の取り込み

AMS AWS CloudFormation 取り込み変更タイプ (CT) を使用すると、いくつかの変更を加えて既存の CloudFormation テンプレートを使用して、AMS マネージド VPC にカスタムスタックをデプロイできます。

トピック

- CloudFormation 取り込みガイドライン、ベストプラクティス、制限事項
- CloudFormation 取り込み: 例
- CloudFormation 取り込みスタックを作成する
- CloudFormation 取り込みスタックを更新する
- CloudFormation 取り込みスタックの変更セットを承認する
- CloudFormation スタック終了保護を更新する
- AMS で CFN 取り込みまたはスタック更新 CTs を使用した自動 IAM デプロイ

AMS CloudFormation 取り込みプロセスには、以下が含まれます。

- カスタム CloudFormation テンプレートを準備して S3 バケットにアップロードするか、RFC の作成時にテンプレートをインラインで指定します。署名付き URL で S3 バケットを使用している場合。詳細については、「 署名」を参照してください。
- CloudFormation の取り込み変更タイプを RFC の AMS に送信します。CFN 取り込み変更タイプのチュートリアルについては、「」を参照してくださいCloudFormation 取り込みスタックを作成する。CFN 取り込みの例については、「」を参照してくださいCloudFormation 取り込み: 例。

- スタックが作成されたら、スタックを更新してドリフトを修正できます。さらに、更新が失敗した場合、明示的に更新を承認して実装できます。これらの手順はすべて、このセクションで説明します。

  CFN ドリフト検出の詳細については、「新しい – CloudFormation ドリフト検出」を参照してください。

> **Note**
>
> - この変更タイプにはバージョン 2.0 が追加されました。バージョン 2.0 は自動化されており、手動では実行されません。これにより、CT 実行をより迅速に実行できます。このバージョンでは、カスタム CloudFormation テンプレートを RFC に貼り付けることができる CloudFormationTemplate と、AMS マルチアカウントランディングゾーンで CloudFormation 取り込みを使用できるようにする Vpcld の 2 つの新しいパラメータが導入されました。 CloudFormation
> - バージョン 1.0 は手動変更タイプです。つまり、AMS 演算子は、変更タイプを正常に終了する前に何らかのアクションを実行する必要があります。少なくとも、レビューが必要です。このバージョンでは、CloudFormationTemplateS3Endpoint パラメータ値を署名付き URL にする必要があります。

CloudFormation 取り込みガイドライン、ベストプラクティス、制限事項

AMS が CloudFormation テンプレートを処理するには、いくつかのガイドラインと制限があります。

ガイドライン

CloudFormation 取り込み中の CloudFormation エラーを減らすには、次のガイドラインに従ってください。

- テンプレートに認証情報やその他の機密情報を埋め込まない – CloudFormation テンプレートは CloudFormation コンソールに表示されるため、認証情報や機密データをテンプレートに埋め込まないでください。テンプレートに機密情報を含めることはできません。次のリソースは、値に AWS Secrets Manager を使用する場合にのみ許可されます。
  - `AWS::RDS::DBInstance` - [MasterUserPassword,TdeCredentialPassword]
  - `AWS::RDS::DBCluster` - [MasterUserPassword]

- `AWS::ElastiCache::ReplicationGroup` - [AuthToken]

> **Note**
>
> リソースプロパティで AWS Secrets Manager シークレットを使用する方法については、AWS CloudFormation テンプレートを使用して AWS Secrets Manager で管理されるシークレットを作成および取得する方法」および「動的リファレンスを使用してテンプレート値を指定する方法」を参照してください。

- Amazon RDS スナップショットを使用して RDS DB インスタンスを作成する – これにより、MasterUserPassword を提供する必要がなくなります。
- 送信するテンプレートに IAM インスタンスプロファイルが含まれている場合は、「customer」というプレフィックスを付ける必要があります。たとえば、example-instance-profile」という名前のインスタンスプロファイルを使用すると、 は失敗します。代わりに、「customer-example-instance-profile」という名前のインスタンスプロファイルを使用します。
- - [UserData] に機密データを含めないでください**`AWS::EC2::Instance`**。UserData には、パスワード、API キー、またはその他の機密データを含めないでください。このタイプのデータは暗号化して S3 バケットに保存し、UserData を使用してインスタンスにダウンロードできます。
- CloudFormation テンプレートを使用した IAM ポリシーの作成は制約付きでサポートされています。IAM ポリシーは AMS SecOps によってレビューおよび承認される必要があります。現在、IAM ロールのデプロイは、事前承認されたアクセス許可を含むインラインポリシーでのみサポートされています。それ以外の場合、IAM ポリシーは AMS SecOps プロセスを上書きするため、CloudFormation テンプレートを使用して作成することはできません。
- SSH KeyPairs はサポートされていません。Amazon EC2 インスタンスには AMS アクセス管理システム経由でアクセスする必要があります。AMS RFC プロセスはユーザーを認証します。SSH キーペアを作成して AMS アクセス管理モデルを上書きするアクセス許可がないため、CloudFormation テンプレートに SSH キーペアを含めることはできません。
- セキュリティグループの進入ルールは制限されています – ソース CIDR 範囲を 0.0.0.0/0 またはパブリックにルーティング可能なアドレス空間に、80 または 443 以外の TCP ポートを含めることはできません。
- CloudFormation リソーステンプレートを記述するときの CloudFormation ガイドラインに従う – リソースの AWS CloudFormation ユーザーガイドを参照して、リソースに適切なデータ型/プロパティ名を使用していることを確認してください。たとえば、AWS::EC2::Instance リソースの SecurityGroupIds プロパティのデータ型は「文字列値のリスト」であるため、["sg-aaaaaaaa"] は OK (角括弧付き) ですが、"sg-aaaaaaaa" は (角括弧なし) ではありません。

詳細については、「AWS リソースタイプとプロパティタイプのリファレンス」を参照してください。

- AMS CloudFormation 取り込み CT で定義されたパラメータを使用するようにカスタム CloudFormation テンプレートを設定する – AMS CloudFormation 取り込み CT で定義されたパラメータを使用するように CloudFormation テンプレートを設定する場合、管理 | カスタムスタック | CloudFormation テンプレートからのCloudFormation テンプレートを再利用して同様のスタックを作成できます。例については、CloudFormation 取り込みの例: リソースの定義を参照してください。
- 署名付き URL を持つ Amazon S3 バケットエンドポイントは期限切れにできません – 署名付き URL を持つ Amazon S3 バケットエンドポイントを使用している場合は、署名付き Amazon S3 URL の有効期限が切れていないことを確認します。期限切れの署名付き Amazon S3 バケット URL で送信された CloudFormation 取り込み RFC は拒否されます。 Amazon S3
- Wait Condition にはシグナルロジックが必要です – Wait Condition は、スタックリソースの作成とスタックの作成の外部にある設定アクションを調整するために使用されます。テンプレートで Wait Condition リソースを使用する場合、 は成功シグナルを CloudFormation 待機し、成功シグナルの数が作成されない場合、スタックの作成を失敗としてマークします。Wait Condition リソースを使用する場合は、シグナルのロジックが必要です。詳細については、「 テンプレートでの待機条件の作成」を参照してください。

ベストプラクティス

AMS CloudFormation 取り込みプロセスを使用してリソースを移行するために使用できるベストプラクティスを以下に示します。

- IAM およびその他のポリシー関連のリソースを 1 つの CT で送信する – CloudFormation Ingest などの自動 CTs を使用して IAM ロールをデプロイできる場合は、そうすることをお勧めします。それ以外の場合は、すべての IAM または他のポリシー関連のリソースを収集し、単一の管理 | その他 | その他 | 変更タイプを作成する (ct-1e1xtak34nx76) で送信することをお勧めします。たとえば、必要なすべての IAM ロール、IAM Amazon EC2 インスタンスプロファイル、既存の IAM ロールの IAM ポリシー更新、Amazon S3 バケットポリシー、Amazon SNS/Amazon SQS ポリシーなどを組み合わせて、ct-1e1xtak34nx76 RFC を送信して、これらの既存のリソースを将来の CloudFormation 取り込みテンプレート内で参照できるようにします。
- EC2 インスタンスはブートストラップされ、ドメインに正常に結合されます。これはベストプラクティスとして自動的に行われます。CloudFormation 取り込みスタックを介して起動された Amazon EC2 インスタンスがブートストラップされ、ドメインに正常に参加するように、AMS に

はAuto Scaling グループリソースの CreationPolicy と UpdatePolicy が含まれます (つまり、これらのポリシーがまだ存在しない場合) 。

- Amazon RDS DB インスタンスパラメータを指定する必要があります – 取り込みを介して Amazon RDS データベースを作成するときは、以前の DB CloudFormation スナップショットから復元するために `DBSnapshotIdentifier`パラメータを指定する必要があります。 CloudFormation 取り込みは現在機密データを処理しないため、これは必要です。

AMS CloudFormation テンプレートの取り込みに CloudFormation テンプレートを使用する方法の例については、「」を参照してくださいCloudFormation 取り込み: 例。

テンプレートの検証

CloudFormation テンプレートは、AMS に送信する前に自己検証できます。

AMS CloudFormation 取り込みに送信されたテンプレートは、AMS アカウント内にデプロイしても安全であることを確認するために検証されます。検証プロセスは以下をチェックします。

- サポートされているリソース – AMS CloudFormation 取り込みがサポートされているリソースのみが使用されます。詳細については、「サポートされているリソース」を参照してください。
- サポートされている AMIs – テンプレートの AMI は AMS がサポートする AMI です。AMS AMIs「」を参照してくださいAMS Amazon マシンイメージ (AMIs)。
- AMS 共有サービスサブネット – テンプレートは、AMS 共有サービスサブネットへのリソースの起動を試みません。
- リソースポリシー – パブリックに読み取り可能な S3 バケットポリシーや書き込み可能な S3 バケットポリシーなど、過度に許可されたリソースポリシーはありません。AMS は、パブリックに読み取り可能または書き込み可能な S3 バケットを許可しません AWS アカウント。

Linter CloudFormation で検証する

CloudFormation Linter ツールを使用して、AMS に送信する前に CloudFormation テンプレートを自己検証できます。

CloudFormation Linter ツールは、リソース/プロパティ名、データ型、関数の検証を提供するため、CloudFormation テンプレートを検証する最善の方法です。詳細については、「aws-cloudformation/cfn-python-lint」を参照してください。

前述のテンプレートの CloudFormation Linter 出力は次のとおりです。

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

CloudFormation テンプレートのオフライン検証を支援するために、AMS は CloudFormation Linter ツール用のプラグイン可能なカスタム検証ルールのセットを開発しました。これらは、AMS コンソールの開発者リソースページにあります。

取り込み CloudFormation 前検証スクリプトを使用するには、次の手順に従います。

1. CloudFormation Linter ツールをインストールします。インストール手順については、「aws-cloudformation / cfn-lint」を参照してください。
2. 検証スクリプトを含む .zip ファイルをダウンロードします。

   CFN Lint カスタムルール。
3. アタッチされたルールを任意のディレクトリに解凍します。
4. 次のコマンドを実行して CloudFormation テンプレートを検証します。

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

CloudFormation 取り込みスタック: CFN 検証の例

これらの例は、取り込みを成功させるためにテンプレートを準備するのに役立ちます。

形式検証

テンプレートに「リソース」セクションが含まれ、テンプレートの下に定義されているすべてのリソースに「タイプ」値があることを確認します。

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

テンプレートのルートキーが許可されていることを確認します。許可されるルートキーは次のとおりです。

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

手動レビューで必要な検証

テンプレートに次のリソースが含まれている場合、自動検証は失敗し、手動によるレビューが必要になります。

表示されるポリシーは、セキュリティの観点から見ると高リスク領域です。たとえば、特定のユーザーまたはグループ以外のユーザーがオブジェクトを作成したり、アクセス許可を書き込んだりすることを許可する S3 バケットポリシーは非常に危険です。そのため、ポリシーを検証し、内容に基づいて承認または拒否します。これらのポリシーは自動作成できません。この問題に対応するための可能なアプローチを調査しています。

現在、以下のリソースに関する自動検証はありません。

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

パラメータ検証

テンプレートパラメータに値が指定されていない場合は、デフォルト値を指定する必要があります。

リソース属性の検証

必須属性チェック: 特定のリソースタイプには、特定の属性が存在している必要があります。

- VPCOptions」は に存在する必要があります AWS::OpenSearch::Domain
- CludsterSubnetGroupName」は に存在する必要があります AWS::Redshift::Cluster

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

許可されていない属性チェック: 特定の属性は、特定のリソースタイプに *存在してはいけません*。

- 「SecretString」は「AWS::SecretsManager::Secret」に存在しません
- 「MongoDbSettings」は「AWS::DMS::Endpoint」に存在してはいけません

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

SSM パラメータチェック: 次のリストの属性の場合、値は Secrets Manager または Systems Manager パラメータストア (Secure String Parameter) を介して指定する必要があります。

```
{
  "RDS::DBInstance": [
    "MasterUserPassword",
    "TdeCredentialPassword"
  ],
  "RDS::DBCluster": [
    "MasterUserPassword"
  ],
  "ElastiCache::ReplicationGroup": [
    "AuthToken"
```

```
  ],
  "DMS::Certificate": [
    "CertificatePem",
    "CertificateWallet"
  ],
  "DMS::Endpoint": [
    "Password"
  ],
  "CodePipeline::Webhook": {
    "AuthenticationConfiguration": [
        "SecretToken"
    ]
  },
  "DocDB::DBCluster": [
    "MasterUserPassword"
  ]
},
```

一部の属性は特定のパターンに準拠する必要があります。たとえば、IAM インスタンスプロファイル名は AMS 予約プレフィックスで始まることはできません。属性値は、次に示すように特定の正規表現と一致する必要があります。

```
{
    "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).+"
      ]
    }
}
```

リソースの検証

テンプレートでは、許可リストに登録されたリソースのみを指定できます。これらのリソースについては、「」を参照してくださいサポートされているリソース。

パッチ適用の制限により、EC2 スタックと Auto Scaling グループ (ASGs) は同じスタックで許可されません。

セキュリティグループの進入ルールの検証

- CFN Ingest Create または Stack Update CT 変更タイプからのリクエストの場合：
  - （ IpProtocol が tcp または 6) かつ (ポートが 80 または 443) の場合、CidrIP値に制限はありません。
  - それ以外の場合、 を 0.0.0.0/0 にCidrIPすることはできません。
- Service Catalog (Service Catalog 製品) からのリクエストの場合：
  - CFN Ingest Create または Stack Update CT の変更タイプ検証に加えて、 のプロトコルmanagement_portsを持つ のポートには、 経由でip_protocolsのみアクセスできます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"]
  }
```

制限

以下の機能は現在、AMS CloudFormation 取り込みプロセスではサポートされていません。

- YAML – サポートされていません。JSON ベースの CloudFormation テンプレートのみがサポートされています。
- ネストされたスタック – 代わりに、単一のテンプレートを使用するようにアプリケーションインフラストラクチャを設計します。または、クロススタック参照を使用して、あるリソースが別のリソースに依存する複数のスタック間でリソースを分離することもできます。詳細については、「チュートリアル: 別の AWS CloudFormation スタックのリソース出力」を参照してください。
- CloudFormation スタックセット – セキュリティ上の影響により、サポートされていません。
- CloudFormation テンプレートを使用した IAM リソースの作成 – セキュリティ上の影響により、IAM ロールのみがサポートされます。
- 機密データ – サポートされていません。テンプレートやパラメータ値に機密データを含めないでください。機密データを参照する必要がある場合は、Secrets Manager を使用してこれらの値を保存および取得します。リソースプロパティで AWS Secrets Managers シークレットを使用する方法については、「AWS CloudFormation テンプレートを使用して AWS Secrets Manager で管理されるシークレットを作成および取得する方法」および「動的リファレンスを使用してテンプレート値を指定する方法」を参照してください。

サポートされているリソース

AMS CloudFormation 取り込みプロセスでは、次の AWS リソースがサポートされています。

CloudFormation Ingest スタック: サポートされているリソース

インスタンスオペレーティングシステムは、AMS ワークロードの取り込みでサポートされる必要があります。ここにリストされている AWS リソースのみがサポートされています。

- 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
  - AWS::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 OpenSearch
  - AWS::OpenSearchService::Domain
- 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::HealthCheck
  - AWS::Route53::HostedZone
  - AWS::Route53::RecordSet
  - AWS::Route53::RecordSetGroup
  - AWS::Route53Resolver::ResolverRule
  - AWS::Route53Resolver::ResolverRuleAssociation
- Amazon S3
  - AWS::S3::Bucket

- 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
- Amazon SQS
  - AWS::SQS::Queue
- Amazon WorkSpaces
  - AWS::WorkSpaces::Workspace
- アプリケーションの 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

  AWS::DMS::Endpoint リソースの MongoDbSettings プロパティは許可されていません。

  次のプロパティは、AWS Secrets Manager によって解決される場合にのみ許可されます。AWS::DMS::Certificate リソースの CertificatePem プロパティと CertificateWallet プロパティ、および AWS::DMS::Endpoint リソースの Password プロパティ。

- 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 Identity and Access Management (IAM)
  - AWS::IAM::Role
- 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 セキュリティハブ
  - 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 リージョン別
  - 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

CloudFormation 取り込み: 例

CloudFormation テンプレート変更タイプでスタックを作成するを使用する方法の詳細な例を以下に示します。

ごとに一連のサンプル CloudFormation テンプレートをダウンロードするには AWS リージョン、「サンプルテンプレート」を参照してください。

CloudFormation リソースのリファレンス情報については、「AWS リソースタイプとプロパティタイプのリファレンス」を参照してください。ただし、AMS は、「」で説明されている、より小さなリソースセットをサポートしていますAMS CloudFormation の取り込み。

> **Note**
>
> AMS では、すべての IAM またはその他のポリシー関連のリソースを収集し、単一の管理 | その他 | その他 | 変更タイプを作成する (ct-1e1xtak34nx76) で送信することをお勧めします。たとえば、必要なすべての IAM ロール、IAM インスタンスプロファイル、既存の IAM ロールの IAM ポリシー更新、S3 バケットポリシー、SNS/SQS ポリシーなどを結合し、ct-1e1xtak34nx76 RFC を送信して、これらの既存のリソースを将来の CFN 取り込みテンプレート内で参照できるようにします。

トピック

- CloudFormation 取り込みの例: リソースの定義

- CloudFormation Ingest の例: 3 層ウェブアプリケーション

CloudFormation 取り込みの例: リソースの定義

AMS CloudFormation 取り込みを使用する場合は、CloudFormation テンプレートをカスタマイズし、CloudFormation 取り込み変更タイプ (ct-36cn2avfrrj9v) を使用して RFC の AMS に送信します。複数回再利用できる CloudFormation テンプレートを作成するには、CloudFormation テンプレートでハードコーディングするのではなく、スタック設定パラメータを CloudFormation 取り込み変更タイプ実行入力に追加します。最大の利点は、テンプレートを再利用できることです。

AMS CloudFormation の取り込み変更タイプ入力スキーマを使用すると、CloudFormation テンプレートで最大 60 個のパラメータを選択し、カスタム値を指定できます。

この例では、AMS CloudFormation 取り込み CT のパラメータとして、さまざまな CloudFormation テンプレートで使用できるリソースプロパティを定義する方法を示します。このセクションの例では、SNS トピックの使用状況を具体的に示しています。

トピック

- 例 1: CloudFormation SNSTopic リソースTopicNameプロパティをハードコードする
- 例 2: SNSTopic リソースを使用して AMS 変更タイプのパラメータを参照する
- 例 3: AMS 取り込み変更タイプで JSON 実行パラメータファイルを送信して SNS トピックを作成する
- 例 4: 同じ CloudFormation テンプレートを参照する新しい変更タイプを送信する
- 例 5: CloudFormation テンプレートでデフォルトのパラメータ値を使用する

例 1: CloudFormation SNSTopic リソース**`TopicName`**プロパティをハードコードする

この例では、CloudFormation テンプレートで CloudFormation SNSTopic リソースTopicNameプロパティをハードコードします。`Parameters` セクションは空であることに注意してください。

新しい CloudFormation テンプレートを作成することなく、新しいスタックの SNSTopic 名の値を変更できる CloudFormation テンプレートを作成するには、CloudFormation 取り込み変更タイプの AMS `Parameters`セクションを使用して、その設定を行うことができます。これにより、後で同じ CloudFormation テンプレートを使用して、別の`SNSTopic`名前の新しいスタックを作成します。

```
{
```

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

例 2: SNSTopic リソースを使用して AMS 変更タイプのパラメータを参照する

この例では、CloudFormation テンプレートで定義されたSNSTopicリソースTopicNameプロパティを使用して、AMS 変更タイプの を参照Parameterします。

```
{
  "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"}
      }
    }
  }
}
```

例 3: AMS 取り込み変更タイプで JSON 実行パラメータファイルを送信して SNS トピックを作成する

この例では、SNS トピック を作成する AMS 取り込み CT を使用して JSON 実行パラメータファイルを送信しますTopicName。SNS トピックは、この例に示す変更可能な方法で CloudFormation テンプレートで定義する必要があります。

```
{
  "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
}
```

例 4: 同じ CloudFormation テンプレートを参照する新しい変更タイプを送信する

この JSON の例では、CloudFormation テンプレートを変更せずに SNS TopicName値を変更します。代わりに、CloudFormation テンプレートから新しいデプロイ | 取り込み | スタックを送信する | 同じ CFN テンプレートを参照する変更タイプを作成します。

```
{
  "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
}
```

例 5: CloudFormation テンプレートでデフォルトのパラメータ値を使用する

この例では、Parameters実行パラメータにTopicName値が指定されていないため、SNS TopicName = 'MyTopicName' が作成されます。Parameters 定義を指定しない場合、CloudFormation テンプレートのデフォルトのパラメータ値が使用されます。

```
{
  "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 の例: 3 層ウェブアプリケーション

標準の 3-Tierウェブアプリケーションの CloudFormation テンプレートを取り込みます。

これには、Application Load Balancer、Application Load Balancer ターゲットグループ、Auto Scaling グループ、Auto Scaling グループ起動テンプレート、MySQL データベースを備えた Amazon Relational Database Service (RDS for SQL Server)、 AWS SSM パラメータストア、 AWS Secrets Manager が含まれます。この例をウォークスルーするには 30～60 分かかります。

前提条件

- Secrets Manager を使用して、対応する値を持つユーザー名とパスワードを含む AWS シークレットを作成します。シークレット名 を含むこのサンプル JSON テンプレート (zip ファイル) を参照しams-shared/myapp/dev/dbsecrets、それをシークレット名に置き換えることができます。AMS で AWS Secrets Manager を使用する方法については、「」を参照してくださいAMS リソースでの AWS Secrets Manager の使用。
- AWS SSM パラメータストア (PS) で必要なパラメータを設定します。この例では、プライベートサブネットVPCIdとパブリックサブネットSubnet-Idの と は、/app/DemoApp/PublicSubnet1a、、PublicSubnet1c、 PrivateSubnet1a PrivateSubnet1c などのパスの SSM PS に保存されますVPCCidr。必要に応じてパス、パラメータ名、値を更新します。
- AWS Secrets Manager および SSM パラメータストアパスへの読み取り権限を持つ IAM Amazon EC2 インスタンスロールを作成します (これらの例で作成および使用される IAM ロールは

ですcustomer-ec2_secrets_manager_instance_profile) 。インスタンスプロファイルロールなどの IAM 標準ポリシーを作成する場合、ロール名は で始まる必要がありますcustomer-。新しい IAM ロールを作成するには、AMS 変更タイプ管理 | アプリケーション | IAM インスタンスプロファイル | (ct-0ixp4ch2tiu04) CT を作成し、必要なポリシーをアタッチします (customer-ec2_secrets_manager_instance_profileなどの名前を付けることができます) 。AMS IAM 標準ポリシー、customer_secrets_manager_policyおよびはcustomer_systemsmanager_parameterstore_policy、IAM AWS コンソールでそのままま たはリファレンスとして使用できます。

標準の 3-Tierウェブアプリケーションの CloudFormation テンプレートを取り込む

1. 添付のサンプル CloudFormation JSON テンプレートを zip ファイル、3-tier-cfn-ingest.zip として S3 バケットにアップロードし、CFN Ingest RFC で使用する署名付き S3 URL を生成します。詳細については、「事前署名」を参照してください。CFN テンプレートは、AMS コンソールから RFC を送信するときに、CFN Ingest RFC にコピー/貼り付けることもできます。
2. AMS コンソールまたは AMS CLI を使用して、CloudFormation テンプレートから CloudFormation Ingest RFC (デプロイ | 取り込み | スタック | 作成 (ct-36cn2avfrrj9v) を作成します。CloudFormation の取り込み自動化プロセスでは、CloudFormation テンプレートを検証して、テンプレートに有効な AMS 対応リソースがあり、セキュリティ標準に準拠していることを確認します。

   - コンソールの使用 - 変更タイプで、CloudFormation テンプレート -> 作成からデプロイ -> 取り込み -> スタックを選択し、次のパラメータを例として追加します (MultiAZDatabase のデフォルトは false であることに注意してください) 。 CloudFormation

     ```
     CloudFormationTemplateS3Endpoint: "https://s3-ap-
     southeast-2.amazonaws.com/amzn-s3-demo-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"
     ```

   - の使用 AWS CLI - を使用した RFCsRFCs」を参照してください。 AWS CLI例えば、以下のコマンドを使用します。

     ```
     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
```

CloudFormation RFC 実行出力で Application Load Balancer URL を見つけて、ウェブサイトにアクセスします。リソースへのアクセスの詳細については、「インスタンスへのアクセス」を参照してください。

CloudFormation 取り込みスタックを作成する

コンソールを使用した CloudFormation 取り込みスタックの作成

コンソールを使用して CloudFormation 取り込みスタックを作成するには

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプ領域を使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、古いバージョンで作成オプションが RFC の作成ボタンの横に表示されます。
   - カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。

3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。

4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。

5. Run parameters エリアを開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した CloudFormation 取り込みスタックの作成

CLI を使用して CloudFormation 取り込みスタックを作成するには

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成し、1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。

2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id` *`ID`* コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id` *`ID`* コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com`

\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

1. スタックの作成に使用する CloudFormation テンプレートを準備し、S3 バケットにアップロードします。重要な詳細については、AWS CloudFormation 取り込みガイドライン」、「ベストプラクティス」、および「制限」を参照してください。
2. RFC を作成して AMS に送信します。

   - 実行パラメータ JSON ファイルを作成して保存し、必要な CloudFormation テンプレートパラメータを含めます。次の例では、CreateCfnParams.json.

     ウェブアプリケーションスタック CreateCfnParams.json ファイルの例：

```
{
  "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"
   }
  ],
}
```

SNS トピック CreateCfnParams.json ファイルの例：

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

3. 次の内容の RFC パラメータ JSON ファイルを作成して保存します。次の例では、CreateCfnRfc.json ファイルという名前を付けます。

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

4. CreateCfnRfc ファイルと CreateCfnParams ファイルを指定して、RFC を作成します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

> **Note**
>
> この変更タイプはバージョン 2.0 であり、自動化されています (手動では実行されません)。これにより、CT 実行がより迅速に実行され、新しいパラメータ CloudFormationTemplate を使用して、カスタム CloudFormation テンプレートを RFC に貼り付けることができます。さらに、このバージョンでは、独自のセキュリティグループを指定した場合、デフォルトの AMS セキュリティグループはアタッチされません。リクエスト

で独自のセキュリティグループを指定しない場合、AMS は AMS のデフォルトのセキュリティグループをアタッチします。CFN Ingest v1.0 では、独自のセキュリティグループを指定したかどうかにかかわらず、常に AMS のデフォルトのセキュリティグループが追加されています。
AMS では、この変更タイプで使用できるように 17 の AMS セルフプロビジョニングサービスが有効になっています。サポートされているリソースの詳細については、CloudFormation Ingest Stack: Supported Resources」を参照してください。

Note

バージョン 2.0 は、署名付き URL ではない S3 エンドポイントを受け入れます。
この CT の以前のバージョンを使用する場合、CloudFormationTemplateS3Endpoint パラメータ値は署名付き URL である必要があります。
署名付き S3 バケット URL (Mac/Linux) を生成するためのコマンドの例 :

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

署名付き S3 バケット URL を生成するためのコマンドの例 (Windows):

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

URLs Amazon S3の作成」も参照してください。

Note

S3 バケットが AMS アカウントに存在する場合は、このコマンドに AMS 認証情報を使用する必要があります。例えば、AMS AWS Security Token Service （ AWS STS) 認証情報を取得`--profile saml`した後に を追加する必要がある場合があります。

関連する変更タイプ: CloudFormation 取り込みスタックの変更セットを承認する、 CloudFormation 取り込みスタックを更新する

AWS CloudFormation の詳細については、「AWS Cloud Formation」を参照してください。CloudFormation テンプレートを表示するには、AWS CloudFormation テンプレートリファレンスを開きます。

CloudFormation 取り込みの検証

テンプレートは、AMS アカウントで作成できることを確認するために検証されます。検証に合格すると、AMS に準拠するために必要なリソースまたは設定が含まれるように更新されます。これには、AMS オペレーションがスタックをモニタリングできるようにするための Amazon CloudWatch アラームなどのリソースの追加が含まれます。

次のいずれかに該当する場合、RFC は拒否されます。

- RFC JSON 構文が正しくないか、指定された形式に従っていません。
- 指定された S3 バケットの署名付き URL が無効です。
- テンプレートは有効な CloudFormation 構文ではありません。
- テンプレートには、すべてのパラメータ値にデフォルトが設定されているわけではありません。
- テンプレートは AMS 検証に失敗します。AMS 検証の手順については、このトピックで後述する情報を参照してください。

リソース作成の問題により CloudFormation スタックの作成に失敗すると、RFC は失敗します。

CFN 検証と検証の詳細については、「テンプレート検証と CloudFormation 取り込みスタック: CFN 検証の例」を参照してください。

CloudFormation 取り込みスタックを更新する

コンソールを使用した CloudFormation 取り込みスタックの更新

コンソールを使用して CloudFormation 取り込みスタックを更新するには

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

CTsソートするには、カードビューまたはテーブルビューですべての変更タイプエリアを使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。

- カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。

3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters エリアを開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した CloudFormation 取り込みスタックの更新

CLI を使用して CloudFormation 取り込みスタックを更新するには

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成し、1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id` *`ID`* コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id` *`ID`* コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"`にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

1. スタックの更新に使用する CloudFormation テンプレートを準備し、S3 バケットにアップロードします。重要な詳細については、AWS CloudFormation 取り込みガイドライン」、「ベストプラクティス」、および「制限」を参照してください。
2. RFC を作成して AMS に送信します。

   - 実行パラメータ JSON ファイルを作成して保存し、必要な CloudFormation テンプレートパラメータを含めます。この例では、UpdateCfnParams.json.

     インラインパラメータ更新を含む UpdateCfnParams.json ファイルの例：

```
{
  "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
}
```

更新された CloudFormation テンプレートを含む S3 バケットエンドポイントを含む UpdateCfnParams.json ファイルの例：

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

3. 次の内容の RFC パラメータ JSON ファイルを作成して保存します。この例では、UpdateCfnRfc.json ファイルという名前を付けます。

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

4. RFC を作成し、UpdateCfnRfc ファイルと UpdateCfnParams ファイルを指定します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

- この変更タイプはバージョン 2.0 になりました。変更には、この CT のバージョン 1.0 で使用された AutoApproveUpdateForResources パラメータの削除と、AutoApproveRiskyUpdates と BypassDriftCheck の 2 つの新しいパラメータの追加が含まれます。
- S3 バケットが AMS アカウントに存在する場合は、このコマンドに AMS 認証情報を使用する必要があります。例えば、AMS AWS Security Token Service （ AWS STS) 認証情報を取得`--profile saml`した後に を追加する必要がある場合があります。
- CloudFormation テンプレートのリソースのすべての`Parameter`値には、デフォルトまたは CT のパラメータセクションのカスタム値のいずれかの値が必要です。パラメータ値を上書きするには、CloudFormation テンプレートリソースを構造化して Parameters キーを参照します。その方法を示す例については、CloudFormation 取り込みスタック: CFN 検証の例」を参照してください。

  重要: フォームで明示的に指定されていないパラメータがありません。デフォルトでは、既存のスタックまたはテンプレートで現在設定されている値になります。
- Ingest CloudFormation を使用して追加できるセルフプロビジョニングサービスのリストについては、CloudFormation Ingest Stack: Supported Resources」を参照してください。

  詳細については CloudFormation、「AWS Cloud Formation」を参照してください。

CloudFormation 取り込みの検証

テンプレートは、AMS アカウントで作成できることを確認するために検証されます。検証に合格すると、AMS に準拠するために必要なリソースまたは設定が含まれるように更新されます。これには、AMS オペレーションがスタックをモニタリングできるようにするための Amazon CloudWatch アラームなどのリソースの追加が含まれます。

次のいずれかに該当する場合、RFC は拒否されます。

- RFC JSON 構文が正しくないか、指定された形式に従っていません。
- 指定された S3 バケットの署名付き URL が無効です。
- テンプレートは有効な CloudFormation 構文ではありません。
- テンプレートには、すべてのパラメータ値にデフォルトが設定されているわけではありません。
- テンプレートは AMS 検証に失敗します。AMS 検証の手順については、このトピックで後述する情報を参照してください。

リソース作成の問題により CloudFormation スタックの作成に失敗すると、RFC は失敗します。

CFN 検証と検証の詳細については、「テンプレート検証と CloudFormation 取り込みスタック: CFN 検証の例」を参照してください。

CloudFormation 取り込みスタックの変更セットを承認する

コンソールを使用した CloudFormation 取り込みスタックの承認と更新

コンソールを使用して CloudFormation 取り込みスタックを承認および更新するには

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプエリアを使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。
   - カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。
3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters エリアを開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した CloudFormation 取り込みスタックの承認と更新

CLI を使用して CloudFormation 取り込みスタックを承認および更新するには

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成し、1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id ID` コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id ID` コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

1. この変更タイプの実行パラメータ JSON スキーマを現在のフォルダ内のファイルに出力します。この例では、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. スキーマを次のように変更して保存します。

   ```
   {
     "StackId": "STACK_ID",
   ```

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

ヒント

> **Note**
>
> スタックに複数のリソースがあり、スタックリソースのサブセットのみを削除する場合は、CloudFormation Update CT を使用します。CloudFormation Ingest Stack: Update」を参照してください。サービスリクエストケースを送信することもでき、必要に応じて AMS エンジニアが変更セットの作成を支援します。

詳細については AWS CloudFormation、「」を参照してくださいAWS CloudFormation。

CloudFormation スタック終了保護を更新する

コンソールを使用した CloudFormation 終了保護スタックの更新

AMS コンソールでこの変更タイプを以下に示します。

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプエリアを使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。

- カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。

3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters 領域を開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した CloudFormation スタック終了保護の更新

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成し、1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id` *`ID`* コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id` *`ID`* コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたとき

に通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

変更するパラメータのみを指定します。パラメータがない場合、既存の値は保持されます。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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\"]}}"
```

テンプレートの作成：

1. この変更タイプの実行パラメータを JSON ファイルに出力します。この例では EnableTermProCFNParams.json:

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

2. EnableTermProCFNParams ファイルを変更して保存し、変更するパラメータのみを保持します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

```
    "TerminationProtectionDesiredState": ["enabled"]
  }
}
```

3. RFC テンプレートを現在のフォルダ内のファイルに出力します。この例では EnableTermProCFNRfc.json:

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

4. EnableTermProCFNRfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

5. EnableTermProCFNRfc ファイルと EnableTermProCFNParams ファイルを指定して、RFC を作成します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

> **Note**
>
> Amazon EC2、EC2 スタックに関連する CT があります。終了保護の更新。

終了保護の詳細については、「スタックが削除されないように保護する」を参照してください。

AMS で CFN 取り込みまたはスタック更新 CTs を使用した自動 IAM デプロイ

これらの AMS 変更タイプを使用して、マルチアカウントランディングゾーン (MALZ) とシングルアカウントランディングゾーン (SALZ) の両方に IAM ロール (`AWS::IAM::Role`リソース) をデプロイできます。

- デプロイ | 取り込み | CloudFormation テンプレートからのスタック | 作成 (ct-36cn2avfrrj9v)
- 管理 | カスタムスタック | CloudFormation テンプレートからのスタック | 更新 (ct-361tlo1k7339x)
- 管理 | カスタムスタック | CloudFormation テンプレートからのスタック | 承認と更新 (ct-1404e21baa2ox)

CFN テンプレートの IAM ロールで実行される検証：

- ManagedPolicyArns: 属性 ManagedPolicyArns は に存在しません`AWS::IAM::Role`。検証では、プロビジョニングするロールに管理ポリシーをアタッチすることはできません。代わりに、ロールのアクセス許可は、 プロパティポリシーを通じてインラインポリシーを使用して管理できます。
- PermissionsBoundary: ロールのアクセス許可の境界を設定するために使用されるポリシーは、AMS が提供する管理ポリシー のみですAWSManagedServices_IAM_PermissionsBoundary。このポリシーは、プロビジョニングされるロールを使用して AMS インフラストラクチャリソースが変更されないように保護するガードレールとして機能します。このデフォルトのアクセス許可の境界では、AMS が提供するセキュリティ上の利点は保持されます。

  `AWSManagedServices_IAM_PermissionsBoundary` (デフォルト) が必要です。必須でない場合、リクエストは拒否されます。
- MaxSessionDuration: IAM ロールに設定できる最大セッション時間は 1～4 時間です。AMS 技術標準では、セッション期間が 4 時間を超える場合、顧客リスクを受け入れる必要があります。
- RoleName: 次の名前空間は AMS によって保持され、IAM ロール名のプレフィックスとして使用することはできません。

```
AmazonSSMRole,
AMS,
Ams,
ams,
AWSManagedServices,
customer_developer_role,
```

```
customer-mc-,
Managed_Services,
MC,
Mc,
mc,
SENTINEL,
Sentinel,
sentinel,
StackSet-AMS,
StackSet-Ams,
StackSet-ams,
StackSet-AWS,
StackSet-MC,
StackSet-Mc,
StackSet-mc
```

- ポリシー: IAM ロールに埋め込まれたインラインポリシーには、AMS によって事前に承認された一連の IAM アクションのみを含めることができます。これは、 (コントロールポリシー) を使用して IAM ロールを作成できるすべての IAM アクションの上限です。コントロールポリシーは、以下で構成されます。
  - すべての AWS のサービス および リソースへの読み取り専用アクセスを提供する AWS 管理ポリシー ReadOnlyAccess のすべてのアクション
  - 次のアクションは、クロスアカウント S3 アクションに制限されます。つまり、許可された S3 アクションは、作成されるロールと同じアカウントに存在するリソースでのみ実行できます。

    ```
    amscm:*,
    amsskms:*,
    lambda:InvokeFunction,
    logs:CreateLogStream,
    logs:PutLogEvents,
    s3:AbortMultipartUpload,
    s3:DeleteObject,
    s3:DeleteObjectVersion,
    s3:ObjectOwnerOverrideToBucketOwner,
    s3:PutObject,
    s3:ReplicateTags,
    secretsmanager:GetRandomPassword,
    sns:Publish
    ```

    CFN 取り込みを通じて作成または更新された IAM ロールは、このコントロールポリシーにリストされているアクション、またはコントロールポリシーにリストされているアクションからス

コープダウンされたアクション (許容値以下) を許可できます。現在、読み取り専用アクションとして分類できるこれらの安全な IAM アクションに加えて、前述の非読み取り専用アクションは CTs では実行できず、AMS 技術標準に従って事前承認されています。

- AssumeRolePolicyDocument: 次のエンティティは事前承認されており、作成中のロールを引き受ける信頼ポリシーに含めることができます。
  - 同じアカウントのすべての IAM エンティティ (ロール、ユーザー、ルートユーザー、STS 引き受けたロールセッション) がロールを引き受けることができます。
  - 次の がロールを引き受け AWS のサービス ることができます。

```
apigateway.amazonaws.com,
autoscaling.amazonaws.com,
cloudformation.amazonaws.com,
codebuild.amazonaws.com,
codedeploy.amazonaws.com,
codepipeline.amazonaws.com,
datapipeline.amazonaws.com,
datasync.amazonaws.com,
dax.amazonaws.com,
dms.amazonaws.com,
ec2.amazonaws.com,
ecs-tasks.amazonaws.com,
ecs.application-autoscaling.amazonaws.com,
elasticmapreduce.amazonaws.com,
es.amazonaws.com,
events.amazonaws.com,
firehose.amazonaws.com,
glue.amazonaws.com,
lambda.amazonaws.com,
monitoring.rds.amazonaws.com,
pinpoint.amazonaws.com,
rds.amazonaws.com,
redshift.amazonaws.com,
s3.amazonaws.com,
sagemaker.amazonaws.com,
servicecatalog.amazonaws.com,
sns.amazonaws.com,
ssm.amazonaws.com,
states.amazonaws.com,
storagegateway.amazonaws.com,
transfer.amazonaws.com,
vmie.amazonaws.com
```

- 同じアカウントの SAML プロバイダーがロールを引き受けることができます。現在、サポートされている SAML プロバイダー名は のみですcustomer-saml。

1 つ以上の検証が失敗すると、RFC は拒否されます。RFC 拒否の理由の例は次のようになります。

```
{"errorMessage":"[ 'LambdaRole: The maximum session duration (in seconds) should be a
 numeric value in the range 3600 to 14400 (i.e. 1 to 4 hours).', 'lambda-policy: Policy
 document is too permissive.']","errorType":"ClientError"}
```

RFC の検証または実行の失敗についてサポートが必要な場合は、RFC 対応を使用して AMS にお問い合わせください。手順については、「RFC の対応とアタッチメント (コンソール)」を参照してください。その他の質問については、サービスリクエストを送信します。ハウツーについては、「サービスリクエストの作成」を参照してください。

> **Note**
>
> 現在、IAM 検証の一環として IAM のベストプラクティスを適用していません。IAM のベストプラクティスについては、「IAM のセキュリティのベストプラクティス」を参照してください。

より寛容なアクションで IAM ロールを作成する、または IAM のベストプラクティスを適用する

次の手動変更タイプを使用して IAM エンティティを作成します。

- デプロイ | 高度なスタックコンポーネント | Identity and Access Management (IAM) | エンティティまたはポリシーの作成 (ct-3dpd8mdd9jn1r)
- 管理 | 高度なスタックコンポーネント | Identity and Access Management (IAM) | エンティティまたはポリシーの更新 (ct-27tuth19k52b4)

これらの手動 RFCs を提出する前に、当社の技術標準を読んで理解することをお勧めします。アクセスについては、「技術標準にアクセスする方法」を参照してください。

> **Note**
>
> これらの手動変更タイプで直接作成された各 IAM ロールは、独自の個々のスタックに属し、CFN Ingest CT を介して他のインフラストラクチャリソースが作成されるのと同じスタックには存在しません。

自動変更タイプを使用して更新を実行できない場合に、手動変更タイプを使用して CFN 取り込みで作成された IAM ロールを更新する

管理 | 高度なスタックコンポーネント | Identity and Access Management (IAM) | 更新エンティティまたはポリシー (ct-27tuth19k52b4) の変更タイプを使用します。

> **Important**
>
> 手動 CT による IAM ロールの更新は CFN スタックテンプレートに反映されず、スタックドリフトが発生します。ロールが手動リクエストによって検証に合格しない状態に更新されると、検証に準拠していない限り、スタック更新 CT (ct-361tlo1k7339x) を使用してロールを再度更新することはできません。更新 CT は、CFN スタックテンプレートが検証に準拠している場合にのみ使用できます。ただし、検証に準拠していない IAM リソースが更新されておらず、CFN テンプレートが検証に合格している限り、スタックは スタック更新 CT (ct-361tlo1k7339x) を介して更新できます。

取り込みによって作成された IAM AWS CloudFormation ロールの削除

スタック全体を削除する場合は、次の自動 Delete Stack 変更タイプを使用します。手順については、「スタックの削除」を参照してください。

- 変更タイプ ID: ct-0q0bic0ywqk6c
- 分類: 管理 | 標準スタック | スタック | 削除と管理 | 高度なスタックコンポーネント | スタック | 削除

スタック全体を削除せずに IAM ロールを削除する場合は、CloudFormation テンプレートから IAM ロールを削除し、更新されたテンプレートを自動 Stack Update 変更タイプへの入力として使用できます。

- 変更タイプ ID: ct-361tlo1k7339x

- 分類: 管理 | カスタムスタック | CloudFormation テンプレートからのスタック | 更新

手順については、AWS CloudFormation 「取り込みスタックの更新」を参照してください。

CodeDeploy リクエスト

AWS CodeDeploy を使用してアプリケーションコンテナを作成し、CodeDeploy アプリケーショングループを介してデプロイできます。CodeDeploy の詳細については、AWS CodeDeploy ドキュメント」を参照してください。

AWS CodeDeploy の使用には、次のプロセスが含まれます。

1. CodeDeploy でアプリケーションを作成します。CodeDeploy アプリケーションは、デプロイ中に正しいリビジョン、デプロイ設定、デプロイグループが参照されるように CodeDeploy が使用する名前またはコンテナです。
2. CodeDeploy デプロイグループを作成します。CodeDeploy デプロイグループは、デプロイの対象となる個々のインスタンスのセットを定義します。AMS には、EC2 の CodeDeploy デプロイグループの個別の変更タイプがあります。
3. CodeDeploy デプロイグループを使用して CodeDeploy アプリケーションをデプロイします。

CodeDeploy アプリケーション

CodeDeploy アプリケーションを作成またはデプロイします。

CodeDeploy アプリケーションを作成する

コンソールを使用した CodeDeploy アプリケーションの作成

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの Browse change types ビューで一般的な変更タイプ (CT) を選択するか、Choose by category ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプ領域を使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。

- カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。

3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定領域を開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters エリアを開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した CodeDeploy アプリケーションの作成

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成し、1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id` *`ID`* コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id` *`ID`* コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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\"}}"
```

テンプレートの作成：

1. CodeDeploy アプリケーション CT の実行パラメータ JSON スキーマを現在のフォルダ内のファイルに出力します。この例では CreateCDAppParams.json:

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

2. 次のように JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

```
    }
}
```

3. CreateRfc の JSON テンプレートを現在のフォルダのファイルに出力します。この例では CreateCDAppRfc.json:

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

4. 次のように JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

5. CreateCDAppRfc ファイルと実行パラメータファイルを指定して、RFC を作成します。

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

   レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

AWS CodeDeploy の詳細については、AWS CodeDeploy でアプリケーションを作成する」を参照してください。

CodeDeploy アプリケーションをデプロイする

コンソールを使用した CodeDeploy アプリケーションのデプロイ

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。

2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTsソートするには、カードビューまたはテーブルビューですべての変更タイプエリアを使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。
   - カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。
3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters 領域を開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した CodeDeploy アプリケーションのデプロイ

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含む`create-rfc`コマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成し、1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力として`create-rfc`コマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id` *`ID`* コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id` *`ID`* コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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\":\"amzn-s3-demo-bucket\",
\"S3BundleType\":\"tar\",\"S3Key\":\"TestKey\"}}}}"Test\"}}"
```

テンプレートの作成：

1. CodeDeploy アプリケーションデプロイ CT の実行パラメータ JSON スキーマを出力します。この例では DeployCDAppParams.json:

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

2. JSON ファイルを次のように変更します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
"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": "amzn-s3-demo-bucket",
        "S3BundleType": "zip",
        "S3Key": "wordpress.zip" }
        }
    }
}
```

3. CreateRfc の JSON テンプレートを現在のフォルダ内のファイルに出力します。この例では DeployCDAppRfc.json:

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

4. DeployCDAppRfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

5. RFC を作成し、実行パラメータファイルと DeployCDAppRfc ファイルを指定します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

詳細については、CodeDeploy を使用してデプロイを作成する」を参照してください。

CodeDeploy デプロイグループ

CodeDeploy アプリケーショングループを作成します。

CodeDeploy デプロイグループを作成する

コンソールを使用した CodeDeploy デプロイグループの作成

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプエリアを使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。
   - カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。
3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。

4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters 領域を開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した CodeDeploy デプロイグループの作成

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成します。1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id ID` コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id ID` コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"`にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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\"}}"
```

テンプレートの作成：

1. 実行パラメータ JSON スキーマを現在のフォルダ内のファイルに出力します。この例では CreateCDDepGroupParams.json:

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

2. JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

   ```
   {
   "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. CreateRfc の JSON テンプレートを現在のフォルダのファイルに出力します。この例では CreateCDDepGroupRfc.json:

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

4. JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

5. CreateCDDepGroupRfc ファイルと実行パラメータファイルを指定して、RFC を作成します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

AWS CodeDeploy デプロイグループの詳細については、AWS CodeDeploy を使用してデプロイグループを作成する」を参照してください。

EC2 の CodeDeploy デプロイグループを作成する

コンソールを使用した EC2 用の CodeDeploy デプロイグループの作成

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。

- 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

  CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプ領域を使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。
- カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。

3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters エリアを開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した EC2 用の CodeDeploy デプロイグループの作成

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含む`create-rfc`コマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成します。1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力として`create-rfc`コマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id` *`ID`* コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id` *`ID`* コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符を エスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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\":\"\"}}"
```

テンプレートの作成：

1. 実行パラメータ JSON スキーマをファイルに出力します。この例では、CreateCDDepGroupEc2Params.json:

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

2. JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
"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. CreateRfc の JSON テンプレートを現在のフォルダ内のファイルに出力します。この例では CreateCDDepGroupEc2Rfc.json:

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

4. JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
"ChangeTypeVersion":    "1.0",
"ChangeTypeId":         "ct-00tlkda4242x7",
"Title":                "CD-Dep-Group-For-Ec2-Stack-RFC"
}
```

5. CreateCDDepGroupEc2Rfc ファイルと実行パラメータファイルを指定して、RFC を作成します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

AWS CodeDeploy デプロイグループの詳細については、AWS CodeDeploy を使用してデプロイグループを作成する」を参照してください。

AWS Database Migration Service (AWS DMS)

AWS Database Migration Service （ AWS DMS) は、データベースを AMS に簡単かつ安全に移行するのに役立ちます。Oracle、MySQL、PostgreSQL などの最も広く使用されている、市販のオープンソースデータベース間でデータを移行できます。このサービスは、Oracle から Oracle への同種移行や、Oracle から PostgreSQL や MySQL から Oracle への異なるデータベースプラットフォーム間の異種移行をサポートします。 AWS DMS は AWS サービスです。AMS CTs は、AMS 管理アカウントで AWS DMS リソースを作成するのに役立ちます。

次の図は、データベース移行のワークフローを示しています。

トピック

- AWS Database Migration Service （ AWS DMS ）、開始する前に
- AWS DMS、セットアップに必要なデータ
- AWS DMS タスクのセットアップ
- AWS DMS 管理

AWS Database Migration Service （ AWS DMS ）、開始する前に

AMS を使用してデータベース移行を計画する場合は AWS DMS、次の点を考慮してください。

- ソースエンドポイントとターゲットエンドポイント: ソースデータベース内のどの情報とテーブルをターゲットデータベースに移行する必要があるかを知る必要があります。AMS は、テーブルやプライマリキーの作成など、基本的なスキーマ移行 AWS DMS をサポートしています。ただし、AMS AWS DMS はターゲットデータベースにセカンダリインデックス、外部キー、アカウントなどを自動的に作成しません。詳細については、「データ移行のソース」と「データ移行のターゲット」を参照してください。
- スキーマ/コード移行: AMS AWS DMS はスキーマまたはコード変換を実行しません。Oracle SQL Developer、MySQL Workbench、または pgAdmin III などのツールを使用してスキーマを変換できます。既存のスキーマを別のデータベースエンジンに変換する場合は、AWS Schema Conversion Tool を使用できます。このツールは、ターゲットスキーマを作成でき、スキーマ全体 (テーブル、

インデックス、ビューなど) を生成および作成することもできます。このツールを使用して PL/SQL または TSQL を PgSQL や他の形式に変換することもできます。

- サポートされていないデータ型: 一部のソースデータ型は、ターゲットデータベースの同等のデータ型に変換する必要があります。

AWS DMS 考慮すべきシナリオ

以下の文書化されたシナリオは、独自のデータベース移行パスを作成するのに役立ちます。

- オンプレミス MySQL サーバーから Amazon RDS MySQL にデータを移行する: AWS ブログ記事「オンプレミス MySQL データを Amazon RDS に移行する (および戻る) 」を参照してください。
- Oracle データベースから Amazon RDS Aurora PostgreSQL データベースへのデータの移行: AWS ブログ記事「Oracle データベースから Amazon Aurora PostgreSQL データベースへの移行の概要」を参照してください。
- RDS MySQL から S3 にデータを移行する: AWS ブログ記事「AWS DMS を使用してリレーショナルデータベースから Amazon Glacier にデータをアーカイブする方法」を参照してください。

データベース移行では、以下の操作を行う必要があります。

- データベース移行を計画します。これには、レプリケーションサブネットグループの設定が含まれます。
- 移行のすべてのプロセスを実行するレプリケーションインスタンスを割り当てます。
- ソースとターゲットのデータベースエンドポイントを指定します。
- 使用するテーブルとレプリケーション プロセスを定義するタスクまたはタスクセットを作成します。
- IAM AWS DMS ロール`dms-cloudwatch-logs-role`と `dms-vpc-role`ロールを作成します。Amazon Redshift をターゲットデータベースとして使用する場合は、IAM ロールを作成して `dms-access-for-endpoint` AWS アカウントに追加する必要があります。詳細については、「AWS CLI および AWS DMS API で使用する IAM ロールの作成」を参照してください。

これらのチュートリアルでは、AMS コンソールまたは AMS CLI を使用して AWS Database Migration Service () を作成する例を示しますAWS DMS。 AWS DMS レプリケーションインスタンス、サブネットグループ、タスク、ソース AWS DMS エンドポイント、ターゲットエンドポイントを作成するための CLI コマンドが用意されています。

AMS の詳細については AWS DMS、一般的な情報AWS Database Migration Serviceについては「」、一般的な質問に対する回答についてはAWS Database Migration Service FAQs」を参照してください。

AWS DMS、セットアップに必要なデータ

以下の各 AWS DMS チュートリアルでは、いくつかの共通データが必要です。

- `Description`: リソースに関する重要な情報。これは他のパラメータ`Description`オプションとは別のものです。
- `VpcId`: 使用する VPC。これは、SKMS API (`list-vpc-summaries` CLI の) の ListVpcSummaries オペレーションを実行するか、AMS コンソールVPCs ページを確認することで確認できます。AMS SKMS API リファレンスについては、AWS Artifact コンソールのレポートタブを参照してください。
- `Name`: スタックまたはスタックコンポーネントの名前。スタック名になります。
- `TimeoutInMinutes`: RFC が失敗するまでにスタックの作成に許可される分数。この設定では RFC の実行は遅延しませんが、十分な時間を与える必要があります (を指定しないなど`"5"`)。
- `ChangeTypeId`、`ChangeTypeVersion`、および `StackTemplateId`: これらは必須ですが、CT ごとに異なり、その値は各関連セクションで以下に示します。

AWS DMS タスクのセットアップ

以下のチュートリアル AWS DMS で をセットアップします。

1: AWS DMS レプリケーションサブネットグループ: 作成

AMS コンソールまたは API/CLI を使用して、AMS AWS DMS レプリケーションサブネットグループを作成できます。

AWS DMS レプリケーションサブネットグループを作成する

コンソールを使用した AWS DMS レプリケーションサブネットグループの作成

> **Note**
>
> IAM `dms-vpc-role` ロールがアカウントに存在しない場合、この CT は失敗します。

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTsソートするには、カードビューまたはテーブルビューですべての変更タイプ領域を使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。
   - カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。
3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters エリアを開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した AWS DMS レプリケーションサブネットグループの作成

> **Note**
>
> IAM `dms-vpc-role` ロールがアカウントに存在しない場合、この CT は失敗します。

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成し、1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id ID` コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id ID` コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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-j637f96ls1h4oy5fj
\"}"
```

テンプレートの作成：

1. この変更タイプの実行パラメータを JSON ファイルに出力します。この例では CreateDmsRsgParams.json:

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

2. 実行パラメータ CreateDmsRsgParams.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
"Description":          "DMSTestRepSG",
"VpcId":                "VPC_ID",
"TimeoutInMinutes":     60,
"StackTemplateId":      "stm-j637f96ls1h4oy5fj",
"Name":                 "Test RSG",
"Parameters":   {
    "Description":            "DESCRIPTION",
    "SubnetIds":              ["SUBNET_ID", "SUBNET_ID"]
    }
}
```

3. JSON テンプレートを現在のフォルダのファイルに出力します。この例では CreateDmsRsgRfc.json:

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

4. CreateDmsRsgRfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

5. RFC を作成し、実行パラメータファイルと CreateDmsRsgRfc ファイルを指定します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

- IAM `dms-vpc-role` ロールがアカウントに存在しない場合、この CT は失敗します。
- 最大 50 個のタグを追加できますが、追加設定ビューを有効にする必要があります。

DMS レプリケーションインスタンスとサブネットグループの詳細については、「レプリケーションインスタンスのネットワークのセットアップ」を参照してください。

2: AWS DMS レプリケーションインスタンス: 作成

AMS コンソールまたは API/CLI を使用して AMS AWS DMS レプリケーションインスタンスを作成できます。

AWS DMS レプリケーションインスタンスを作成する

コンソールを使用した AWS DMS レプリケーションインスタンスの作成

AMS コンソールでのこの変更タイプのスクリーンショット：

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの Browse change types ビューで一般的な変更タイプ (CT) を選択するか、Choose by category ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプ領域を使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。

- カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。

3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定領域を開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters エリアを開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した AWS DMS レプリケーションインスタンスの作成

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成し、1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id` *`ID`* コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id` *`ID`* コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたとき

に通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符を エスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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\"}"
```

レプリケーションのインスタンスが作成される間、ソースおよびターゲットデータストアを指定できます。ソースデータストアとターゲットデータストアは、Amazon Elastic Compute Cloud (Amazon EC2) インスタンス、AWS S3 バケット、Amazon Relational Database Service (Amazon RDS) DB インスタンス、またはオンプレミスデータベース上に配置できます。

テンプレートの作成：

1. この変更タイプの実行パラメータを JSON ファイルに出力します。この例では CreateDmsRiParams.json:

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

2. 実行パラメータ CreateDmsRiParams.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

   ```
   {
   "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. JSON テンプレートを現在のフォルダ内のファイルに出力します。この例では CreateDmsRiRfc.json:

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

4. CreateDmsRiRfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

5. RFC を作成し、実行パラメータファイルと CreateDmsRiRfc ファイルを指定します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

- 最大 50 個のタグを追加できますが、追加設定ビューを有効にする必要があります。
- AMS VPC の EC2 インスタンスに、ソースデータベースからターゲットデータベースにデータを割り当てて移行するタスクを実行するのに十分なストレージと処理能力を持つレプリケーションインスタンスを作成する必要があります。このインスタンスの必要なサイズは、移行する必要のあるデータの量、および、インスタンスが実行するタスクにより異なります。レプリケーションインスタンスは、 MultiAZオプションを選択すると、マルチ AZ 配置を使用した高可用性とフェイルオーバーのサポートを提供します。レプリケーションインスタンスの詳細については、「AWS DMS レプリケーションインスタンスの使用」を参照してください。

3: AWS DMS ソースエンドポイント: Mongo DB 用に作成、S3 用に作成

AMS コンソールまたは API/CLI を使用して、さまざまなデータベースの AMS DMS ソースエンドポイントを作成できます。3 つの例を示します。

DMS ソースエンドポイント: 作成

コンソールを使用した DMS ソースエンドポイントの作成

AMS コンソールでのこの変更タイプのスクリーンショット：

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプエリアを使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。
   - カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。
3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。

5. Run parameters 領域を開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した DMS ソースエンドポイントの作成

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成します。1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id ID` コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id ID` コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"`にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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\"}"
```

テンプレートの作成：

1. この変更タイプの実行パラメータを CreateDmsSeParams.json.

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

2. 実行パラメータ JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
"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. JSON テンプレートを現在のフォルダ内のファイルに出力します。この例では CreateDmsSeRfc.json:

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

4. CreateDmsSeRfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

5. RFC を作成し、実行パラメータファイルと CreateDmsSeRfc ファイルを指定します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

DMS エンドポイントを作成する前に、パスワードにサポートされていない文字が含まれていないことを確認してください。詳細については、AWS Database Migration Service 「 ユーザーガイド」の「ソースエンドポイントとターゲットエンドポイントの作成」を参照してください。

詳細については、「データ移行のソース」を参照してください。

S3 ソースエンドポイントについては、「」を参照してくださいS3 の DMS ソースエンドポイント: 作成。

Mongo DB ソースエンドポイントについては、「」を参照してくださいMongoDB の DMS ソースエンドポイント: 作成。

MongoDB の DMS ソースエンドポイント: 作成

コンソールを使用した DMS Mongo DB ソースエンドポイントの作成

AMS コンソールでのこの変更タイプのスクリーンショット：

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。

2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプエリアを使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。
   - カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。
3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters エリアを開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した DMS Mongo DB ソースエンドポイントの作成

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含む`create-rfc`コマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成します。1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力として`create-rfc`コマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id` *`ID`* コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id` *`ID`* コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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\"}}"
```

テンプレートの作成：

1. この変更タイプの実行パラメータを CreateDmsSeMongoParams.json.

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

2. 実行パラメータ JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
"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. JSON テンプレートを現在のフォルダ内のファイルに出力します。この例では CreateDmsSeMongoRfc.json:

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

4. CreateDmsSeMongoRfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

5. RFC を作成し、実行パラメータファイルと CreateDmsSeMongoRfc ファイルを指定します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

> **Note**
>
> 最大 50 個のタグを追加できますが、追加設定ビューを有効にする必要があります。

AMS DMS は、Mongo または任意のリレーショナルデータベースサービス (RDS) をソースエンドポイントとして使用できます。S3 ソースエンドポイントについては、「」を参照してくださいS3 の DMS ソースエンドポイント: 作成。

S3 の DMS ソースエンドポイント: 作成

コンソールを使用した DMS S3 ソースエンドポイントの作成

AMS コンソールでのこの変更タイプのスクリーンショット：

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプエリアを使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。
   - カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。
3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。

4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters 領域を開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した DMS S3 ソースエンドポイントの作成

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成します。1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id ID` コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id ID` コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"`にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符を エスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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\":\"amzn-s3-
demo-bucket\",\"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::123456789101:role/ams-
ops-ct-authors-dms-s3-test-role\"},\"TimeoutInMinutes\":60,\"StackTemplateId\":\"stm-
pud4ghhkp7395n9bc\"}"
```

テンプレートの作成：

1. この変更タイプの実行パラメータを CreateDmsSeS3Params.json.

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

2. 実行パラメータ JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

   ```
   {
   "Description":          "TestS3DMS-SE",
   "VpcId":                "VPC_ID",
   "Name":                 "S3-DMS-SE",
   "StackTemplateId":      "stm-pud4ghhkp7395n9bc",
   "TimeoutInMinutes":     60,
   "Parameters":   {
       "EngineName":               "s3",
       "S3BucketName":              "amzn-s3-demo-bucket",
   ```

```
    "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"}],"TableColumnsTot
    "S3ServiceAccessRoleArn":      "arn:aws:iam::123456789101:role/ams-ops-ct-
authors-dms-s3-test-role",
      }
}
```

3. JSON テンプレートを現在のフォルダ内のファイルに出力します。この例では CreateDmsSeS3Rfc.json:

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

4. CreateDmsSeS3Rfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

5. RFC を作成し、実行パラメータファイルと CreateDmsSeS3Rfc ファイルを指定します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

> **Note**
>
> 最大 50 個のタグを追加できますが、追加設定ビューを有効にする必要があります。

AMS DMS は、S3 または任意のリレーショナルデータベースサービス (RDS) ソースエンドポイントを使用できます。Mongo DB ソースエンドポイントについては、「」を参照してくださいMongoDB の DMS ソースエンドポイント: 作成。

4: AWS DMS ターゲットエンドポイント: S3 用に作成、作成

AMS コンソールまたは API/CLI を使用して、さまざまなデータベースの AMS DMS ターゲットエンドポイントを作成できます。2 つの例を示します。

DMS ターゲットエンドポイント: 作成

AMS DMS は、ターゲットエンドポイントとして MySQL、MariaDB、Oracle、Postgresql、または Microsoft SQL で S3 または任意のリレーショナルデータベースサービス (RDS) を使用できます。

コンソールを使用した DMS ターゲットエンドポイントの作成

AMS コンソールでのこの変更タイプのスクリーンショット：

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの Browse change types ビューで一般的な変更タイプ (CT) を選択するか、Choose by category ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプ領域を使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。

- カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。

3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定領域を開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters エリアを開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した DMS ターゲットエンドポイントの作成

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成し、1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id` *`ID`* コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id` *`ID`* コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたとき

に通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

テンプレートの作成：

1. この変更タイプの実行パラメータを CreateDmsTeParams.json.

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

2. 実行パラメータ JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

```
    "Password":         "DB-PW",}
    }
}
```

3. JSON テンプレートを現在のフォルダ内のファイルに出力します。この例では CreateDmsTeRfc.json:

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

4. CreateDmsTeRfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

5. RFC を作成し、実行パラメータファイルと CreateDmsTeRfc ファイルを指定します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

- この変更タイプはバージョン 2.0 になりました。
- AMS DMS は、ターゲットエンドポイントとして MySQL、MariaDB、Oracle、Postgresql、または Microsoft SQL で S3 または任意のリレーショナルデータベースサービス (RDS) を使用できます。S3 ターゲットエンドポイントについては、「」を参照してくださいS3 の DMS ターゲットエンドポイント: 作成。
- 詳細については、「データ移行のターゲット」を参照してください。
- 最大 50 個のタグを追加できますが、追加設定ビューを有効にする必要があります。

S3 の DMS ターゲットエンドポイント: 作成

コンソールを使用した DMS S3 ターゲットエンドポイントの作成

AMS コンソールでのこの変更タイプのスクリーンショット：

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプエリアを使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。
   - カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。
3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters 領域を開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した DMS S3 ターゲットエンドポイントの作成

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成し、1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。
2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id ID` コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id ID` コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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\":\"amzn-s3-demo-bucket\",
\"S3ServiceAccessRoleArn\":\"arn:aws:iam::123456789123:role/my-s3-role\"}}"
```

テンプレートの作成：

1. この変更タイプの実行パラメータを JSON ファイルに出力します。この例では、CreateDmsTeS3Params.json:

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

2. 実行パラメータ CreateDmsTeS3Params.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
"Description":          "TestS3DMS-TE",
"VpcId":                "VPC_ID",
"StackTemplateId":      "stm-knghtmmgefafdq89u",
"Name":                 "DMS-S3-TE",
"TimeoutInMinutes":     60,
"Parameters":   {
    "EngineName":       "s3",
    "S3BucketName":      "amzn-s3-demo-bucket",
    "S3ServiceAccessRoleArn":       "arn:aws:iam::123456789101:role/ams-ops-ct-
authors-dms-s3-test-role"
      }
}
```

3. JSON テンプレートを現在のフォルダ内のファイルに出力します。この例では CreateDmsTeS3Rfc.json:

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

4. CreateDmsTeS3Rfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

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

5. RFC を作成し、実行パラメータファイルと CreateDmsTeS3Rfc ファイルを指定します。

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

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

> **Note**
>
> 最大 50 個のタグを追加できますが、追加設定ビューを有効にする必要があります。

AMS には、S3 のターゲットエンドポイントを作成するための別の変更タイプが用意されています。詳細については、AWS Database Migration Service のターゲットとしての Amazon S3 の使用」および「AWS DMS のターゲットとして Amazon S3 を使用する場合の追加の接続属性」を参照してください。

5: AWS DMS レプリケーションタスク: 作成

AMS コンソールまたは API/CLI を使用して AMS AWS DMS レプリケーションタスクを作成できます。

AWS DMS レプリケーションタスクの作成

コンソールを使用した AWS DMS レプリケーションタスクの作成

AMS コンソールでのこの変更タイプのスクリーンショット：

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。

- 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

  CTsソートするには、カードビューまたはテーブルビューですべての変更タイプ領域を使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。

- カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。

3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。

4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。

5. Run parameters エリアを開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した AWS DMS レプリケーションタスクの作成

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含む`create-rfc`コマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成し、1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力として`create-rfc`コマンドを発行します。どちらの方法もここで説明します。

2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id` *`ID`* コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id` *`ID`* コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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}"
```

テンプレートの作成：

1. この変更タイプの実行パラメータを JSON ファイルに出力します。この例では、CreateDmsRtParams.json:

   ```
   aws amscm get-change-type-version --change-type-id "ct-1d2fml15b9eth" --query
    "ChangeTypeVersion.ExecutionInputSchema" --output text > CreateDmsRtParams.json
   ```

2. 実行パラメータ JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
"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. JSON テンプレートを現在のフォルダ内のファイルに出力します。この例では CreateDmsRtRfc.json:

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

4. CreateDmsRtRfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
"ChangeTypeVersion":    "1.0",
"ChangeTypeId":         "ct-1d2fml15b9eth",
"Title":                "DMS-RI-Create-RFC"
}
```

5. RFC を作成し、実行パラメータファイルと CreateDmsRtRfc ファイルを指定します。

```
aws amscm create-rfc --cli-input-json file://CreateDmsRtRfc.json --execution-
parameters file://CreateDmsRtParams.json
```

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

3 つの異なるタイプの変更またはデータをキャプチャする AWS DMS タスクを作成できます。詳細については、「AWS DMS タスクの使用」、「タスクの作成」、「AWS DMS を使用した継続的なレプリケーション用のタスクの作成」を参照してください。

AWS DMS 管理

AWS DMS 管理の例。

AWS DMS レプリケーションタスクを開始する

コンソールで AWS DMS レプリケーションタスクを開始する

AMS コンソールでのこの変更タイプのスクリーンショット：

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプエリアを使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。
   - カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。

3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。

4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。

5. Run parameters 領域を開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI で AWS DMS レプリケーションタスクを開始する

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成します。1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。

2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id` *`ID`* コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id` *`ID`* コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) --notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com

\"]}}"にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符を エスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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\":[\"\"]}}"
```

テンプレートの作成：

1. この変更タイプの実行パラメータを JSON ファイルに出力します。この例では StartDmsRtParams.json:

   ```
   aws amscm get-change-type-version --change-type-id "ct-1yq7hhqse71yg" --query
    "ChangeTypeVersion.ExecutionInputSchema" --output text > StartDmsRtParams.json
   ```

2. 実行パラメータ JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

   ```
   {
     "DocumentName": "AWSManagedServices-StartDmsTask",
     "Region": "us-east-1",
     "Parameters": {
         "ReplicationTaskArn": [
           "TASK_ARN"
         ],
         "StartReplicationTaskType": [
           "start-replication"
         ],
         "CdcStartPosition": [
           ""
         ],
         "CdcStopPosition": [
   ```

```
        ""
      ]
  }
}
```

3. JSON テンプレートを現在のフォルダ内のファイルに出力します。この例では StartDmsRtRfc.json:

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

4. StartDmsRtRfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
  "ChangeTypeId": "ct-1yq7hhqse71yg",
  "ChangeTypeVersion": "1.0",
  "Title": "Start DMS Replication Task"
}
```

5. RFC を作成し、実行パラメータファイルと StartDmsRtRfc ファイルを指定します。

```
aws amscm create-rfc --cli-input-json file://StartDmsRtRfc.json --execution-
parameters file://StartDmsRtParams.json
```

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

AMS コンソールまたは AMS API/CLI を使用して、 AWS DMS レプリケーションタスクを開始できます。詳細については、「AWS DMS タスクの使用」を参照してください。

AWS DMS レプリケーションタスクを停止する

コンソールでの AWS DMS レプリケーションタスクの停止

AMS コンソールでのこの変更タイプのスクリーンショット：

仕組み：

1. RFC の作成ページに移動します。AMS コンソールの左側のナビゲーションペインでRFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。
2. デフォルトの変更タイプ参照ビューで一般的な変更タイプ (CT) を選択するか、カテゴリ別選択ビューで CT を選択します。
   - 変更タイプ別に参照: クイック作成エリアで一般的な CT をクリックすると、すぐに RFC の実行ページを開くことができます。クイック作成で古い CT バージョンを選択することはできません。

     CTs をソートするには、カードビューまたはテーブルビューですべての変更タイプ領域を使用します。どちらのビューでも、CT を選択し、RFC の作成をクリックして RFC の実行ページを開きます。必要に応じて、RFC の作成ボタンの横に古いバージョンで作成オプションが表示されます。
   - カテゴリ別に選択: カテゴリ、サブカテゴリ、項目、オペレーションを選択すると、CT 詳細ボックスが開き、必要に応じて古いバージョンで作成するオプションが表示されます。RFC の作成をクリックして、RFC の実行ページを開きます。
3. RFC の実行ページで、CT 名エリアを開き、CT の詳細ボックスを表示します。件名は必須です (変更タイプの参照ビューで CT を選択した場合は入力されます) 。追加設定エリアを開き、RFC に関する情報を追加します。

   実行設定領域で、使用可能なドロップダウンリストを使用するか、必要なパラメータの値を入力します。オプションの実行パラメータを設定するには、追加設定エリアを開きます。
4. 完了したら、実行 をクリックします。エラーがない場合、RFC が正常に作成されたページに、送信された RFC の詳細と最初の実行出力が表示されます。
5. Run parameters エリアを開き、送信した設定を確認します。ページを更新して RFC 実行ステータスを更新します。必要に応じて、RFC をキャンセルするか、ページ上部のオプションを使用してコピーを作成します。

CLI を使用した AWS DMS レプリケーションタスクの停止

仕組み：

1. インライン作成 (すべての RFC と実行パラメータを含むcreate-rfcコマンドを発行) またはテンプレート作成 (2 つの JSON ファイルを作成します。1 つは RFC パラメータ用、もう 1 つは実行パラメータ用) のいずれかを使用し、2 つのファイルを入力としてcreate-rfcコマンドを発行します。どちらの方法もここで説明します。

2. 返された RFC ID を使用して RFC: `aws amscm submit-rfc --rfc-id ID` コマンドを送信します。

   RFC: `aws amscm get-rfc --rfc-id ID` コマンドをモニタリングします。

変更タイプのバージョンを確認するには、次のコマンドを使用します。

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

> **Note**
>
> 変更タイプのスキーマの一部であるかどうかにかかわらず、任意の RFC で任意のCreateRfcパラメータを使用できます。たとえば、RFC ステータスが変更されたときに通知を受け取るには、リクエストの RFC パラメータ部分 (実行パラメータではなく) `--notification "{\"Email\": {\"EmailRecipients\" : [\"email@example.com\"]}}"`にこの行を追加します。すべての CreateRfc パラメータのリストについては、AMS 変更管理 API リファレンスを参照してください。

インライン作成：

インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を指定して create RFC コマンドを発行し、返された RFC ID を送信します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
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\"]}}"
```

テンプレートの作成：

1. この変更タイプの実行パラメータを JSON ファイルに出力します。この例では、StopDmsRtParams.json:

   ```
   aws amscm get-change-type-version --change-type-id "ct-1vd3y4ygbqmfk" --query
    "ChangeTypeVersion.ExecutionInputSchema" --output text > StopDmsRtParams.json
   ```

2. 実行パラメータ JSON ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
  "DocumentName": "AWSManagedServices-StopDmsTask",
  "Region": "us-east-1",
  "Parameters": {
      "ReplicationTaskArn": [
        "TASK_ARN"
      ]
  }
}
```

3. JSON テンプレートを現在のフォルダ内のファイルに出力します。この例では StopDmsRtRfc.json:

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

4. StopDmsRtRfc.json ファイルを変更して保存します。たとえば、コンテンツを次のような内容に置き換えることができます。

```
{
  "ChangeTypeId": "ct-1vd3y4ygbqmfk",
  "ChangeTypeVersion": "1.0",
  "Title": "Stop DMS Replication Task"
}
```

5. RFC を作成し、実行パラメータファイルと StopDmsRtRfc ファイルを指定します。

```
aws amscm create-rfc --cli-input-json file://StopDmsRtRfc.json --execution-
parameters file://StopDmsRtParams.json
```

レスポンスで新しい RFC の ID を受け取り、それを使用して RFC を送信およびモニタリングできます。送信するまで、RFC は編集状態のままであり、開始されません。

ヒント

AMS コンソールまたは AMS API/CLI を使用して、DMS レプリケーションタスクを停止できます。詳細については、「AWS DMS タスクの使用」を参照してください。

AMS RDS for Microsoft SQL Server へのデータベース (DB) のインポート

> **Note**
>
> AMS API/CLI (amscm および amsskms) エンドポイントは、AWS N. Virginia リージョン にありますus-east-1。認証の設定方法や、アカウントとリソースが存在する AWS リージョンによっては、コマンドの発行--region us-east-1時に追加が必要になる場合があります。認証方法である場合は--profile saml、 を追加する必要がある場合もあります。

AMS RDS for SQL Server への DB インポート、プロセスは、変更リクエスト (RFCsCTs) に依存し、Amazon RDS API パラメータを入力として使用します。MicroSoft SQL Server は、リレーショナルデータベース管理システム (RDBMS) です。詳細については、Amazon Relational Database Service (Amazon RDS)」および「rds」または「Amazon RDS API リファレンス」も参照してください。

> **Note**
>
> 次のステップに進む前に、各 RFC が正常に完了していることを確認してください。

高レベルのインポートステップ：

1. ソースのオンプレミス MS SQL データベースを .bak (バックアップ) ファイルにバックアップする
2. .bak ファイルをトランジット (暗号化) Amazon Simple Storage Service (S3) バケットにコピーします。
3. ターゲット Amazon RDS MS SQL インスタンスの新しい DB に .bak をインポートする

要件:

- AMS の MS SQL RDS スタック
- 復元オプション付きの RDS スタック (SQLSERVER_BACKUP_RESTORE)
- トランジット S3 バケット
- Amazon RDS がロールを引き受けることができるバケットアクセスを持つ IAM ロール

- RDS を管理するために MS SQL Management Studio がインストールされている EC2 インスタンス (オンプレミスのワークステーションでもかまいません)

セットアップ

インポートプロセスを開始するには、これらのタスクを完了します。

1. RFC を送信して、デプロイ | 高度なスタックコンポーネント | RDS データベーススタック | 作成 (ct-2z60dyvto9g6c) を使用して RDS スタックを作成します。作成リクエストでターゲット DB 名 (パラメータ) を使用しないでください。ターゲット DB はインポート中に作成されます。`RDSDBName`十分なスペース (`RDSAllocatedStorage` パラメータ) を確保してください。これを行う方法の詳細については、「AMS 変更管理ガイド RDS DB スタック | 作成」を参照してください。
2. デプロイ | 高度なスタックコンポーネント | S3 ストレージ | 作成 (ct-1a68ck03fn98r) を使用して、RFC を送信してトランジット S3 バケットを作成します (まだ存在しない場合) 。これを行う方法の詳細については、「AMS 変更管理ガイド S3 ストレージ | 作成」を参照してください。
3. 管理を送信する | その他 | その他 | RFC (ct-1e1xtak34nx76) を更新して、以下の詳細`customer_rds_s3_role`を含む を実装します。

   コンソールで:

   - 件名 :「MS SQL Server データベースのインポートをサポートするには、このアカウントに を実装`customer_rds_s3_role`します。
   - トランジット S3 バケット名: *`BUCKET_NAME`*。
   - 連絡先情報: *`EMAIL`*。

   CLI の ImportDbParams.json ファイルの場合 :

   ```
   {
          "Comment": "{"Transit S3 bucket name":"BUCKET_NAME"}",
          "Priority": "High"
        }
   }
   ```

4. 管理を送信する | その他 | その他 | ステップ 1 で作成した RDS に`SQLSERVER_BACKUP_RESTORE`オプションを設定するように AMS にリクエストする RFC を更新します (このリクエストでは、ステップ 1 の出力のスタック ID と、このリクエストの `customer_rds_s3_role` IAM ロールを使用します) 。

5. RFC を送信して EC2 インスタンスを作成し (既存の EC2 またはオンプレミスのワークステーション/インスタンスを使用できます)、インスタンスに Microsoft SQL Management Studio をインストールします。

データベースのインポート

データベース (DB) をインポートするには、次の手順に従います。

1. MS SQL ネイティブバックアップと復元を使用して、ソースのオンプレミスデータベースをバックアップします (SQL Server でのネイティブバックアップと復元のサポートを参照)。そのオペレーションを実行した結果、.bak (バックアップ) ファイルが必要です。
2. AWS S3 CLI または AWS S3 コンソールを使用して、.bak ファイルを および既存のトランジット S3 バケットにアップロードします。トランジット S3 バケットの詳細については、「暗号化を使用したデータの保護」を参照してください。
3. .bak ファイルをターゲット RDS for SQL Server MS SQL インスタンスの新しい DB にインポートします (タイプの詳細については、「Amazon RDS for MySQL インスタンスタイプ」を参照してください)。

    a. EC2 インスタンス (オンプレミスワークステーション) にログインし、MS SQL Management Studio を開きます。

    b. ステップ 1 で前提条件として作成されたターゲット RDS インスタンスに接続します。接続するには、次の手順に従います。Microsoft SQL Server データベースエンジンを実行している DB インスタンスへの接続

    c. 新しい構造化クエリ言語 (SQL) クエリを使用してインポート (復元) ジョブを開始します (SQL クエリの詳細については、「SQL の概要」を参照してください)。ターゲットデータベース名は新しい名前にする必要があります (以前に作成したデータベースと同じ名前を使用しないでください)。暗号化なしの例：

    ```
    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. このクエリを別のウィンドウで実行して、インポートジョブのステータスを定期的に確認します。

    ```
    exec msdb.dbo.rds_task_status;
    ```

ステータスが Failed に変わった場合は、メッセージで失敗の詳細を探します。

クリーンアップ

データベースをインポートしたら、不要なリソースを削除できます。次のステップに従います。

1. S3 バケットからバックアップファイル (.bak) を削除します。これを行うには、S3 コンソールを使用します。S3 バケットからオブジェクトを削除する CLI コマンドについては、AWS CLI コマンドリファレンスの「rm」を参照してください。
2. 使用する予定がない場合は、S3 バケットを削除します。その手順については、「スタックの削除」を参照してください。
3. MS SQL インポートを実行する予定がない場合は、管理 | その他 | その他 | 更新 (ct-0xdawir96cy7k) RFC を送信し、AMS に IAM ロール の削除をリクエストしますcustomer_rds_s3_role。

AMS でのアプリデプロイの階層化と階層化

階層と階層のデプロイでは、個別の RFCs を使用してスタックのリソースを個別に作成、設定、デプロイし、進行時にスタックコンポーネントの IDs を使用して相互に関連付けます。

たとえば、高可用性 (冗長) ウェブサイトをロードバランサーの背後、データベースをデプロイするには、Tier と Tier のアプローチを使用して、データベースとロードバランサーの RFCs、および 2 つの EC2 インスタンスまたは Auto Scaling グループを送信し、作成した ELB の ID を使用して EC2 インスタンスまたは Auto Scaling グループを設定します。

リソースがデプロイされたら、セキュリティグループの作成変更を送信して、リソースがデータベースと通信できるようにします。セキュリティグループの作成の詳細については、「セキュリティグループの作成」を参照してください。

AMS でのフルスタックアプリケーションのデプロイ

フルスタックデプロイでは、必要なすべてを一度に作成して設定する CT を使用して RFC を送信します。例えば、前述の高可用性ウェブサイト (EC2 インスタンス、ロードバランサー、データベース) をデプロイするには、Auto Scaling グループ、ロードバランサー、データベース、およびすべてのインスタンスがスタックとして機能するために必要なセキュリティグループ設定を作成および設定した CT を使用します。これを行う 2 つの AMS CTs の例を次に示します。

- 高可用性 2 層スタック (ct-06mjngx5flwto): この変更タイプでは、スタックを作成し、Auto Scaling グループ、RDS-backed データベース、Load Balancer、CodeDeploy アプリケーションと設定を設定できます。ロードバランサーは、ネットワークアプライアンスとして複数のアプリケーション間で共有され、CodeDeploy 関数もアプライアンスと見なされるため、階層とは見なされないことに注意してください。さらに、アプリケーションのデプロイに使用できる CodeDeploy デプロイグループ (CodeDeploy アプリケーションに付けた名前) を作成します。リソースを一緒に機能させるセキュリティグループ設定が自動的に作成されます。
- 高可用性 1 層スタック (ct-09t6q7j9v5hrn): この変更タイプでは、スタックを作成し、Auto Scaling グループと Application Load Balancer を設定できます。リソースを一緒に機能させるセキュリティグループ設定が自動的に作成されます。

プロビジョニング変更タイプ (CTsの使用

AMS はマネージドインフラストラクチャに責任を負います。変更を加えるには、正しい CT 分類 (カテゴリ、サブカテゴリ、項目、オペレーション) の RFC を送信する必要があります。このセクションでは、CTs を検索し、ニーズに合ったものがあるかどうかを判断し、ニーズがない場合は新しい CT をリクエストする方法について説明します。

既存の CT が要件を満たしているかどうかを確認する

AMS でデプロイする内容を決定したら、次のステップとして、既存の CTs と CloudFormation テンプレートを調べて、ソリューションがすでに存在するかどうかを確認します。

RFC を作成するときは、CT を指定する必要があります。 AWS マネジメントコンソール または AMS API/CLI を使用できます。両方を使用する例を次に示します。

コンソールまたは API/CLI を使用して、変更タイプ ID (CT) またはバージョンを検索できます。検索または分類の選択の 2 つの方法があります。どちらの選択タイプでも、最も頻繁に使用される、最近使用された、またはアルファベットのいずれかを選択して検索をソートできます。

YouTube 動画: AWS Managed Services CLI を使用して RFC を作成する方法と、CT スキーマの場所を教えてください。

AMS コンソールの RFCs -> RFC の作成ページで、次の操作を行います。

- 変更タイプによる参照 (デフォルト) を選択した場合、次のいずれかを実行します。

- クイック作成エリアを使用して、AMS の最も一般的な CTs から選択します。ラベルをクリックすると、RFC の実行ページが開き、件名オプションが自動的に入力されます。必要に応じて残りのオプションを完了し、実行をクリックして RFC を送信します。
- または、すべての変更タイプ領域までスクロールダウンし、オプションボックスに CT 名を入力し始めます。変更タイプ名を完全または完全に設定する必要はありません。関連する単語を入力して、変更タイプ ID、分類、または実行モード (自動または手動) で CT を検索することもできます。

  デフォルトのカードビューを選択すると、入力時に一致する CT カードが表示され、カードを選択して RFC の作成をクリックします。テーブルビューを選択し、関連する CT を選択し、RFC の作成をクリックします。どちらの方法でも、RFC の実行ページが開きます。

- または、変更タイプの選択肢を確認するには、ページ上部の「カテゴリ別に選択」をクリックして、一連のドロップダウンオプションボックスを開きます。
- カテゴリ、サブカテゴリ、項目、オペレーションを選択します。その変更タイプの情報ボックスは、ページの下部にパネルが表示されます。
- 準備ができたら、Enter キーを押すと、一致する変更タイプのリストが表示されます。
- リストから変更タイプを選択します。その変更タイプの情報ボックスがページの下部に表示されます。
- 正しい変更タイプを取得したら、RFC の作成を選択します。

> **Note**
>
> これらのコマンドを機能させるには、AMS CLI をインストールする必要があります。AMS API または CLI をインストールするには、AMS コンソールのデベロッパーリソースページに移動します。AMS CM API または AMS SKMS API のリファレンスマテリアルについては、「 ユーザーガイド」の「AMS 情報リソース」セクションを参照してください。認証--profileのオプションを追加する必要がある場合があります。例: `aws amsskms ams-cli-command --profile SAML`。また、 など、すべての AMS コマンドが us-east-1 を使い果たすため、 `--region`オプションを追加する必要がある場合がありますaws amscm `ams-cli-command --region=us-east-1`。

> **Note**
>
> AMS API/CLI (amscm および amsskms) エンドポイントは、AWS N. Virginia リージョン にありますus-east-1。認証の設定方法や、アカウントとリソースが存在する AWS リージョンによっては、コマンドの発行--region us-east-1時に追加が必要になる場合があります。認証方法である場合は--profile saml、 を追加する必要がある場合もあります。

AMS CM API (ListChangeTypeClassificationSummaries を参照) または CLI を使用して変更タイプを検索するには：

フィルターまたはクエリを使用して検索できます。ListChangeTypeClassificationSummaries オペレーションには、Category、、SubcategoryItem、および のフィルターオプションがありますがOperation、値は既存の値と完全に一致する必要があります。CLI を使用する際のより柔軟な結果を得るには、 --queryオプションを使用できます。

AMS CM API/CLI を使用した変更タイプのフィルタリング

属性	有効値	有効/デフォルト条件	注意
ChangeTypeId	ChangeTypeId を表す任意の文字列 (例: ct-abc123xyz7890)	Equals (等しい)	変更タイプ IDs「変更タイプリファレンス」を参照してください。 変更タイプ IDs「変更タイプまたは CSIO の検索」を参照してください。
カテゴリ	任意の自由形式のテキスト	を含む	個々のフィールドの正規表現はサポートされていません。大文字と小文字を区別しない検索
サブカテゴリ			
Item			
Operation			

1. 変更タイプの分類を一覧表示する例をいくつか示します。

次のコマンドは、すべての変更タイプカテゴリを一覧表示します。

```
aws amscm list-change-type-categories
```

次の コマンドは、指定されたカテゴリに属するサブカテゴリを一覧表示します。

```
aws amscm list-change-type-subcategories --category CATEGORY
```

次の コマンドは、指定されたカテゴリとサブカテゴリに属する項目を一覧表示します。

```
aws amscm list-change-type-items --category CATEGORY --subcategory SUBCATEGORY
```

2. CLI クエリで変更タイプを検索する例をいくつか示します。

   次のコマンドは、項目名に「S3」が含まれている CT 分類の概要を検索し、カテゴリ、サブカテゴリ、項目、オペレーション、変更タイプ ID の出力をテーブル形式で作成します。

```
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. その後、変更タイプ ID を使用して CT スキーマを取得し、パラメータを調べることができます。次のコマンドは、CreateS3Params.schema.json.

```
aws amscm get-change-type-version --change-type-id "ct-1a68ck03fn98r"
 --query "ChangeTypeVersion.ExecutionInputSchema" --output text >
 CreateS3Params.schema.json
```

CLI クエリの使用の詳細については、「--query Option で出力をフィルタリングする方法」および「クエリ言語リファレンス」のJMESPath Specification」を参照してください。

4. 変更タイプ ID を取得したら、変更タイプのバージョンを検証して最新バージョンであることを確認することをお勧めします。次のコマンドを使用して、指定された変更タイプのバージョンを検索します。

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

AutomationStatus 特定の変更タイプの を検索するには、次のコマンドを実行します。

```
aws amscm --profile saml get-change-type-version --change-type-id CHANGE_TYPE_ID --
query "ChangeTypeVersion.{AutomationStatus:AutomationStatus.Name}"
```

ExpectedExecutionDurationInMinutes 特定の変更タイプの を検索するには、次のコマンドを実行します。

```
aws amscm --profile saml get-change-type-version --change-type-id ct-14027q0sjyt1h
 --query "ChangeTypeVersion.{ExpectedDuration:ExpectedExecutionDurationInMinutes}"
```

適切と思われる CT を見つけたら、それに関連付けられた実行パラメータ JSON スキーマを調べて、ユースケースに対処しているかどうかを確認します。

このコマンドを使用して、CT スキーマを CT にちなんで という名前の JSON ファイルに出力します。この例では、Create S3 storage schema を出力します。

```
aws amscm get-change-type-version --change-type-id "ct-1a68ck03fn98r"
--query "ChangeTypeVersion.ExecutionInputSchema" --output text >
CreateBucketParams.json
```

このスキーマが提供する内容を詳しく見てみましょう。

S3 バケットスキーマの作成

```
{
  "$schema": "http://json-schema.org/draft-04/sch
ema#",
"name": "Create S3 Storage
"description": "Use to create an Amazon Simple
 Storage Service stack.",
  "type": "object",
```

スキーマは、スキーマの目的を示す CT (「説明」) で始まります。この場合、S3 ストレージスタックを作成します。

```
  "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-s2b72
beb000000000.",
      "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
```

次に、指定できる必須プロパティとオプションプロパティがあります。デフォルトのプロパティ値が指定されています。必要なプロパティは、スキーマの最後に一覧表示されます。

StackTemplateId エリアでは、この CT とスキーマに 1 つの特定のスタックテンプレートがあり、その ID が必須のプロパティ値であることがわかります。

スキーマを使用すると、内部の簿記目的で、作成するスタックにタグを付けることができます。さらに、バックアップなどの一部のオプションでは、Key:backup と Value:true のタグが必要です。詳細については、Amazon EC2 リソースのタグ付け」を参照してください。

```
          }
        },
        "additionalProperties": false,
        "required": [
          "Key",
          "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-z
0-9]$",
          "minLength": 3,
          "maxLength": 63
        }
```

CT JSON スキーマの Parameters セクションでは、実行パラメータを指定します。

このスキーマでは、ACL と BucketName のみが必須の実行パラメータです。

```
      },
      "additionalProperties": false,
      "required": [
        "AccessControl",
        "BucketName"
      ]
    }
  },
  "additionalProperties": false,
  "required": [
    "Description",
    "VpcId",
    "StackTemplateId",
    "Name",
    "TimeoutInMinutes",
    "Parameters"
  ]
}
```

新しい CT をリクエストする

スキーマを調べた後、必要なデプロイを作成するのに十分なパラメータが提供されていないと判断できます。その場合は、既存の CloudFormation テンプレートを調べて、目的に近いものを見つけます。必要な追加パラメータがわかったら、管理 | その他 | その他 | CT を作成します。

> **Note**
>
> All Other | Other Create and Update CTs は、AMS オペレーターから通知を受け、オペレーターから連絡を受けて新しい CT について話し合います。

新しい CT のリクエストを送信するには、通常の から AMS コンソールにアクセスしAWS マネジメントコンソール、以下の手順に従います。

1. 左側のナビゲーションで、RFCsをクリックします。

   RFCsページが開きます。
2. [Create (作成)] をクリックします。

   変更リクエストの作成ページが開きます。

3. 「カテゴリ」ドロップダウンリストの「管理」を選択し、「サブカテゴリと項目」の「その他」を選択します。 オペレーションで、作成 を選択します。RFC を実装するには、事前に承認が必要です。
4. CT が必要な理由の情報を入力します。たとえば、既存の Create S3 storage CT に基づいてカスタム ACLs を許可する、変更された Create S3 storage CT をリクエストします。これにより、新しい CT: Deployment | Advanced Stack Components | S3 ストレージ | Create S3 custom ACL が作成されます。この新しい CT は公開されている可能性があります。
5. 送信 をクリックします。

   RFC ダッシュボードに RFC が表示されます。

新しい CT をテストする

AWS Managed Services がその新しい CT を作成したら、RFC を送信してテストします。AMS を使用して新しい CT を事前承認した場合は、標準の RFC 送信に従って結果を監視できます (RFCs」を参照してください)。 https://docs.aws.amazon.com/managedservices/latest/userguide/create-rfcs.html新しい CT が事前承認されていない場合 (明示的な承認なしに実行されないようにしたい場合)、その実装を実行するたびに AMS と話し合う必要があります。

クイックスタート

トピック

AMS 変更タイプを組み合わせて使用すると、複雑なタスクを実行できます。

AMS 変更管理システムを使用して、マルチアカウントランディングゾーン (MALZ) またはシングルアカウントランディングゾーン (SALZ) アカウントに AMS リソーススケジューラを設定できます。プロセスは異なります。また、ファイル転送とクロスアカウントスナップショットも実行します。

AMS Resource Scheduler クイックスタート

このクイックスタートガイドを使用して、AMS リソーススケジューラを実装します。これは、AMS Advanced のコストを節約するためのタグベースのインスタンススケジューラです。

AMS リソーススケジューラは AWS インスタンススケジューラに基づいています。

AMS Resource Scheduler の用語

開始する前に、AMS Resource Scheduler の用語を理解しておくことをお勧めします。

- period: 各スケジュールには、インスタンスが実行する時間を定義する期間が少なくとも 1 つ含まれている必要があります (複数可)。スケジュールには複数の期間を含めることができます。スケジュールで複数の期間を使用する場合、Resource Scheduler は、期間ルールの少なくとも 1 つが true であるときに適切な開始アクションを適用します。
- timezone: 後で参照する DefaultTimezone パラメータで使用する許容可能なタイムゾーン値のリストについては、TZ データベースタイムゾーンのリストの TZ 列を参照してください。
- 休止： 休止が有効で、休止要件を満たす true EC2 インスタンスに設定すると、休止状態になります (suspend-to-disk)。EC2 コンソールで、インスタンスで休止が有効になっているかどうかを確認します。Amazon Linux を実行している停止した Amazon EC2 インスタンスには休止を使用します。
- enforced: true に設定すると、定義されたスケジュールに基づいて、Resource Scheduler は実行中のリソースが実行期間外に手動で開始された場合に実行中のリソースを停止し、実行中に手動で停止された場合にリソースを開始します。

- retain_running: true に設定すると、インスタンスが期間の開始前に手動で開始された場合、Resource Scheduler は実行期間の終了時にインスタンスを停止できなくなります。例えば、午前 9 時から午後 5 時まで実行される期間が設定されたインスタンスが午前 9 時までに手動で起動された場合、Resource Scheduler は午後 5 時にインスタンスを停止しません。
- ssm-maintenance-window: スケジュールに実行期間として AWS Systems Manager メンテナンスウィンドウを追加します。デプロイされたスタックと同じアカウントと AWS リージョンに存在するメンテナンスウィンドウの名前を指定して Amazon EC2 インスタンスをスケジュールすると、Resource Scheduler はメンテナンスウィンドウの開始前にインスタンスを起動し、メンテナンスウィンドウの最後にインスタンスを停止します。他の実行期間でインスタンスの実行が指定されていない場合、およびメンテナンスイベントが完了した場合です。

  Resource Scheduler は、初期設定時に指定した AWS Lambda 頻度を使用して、メンテナンスウィンドウがインスタンスを起動するまでの時間を決定します。頻度 AWS CloudFormation パラメータを 10 分以下に設定した場合、Resource Scheduler はメンテナンスウィンドウの 10 分前にインスタンスを起動します。頻度を 10 分以上に設定すると、Resource Scheduler は指定した頻度と同じ分数でインスタンスを起動します。例えば、Systems Manager メンテナンスウィンドウの頻度を 30 分に設定すると、Resource Scheduler はメンテナンスウィンドウの 30 分前にインスタンスを起動します。

  詳細については、AWS Systems Manager 「メンテナンスウィンドウ」を参照してください。
- override-status: Resource Scheduler の設定済みスケジュールの開始および停止アクションを一時的に上書きします。フィールドを実行に設定した場合、Resource Scheduler は該当するインスタンスを起動しますが、停止はしません。インスタンスは、手動で停止するまで実行されます。override-status を停止に設定すると、Resource Scheduler は停止しますが、該当するインスタンスは起動しません。インスタンスは、手動で起動するまで実行されません。

AMS Resource Scheduler の実装

AMS リソーススケジューラソリューションをデプロイするには、次の手順に従います。

1. デプロイを送信する | AMS リソーススケジューラ | ソリューション | RFC (ct-0ywnhc8e5k9z5) をデプロイし、次のパラメータを指定します。

   - SchedulingActive: リソーススケジューリングを有効にするにははい、無効にするにはいいえ。デフォルトは [Yes] (はい) です。

- ScheduledServices: リソースをスケジュールするサービスのカンマ区切りリストを入力します。有効な値には、Auto Scaling、ec2、rds の組み合わせが含まれます。デフォルトは Auto Scaling,ec2,rds です。
- TagName: リソーススケジュールスキーマをサービスリソースに関連付けるタグキーの名前。デフォルトはスケジュールです。

> **Note**
>
> Resource Scheduler のデプロイは、このタグを持つリソースでのみ動作します。

- DefaultTimezone: デフォルトのタイムゾーンとして使用する米国/太平洋の形式のタイムゾーンの名前。デフォルトは UTC です。

2. ステップ 1 の RFC が正常に実行されたという確認を受け取ったら、期間 | 変更タイプを追加 を送信できます。
3. 最後に、RFC を送信して、ステップ 2 で作成された期間にスケジュールを追加します。スケジュールを使用する | 変更タイプを追加します。

AMS Resource Scheduler の実装と使用FAQs

AMS Resource Scheduler に関するよくある質問。

Q: 休止を有効にしても EC2 インスタンスがサポートしていない場合はどうなりますか？

A: 休止は、インスタンスメモリ (RAM) から Amazon Elastic Block Store (Amazon EBS) ルートボリュームにコンテンツを保存します。このフィールドが true に設定されている場合、Resource Scheduler がインスタンスを停止すると、インスタンスは休止状態になります。

休止を使用するように Resource Scheduler を設定しても、インスタンスが休止に対して有効になっていないか、休止の前提条件を満たしていない場合、Resource Scheduler は警告を口グに記録し、インスタンスは休止なしで停止します。詳細については、「インスタンスの休止」を参照してください。

Q: override_status と enforced の両方を設定した場合どうなりますか？

A: override_status を running に設定し、forced を true に設定すると (インスタンスが実行期間外に手動で開始されるのを防ぐ) 、Resource Scheduler はインスタンスを停止します。

override_status を停止に設定し、enforced を true に設定すると (実行中にインスタンスが手動で停止するのを防ぐ) 、Resource Scheduler はインスタンスを再起動します。

> **Note**
>
> 強制が false の場合、設定されたオーバーライド動作が適用されます。

Q: AMS Resource Scheduler がデプロイされた後、アカウントでリソーススケジューラを無効化または有効にする方法を教えてください。

A: AMS Resource Scheduler を無効または有効にするには：

- 無効にするには: State | Disable を使用して RFC を作成します。SchedulerState を DISABLE に設定してください
- 有効にするには: State | Enable を使用して RFC を作成します。SchedulerState を ENABLE に設定してください

Q AMS Resource Scheduler の期間がパッチ適用メンテナンスウィンドウ内にある場合はどうなりますか？

A: Resource Scheduler は、設定されたスケジュールに基づいて動作します。パッチ適用が実行中にインスタンスを停止するように設定されている場合、パッチ適用ウィンドウがパッチ適用を開始する前にスケジュールに期間として追加されない限り、インスタンスは停止します。つまり、Resource Scheduler は、指定された期間が設定されていない限り、パッチ適用のために停止したインスタンスを自動起動しません。パッチ適用メンテナンスウィンドウとの競合を回避するには、パッチ適用に割り当てられた時間枠を期間として Resource Scheduler スケジュールに追加します。既存のスケジュールに期間を追加するには、Period | Add を使用して RFC を作成します。

Q EC2 インスタンスごとに異なるスケジュールが必要な場合は、アカウント内で複数のスケジュールを設定できますか？

A: はい、複数のスケジュールを作成できます。各スケジュールは、要件に基づいて複数の期間を持つことができます。アカウントで AMS リソーススケジューラが有効になっている場合、タグキーが設定されます。例えば、タグキーが「スケジュール」の場合、タグ値は AMS リソーススケジューラのスケジュール名に対応するさまざまなスケジュールに基づいて異なる場合があります。新しいスケジュールを追加するには、 管理 | AMS リソーススケジューラ | スケジュール | (ct-2bxelbn765ive) 変更タイプを使用して RFC を作成できます。スケジュール | 追加を参照してください。

Q: AMS Resource Scheduler でサポートされているさまざまな変更タイプはどこにありますか？

A: AMS には、AMS リソーススケジューラをアカウントにデプロイするための Resource Scheduler の変更タイプがあり、それを有効または無効にし、使用するスケジュールと期間を定義、追加、更新、削除し、スケジュールと期間を記述 (詳細な説明を取得) します。

クロスアカウントバックアップの設定 (リージョン内)

AWS Backup は、2 つのアカウントが同じ AWS Organization 内にある限り、1 つのアカウントから同じ AWS リージョン内の別のアカウントにスナップショットをコピーする機能をサポートします。例えば、AMS Advanced マルチアカウントランディングゾーン (MALZ) では、このクイックスタートを使用して、同じ AWS リージョン内にクロスアカウントスナップショットコピーを設定できます。

詳細については、AWS Backup」とAWS Organizations Bring cross-account backup feature」を参照してください。

ディザスタリカバリ (DR) 用にスナップショットのクロスアカウントをコピーします。データ保護のために、スナップショットを同じ AWS リージョン内に保持し、アカウントの境界を越えて保持する必要がある場合があります。

概要：

大まかに言うと、AMS 内のクロスアカウントバックアップの手順は次のとおりです。

- AMS ランディングゾーンがホストされている AWS リージョンでバックアップをホストする送信先アカウントを作成する (ステップ 1)
- 送信先アカウントでバックアップを暗号化するための KMS キーを作成する (ステップ 3)
- AMS Advanced ランディングゾーンと同じリージョンの送信先アカウントにバックアップボールトを作成する (ステップ 4)
- 管理アカウントでクロスアカウント設定を有効にする (ステップ 5)
- ソースアカウントのバックアッププランとルール (複数可) を作成または変更する (ステップ 6)

> **Note**
>
> 送信元アカウントと送信先アカウントの両方が同じリージョンにあることを確認します。リージョン間でバックアップをコピーする場合は、CA または CSDM にお問い合わせください。

クロスアカウントバックアップを有効にして設定するには：

1. バックアップをホストする送信先アカウントを作成します。そのようなアカウントがすでにある場合は、このステップをスキップできます。アカウントを作成するには、デプロイ | マネージドランディングゾーン | 管理アカウント | アプリケーションアカウントの作成 (VPC を使用） 変更タイプ (ct-1zdasmc2ewzrs) を使用して、Management Payer アカウントから RFC を送信します。
2. 〔オプション] リソースまたはスナップショットがソースアカウント (Prod など) で暗号化されている場合は、暗号化に使用される KMS キーを送信先アカウントと共有します。これを行うには、 管理 | 高度なスタックコンポーネント | KMS キー | 更新変更タイプ (ct-3ovo7px2vsa6n) を使用して RFC を送信します。
3. 送信先アカウントで、バックアップボールトの暗号化に使用する KMS キーを作成します。これを行うには、Deployment | Advanced スタックコンポーネント | KMS キー | Create (auto) change type (ct-1d84keiri1jhg) を使用して RFC を送信します。
4. 送信先アカウントで、前に作成したキーを使用して Backup Vault を作成します。AWS Backup Vault は、CFN 取り込み自動変更タイプ、デプロイ | 取り込み | CloudFormation テンプレートからのスタック | 作成 (ct-36cn2avfrrj9v) を使用して作成できます。同じリクエストで、ソースアカウント (複数可) にボールトへのアクセスを許可するには、ボールトアクセスポリシーを変更する必要があります。ポリシーの例を次に示します。

   Backup Vault の CloudFormation テンプレートの例：

```
{
  "Description": "Test infrastructure",
  "Resources": {
  "BackupVaultForTesting": {
    "Type": "AWS::Backup::BackupVault",
    "Properties": {
      "BackupVaultName": "backup-vault-for-test",
      "EncryptionKeyArn" : "arn:aws:kms:us-east-2:123456789012:key/227d8xxx-
aefx-44ex-a09x-b90c487b4xxx",
        "AccessPolicy" : {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Sid": "AllowSrcAccountPermissionsToCopy",
              "Effect": "Allow",
              "Action": "backup:CopyIntoBackupVault",
              "Resource": "*",
              "Principal": {
```

```
                "AWS": ["arn:aws:iam::987654321098:root"]
              }
            }
          ]
        }
      }
    }
  }
}
```

5. Management Payer アカウントから、クロスアカウントバックアップを有効にします。これを行うには、 Management | AWS Backup | Backup plan | Enable cross account copy (Management account) change type (ct-2yja7ihh30ply) を使用して RFC を送信します。
6. 最後に、バックアップのソースとなるソースアカウントから、バックアップを管理するバックアッププランのルールを作成して、スナップショットをクロスアカウントにコピーします。これを行うには、デプロイ | AWS Backup | Backup plan | Create change type(ct-2hyozbpa0sx0m) を使用して RFC を送信します。既存のバックアッププランを更新する必要がある場合は、 管理 | その他 | その他 | 更新変更タイプ (ct-0xdawir96cy7k) を使用して、この情報を使用して RFC を送信します。

   1. 更新するバックアッププラン名とルール名。
   2. 送信先/ICE アカウントのバックアップボールト ARN。
   3. スナップショットをターゲット ICE ボールトに保持する保持日/月。

チュートリアル

トピック

以下のチュートリアルでは、 CLI を使用してコンソールを使用し、Linux または RHEL Amazon EC2 Auto Scaling グループ (ASG) をデプロイして、高可用性 (ct-06mjngx5flwto) で 2 層スタックを作成する手順について詳しく説明します。 Auto Scaling 同様のtier-and-tieチュートリアルでは、それぞれ (コンソールの場合は 1 つ、CLI の場合は 1 つ) が続きます。このチュートリアルでは、個別の CTs を使用して、作成されたリソースを結合できる順序で作成されます。

ChangeTypeId を含むすべての CT オプションの説明は、managedservices/latest/ctref/ Change Type Reference にあります。

コンソールチュートリアル: 高可用性 2 層スタック (Linux/RHEL)

このセクションでは、AMS コンソールを使用して高可用性 (HA) WordPress サイトを AMS 環境にデプロイする方法について説明します。

> **Note**
>
> このデプロイのチュートリアルは、AMZN Linux および RHEL 環境でテストされています。

タスクと必要な RFCs の概要：

1. インフラストラクチャの作成 (HA 2 層スタック)
2. CodeDeploy アプリケーション用の S3 バケットを作成する
3. WordPress アプリケーションバンドルを作成し、S3 バケットにアップロードする
4. CodeDeploy を使用してアプリケーションをデプロイする
5. WordPress サイトにアクセスしてログインし、デプロイを検証します。
6. デプロイをティアダウンする

ChangeTypeId を含むすべての CT オプションの説明は、AMS 変更タイプリファレンスで確認できます。

開始する前に

Deployment | Advanced Stack Components | High Availability Two Tier Stack | Create CT は、Auto Scaling グループ、ロードバランサー、データベース、CodeDeploy アプリケーション名とデプロイグループ (アプリケーションと同じ名前) を作成します。CodeDeploy の詳細については、CodeDeploy とは」を参照してください。

このチュートリアルでは、CodeDeploy がデプロイできる WordPress バンドルの作成方法UserDataを含む高可用性 2 層スタック RFC を使用します。

この例UserDataに示す は、http://169.254.169.254/latest/meta-data/ で利用可能な EC2 インスタンスメタデータサービスをクエリすることで、実行中のインスタンス内からインスタンス ID、リージョンなどのインスタンスメタデータを取得します。ユーザーデータスクリプトのこの行: はREGION=$(curl 169.254.169.254/latest/meta-data/placement/availability-zone/ | sed 's/[a-z]$//')、メタデータサービスからサポートされているリージョンの $REGION 変数にアベイラビリティーゾーン名を取得し、それを使用して CodeDeploy エージェントをダウンロードする S3 バケットの URL を完了します。169.254.169.254 IP は VPC 内でのみルーティング可能です (すべての VPCsはサービスをクエリできます) 。サービスの詳細については、「インスタンスメタデータとユーザーデータ」を参照してください。また、UserData として入力されたスクリプトは「ルート」ユーザーとして実行され、「sudo」コマンドを使用する必要はありません。

このチュートリアルでは、以下のパラメータをデフォルト値 (表示) のままにします。

- Auto Scaling グループ: Cooldown=300, DesiredCapacity=2, EBSOptimized=false, HealthCheckGracePeriod=600, IAMInstanceProfile=customer-mc-ec2-instance-profile, InstanceDetailedMonitoring=true, InstanceRootVolumeIops=0, InstanceRootVolumeType=standard, InstanceType=m3.medium, MaxInstances=2, MinInstances=2, ScaleDownPolicyCooldown=300, ScaleDownPolicyEvaluationPeriods=4, ScaleDownPolicyPeriod=60, ScaleDownPolicyScalingAdjustment=-1, ScaleDownPolicyStatistic=Average, ScaleDownPolicyThreshold=35, ScaleMetricName=CPUUtilization, ScaleUpPolicyCooldown=60, ScaleUpPolicyEvaluationPeriods=2, ScaleUpPolicyPeriod=60, ScaleUpPolicyScalingAdjustment=2, ScaleUpPolicyStatistic=Average, ScaleUpPolicyThreshold=75。

- Load Balancer: `HealthCheckInterval=30, HealthCheckTimeout=5`。
- データベース: `BackupRetentionPeriod=7, Backups=true, InstanceType=db.m3.medium, IOPS=0, MultiAZ=true, PreferredBackupWindow=22:00-23:00, PreferredMaintenanceWindow=wed:03:32-wed:04:02, StorageEncrypted=false, StorageEncryptionKey="", StorageType=gp2`。
- アプリケーション: `DeploymentConfigName=CodeDeployDefault.OneAtATime`。

変数パラメータ：

コンソールには開始時刻の ASAP オプションが用意されており、このチュートリアルではそれを使用することをお勧めします。ASAP では、承認に合格するとすぐに RFC が実行されます。

> **Note**
>
> 以下に示すものとは異なる設定を選択できるパラメータが多数あります。この例に示すパラメータの値はテスト済みですが、適切ではない可能性があります。必須値のみが例に表示されます。`#######`フォントの値は、アカウント固有の値として変更する必要があります。

インフラストラクチャを作成する

この手順では、高可用性 2 層スタック CT の後に S3 ストレージ CT を作成します。

開始する前に次のデータを収集すると、デプロイがより速くなります。

必須データのスタック：

- AutoScalingGroup：
  - UserData: この値は、このチュートリアルで提供されています。これには、CodeDeploy のリソースをセットアップし、CodeDeploy エージェントを起動するためのコマンドが含まれています。
  - AMI-ID: この値は、Auto Scaling グループ (ASG) がスピンアップする EC2 インスタンスのオペレーティングシステムを決定します。アカウントで「customer-」で始まり、目的のオペレーティングシステムの AMI を選択します。AMS コンソール VPCs -> VPCsの詳細ページで AMI IDs を検索します。このチュートリアルは、Amazon Linux または RHEL AMI を使用するように設定された ASGs を対象としています。
- データベース：

- これらのパラメータ DBEngine、EngineVersion、LicenseModel は、例に示す値がテスト済みですが、状況に応じて設定する必要があります。このチュートリアルでは、それぞれ *MySQL*、*8.0.16*、 *general-public-license* の値を使用します。
- これらのパラメータ、DBName、MasterUserPassword、MasterUsername は、アプリケーションバンドルをデプロイするときに必要です。このチュートリアルでは、*wordpressDB*、*p4ssw0rd*、*admin* の各値を使用します。DBName には英数字のみを含めることができます。
- RDS DB の MasterUsername を入力すると、クリアテキストで表示されるため、できるだけ早くデータベースにログインし、パスワードを変更してセキュリティを確保します。
- RDSSubnetIds には、2 つのプライベートサブネットを使用します。それぞれの後に「Enter」を押すたびに 1 つずつ入力します。AMS SKMS API リファレンスでサブネット IDs を検索するには、AWS Artifact コンソールのレポートタブを参照してください。 オペレーション (CLI: list-subnet-summaries) または AMS コンソール VPCs-> VPC の詳細ページ。

- LoadBalancer：
  - チュートリアルではパブリック ELB サブネットを使用するため、このパラメータ Public を true に設定します。
  - ELBSubnetIds: 2 つのパブリックサブネットを使用します。それぞれの後に「Enter」を押すたびに 1 つずつ入力します。AMS SKMS API リファレンスでサブネット IDs を検索するには、AWS Artifact コンソールのレポートタブを参照してください。 オペレーション (CLI: list-subnet-summaries) または AMS コンソール VPCs-> VPC の詳細ページ。
- アプリケーション: ApplicationName 値は、CodeDeploy アプリケーション名と CodeDeploy デプロイグループ名を設定します。これを使用してアプリケーションをデプロイします。アカウント内で一意である必要があります。アカウントで CodeDeploy 名を確認するには、CodeDeploy コンソールを参照してください。この例では *WordPress* を使用していますが、その値を使用する場合は、まだ使用されていないことを確認してください。

1. 高可用性スタックを起動します。

   a. RFC の作成ページで、リストから「デプロイ」、「サブカテゴリ標準スタック」、「高可用性 2 層スタック」、「Create」を選択します。

   b. 重要: Advanced を選択し、図のように値を設定します。

      星付き (*) オプションの値を入力するだけで、テストされた値は例に示されています。不要な空のオプションは空白のままにすることができます。

   c. RFC の説明セクションの場合：

```
Subject: WP-HA-2-Tier-RFC
```

d. リソース情報セクションで、AutoScalingGroup、 Database、LoadBalancer、Application、および Tags のパラメータを設定します。

また、AppName」タグキーの目的は、EC2 コンソールで ASG インスタンスを簡単に検索できるようにすることです。このタグキー「Name」または他の任意のキー名を呼び出すことができます。最大 50 個のタグを追加できます。

```
UserData:
    #!/bin/bash
    REGION=$(curl 169.254.169.254/latest/meta-data/placement/availability-zone/
 | sed 's/[a-z]$//')
    yum -y install ruby httpd
    chkconfig httpd on
    service httpd start
    touch /var/www/html/status
    cd /tmp
    curl -O https://aws-codedeploy-$REGION.s3.amazonaws.com/latest/install
    chmod +x ./install
    ./install auto
    chkconfig codedeploy-agent on
    service codedeploy-agent start
AmiId:               AMI-ID
Description:         WP-HA-2-Tier-Stack

Database:
    LicenseModel:    general-public-license (USE RADIO BUTTON)
    EngineVersion:   8.0.16
    DBEngine:        MySQL
    RDSSubnetIds:    PRIVATE_AZ1 PRIVATE_AZ2 (ENTER ONE AT A TIME PRESSING
 "ENTER" AFTER EACH)
    MasterUserPassword:  p4ssw0rd
    MasterUsername:      admin
    DBName:              wordpressDB

LoadBalancer:
    Public:              true (USE RADIO BUTTON)
    ELBSubnetIds:        PUBLIC_AZ1 PUBLIC_AZ2

Application:
    ApplicationName:     WordPress
```

```
Tags:
    Name:                   WP-Rhel-Stack
```

e. 完了したら、送信 をクリックします。

2. 作成したデータベースにログインし、パスワードを変更します。
3. S3 バケットスタックを起動します。

   開始する前に次のデータを収集すると、デプロイがより速くなります。

   必須データ S3 バケット：

   - VPC-ID: この値により、S3 バケットの場所が決まります。AMS SKMS API リファレンスで VPC IDs を検索するには、AWS Artifact コンソールのレポートタブを参照してください。 オペレーション (CLI: list-vpc-summaries) または AMS コンソール VPCsページ。
   - BucketName: この値は S3 バケット名を設定します。これを使用してアプリケーションバンドルをアップロードします。アカウントのリージョン全体で一意である必要があり、大文字を含めることはできません。BucketName の一部としてアカウント ID を含めることは必須ではありませんが、後でバケットを識別しやすくなります。アカウントに存在する S3 バケット名を確認するには、アカウントの Amazon S3 コンソールに移動します。

   a. RFC の作成ページで、RFC CT ピックリストから、カテゴリデプロイ、サブカテゴリアドバンストスタックコンポーネント、項目 S3 ストレージ、およびオペレーションの作成を選択します。

   b. デフォルトの基本オプションを保持し、図のように値を設定します。

```
Subject:                S3-Bucket-WP-HA-RFC
Description:            S3BucketForWordPressBundles
BucketName:             ACCOUNT_ID-BUCKET_NAME
AccessControl:          Private
VpcId:                  VPC_ID
Name:                   S3-Bucket-WP-HA-Stack
TimeoutInMinutes:       60
```

   c. 完了したら送信をクリックします。この変更タイプでデプロイされたバケットは、アカウント全体への完全な読み取り/書き込みアクセスを許可します。

アプリケーションの作成、アップロード、デプロイ

まず、WordPress アプリケーションバンドルを作成し、CodeDeploy CT を使用してアプリケーションを作成してデプロイします。 CTs

1. WordPress をダウンロードし、ファイルを抽出して ./scripts ディレクトリを作成します。

   Linux コマンド：

   ```
   wget https://github.com/WordPress/WordPress/archive/master.zip
   ```

   Windows: ブラウザウィンドウ`https://github.com/WordPress/WordPress/archive/master.zip`に貼り付け、zip ファイルをダウンロードします。

   パッケージをアセンブルする一時ディレクトリを作成します。

   Linux:

   ```
   mkdir /tmp/WordPress
   ```

   Windows: WordPress」ディレクトリを作成します。後でディレクトリパスを使用します。
2. WordPress ソースをWordPress」ディレクトリに抽出し、./scripts ディレクトリを作成します。

   Linux:

   ```
   unzip master.zip -d /tmp/WordPress_Temp
   cp -paf /tmp/WordPress_Temp/WordPress-master/* /tmp/WordPress
   rm -rf /tmp/WordPress_Temp
   rm -f master
   cd /tmp/WordPress
   mkdir scripts
   ```

   Windows: 作成したWordPress」ディレクトリに移動し、そこに「scripts」ディレクトリを作成します。

   Windows 環境の場合は、スクリプトファイルのブレークタイプを Unix (LF) に設定してください。Notepad ++ では、これはウィンドウの右下にあるオプションです。
3. WordPress ディレクトリに CodeDeploy appspec.yml ファイルを作成します (例をコピーする場合は、インデントと各スペース数を確認します) 。重要: WordPress ファイル (この場合

は WordPress ディレクトリ) を目的の宛先 (/var/www/html/WordPress) にコピーするための WordPress 「ソース」パスが正しいことを確認します。この例では、appspec.yml ファイルは WordPress ファイルを含む ディレクトリにあるため、「/」のみが必要です。また、Auto Scaling グループに RHEL AMI を使用した場合でも、「os: linux」行はそのままにしておきます。appspec.yml ファイルの例：

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/config_wordpress.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

4. WordPress ./scripts ディレクトリに bash ファイルスクリプトを作成します。

   まず、次の内容`config_wordpress.sh`で を作成します (必要に応じて、wp-config.php ファイルを直接編集できます) 。

   > **Note**
   >
   > *`DBName`* を HA スタック RFC で指定された値 (例:) に置き換えます`wordpress`。
   > *`DB_MasterUsername`* を HA スタック RFC で指定された`MasterUsername`値 (など`admin`) に置き換えます。
   > *`DB_MasterUserPassword`* を HA スタック RFC で指定された`MasterUserPassword`値 (例:) に置き換えます`p4ssw0rd`。

DB_ENDPOINT を HA スタック RFC の実行出力のエンドポイント DNS 名 (例:) に置き換えますsrt1cz23n45sfg.clgvd67uvydk.us-east-1.rds.amazonaws.com。これは、GetRfc オペレーション (CLI: get-rfc --rfc-id RFC_ID) または以前に送信した HA スタック RFC の AMS コンソール RFC の詳細ページで確認できます。

```
#!/bin/bash
chmod -R 755 /var/www/html/WordPress
cp /var/www/html/WordPress/wp-config-sample.php /var/www/html/WordPress/wp-
config.php
cd /var/www/html/WordPress
sed -i "s/database_name_here/DBName/g" wp-config.php
sed -i "s/username_here/DB_MasterUsername/g" wp-config.php
sed -i "s/password_here/DB_MasterUserPassword/g" wp-config.php
sed -i "s/localhost/DB_ENDPOINT/g" wp-config.php
```

5. 同じディレクトリに、次の内容install_dependencies.shで を作成します。

```
#!/bin/bash
yum install -y php
yum install -y php-mysql
yum install -y mysql
service httpd restart
```

> **Note**
>
> HTTPS は、ヘルスチェックを最初から機能させるために、起動時にユーザーデータの一部としてインストールされます。

6. 同じディレクトリに、次の内容start_server.shで を作成します。

   - Amazon Linux インスタンスの場合は、以下を使用します。

     ```
     #!/bin/bash
     service httpd start
     ```

   - RHEL インスタンスの場合は、以下を使用します (追加のコマンドは、SELINUX が WordPress を受け入れることを許可するポリシーです)。

     ```
     #!/bin/bash
     ```

```
setsebool -P  httpd_can_network_connect_db 1
setsebool -P  httpd_can_network_connect 1
chcon -t httpd_sys_rw_content_t /var/www/html/WordPress/wp-content -R
restorecon -Rv /var/www/html
service httpd start
```

7. 同じディレクトリに、次の内容stop_server.shで を作成します。

```
#!/bin/bash
service httpd stop
```

8. zip バンドルを作成します。

   Linux:

```
$ cd /tmp/WordPress
$ zip -r wordpress.zip .
```

   Windows: WordPress」ディレクトリに移動し、すべてのファイルを選択して zip ファイルを作成します。必ず wordpress.zip という名前を付けます。

1. アプリケーションバンドルを S3 バケットにアップロードする

   スタックのデプロイを続行するには、パッケージを設定する必要があります。

   作成した S3 バケットインスタンスに自動的にアクセスできます。踏み台 (インスタンスへのアクセスを参照) または S3 コンソールからアクセスし、drag-and-dropで CodeDeploy パッケージをアップロードするか、 ファイルを参照して選択できます。

   シェルウィンドウで次のコマンドを使用することもできます。zip ファイルへの正しいパスがあることを確認してください。

```
aws s3 cp wordpress/wordpress.zip s3://BUCKET_NAME/
```

2. WordPress CodeDeploy アプリケーションバンドルをデプロイする

   必須データコードアプリケーションのデプロイ：

   - CodeDeployApplicationName: CodeDeploy アプリケーションに付けた名前。

- CodeDeployGroupName: CodeDeploy アプリケーションとグループはどちらも HA スタック RFC で CodeDeploy アプリケーションに付けた名前から作成されたため、これは CodeDeployApplicationName と同じ名前です。
- S3Bucket: S3 バケットに付けた名前。
- S3BundleType と S3Key: これらはデプロイした WordPress アプリケーションバンドルの一部です。
- VpcId: 関連する VPC。

a. RFC の作成ページで、RFC CT ピックリストからカテゴリデプロイ、サブカテゴリアプリケーション、項目 CodeDeploy アプリケーション、およびオペレーションデプロイを選択します。

b. デフォルトの基本オプションのままにし、図のように値を設定します。

> **Note**
>
> 以前に作成した CodeDeploy アプリケーション、CodeDeploy デプロイグループ、S3 バケット、バンドルを参照してください。

```
Subject:                                    WP-CD-Deploy-RFC
Description:                                DeployWordPress
S3Bucket:                                   BUCKET_NAME
S3Key:                                      wordpress.zip
S3BundleType:                               zip
CodeDeployApplicationName:                  WordPress
CodeDeployDeploymentGroupName:              WordPress
CodeDeployIgnoreApplicationStopFailures:    false
RevisionType:                               S3



VpcId:                                      VPC_ID
Name:                                       WP-CD-Deploy-Op
TimeoutInMinutes:                           60
```

c. 完了したら、送信 をクリックします。

アプリケーションのデプロイを検証する

WordPress デプロイパス: /WordPress を使用して、以前に作成したロードバランサーのエンドポイント (LoadBalancerCName) WordPress に移動します。例:

```
http://stack-ID-FOR-ELB.us-east-1.elb.amazonaws.com/WordPress
```

次のようなページが表示されます。

高可用性デプロイのティアダウン

デプロイを破棄するには、HA 2 階層スタックと S3 バケットに対して Delete Stack CT を送信し、RDS スナップショットの削除をリクエストできます (10 日後に自動的に削除されますが、その間は少額のコストがかかります)。HA スタックと S3 バケットのスタック IDs を収集し、以下の手順に従います。「スタック | 削除」を参照してください。

コンソールチュートリアル: 階層のデプロイと WordPress ウェブサイトの絞り込み

このセクションでは、AMS コンソールを使用して高可用性 (HA) WordPress サイトを AMS 環境にデプロイする方法について説明します。この一連の手順には、必要な WordPress CodeDeploy 互換パッケージ (zip など) ファイルを作成する例が含まれています。リソースのプロビジョニングは、それらを結合して「階層」を形成することができる順序に従います。

> **Note**
>
> このデプロイチュートリアルは、AMZN Linux OS で使用するように設計されています。
> 必須変数パラメータは######と表記されますが、状況に合わせて他のパラメータを変更することもできます。

タスクと必要な RFCs の概要：

1. インフラストラクチャを作成します。
   a. MySQL RDS データベースクラスターを作成する
   b. ロードバランサーの作成

   c. Auto Scaling グループを作成し、ロードバランサーに関連付けます。
   d. CodeDeploy アプリケーション用の S3 バケットを作成する
2. WordPress アプリケーションバンドルを作成する (RFC は不要)
3. CodeDeploy を使用して WordPress アプリケーションバンドルをデプロイします。
   a. CodeDeploy アプリケーションを作成する
   b. CodeDeploy デプロイグループを作成する
   c. WordPress アプリケーションバンドルを S3 バケットにアップロードする (RFC は必要ありません)
   d. CodeDeploy アプリケーションをデプロイする
4. デプロイを検証する
5. デプロイをティアダウンする

ChangeTypeId を含むすべての CT オプションの説明は、AMS 変更タイプリファレンスで確認できます。

コンソールを使用した RFC の作成 (基本)

以下は、 コンソールを使用して RFC を作成するたびに実行する必要があるステップです。

1. 左側のナビゲーションペインの RFCs をクリックして RFCsリストページを開き、RFC の作成をクリックします。

   RFC の作成ページが開きます。
2. 変更タイプを参照 (デフォルト) またはカテゴリ別に選択します。
3. 変更タイプを参照します。

   a. クイック作成オプションをクリックして、最も使用されている変更タイプの 1 つで RFC を開始します。

      その変更タイプの一般的な設定領域が開き、件名が入力されます。変更タイプの詳細を表示するには、ページの上部にある領域を開きます。
   b. すべての変更タイプエリアを使用します。

      カードまたはテーブルビューをフィルタリング、切り替え、または変更タイプをソートします。目的のものを見つけたら、それを選択し、ページ上部の「RFC の作成」をクリックします。

その変更タイプの一般的な設定領域が開き、件名が入力されます。変更タイプの詳細を表示するには、ページの上部にある領域を開きます。

4. カテゴリで選択します。

   a. 適切なカテゴリ、サブカテゴリ、項目、オペレーションを選択します。

      変更タイプの詳細ボックスがページの下部に表示されます。

   b. ページの下部にある「RFC の作成」をクリックします。

   c. その変更タイプの一般的な設定領域が開き、件名が入力されます。変更タイプの詳細を表示するには、ページの上部にある領域を開きます。

5. 特定のユーザーに RFC の進行状況の通知を確実に受け取るには、E メールアドレスを入力します。変更タイプの詳細を追加するには、説明を入力します。追加設定エリアを開き、RFC の詳細を追加します。

6. スケジューリングでは、この変更をできるだけ早く実行するか、この変更をスケジュールするかを選択します。この変更をできるだけ早く実行するを選択すると、RFC は承認に合格するとすぐに実行されます。この変更タイプのスケジュールを選択すると、ピックカレンダー、時刻、タイムゾーンが表示され、送信後に RFC がスケジュールどおりに開始されます。

7. 実行設定エリアで、変更タイプのパラメータを設定します。オプションのパラメータを表示するには、追加設定エリアを開きます。

8. 準備ができたら、実行 をクリックします。

インフラストラクチャの作成

ターゲット AMS アカウントの AWS コンソールにログインし、次にアカウントの AMS コンソールにログインします。

次の手順では、リソース IDs を使用してインフラストラクチャを構築する方法で RDS データベース、ロードバランサー、Auto Scaling グループを作成する方法について説明します。

RDS スタックを作成する

「RDS スタック | 作成」を参照してください。

ELB スタックを作成する

パブリック ELB を起動します。

必須データ：

- VpcId: 使用している VPC。これは以前に使用した VPC と同じである必要があります。
- ELBSubnetIds: ロードバランサーがトラフィックを分散するサブネットの配列。パブリックサブネットまたはプライベートサブネットを選択します。AMS SKMS API リファレンスでサブネット IDs を検索するには、AWS Artifact コンソールのレポートタブを参照してください。 オペレーション (CLI: list-subnet-summaries) または AMS コンソール VPCs-> VPC の詳細ページ。
- VpcId: 使用している VPC。これは以前に使用した VPC と同じである必要があります。

1. RFC の作成ページで、カテゴリデプロイ、サブカテゴリアドバンストスタックコンポーネント、項目ロードバランサー (ELB) スタックを選択し、作成をクリックします。Advanced を選択し、次に示すものを除くすべてのデフォルト (値のないものを含む) を受け入れます。

```
Subject:                        WP-ELB-RFC
ELBSubnetIds:                   PUBLIC_AZ1
                              PUBLIC_AZ2
ELBScheme                       true
ELBCookieExpirationPeriod       600
VpcId:                          VPC_ID
Name:                           WP-Public-ELB
```

2. 完了したら送信をクリックします。

Auto Scaling グループスタックを作成する

Auto Scaling グループを起動します。

必須データ：

- VpcId: 使用している VPC。これは以前に使用した VPC と同じである必要があります。
- AMI-ID: この値は、Auto Scaling グループ (ASG) がスピンアップする EC2 インスタンスの種類を決定します。アカウントで「customer-」で始まり、目的のオペレーティングシステムの AMI を選択してください。AMS SKMS API リファレンスで AMI IDs を検索するには、AWS Artifact コンソールのレポートタブを参照してください。 オペレーション (CLI: list-amis) または AMS コンソール VPCs-> VPCsの詳細ページ。このチュートリアルは、Linux AMI を使用するように設定された ASGs を対象としています。

- ASGLoadBalancerNames: 以前に作成したロードバランサー。EC2 コンソール -> Load Balancer (左側のナビゲーション) を見て名前を見つけます。これは、以前に ELB を作成したときに指定した「名前」ではないことに注意してください。

1. RFC の作成ページで、カテゴリデプロイ、サブカテゴリアドバンストスタックコンポーネント、項目自動スケーリンググループを選択し、作成をクリックします。Advanced を選択し、次に示すものを除くすべてのデフォルト (値のないものを含む) を受け入れます。

> **Note**
>
> 最新の AMS AMI を指定します。以前に作成した ELB を指定します。

```
Subject:                        WP-ASG-RFC
ASGSubnetIds:                   PRIVATE_AZ1
                                                    PRIVATE_AZ2
ASGAmiId:                       AMI_ID
VpcId:                          VPC_ID
Name:                           WP_ASG
ASGLoadBalancerNames:           ELB_NAME
ASGUserData:
#!/bin/bash
REGION=$(curl 169.254.169.254/latest/meta-data/placement/availability-zone/ | sed
 's/[a-z]$//')
yum -y install ruby httpd
chkconfig httpd on
service httpd start
touch /var/www/html/status
cd /tmp
curl -O https://aws-codedeploy-$REGION.s3.amazonaws.com/latest/install
chmod +x ./install
./install auto
chkconfig codedeploy-agent on
service codedeploy-agent start
```

2. 完了したら、送信 をクリックします。

S3 スタックを作成する

S3 バケットを起動します。S3 バケットは、作成したアプリケーションバンドルをアップロードする場所です。

必須データ：

- VPC-ID: この値は S3 バケットの場所を決定します。これは以前に使用した VPC と同じである必要があります。
- AccessControl: プリセット AccessControl リスト (ACL) オプションは、Private、、および ですPublicRead。詳細については、「Amazon Simple Storage Service の既定 ACL」を参照してください。
- BucketName: この値は S3 バケット名を設定します。これを使用してアプリケーションバンドルをアップロードします。アカウントのリージョン全体で一意である必要があり、大文字を含めることはできません。BucketName の一部としてアカウント ID を含めることは必須ではありませんが、後でバケットを識別しやすくなります。アカウントに存在する S3 バケット名を確認するには、アカウントの Amazon S3 コンソールに移動します。

1. RFC の作成ページで、カテゴリデプロイ、サブカテゴリアドバンストスタックコンポーネント、項目 S3 ストレージを選択し、作成をクリックします。

   デフォルトのパラメータオプションは Basic のままにして、説明に従ってデフォルトを受け入れることができます。異なる値を設定するには、アドバンスト を選択します。

   > **Note**
   >
   > この変更タイプでデプロイされたバケットは、アカウント全体への完全な読み取り/書き込みアクセスを許可します。アクセス許可をより制限するには、新しい変更タイプが必要になる場合があります。

   ```
   Subject:            S3-Bucket-RFC
   BucketName:         ACCOUNT_ID-codedeploy-bundles
   AccessControl:      Private

   VpcId:              VPC_ID
   Name:               S3BucketForWP
   ```

2. 完了したら、送信 をクリックします。

WordPress CodeDeploy バンドルを作成する

このセクションでは、アプリケーションデプロイバンドルを作成する例を示します。

1. WordPress をダウンロードし、ファイルを抽出して ./scripts ディレクトリを作成します。

   Linux コマンド：

   ```
   wget https://github.com/WordPress/WordPress/archive/master.zip
   ```

   Windows: ブラウザウィンドウhttps://github.com/WordPress/WordPress/archive/master.zipに貼り付け、zip ファイルをダウンロードします。

   パッケージをアセンブルする一時ディレクトリを作成します。

   Linux:

   ```
   mkdir /tmp/WordPress
   ```

   Windows: WordPress」ディレクトリを作成します。後でディレクトリパスを使用します。
2. WordPress ソースをWordPress」ディレクトリに抽出し、./scripts ディレクトリを作成します。

   Linux:

   ```
   unzip master.zip -d /tmp/WordPress_Temp
   cp -paf /tmp/WordPress_Temp/WordPress-master/* /tmp/WordPress
   rm -rf /tmp/WordPress_Temp
   rm -f master
   cd /tmp/WordPress
   mkdir scripts
   ```

   Windows: 作成したWordPress」ディレクトリに移動し、そこに「scripts」ディレクトリを作成します。

   Windows 環境の場合は、スクリプトファイルのブレークタイプを Unix (LF) に設定してください。Notepad ++ では、これはウィンドウの右下にあるオプションです。

3. WordPress ディレクトリに CodeDeploy appspec.yml ファイルを作成します (例をコピーする場合は、インデントを確認し、各スペース数を確認します)。重要: WordPress ファイル (この場合は WordPress ディレクトリ) を目的の宛先 (/var/www/html/WordPress) にコピーするには、 WordPress 「ソース」パスが正しいことを確認してください。この例では、appspec.yml ファイルは WordPress ファイルを含む ディレクトリにあるため、「/」のみが必要です。また、Auto Scaling グループに RHEL AMI を使用した場合でも、「os: linux」行はそのままにしておきます。appspec.yml ファイルの例 :

```
version: 0.0
os: linux
files:
  - source: /
    destination: /var/www/html/WordPress
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  AfterInstall:
    - location: scripts/config_wordpress.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

4. WordPress ./scripts ディレクトリに bash ファイルスクリプトを作成します。

   まず、次の内容`config_wordpress.sh`で を作成します (必要に応じて、wp-config.php ファイルを直接編集できます)。

> **Note**
>
> *`DBName`* を HA スタック RFC で指定された値 (例:) に置き換えます`wordpress`。
> *`DB_MasterUsername`* を HA スタック RFC で指定された`MasterUsername`値 (など`admin`) に置き換えます。

DB_MasterUserPassword を HA スタック RFC で指定されたMasterUserPassword値 (例:) に置き換えますp4ssw0rd。*DB_ENDPOINT* を HA スタック RFC の実行出力のエンドポイント DNS 名 (例:) に置き換えますsrt1cz23n45sfg.clgvd67uvydk.us-east-1.rds.amazonaws.com。これは、GetRfc オペレーション (CLI: get-rfc --rfc-id RFC_ID) または以前に送信した HA スタック RFC の AMS コンソール RFC の詳細ページで確認できます。

```
#!/bin/bash
chmod -R 755 /var/www/html/WordPress
cp /var/www/html/WordPress/wp-config-sample.php /var/www/html/WordPress/wp-
config.php
cd /var/www/html/WordPress
sed -i "s/database_name_here/DBName/g" wp-config.php
sed -i "s/username_here/DB_MasterUsername/g" wp-config.php
sed -i "s/password_here/DB_MasterUserPassword/g" wp-config.php
sed -i "s/localhost/DB_ENDPOINT/g" wp-config.php
```

5. 同じディレクトリに、次の内容install_dependencies.shで を作成します。

```
#!/bin/bash
yum install -y php
yum install -y php-mysql
yum install -y mysql
service httpd restart
```

> **Note**
>
> HTTPS は、ヘルスチェックを最初から機能させるために、起動時にユーザーデータの一部としてインストールされます。

6. 同じディレクトリに、次の内容start_server.shで を作成します。

   - Amazon Linux インスタンスの場合は、以下を使用します。

     ```
     #!/bin/bash
     service httpd start
     ```

   - RHEL インスタンスの場合は、以下を使用します (追加のコマンドは、SELINUX が WordPress を受け入れることを許可するポリシーです) 。

```
#!/bin/bash
setsebool -P  httpd_can_network_connect_db 1
setsebool -P  httpd_can_network_connect 1
chcon -t httpd_sys_rw_content_t /var/www/html/WordPress/wp-content -R
restorecon -Rv /var/www/html
service httpd start
```

7. 同じディレクトリに、次の内容stop_server.shで を作成します。

```
#!/bin/bash
service httpd stop
```

8. zip バンドルを作成します。

   Linux:

```
$ cd /tmp/WordPress
$ zip -r wordpress.zip .
```

   Windows: WordPress」ディレクトリに移動し、すべてのファイルを選択して zip ファイルを作成します。必ず wordpress.zip という名前を付けます。

CodeDeploy を使用して WordPress アプリケーションバンドルをデプロイする

CodeDeploy は、Amazon EC2 インスタンスへのアプリケーションのデプロイを自動化する AWS デプロイサービスです。プロセスのこの部分では、CodeDeploy アプリケーションを作成し、CodeDeploy デプロイグループを作成し、CodeDeploy を使用してアプリケーションをデプロイします。

CodeDeploy アプリケーションを作成する

CodeDeploy アプリケーションは、AWS CodeDeploy が使用する名前またはコンテナであり、デプロイ中に正しいリビジョン、デプロイ設定、デプロイグループが参照されるようにします。この場合、デプロイ設定は、以前に作成した WordPress バンドルです。

必須データ：

- VpcId: 使用している VPC。これは以前に使用した VPC と同じである必要があります。

- CodeDeployApplicationName: アカウント内で一意である必要があります。CodeDeploy コンソールで、既存のアプリケーション名を確認します。

1. WordPress 用の CodeDeploy アプリケーションを作成する

   RFC の作成ページで、RFC CT ピックリストから、カテゴリデプロイ、サブカテゴリアプリケーション、項目 CodeDeploy アプリケーション、およびオペレーションの作成を選択します。Basic を選択し、図のように値を設定します。完了したら送信をクリックします。

   ```
   Subject:                    CD-WP-App-RFC
   CodeDeployApplicationName:  WordPress
   VpcId:                      VPC_ID
   Name:                       WP-CD-App
   ```

2. 完了したら送信をクリックします。

CodeDeploy デプロイグループを作成する

CodeDeploy デプロイグループを作成します。

CodeDeploy デプロイグループは、デプロイの対象となる個々のインスタンスのセットを定義します。

必須データ：

- VpcId: 使用している VPC。これは以前に使用した VPC と同じである必要があります。
- CodeDeployApplicationName: 以前に作成した値を使用します。
- CodeDeployAutoScalingGroups: 前に作成した Auto Scaling グループの名前を使用します。
- CodeDeployDeploymentGroupName: デプロイグループの名前。この名前はデプロイグループに関連付けられる各アプリケーションで一意にする必要があります。
- CodeDeployServiceRoleArn: 例に示す式を使用します。

1. RFC の作成ページで、RFC CT ピックリストからカテゴリデプロイ、サブカテゴリアプリケーション、アイテム CodeDeploy デプロイグループ、およびオペレーションの作成を選択します。Advanced を選択し、図のように値を設定します (RFC に必要なのはサブジェクトのみです) 。完了したら、送信 をクリックします。

> **Note**
>
> この形式で CodeDeploy サービスロール ARN を参照"arn:aws:iam::085398962942:role/aws-codedeploy-role"し、以前に作成した「ASG_NAME」の Auto Scaling グループ名を使用します。

```
Description:                    Create CodeDeploy Deployment Group for WP
CodeDeployApplicationName:      WordPress
CodeDeployAutoScalingGroups:    ASG_NAME
CodeDeployDeploymentConfigName: CodeDeployDefault.HalfAtATime
CodeDeployDeploymentGroupName:  WP CD Group
CodeDeployServiceRoleArn:       arn:aws:iam::ACCOUNT_ID:role/aws-codedeploy-role

VpcId:                          VPC_ID
Name:                           WP Deployment Group
```

2. 完了したら、送信 をクリックします。

WordPress アプリケーションをアップロードする

作成した S3 バケットインスタンスに自動的にアクセスできます。踏み台 (「インスタンスへのアクセス」を参照) または S3 コンソールからアクセスし、CodeDeploy バンドルをアップロードできます。スタックのデプロイを続行するには、バンドルを設定する必要があります。この例では、以前に作成したバケット名を使用します。

この AWS コマンドを使用してバンドルを圧縮できます。

```
aws s3 cp wordpress/wordpress.zip s3://ACCOUNT_ID-codedeploy-bundles/
```

CodeDeploy を使用して WordPress アプリケーションをデプロイする

CodeDeploy アプリケーションをデプロイします。

必須データ：

- VPC-ID: 使用している VPC。これは以前に使用した VPC と同じである必要があります。
- CodeDeployApplicationName: 以前に作成した CodeDeploy アプリケーションの名前を使用します。

- CodeDeployDeploymentGroupName: 前に作成した CodeDeploy デプロイグループの名前を使用します。
- S3Location (アプリケーションバンドルをアップロードした場所): S3Bucket: 以前に作成した BucketName、S3BundleTypeおよび S3Key: S3 ストアに配置したバンドルのタイプと名前。

1. WordPress CodeDeploy アプリケーションバンドルをデプロイする

   RFC の作成ページで、RFC CT ピックリストから、カテゴリデプロイ、サブカテゴリアプリケーション、項目 CodeDeploy アプリケーション、およびオペレーションデプロイを選択します。Basic を選択し、図のように値を設定します。完了したら送信をクリックします。

   > **Note**
   >
   > 以前に作成した CodeDeploy アプリケーション、CodeDeploy デプロイグループ、S3 バケット、バンドルを参照してください。

   Subject: WP-CD-Deploy-RFC
   CodeDeployApplicationName: *WordPress*
   CodeDeployDeploymentGroupName: *WPCDGroup*
   RevisionType: S3
   S3Bucket: *ACCOUNT_ID-codedeploy-bundles*
   S3BundleType: zip
   S3Key: wordpress.zip
   VpcId: *VPC_ID*
   Name: WordPress

2. 完了したら送信をクリックします。

アプリケーションのデプロイを検証する

WordPress デプロイパス: /WordPress を使用して、以前に作成したロードバランサーのエンドポイント (ELB CName) WordPress に移動します。例：

```
http://stack-ID-FOR-ELB.us-east-1.elb.amazonaws.com/WordPress
```

アプリケーションのデプロイをティアダウンする

デプロイを破棄するには、RDS データベーススタック、アプリケーションロードバランサー、Auto Scaling グループ、S3 バケット、および Code Deploy アプリケーションと group-six RFCs に対して Delete Stack CT を送信します。さらに、削除する RDS スナップショットのサービスリクエストを送信できます (10 日後に自動的に削除されますが、その間は少額のコストがかかります)。すべてのスタック IDsを収集し、以下の手順に従います。「スタック | 削除」を参照してください。

CLI チュートリアル: 高可用性 2 層スタック (Linux/RHEL)

このセクションでは、AMS CLI を使用して高可用性 (HA) 2 層スタックを AMS 環境にデプロイする方法について説明します。

> **Note**
>
> このデプロイのチュートリアルは、AMZN Linux および RHEL 環境でテストされています。

タスクと必要な RFCs の概要：

1. インフラストラクチャの作成 (HA 2 層スタック)
2. CodeDeploy アプリケーション用の S3 バケットを作成する
3. WordPress アプリケーションバンドルを作成し、S3 バケットにアップロードする
4. CodeDeploy を使用してアプリケーションをデプロイする
5. WordPress サイトにアクセスしてログインし、デプロイを検証します。

開始する前に

Deployment | Advanced Stack Components | High Availability Two Tier Stack Advanced | Create CT は、Auto Scaling グループ、ロードバランサー、データベース、CodeDeploy アプリケーション名とデプロイグループ (アプリケーションと同じ名前) を作成します。CodeDeploy の詳細については、CodeDeploy とは」を参照してください。

このチュートリアルでは、UserData を含む高可用性 2 層スタック (アドバンスト) RFC を使用し、CodeDeploy がデプロイできる WordPress バンドルを作成する方法について説明します。

この例`UserData`に示す は、http://169.254.169.254/latest/meta-data/ で利用可能な EC2 インスタンスメタデータサービスをクエリすることで、実行中のインスタンス内からインスタンス ID、

リージョンなどのインスタンスメタデータを取得します。ユーザーデータスクリプトのこの行: はREGION=$(curl 169.254.169.254/latest/meta-data/placement/availability-zone/ | sed 's/[a-z]$//')、メタデータサービスからサポートされているリージョンの $REGION 変数にアベイラビリティーゾーン名を取得し、それを使用して CodeDeploy エージェントをダウンロードする S3 バケットの URL を完了します。169.254.169.254 IP は VPC 内でのみルーティング可能です (すべての VPCsはサービスをクエリできます) 。サービスの詳細については、「インスタンスメタデータとユーザーデータ」を参照してください。また、UserData として入力されたスクリプトは「ルート」ユーザーとして実行され、「sudo」コマンドを使用する必要はありません。

このチュートリアルでは、以下のパラメータをデフォルト値 (表示) のままにします。

- Auto Scaling グループ: Cooldown=300, DesiredCapacity=2, EBSOptimized=false, HealthCheckGracePeriod=600, IAMInstanceProfile=customer-mc-ec2-instance-profile, InstanceDetailedMonitoring=true, InstanceRootVolumeIops=0, InstanceRootVolumeType=standard, InstanceType=m3.medium, MaxInstances=2, MinInstances=2, ScaleDownPolicyCooldown=300, ScaleDownPolicyEvaluationPeriods=4, ScaleDownPolicyPeriod=60, ScaleDownPolicyScalingAdjustment=-1, ScaleDownPolicyStatistic=Average, ScaleDownPolicyThreshold=35, ScaleMetricName=CPUUtilization, ScaleUpPolicyCooldown=60, ScaleUpPolicyEvaluationPeriods=2, ScaleUpPolicyPeriod=60, ScaleUpPolicyScalingAdjustment=2, ScaleUpPolicyStatistic=Average, ScaleUpPolicyThreshold=75。
- Load Balancer: HealthCheckInterval=30, HealthCheckTimeout=5。
- データベース: BackupRetentionPeriod=7, Backups=true, InstanceType=db.m3.medium, IOPS=0, MultiAZ=true, PreferredBackupWindow=22:00-23:00, PreferredMaintenanceWindow=wed:03:32-wed:04:02, StorageEncrypted=false, StorageEncryptionKey="", StorageType=gp2。
- アプリケーション: DeploymentConfigName=CodeDeployDefault.OneAtATime。
- S3 バケット: AccessControl=Private。

その他の設定：

RequestedStartTime RFC をスケジュールRequestedEndTimeする場合は、 を使用して正しい UTC 時間Time.isを決定できます。提供された例は適切に調整する必要があります。開始時刻が経過

すると、RFC は続行できません。または、これらの値をオフのままにして、承認が渡されるとすぐに実行される ASAP RFC を作成することもできます。

> **Note**
>
> 以下に示すものとは異なる設定を選択できるパラメータが多数あります。この例に示すパラメータの値はテスト済みですが、適切ではない可能性があります。

インフラストラクチャを作成する

開始する前に次のデータを収集すると、デプロイがより速くなります。

必須データにはスタックがあります。

- AutoScalingGroup:
  - `UserData`: この値は、このチュートリアルで提供されています。これには、CodeDeploy のリソースをセットアップし、CodeDeploy エージェントを起動するためのコマンドが含まれています。
  - `AMI-ID`: この値は、Auto Scaling グループ (ASG) がスピンアップする EC2 インスタンスの種類を決定します。アカウントで「customer-」で始まり、目的のオペレーティングシステムの AMI を選択してください。AMS SKMS API リファレンスで AMI IDs を検索するには、AWS Artifact コンソールのレポートタブを参照してください。 オペレーション (CLI: list-amis) または AMS コンソール VPCs-> VPCsの詳細ページ。このチュートリアルは、Linux AMI を使用するように設定された ASGs を対象としています。
- データベース:
  - これらのパラメータ 、`DBEngine`、 は、例に示す値がテスト済みですが`EngineVersion`、状況に応じて設定`LicenseModel`する必要があります。
  - これらのパラメータ 、`DBName`、`MasterUsername`、`MasterUserPassword`は`RDSSubnetIds`、アプリケーションバンドルをデプロイするときに必要です。RDSSubnetIds には、2 つのプライベートサブネットを使用します。
- LoadBalancer:
  - これらのパラメータ 、`DBEngine`、 は、例に示す値がテスト済みですが`EngineVersion`、状況に応じて設定`LicenseModel`する必要があります。
  - `ELBSubnetIds`: 2 つのパブリックサブネットを使用します。

- アプリケーション: ApplicationName値は CodeDeploy アプリケーション名と CodeDeploy デプロイグループ名を設定します。これを使用してアプリケーションをデプロイします。アカウント内で一意である必要があります。アカウントで CodeDeploy 名を確認するには、CodeDeploy コンソール」を参照してください。この例では「WordPress」を使用していますが、その値を使用する場合は、まだ使用されていないことを確認してください。

この手順では、高可用性 2 層スタック (アドバンスド) CT (ct-06mjngx5flwto) と Create S3 storage CT (ct-1a68ck03fn98r) を使用します。認証されたアカウントから、コマンドラインで以下のステップに従います。

1. インフラストラクチャスタックを起動します。

   a. HA 2 層スタック CT の実行パラメータ JSON スキーマを、CreateStackParams.json.

```
aws amscm get-change-type-version --change-type-id "ct-06mjngx5flwto"
 --query "ChangeTypeVersion.ExecutionInputSchema" --output text >
 CreateStackParams.json
```

   b. スキーマを変更します。必要に応じて##を置き換えます。たとえば、ASG が作成する EC2 インスタンスに必要な OS を使用します。後でアプリケーションのデプロイに使用するApplicationNameため、 を記録します。最大 50 個のタグを追加できます。

```
{
"Description":      "HA two tier stack for WordPress",
"Name":             "WordPressStack",
"TimeoutInMinutes":  360,
"Tags": [
        {
            "Key": "ApplicationName",
            "Value": "WordPress"
        }
    ],
"AutoScalingGroup": {
            "AmiId":    "AMI-ID",
            "UserData": "#!/bin/bash \n
            REGION=$(curl 169.254.169.254/latest/meta-data/placement/
availability-zone/ | sed 's/[a-z]$//') \n
            yum -y install ruby httpd \n
            chkconfig httpd on \n
            service httpd start \n
            touch /var/www/html/status \n
```

```
            cd /tmp \n
            curl -O https://aws-codedeploy-$REGION.s3.amazonaws.com/latest/
install \n
            chmod +x ./install \n
            ./install auto \n
            chkconfig codedeploy-agent on \n
            service codedeploy-agent start"
    },
    "LoadBalancer": {
        "Public":               true,
        "HealthCheckTarget":    "HTTP:80/status"
    },
    "Database":     {
        "DBEngine":             "MySQL",
        "DBName":               "wordpress",
        "EngineVersion":        "8.0.16 ",
        "LicenseModel":         "general-public-license",
        "MasterUsername":       "admin",
        "MasterUserPassword":   "p4ssw0rd"
    },
    "Application":  {
    "ApplicationName":  "WordPress"
        }
}
```

c. CreateStackRfc.CreateStackRfc.json:

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

d. RFC テンプレートを次のように変更して保存します。コンテンツを削除して置き換えることができます。RequestedStartTime と RequestedEndTimeはオプションになりました。これらを除外すると、承認されるとすぐに実行される ASAP RFC が作成されます (通常は自動的に実行されます) 。スケジュールされた RFC を送信するには、これらの値を追加します。

```
{
"ChangeTypeVersion":    "3.0",
"ChangeTypeId":         "ct-06mjngx5flwto",
"Title":                "HA-Stack-For-WP-RFC"
}
```

e. CreateStackRfc.json ファイルと CreateStackParams.json 実行パラメータファイルを指定して、RFC を作成します。

```
aws amscm create-rfc --cli-input-json file://CreateStackRfc.json --execution-
parameters file://CreateStackParams.json
```

レスポンスで RFC ID を受け取ります。後続のステップの ID を保存します。

f. RFC を送信します。

```
aws amscm submit-rfc --rfc-id  RFC_ID
```

RFC が成功した場合、出力は受信されません。

g. RFC ステータスを確認するには、 を実行します。

```
aws amscm get-rfc --rfc-id RFC_ID
```

RFC ID を書き留めておきます。

2. S3 バケットを起動する

開始する前に次のデータを収集すると、デプロイがより速くなります。

必須データ S3 バケット：

- VPC-ID: この値により、S3 バケットの場所が決まります。以前に使用したものと同じ VPC ID を使用します。
- BucketName: この値は S3 バケット名を設定します。これを使用してアプリケーションバンドルをアップロードします。アカウントのリージョン全体で一意である必要があり、大文字を含めることはできません。BucketName の一部としてアカウント ID を含めることは必須ではありませんが、後でバケットを識別しやすくなります。アカウントに存在する S3 バケット名を確認するには、アカウントの Amazon S3 コンソールに移動します。

a. S3 ストレージ作成 CT の実行パラメータ JSON スキーマを CreateS3StoreParams.json.

```
aws amscm get-change-type-version --change-type-id "ct-1a68ck03fn98r"
 --query "ChangeTypeVersion.ExecutionInputSchema" --output text >
 CreateS3StoreParams.json
```

b. スキーマを次のように変更し、コンテンツを削除して置き換えることができます。*VPC_ID* を適切に置き換えます。この例の値はテスト済みですが、適切ではない可能性があります。

> **Tip**
>
> はアカウントのリージョン全体で一意BucketNameである必要があり、大文字を含めることはできません。BucketName の一部としてアカウント ID を含めることは必須ではありませんが、後でバケットを識別しやすくなります。アカウントに存在する S3 バケット名を確認するには、アカウントの Amazon S3 コンソールに移動します。

```
{
"Description":      "S3BucketForWordPressBundle",
"VpcId":            "VPC_ID",
"StackTemplateId":  "stm-s2b72beb000000000",
"Name":             "S3BucketForWP",
"TimeoutInMinutes":  60,
"Parameters":   {
    "AccessControl":    "Private",
    "BucketName":       "ACCOUNT_ID-BUCKET_NAME"
    }
}
```

c. CreateRfc の JSON テンプレートを、CreateS3StoreRfc.json:

```
aws amscm create-rfc --generate-cli-skeleton > CreateS3StoreRfc.json
```

d. CreateS3StoreRfc.json ファイルを変更して保存し、コンテンツを削除して置き換えることができます。RequestedStartTime と RequestedEndTimeはオプションになりました。これらを除外すると、承認されるとすぐに実行される ASAP RFC が作成されます (通常は自動的に実行されます) 。スケジュールされた RFC を送信するには、これらの値を追加します。

```
{
"ChangeTypeVersion":    "1.0",
"ChangeTypeId":         "ct-1a68ck03fn98r",
"Title":                "S3-Stack-For-WP-RFC"
}
```

e. CreateS3StoreRfc.json ファイルと CreateS3StoreParams.json 実行パラメータファイルを指定して、RFC を作成します。

```
aws amscm create-rfc --cli-input-json file://CreateS3StoreRfc.json  --
execution-parameters file://CreateS3StoreParams.json
```

レスポンスで新しい RFC の RfcId を受け取ります。後続のステップの ID を保存します。

f. RFC を送信します。

```
aws amscm submit-rfc --rfc-id RFC_ID
```

RFC が成功した場合、出力は受信されません。

g. RFC ステータスを確認するには、 を実行します。

```
aws amscm get-rfc --rfc-id RFC_ID
```

アプリケーションの作成、アップロード、デプロイ

まず、WordPress アプリケーションバンドルを作成し、CodeDeploy CT を使用してアプリケーションを作成してデプロイします。 CTs

1. WordPress をダウンロードし、ファイルを抽出して ./scripts ディレクトリを作成します。

   Linux コマンド：

   ```
   wget https://github.com/WordPress/WordPress/archive/master.zip
   ```

   Windows: ブラウザウィンドウ`https://github.com/WordPress/WordPress/archive/master.zip`に貼り付け、zip ファイルをダウンロードします。

   パッケージをアセンブルする一時ディレクトリを作成します。

   Linux:

   ```
   mkdir /tmp/WordPress
   ```

   Windows: WordPress」ディレクトリを作成します。後でディレクトリパスを使用します。

2. WordPress ソースをWordPress」ディレクトリに抽出し、./scripts ディレクトリを作成します。

   Linux:

   ```
   unzip master.zip -d /tmp/WordPress_Temp
   cp -paf /tmp/WordPress_Temp/WordPress-master/* /tmp/WordPress
   rm -rf /tmp/WordPress_Temp
   rm -f master
   cd /tmp/WordPress
   mkdir scripts
   ```

   Windows: 作成したWordPress」ディレクトリに移動し、そこに「scripts」ディレクトリを作成します。

   Windows 環境の場合は、スクリプトファイルのブレークタイプを Unix (LF) に設定してください。Notepad ++ では、これはウィンドウの右下にあるオプションです。

3. WordPress ディレクトリに CodeDeploy appspec.yml ファイルを作成します (例をコピーする場合は、インデント、各スペース数を確認します) 。重要: WordPress ファイル (この場合は WordPress ディレクトリ) を目的の宛先 (/var/www/html/WordPress) にコピーするための WordPress 「ソース」パスが正しいことを確認します。この例では、appspec.yml ファイルは WordPress ファイルを含む ディレクトリにあるため、「/」のみが必要です。また、Auto Scaling グループに RHEL AMI を使用した場合でも、「os: linux」行はそのままにしておきます。appspec.yml ファイルの例 :

   ```
   version: 0.0
   os: linux
   files:
     - source: /
       destination: /var/www/html/WordPress
   hooks:
     BeforeInstall:
       - location: scripts/install_dependencies.sh
         timeout: 300
         runas: root
     AfterInstall:
       - location: scripts/config_wordpress.sh
         timeout: 300
         runas: root
     ApplicationStart:
       - location: scripts/start_server.sh
         timeout: 300
   ```

```
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

4. WordPress ./scripts ディレクトリに bash ファイルスクリプトを作成します。

   まず、次の内容config_wordpress.shで を作成します (必要に応じて、wp-config.php ファイルを直接編集できます) 。

> **Note**
>
> *DBName* を HA スタック RFC で指定された値 (例:) に置き換えますwordpress。
> *DB_MasterUsername* を HA スタック RFC で指定されたMasterUsername値 (などadmin) に置き換えます。
> *DB_MasterUserPassword* を HA スタック RFC で指定されたMasterUserPassword値 (などp4ssw0rd) に置き換えます。
> *DB_ENDPOINT* を HA スタック RFC の実行出力のエンドポイント DNS 名 (例:) に置き換えますsrt1cz23n45sfg.clgvd67uvydk.us-east-1.rds.amazonaws.com。これは、GetRfc オペレーション (CLI: get-rfc --rfc-id RFC_ID) または以前に送信した HA スタック RFC の AMS コンソール RFC の詳細ページで確認できます。

```
#!/bin/bash
chmod -R 755 /var/www/html/WordPress
cp /var/www/html/WordPress/wp-config-sample.php /var/www/html/WordPress/wp-
config.php
cd /var/www/html/WordPress
sed -i "s/database_name_here/DBName/g" wp-config.php
sed -i "s/username_here/DB_MasterUsername/g" wp-config.php
sed -i "s/password_here/DB_MasterUserPassword/g" wp-config.php
sed -i "s/localhost/DB_ENDPOINT/g" wp-config.php
```

5. 同じディレクトリに、次の内容install_dependencies.shで を作成します。

```
#!/bin/bash
yum install -y php
yum install -y php-mysql
yum install -y mysql
```

```
service httpd restart
```

> **Note**
>
> HTTPS は、ヘルスチェックを最初から機能させるために、起動時にユーザーデータの一部としてインストールされます。

6. 同じディレクトリに、次の内容`start_server.sh`で を作成します。

   - Amazon Linux インスタンスの場合は、以下を使用します。

   ```
   #!/bin/bash
   service httpd start
   ```

   - RHEL インスタンスの場合は、以下を使用します (追加のコマンドは、SELINUX が WordPress を受け入れることを許可するポリシーです) 。

   ```
   #!/bin/bash
   setsebool -P  httpd_can_network_connect_db 1
   setsebool -P  httpd_can_network_connect 1
   chcon -t httpd_sys_rw_content_t /var/www/html/WordPress/wp-content -R
   restorecon -Rv /var/www/html
   service httpd start
   ```

7. 同じディレクトリに、次の内容`stop_server.sh`で を作成します。

   ```
   #!/bin/bash
   service httpd stop
   ```

8. zip バンドルを作成します。

   Linux:

   ```
   $ cd /tmp/WordPress
   $ zip -r wordpress.zip .
   ```

   Windows: WordPress」ディレクトリに移動し、すべてのファイルを選択して zip ファイルを作成します。必ず wordpress.zip という名前を付けます。

1. アプリケーションバンドルを S3 バケットにアップロードします。

スタックのデプロイを続行するには、バンドルを設定する必要があります。

作成した S3 バケットインスタンスに自動的にアクセスできます。踏み台または S3 コンソールからアクセスし、WordPress バンドルをdrag-and-dropでアップロードするか、zip ファイルを参照して選択できます。

シェルウィンドウで次のコマンドを使用することもできます。zip ファイルへの正しいパスがあることを確認してください。

```
aws s3 cp wordpress.zip s3://BUCKET_NAME/
```

2. WordPress アプリケーションバンドルをデプロイします。

開始する前に次のデータを収集すると、デプロイがより速くなります。

必須データ：

- VPC-ID: この値により、S3 バケットの場所が決まります。以前に使用したものと同じ VPC ID を使用します。
- CodeDeployApplicationName および CodeDeployApplicationName: HA 2-Tier スタック RFC で使用したApplicationName値は、CodeDeployApplicationName と CodeDeployDeploymentGroupName を設定します。この例では「WordPress」を使用していますが、別の値を使用した可能性があります。
- S3Location: にはS3Bucket、BucketName以前に作成した を使用します。S3BundleType と S3Keyは、S3 ストアに配置したバンドルのものです。

a. CodeDeploy アプリケーションデプロイ CT の実行パラメータ JSON スキーマを DeployCDAppParams.json.

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

b. スキーマを次のように変更し、 として保存します。コンテンツを削除して置き換えることができます。

```
{
"Description":                          "DeployWPCDApp",
"VpcId":                                "VPC_ID",
```

```
"Name":                                "WordPressCDAppDeploy",
"TimeoutInMinutes":                    60,
"Parameters":   {
    "CodeDeployApplicationName":                "WordPress",
    "CodeDeployDeploymentGroupName":            "WordPress",
    "CodeDeployIgnoreApplicationStopFailures":   false,
    "CodeDeployRevision": {
      "RevisionType": "S3",
      "S3Location": {
        "S3Bucket":     "BUCKET_NAME",
        "S3BundleType": "zip",
        "S3Key":        "wordpress.zip" }
        }
    }
}
```

c. CreateRfc の JSON テンプレートを、現在のフォルダの DeployCDAppRfc.json:

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

d. DeployCDAppRfc.json ファイルを変更して保存し、コンテンツを削除して置き換えることができます。RequestedStartTime と RequestedEndTimeはオプションになりました。これらを除外すると、承認されるとすぐに実行される ASAP RFC が作成されます (通常は自動的に実行されます) 。スケジュールされた RFC を送信するには、これらの値を追加します。

```
{
"ChangeTypeVersion":    "1.0",
"ChangeTypeId":         "ct-2edc3sd1sqmrb",
"Title":                "CD-Deploy-For-WP-RFC"
}
```

e. DeployCDAppRfc ファイルと DeployCDAppParams 実行パラメータファイルを指定して、RFC を作成します。

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

レスポンスで新しい RFC の RfcId を受け取ります。後続のステップの ID を保存します。

f. RFC を送信します。

```
aws amscm submit-rfc --rfc-id RFC_ID
```

RFC が成功した場合、出力は受信されません。

g. RFC ステータスを確認するには、 を実行します。

```
aws amscm get-rfc --rfc-id RFC_ID
```

アプリケーションのデプロイを検証する

WordPress デプロイパス: /WordPress を使用して、以前に作成したロードバランサーのエンドポイント (ELB CName) WordPress に移動します。例：

```
http://stack-ID-FOR-ELB.us-east-1.elb.amazonaws.com/WordPress
```

アプリケーションのデプロイをティアダウンする

チュートリアルが終了したら、リソースに対して課金されないようにデプロイを破棄します。

以下は、一般的なスタック削除オペレーションです。HA 2-Tierスタックの場合は 1 回、S3 バケットスタックの場合は 1 回、2 回送信します。最後のフォローアップとして、S3 バケットのすべてのスナップショット (サービスリクエストに S3 バケットスタック ID を含める) を削除するサービスリクエストを送信します。これらは 10 日後に自動的に削除されますが、早期に削除すると多少のコストを節約できます。

このチュートリアルでは、AMS コンソールを使用して S3 スタックを削除する例を示します。この手順は、AMS コンソールを使用してスタックを削除する場合に適用されます。

> **Note**
>
> S3 バケットを削除する場合は、まずオブジェクトを空にする必要があります。

必須データ：

- `StackId`: 使用するスタック。これは、左側のナビゲーションのリンクから入手できる AMS コンソールスタックページから確認できます。AMS SKMS API/CLI を使用して、AMS SKMS API

リファレンスについては、AWS Artifact Console のレポートタブを参照してください。 オペレーション (`list-stack-summaries` CLI の)。

- このウォークスルーの変更タイプ ID は でct-0q0bic0ywqk6c、バージョンは「1.0」です。最新バージョンを確認するには、次のコマンドを実行します。

```
aws amscm list-change-type-version-summaries --filter
 Attribute=ChangeTypeId,Value=ct-0q0bic0ywqk6c
```

インライン作成：

- インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を使用して、create RFC コマンドを発行します。E

```
aws amscm create-rfc --change-type-id "ct-0q0bic0ywqk6c" --change-type-version "1.0"
 --title "Delete My Stack" --execution-parameters "{\"StackId\":\"STACK_ID\"}"
```

- RFC の作成オペレーションで返された RFC ID を使用して RFC を送信します。送信されるまで、RFC は `Editing`状態のままで、 に対して動作しません。

```
aws amscm submit-rfc --rfc-id RFC_ID
```

- RFC ステータスをモニタリングし、実行出力を表示します。

```
aws amscm get-rfc --rfc-id RFC_ID
```

テンプレートの作成：

1. RFC テンプレートを現在のフォルダ内のファイルに出力します。例名は DeleteStackRfc.json:

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

2. DeleteStackRfc.json ファイルを変更して保存します。スタックの削除には実行パラメータが 1 つしかないため、実行パラメータは DeleteStackRfc.json ファイル自体に含めることができます (実行パラメータを使用して別の JSON ファイルを作成する必要はありません)。

   ExecutionParameters JSON 拡張機能の内部引用符は、バックスラッシュ (\) で エスケープする必要があります。開始時刻と終了時刻のない例：

```
{
"ChangeTypeVersion":    "1.0",
"ChangeTypeId":         "ct-0q0bic0ywqk6c",
"Title":                "Delete-My-Stack-RFC"
"ExecutionParameters":  "{
        \"StackId\":\"STACK_ID\"}"
}
```

3. RFC を作成します。

```
aws amscm create-rfc --cli-input-json file://DeleteStackRfc.json
```

レスポンスで新しい RFC の RfcId を受け取ります。例:

```
{
"RfcId": "daaa1867-ffc5-1473-192a-842f6b326102"
}
```

後続のステップの ID を保存します。

4. RFC を送信します。

```
aws amscm submit-rfc --rfc-id RFC_ID
```

RFC が成功した場合、コマンドラインに確認は送信されません。

5. リクエストのステータスをモニタリングし、実行出力を表示するには :

```
aws amscm get-rfc --rfc-id RFC_ID --query "Rfc.
{Status:Status.Name,Exec:ExecutionOutput}" --output table
```

CLI チュートリアル: 階層のデプロイと WordPress ウェブサイトの絞り込み

このセクションでは、AMS CLI を使用して高可用性 (HA) WordPress サイトを AMS 環境にデプロイする方法について説明します。この一連の手順には、必要な WordPress CodeDeploy 互換パッケージ (zip など) ファイルを作成する例が含まれています。

> **Note**
>
> このデプロイチュートリアルは、AMZN Linux 環境で使用するために設計されています。必須変数パラメータは######と表記されますが、状況に合わせて他のパラメータを変更することもできます。

タスクと必要な RFCs の概要：

1. インフラストラクチャを作成します。
   a. RDS スタックを作成する (CLI)
   b. ロードバランサーの作成
   c. Auto Scaling グループを作成し、ロードバランサーに結び付ける
   d. CodeDeploy アプリケーション用の S3 バケットを作成する
2. WordPress アプリケーションバンドルを作成する (RFC は必要ありません)
3. CodeDeploy を使用して WordPress アプリケーションバンドルをデプロイします。
   a. CodeDeploy アプリケーションを作成する
   b. CodeDeploy デプロイグループを作成する
   c. WordPress アプリケーションバンドルを S3 バケットにアップロードする (RFC は必要ありません)
   d. CodeDeploy アプリケーションをデプロイする
4. デプロイを検証する
5. デプロイをティアダウンする

認証されたアカウントからコマンドラインにあるすべてのステップに従います。

CLI を使用した RFC の作成

RFC の作成の詳細については、RFCs」を参照してください。一般的な RFC パラメータの説明については、「RFC 共通パラメータ」を参照してください。

インフラストラクチャを作成する

次の手順では、リソース IDs を使用してインフラストラクチャを構築する方法で RDS データベース、ロードバランサー、Auto Scaling グループを作成する方法について説明します。

RDS スタックを作成する (CLI)

「RDS スタック | 作成」を参照してください。

ELB スタックを作成する

パブリックロードバランサー (ELB) を起動します。Load Balancer (ELB) スタック | 作成」を参照してください。

Auto Scaling グループスタックを作成する

Auto Scaling グループを起動します。

Auto Scaling グループ | 作成」を参照してください。

S3 ストアを作成する

S3 バケットを起動します。S3 バケットは、作成したアプリケーションバンドルをアップロードする場所です。S3 ストレージ | 作成」を参照してください。

CodeDeploy 用の WordPress アプリケーションバンドルを作成する

このセクションでは、アプリケーションデプロイバンドルを作成する例を示します。

1. WordPress をダウンロードし、ファイルを抽出して ./scripts ディレクトリを作成します。

   Linux コマンド：

   ```
   wget https://github.com/WordPress/WordPress/archive/master.zip
   ```

   Windows: ブラウザウィンドウ`https://github.com/WordPress/WordPress/archive/master.zip`に貼り付け、zip ファイルをダウンロードします。

   パッケージをアセンブルする一時ディレクトリを作成します。

   Linux:

   ```
   mkdir /tmp/WordPress
   ```

   Windows: WordPress」ディレクトリを作成します。後でディレクトリパスを使用します。

2. WordPress ソースをWordPress」ディレクトリに抽出し、./scripts ディレクトリを作成します。

   Linux:

   ```
   unzip master.zip -d /tmp/WordPress_Temp
   cp -paf /tmp/WordPress_Temp/WordPress-master/* /tmp/WordPress
   rm -rf /tmp/WordPress_Temp
   rm -f master
   cd /tmp/WordPress
   mkdir scripts
   ```

   Windows: 作成したWordPress」ディレクトリに移動し、そこに「scripts」ディレクトリを作成します。

   Windows 環境の場合は、スクリプトファイルのブレークタイプを Unix (LF) に設定してください。Notepad ++ では、これはウィンドウの右下にあるオプションです。

3. WordPress ディレクトリに CodeDeploy appspec.yml ファイルを作成します (例をコピーする場合は、インデントを確認し、各スペース数を確認します) 。重要: WordPress ファイル (この場合は WordPress ディレクトリ) を目的の宛先 (/var/www/html/WordPress) にコピーするには、 WordPress 「ソース」パスが正しいことを確認してください。この例では、appspec.yml ファイルは WordPress ファイルを含む ディレクトリにあるため、「/」のみが必要です。また、Auto Scaling グループに RHEL AMI を使用した場合でも、「os: linux」行はそのままにしておきます。appspec.yml ファイルの例 :

   ```
   version: 0.0
   os: linux
   files:
     - source: /
       destination: /var/www/html/WordPress
   hooks:
     BeforeInstall:
       - location: scripts/install_dependencies.sh
         timeout: 300
         runas: root
     AfterInstall:
       - location: scripts/config_wordpress.sh
         timeout: 300
         runas: root
     ApplicationStart:
       - location: scripts/start_server.sh
         timeout: 300
   ```

```
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

4. WordPress ./scripts ディレクトリに bash ファイルスクリプトを作成します。

   まず、次の内容config_wordpress.shで を作成します (必要に応じて、wp-config.php ファイルを直接編集できます) 。

> **Note**
>
> *DBName* を HA スタック RFC で指定された値 (例:) に置き換えますwordpress。
> *DB_MasterUsername* を HA スタック RFC で指定されたMasterUsername値 (などadmin) に置き換えます。
> *DB_MasterUserPassword* を HA スタック RFC で指定されたMasterUserPassword値 (など) に置き換えますp4ssw0rd。
> *DB_ENDPOINT* を HA スタック RFC の実行出力のエンドポイント DNS 名 (例:) に置き換えますsrt1cz23n45sfg.clgvd67uvydk.us-east-1.rds.amazonaws.com。これは、GetRfc オペレーション (CLI: get-rfc --rfc-id RFC_ID) または以前に送信した HA スタック RFC の AMS コンソール RFC の詳細ページで確認できます。

```
#!/bin/bash
chmod -R 755 /var/www/html/WordPress
cp /var/www/html/WordPress/wp-config-sample.php /var/www/html/WordPress/wp-
config.php
cd /var/www/html/WordPress
sed -i "s/database_name_here/DBName/g" wp-config.php
sed -i "s/username_here/DB_MasterUsername/g" wp-config.php
sed -i "s/password_here/DB_MasterUserPassword/g" wp-config.php
sed -i "s/localhost/DB_ENDPOINT/g" wp-config.php
```

5. 同じディレクトリに、次の内容install_dependencies.shで を作成します。

```
#!/bin/bash
yum install -y php
yum install -y php-mysql
yum install -y mysql
```

```
service httpd restart
```

> **Note**
>
> HTTPS は、ヘルスチェックを最初から機能させるために、起動時にユーザーデータの一部としてインストールされます。

6. 同じディレクトリに、次の内容`start_server.sh`で を作成します。

   - Amazon Linux インスタンスの場合は、以下を使用します。

     ```
     #!/bin/bash
     service httpd start
     ```

   - RHEL インスタンスの場合は、以下を使用します (追加のコマンドは、SELINUX が WordPress を受け入れることを許可するポリシーです) 。

     ```
     #!/bin/bash
     setsebool -P  httpd_can_network_connect_db 1
     setsebool -P  httpd_can_network_connect 1
     chcon -t httpd_sys_rw_content_t /var/www/html/WordPress/wp-content -R
     restorecon -Rv /var/www/html
     service httpd start
     ```

7. 同じディレクトリに、次の内容`stop_server.sh`で を作成します。

   ```
   #!/bin/bash
   service httpd stop
   ```

8. zip バンドルを作成します。

   Linux:

   ```
   $ cd /tmp/WordPress
   $ zip -r wordpress.zip .
   ```

   Windows: WordPress」ディレクトリに移動し、すべてのファイルを選択して zip ファイルを作成します。必ず wordpress.zip という名前を付けます。

CodeDeploy を使用して WordPress アプリケーションバンドルをデプロイする

CodeDeploy は、Amazon EC2 インスタンスへのアプリケーションのデプロイを自動化する AWS デプロイサービスです。プロセスのこの部分では、CodeDeploy アプリケーションを作成し、CodeDeploy デプロイグループを作成し、CodeDeploy を使用してアプリケーションをデプロイします。

CodeDeploy アプリケーションを作成する

CodeDeploy アプリケーションは、AWS CodeDeploy が使用する名前またはコンテナであり、デプロイ中に正しいリビジョン、デプロイ設定、デプロイグループが参照されるようにします。この場合、デプロイ設定は、以前に作成した WordPress バンドルです。

必須データ：

- `VpcId`: 使用している VPC。これは以前に使用した VPC と同じである必要があります。
- `CodeDeployApplicationName`: アカウント内で一意である必要があります。CodeDeploy コンソールで、既存のアプリケーション名を確認します。
- `ChangeTypeId` および `ChangeTypeVersion`: このチュートリアルの変更タイプ ID は です。最新バージョン`ct-0ah3gwb9seqk2`を確認するには、次のコマンドを実行します。

```
aws amscm list-change-type-version-summaries --filter
 Attribute=ChangeTypeId,Value=ct-0ah3gwb9seqk2
```

1. CodeDeploy アプリケーション CT の実行パラメータ JSON スキーマを現在のフォルダ内のファイルに出力します。例では CreateCDAppParams.json.

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

2. 次のように JSON ファイルを変更して保存します。コンテンツを削除して置き換えることができます。

```
{
"Description":                    "Create WordPress CodeDeploy App",
"VpcId":                          "VPC_ID",
"StackTemplateId":                "stm-sft6rv00000000000",
```

```
"Name":                                "WordPressCDApp",
"TimeoutInMinutes":                    60,
"Parameters":   {
    "CodeDeployApplicationName":    "WordPressCDApp"
    }
}
```

3. CreateRfc の JSON テンプレートを現在のフォルダ内のファイルに出力します。例では CreateCDAppRfc.json.

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

4. 次のように JSON ファイルを変更して保存します。コンテンツを削除して置き換えることができます。RequestedStartTime と RequestedEndTimeはオプションになりました。これらを除外すると、RFC は承認されるとすぐに実行されることに注意してください (通常は自動的に実行されます) 。「スケジュールされた」 RFC を送信するには、これらの値を追加します。

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

5. CreateCDAppRfc ファイルと実行パラメータファイルを指定して、RFC を作成します。

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

レスポンスで新しい RFC の RFC ID を受け取ります。後続のステップの ID を保存します。

6. RFC を送信します。

```
aws amscm submit-rfc --rfc-id RFC_ID
```

RFC が成功した場合、出力は受信されません。

7. RFC を送信します。

```
aws amscm get-rfc --rfc-id RFC_ID
```

CodeDeploy デプロイグループを作成する

CodeDeploy デプロイグループを作成します。

CodeDeploy デプロイグループは、デプロイの対象となる個々のインスタンスのセットを定義します。

必須データ：

- VpcId: 使用している VPC。これは以前に使用した VPC と同じである必要があります。
- CodeDeployApplicationName: 以前に作成した値を使用します。
- CodeDeployAutoScalingGroups: 前に作成した Auto Scaling グループの名前を使用します。
- CodeDeployDeploymentGroupName: デプロイグループの名前。この名前はデプロイグループに関連付けられる各アプリケーションで一意にする必要があります。
- CodeDeployServiceRoleArn: 例に示す式を使用します。
- ChangeTypeId および ChangeTypeVersion: このチュートリアルの変更タイプ ID は です。最新バージョンct-2gd0u847qd9d2を確認するには、次のコマンドを実行します。

```
aws amscm list-change-type-version-summaries --filter
 Attribute=ChangeTypeId,Value=ct-2gd0u847qd9d2
```

1. 実行パラメータ JSON スキーマを現在のフォルダ内のファイルに出力します。例では CreateCDDepGroupParams.json.

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

2. 次のように JSON ファイルを変更して保存します。コンテンツを削除して置き換えることができます。

```
{
"Description":                    "CreateWPCDDeploymentGroup",
"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. CreateRfc の JSON テンプレートを現在のフォルダ内のファイルに出力します。例では CreateCDDepGroupRfc.json.

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

4. 次のように JSON ファイルを変更して保存します。コンテンツを削除して置き換えることができます。RequestedStartTime と RequestedEndTimeはオプションになりました。これらを除外すると、RFC は承認されるとすぐに実行されることに注意してください (通常は自動的に実行されます) 。「スケジュールされた」 RFC を送信するには、これらの値を追加します。

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

5. CreateCDDepGroupRfc ファイルと実行パラメータファイルを指定して、RFC を作成します。

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

レスポンスで新しい RFC の RFC ID を受け取ります。後続のステップの ID を保存します。

6. RFC を送信します。

```
aws amscm submit-rfc --rfc-id RFC_ID
```

RFC が成功した場合、出力は受信されません。

7. RFC ステータスを確認します。

```
aws amscm get-rfc --rfc-id RFC_ID
```

WordPress アプリケーションをアップロードする

作成した S3 バケットインスタンスに自動的にアクセスできます。踏み台 (「インスタンスへのアクセス」を参照) または S3 コンソールからアクセスし、CodeDeploy バンドルをアップロードできます。スタックのデプロイを続行するには、バンドルを設定する必要があります。この例では、以前に作成したバケット名を使用します。

```
aws s3 cp wordpress/wordpress.zip s3://ACCOUNT_ID-codedeploy-bundles/
```

CodeDeploy を使用して WordPress アプリケーションをデプロイする

CodeDeploy アプリケーションをデプロイします。

CodeDeploy アプリケーションバンドルとデプロイグループを取得したら、この RFC を使用してアプリケーションをデプロイします。

必須データ：

- VPC-ID: 使用している VPC。これは以前に使用した VPC と同じである必要があります。
- CodeDeployApplicationName: 以前に作成した CodeDeploy アプリケーションの名前を使用します。
- CodeDeployDeploymentGroupName: 前に作成した CodeDeploy デプロイグループの名前を使用します。
- S3Location (アプリケーションバンドルをアップロードした場所): S3Bucket: 以前に作成した BucketName、S3BundleTypeおよび S3Key: S3 ストアに配置したバンドルのタイプと名前。
- ChangeTypeId および ChangeTypeVersion: このチュートリアルの変更タイプ ID は です。最新バージョンct-2edc3sd1sqmrbを確認するには、次のコマンドを実行します。

```
aws amscm list-change-type-version-summaries --filter
 Attribute=ChangeTypeId,Value=ct-2edc3sd1sqmrb
```

1. CodeDeploy アプリケーションデプロイ CT の実行パラメータ JSON スキーマを現在のフォルダ内のファイルに出力します。例では DeployCDAppParams.json.

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

2. 次のように JSON ファイルを変更します。コンテンツを削除して置き換えることができます。にはS3Bucket、BucketName以前に作成した を使用します。

```
{
"Description":                          "Deploy WordPress CodeDeploy Application",
"VpcId":                                "VPC_ID",
"Name":                                 "WP CodeDeploy Deployment Group",
"TimeoutInMinutes":                     60,
"Parameters":   {
    "CodeDeployApplicationName":        "WordPressCDApp",
    "CodeDeployDeploymentGroupName":    "WordPressCDDepGroup",
    "CodeDeployIgnoreApplicationStopFailures": false,
    "CodeDeployRevision": {
      "RevisionType": "S3",
      "S3Location": {
        "S3Bucket": "ACCOUNT_ID.BUCKET_NAME",
        "S3BundleType": "zip",
        "S3Key": "wordpress.zip" }
        }
    }
}
```

3. CreateRfc の JSON テンプレートを現在のフォルダのファイルに出力します。例の名前は DeployCDAppRfc.json:

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

4. DeployCDAppRfc.json ファイルを変更して保存します。コンテンツを削除して置き換えることができます。

```
{
"ChangeTypeVersion":    "1.0",
"ChangeTypeId":         "ct-2edc3sd1sqmrb",
"Title":                "CD-Deploy-For-WP-Stack-RFC",
"RequestedStartTime":   "2017-04-28T22:45:00Z",
"RequestedEndTime":     "2017-04-28T22:45:00Z"
}
```

5. RFC を作成し、実行パラメータファイルと DeployCDAppRfc ファイルを指定します。

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

レスポンスで新しい RFC の RfcId を受け取ります。後続のステップの ID を保存します。

6. RFC を送信します。

```
aws amscm submit-rfc --rfc-id RFC_ID
```

RFC が成功した場合、出力は受信されません。

アプリケーションのデプロイを検証する

WordPress デプロイパス: /WordPress を使用して、以前に作成したロードバランサーのエンドポイント (ELB CName) WordPress に移動します。例:

```
http://stack-ID-FOR-ELB.us-east-1.elb.amazonaws.com/WordPress
```

アプリケーションのデプロイをティアダウンする

デプロイを破棄するには、RDS データベーススタック、アプリケーションロードバランサー、Auto Scaling グループ、S3 バケット、および Code Deploy アプリケーションと group-six RFCs に対して Delete Stack CT を送信します。さらに、削除する RDS スナップショットのサービスリクエストを送信できます (10 日後に自動的に削除されますが、その間は少額のコストがかかります) 。すべてのスタック IDsを収集し、以下の手順に従います。

このチュートリアルでは、AMS コンソールを使用して S3 スタックを削除する例を示します。この手順は、AMS コンソールを使用してスタックを削除する場合に適用されます。

> **Note**
>
> S3 バケットを削除する場合は、まずオブジェクトを空にする必要があります。

必須データ：

- `StackId`: 使用するスタック。これは、左側のナビゲーションのリンクから入手できる AMS コンソールスタックページから確認できます。AMS SKMS API/CLI を使用して、AMS SKMS API リファレンスの を実行します。AWS Artifact Console のレポートタブを参照してください。 オペレーション (`list-stack-summaries` CLI の)。

- このウォークスルーの変更タイプ ID は でct-0q0bic0ywqk6c、バージョンは「1.0」です。最新バージョンを確認するには、次のコマンドを実行します。

```
aws amscm list-change-type-version-summaries --filter
 Attribute=ChangeTypeId,Value=ct-0q0bic0ywqk6c
```

インライン作成：

- インラインで指定された実行パラメータ (インラインで実行パラメータを指定する場合は引用符をエスケープ) を使用して、create RFC コマンドを発行します。E

```
aws amscm create-rfc --change-type-id "ct-0q0bic0ywqk6c" --change-type-version "1.0"
 --title "Delete My Stack" --execution-parameters "{\"StackId\":\"STACK_ID\"}"
```

- RFC の作成オペレーションで返された RFC ID を使用して RFC を送信します。送信されるまで、RFC は Editing状態のままであり、 は動作しません。

```
aws amscm submit-rfc --rfc-id RFC_ID
```

- RFC ステータスをモニタリングし、実行出力を表示します。

```
aws amscm get-rfc --rfc-id RFC_ID
```

テンプレートの作成：

1. RFC テンプレートを現在のフォルダ内のファイルに出力します。例名は DeleteStackRfc.json:

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

2. DeleteStackRfc.json ファイルを変更して保存します。スタックの削除には実行パラメータが 1 つしかないため、実行パラメータは DeleteStackRfc.json ファイル自体に含めることができます (実行パラメータを使用して別の JSON ファイルを作成する必要はありません)。

   ExecutionParameters JSON 拡張機能の内部引用符は、バックスラッシュ (\) で エスケープする必要があります。開始時刻と終了時刻のない例：

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

```
"ChangeTypeId":          "ct-0q0bic0ywqk6c",
"Title":                 "Delete-My-Stack-RFC"
"ExecutionParameters":   "{
        \"StackId\":\"STACK_ID\"}"
}
```

3. RFC を作成します。

```
aws amscm create-rfc --cli-input-json file://DeleteStackRfc.json
```

レスポンスで新しい RFC の RfcId を受け取ります。例:

```
{
"RfcId": "daaa1867-ffc5-1473-192a-842f6b326102"
}
```

後続のステップの ID を保存します。

4. RFC を送信します。

```
aws amscm submit-rfc --rfc-id RFC_ID
```

RFC が成功した場合、コマンドラインに確認は送信されません。

5. リクエストのステータスをモニタリングし、実行出力を表示するには :

```
aws amscm get-rfc --rfc-id RFC_ID --query "Rfc.
{Status:Status.Name,Exec:ExecutionOutput}" --output table
```

アプリケーションのメンテナンス

インフラストラクチャがデプロイされたら、QA からステージング、本番稼働まで、すべての AMS 環境で一貫した方法でインフラストラクチャを更新するのが課題です。

このセクションでは、AMS ワークロードの取り込みプロセスの概要と、クラウドインフラストラクチャレイヤーを最新の状態に保つために使用できるさまざまな方法の例をいくつか紹介します。

アプリケーションメンテナンス戦略

アプリケーションのデプロイ方法は、アプリケーションのメンテナンス方法に影響します。このセクションでは、アプリケーションメンテナンスのいくつかの戦略について説明します。

環境の更新には、次のいずれかの変更が含まれる場合があります。

- セキュリティ更新
- アプリケーションの新しいバージョン
- アプリケーション設定の変更
- 依存関係の更新

> **Note**
>
> アプリケーションのデプロイでは、 メソッドに関係なく、必ず事前にサービスリクエストを提出して、アプリケーションをデプロイすることを AMS に通知します。

イミュータブルアプリケーションとミュータブルアプリケーションのインストール例

コンピューティングインスタンスのミュータビリティ	アプリのインストール方法	AMI
Mutable	CodeDeploy を使用する場合	AMS 提供
	手動	
	Chef または Puppet を使用したプルベース	
	Ansible または Salt を使用したプッシュベース	

コンピューティングインスタンスのミュータビリティ	アプリのインストール方法	AMI
Immutable	Golden AMI の使用	カスタム (AMS 提供)

CodeDeploy 対応 AMI を使用したミュータブルデプロイ

AWS CodeDeploy は、Amazon EC2 インスタンスやオンプレミスで実行されているインスタンスなど、すべてのインスタンスへのコードデプロイを自動化するサービスです。AMS で CodeDeploy を使用して、CodeDeploy アプリケーションを作成およびデプロイできます。AMS は CodeDeploy アプリケーションのデフォルトのインスタンスプロファイルを提供することに注意してください。

- Amazon Linux (バージョン 1)
- Amazon Linux 2
- RedHat 7
- CentOS 7

CodeDeploy を初めて使用する前に、いくつかのセットアップ手順を完了する必要があります。

1. AWS CLI のインストールまたはアップグレード
2. AWS CodeDeploy のサービスロールを作成し、デプロイでサービスロール ARN を使用する

すべての CT オプションの IDs「変更タイプリファレンス」に記載されています。

> **Note**
>
> 現在、このソリューションでは Amazon S3 ストレージを使用する必要があります。

ここでは基本的な手順について説明し、手順の詳細については AMS ユーザーガイドを参照してください。

1. Amazon S3 ストレージバケットを作成します。CT: ct-1a68ck03fn98r。S3 バケットではバージョニングが有効になっている必要があります (これを行う方法については、「バケットバージョニングの有効化」を参照してください)。

2. バンドルされた CodeDeploy アーティファクトを配置します。これは、AMS 経由でアクセスを Amazon S3 コンソールで行うことができます。または、このコマンドのバリエーションを使用します。

```
aws s3 cp ZIP_FILEPATH_AND_NAME s3://S3BUCKET_NAME/
```

3. AMS AMI `customer-` を見つけます。次のいずれかを使用します。

   - AMS コンソール: 関連する VPC の VPC 詳細ページ
   - AMS API AMS SKMS API リファレンスについては、AWS Artifact コンソールのレポートタブを参照してください。または CLI: `aws amsskms list-amis`

4. Autoscaling グループ (ASG) を作成します。CT: ct-2tylseo8rxfsc。AMS AMI を指定し、ポートを開くようにロードバランサーを設定し、 `customer-mc-ec2-instance-profile` に を指定しますASGIAMInstanceProfile。

5. CodeDeploy アプリケーションを作成します。CT: ct-0ah3gwb9seqk2。パラメータには、 などのアプリケーション名が含まれますWordpressProd。

6. CodeDeploy デプロイグループを作成します。CT: ct-2gd0u847qd9d2。パラメータには、CodeDeploy アプリケーション名、ASG 名、設定タイプ名、サービスロール ARN が含まれます。

7. CodeDeploy アプリケーションをデプロイします。CT: ct-2edc3sd1"rb。パラメータには、CodeDeploy アプリケーション名、設定タイプ名、デプロイグループ名、リビジョンタイプ、および CodeDeploy アーティファクトがある S3 バケットの場所が含まれます。

ミュータブルデプロイ、手動で設定および更新されたアプリケーションインスタンス

このアプリケーションデプロイ戦略は、アプリケーションインスタンスのシンプルで手動の更新です。これらは基本的なステップです。

すべての CT オプションの IDs「変更タイプリファレンス」に記載されています。

> **Note**
>
> 現在、このソリューションでは Amazon S3 ストレージを使用する必要があります。

基本的な手順については、「AMS ユーザーガイド」で詳しく説明します。

1. Amazon S3 ストレージバケットを作成します。CT: ct-1a68ck03fn98r。S3 バケットではバージョニングが有効になっている必要があります (これを行う方法については、「バケットバージョニングの有効化」を参照してください)。
2. バンドルされたアプリケーションアーティファクトを配置します (起動時にアプリケーションを起動して動作させる必要があるすべて)。これは、AMS 経由でアクセスをAmazon S3 コンソールで行うことができます。または、このコマンドのバリエーションを使用します。

```
aws s3 cp ZIP_FILEPATH_AND_NAME s3://S3BUCKET_NAME/
```

3. AMS AMI を検索すると、すべてに CodeDeploy が付けられます。「customer-」AMI を検索するには、次のいずれかを使用します。
   - AMS コンソール: 関連する VPC の VPC 詳細ページ
   - AMS API AMS SKMS API リファレンスについては、AWS Artifact コンソールのレポートタブを参照してください。または CLI: `aws amsskms list-amis`
4. その AMI を使用して EC2 インスタンスを作成します。CT: ct-14027q0sjyt1h。AMS AMI を指定`Key=backup, Value=true`し、タグを設定し、 `InstanceProfile`パラメータ`customer-mc-ec2-instance-profile`の を指定します。返されるインスタンス ID を書き留めます。
5. インスタンスへの管理者アクセスをリクエストします。CT: ct-1dmlg9g1l91h6。アカウントには FQDN が必要です。FQDN が不明な場合は、次の方法で確認できます。
   - ディレクトリサービス用の AWS マネジメントコンソールの使用 (セキュリティとアイデンティティの下) ディレクトリ名タブ。
   - これらのコマンド (戻りディレクトリクラス、DC+DC+DC=FQDN): Windows: `whoami / fqdn`または Linux: のいずれかを実行します`hostname --fqdn`。
6. インスタンスにログインします。「AMS ユーザーガイド」の「踏み台によるインスタンスへのアクセス」を参照してください。
7. バンドルされたアプリケーションファイルを S3 バケットからインスタンスにダウンロードします。
8. AMS へのサービスリクエストで即時バックアップをリクエストするには、インスタンス ID を知る必要があります。
9. アプリケーションを更新する必要がある場合は、新しいファイルを S3 バケットにロードし、ステップ 3 ~ 8 に従います。

プルベースのデプロイツール設定 AMI を使用したミュータブルデプロイ

この戦略は、 Managed Services Create EC2 CT の `InstanceUserData`パラメータに依存します。このパラメータの使用の詳細については、「ユーザーデータを使用したインスタンスの設定」を参照してください。この例では、Chef や Puppet などのプルベースのアプリケーションデプロイツールを想定しています。

CodeDeploy エージェントは、すべての AMS AMIs。サポートされている AMIs。

- Amazon Linux (バージョン 1)
- Amazon Linux 2
- RedHat 7
- CentOS 7

すべての CT オプションの IDs「変更タイプリファレンス」に記載されています。

> **Note**
>
> 現在、このソリューションでは Amazon S3 ストレージを使用する必要があります。

ここでは基本的な手順について説明し、手順の詳細については AMS ユーザーガイドを参照してください。

1. Amazon S3 ストレージバケットを作成します。CT: ct-1a68ck03fn98r。S3 バケットではバージョニングが有効になっている必要があります (これを行う方法については、「バケットバージョニングの有効化」を参照してください) 。
2. バンドルされた CodeDeploy アーティファクトを配置します。これは、AMS 経由でアクセスを Amazon S3 コンソールで行うことができます。または、このコマンドのバリエーションを使用します。

   ```
   aws s3 cp ZIP_FILEPATH_AND_NAME s3://S3BUCKET_NAME/
   ```

3. AMS AMI `customer-` を見つけます。次のいずれかを使用します。

   - AMS コンソール: 関連する VPC の VPC 詳細ページ

- AMS API AMS SKMS API リファレンスについては、AWS Artifact コンソールのレポートタブを参照してください。または CLI: `aws amsskms list-amis`

4. EC2 インスタンスを作成します。CT: ct-14027q0sjyt1h; タグ を設定し`Key=backup,Value=true`、 `InstanceUserData`パラメータを使用してブートストラップやその他のスクリプト (Chef/Puppet エージェントのダウンロードなど) を指定し、必要な認可キーを含めます。これを行う例については、「AMS ユーザーガイド」の「変更管理」セクションの「HA 2 層デプロイの作成例」を参照してください。または、インスタンスへのアクセスをリクエストしてログインし、必要なデプロイアーティファクトで設定します。プルベースのデプロイコマンドは、インスタンスのエージェントから企業のマスターサーバーに送信され、踏み台を通過するための認可が必要になる場合があることに注意してください。踏み台なしでセキュリティグループ/AD グループアクセスをリクエストするには、AMS へのサービスリクエストが必要になる場合があります。
5. ステップ 4 を繰り返して別の EC2 インスタンスを作成し、デプロイツールのマスターサーバーで設定します。
6. アプリケーションを更新する必要がある場合は、デプロイツールを使用してインスタンスに更新をロールアウトします。

プッシュベースのデプロイツール設定 AMI を使用したミュータブルデプロイ

この戦略は、 Managed Services Create EC2 CT の `InstanceUserData`パラメータに依存します。このパラメータの使用の詳細については、「ユーザーデータを使用したインスタンスの設定」を参照してください。この例では、Chef や Puppet などのプルベースのアプリケーションデプロイツールを想定しています。

すべての CT オプションの IDs「変更タイプリファレンス」にあります。

> **Note**
>
> 現在、このソリューションでは Amazon S3 ストレージを使用する必要があります。

ここでは基本的な手順について説明し、手順の詳細については AMS ユーザーガイドを参照してください。

1. Amazon S3 ストレージバケットを作成します。CT: ct-1a68ck03fn98r。S3 バケットではバージョニングが有効になっている必要があります (これを行う方法については、「バケットバージョニングの有効化」を参照してください)。
2. バンドルされた CodeDeploy アーティファクトを配置します。これは、AMS 経由でアクセスを Amazon S3 コンソールで行うことができます。または、このコマンドのバリエーションを使用します。

```
aws s3 cp ZIP_FILEPATH_AND_NAME s3://S3BUCKET_NAME/
```

3. AMS AMI を検索すると、すべてに CodeDeploy が付けられます。「customer-」AMI を検索するには、次のいずれかを使用します。
   - AMS コンソール: 関連する VPC の VPC 詳細ページ
   - AMS API AMS SKMS API リファレンスについては、AWS Artifact コンソールのレポートタブを参照してください。または CLI: `aws amsskms list-amis`
4. EC2 インスタンスを作成します。CT: ct-14027q0sjyt1h; タグ を設定し`Key=backup, Value=true`、 `InstanceUserData`パラメータを使用して、認可キー、SALT スタック (ミニオンのブートストラップ - 詳細については、「Cloud-Init を使用した Linux EC2 でのソルトのブートストラップ」を参照してください)、または Ansible (キーペアのインストール - 詳細については、「Ansible と動的 Amazon EC2 インベントリ管理の開始方法」を参照してください) などのブートストラップとその他のスクリプトを実行します。または、インスタンスへのアクセスをリクエストしてログインし、必要なデプロイアーティファクトで設定します。プッシュベースのコマンドは企業サブネットからインスタンスに送信されるため、踏み台を通過するように認可を設定する必要がある場合があります。踏み台なしでセキュリティグループ/AD グループアクセスをリクエストするには、AMS へのサービスリクエストが必要になる場合があります。
5. ステップ 4 を繰り返して別の EC2 インスタンスを作成し、デプロイツールマスターサーバーで設定します。
6. アプリケーションを更新する必要がある場合は、デプロイツールを使用してインスタンスに更新をロールアウトします。

Golden AMI を使用したイミュータブルなデプロイ

この戦略では、すべてのアプリケーションインスタンスの目的どおりに動作するように設定した「ゴールデン」AMI を使用します。例えば、このゴールデン AMI で作成されたインスタンスは、正しいドメインと DNS に自己参加し、自己設定し、再起動して、必要なすべてのシステムを起動しま

す。アプリケーションインスタンスを更新する場合は、ゴールデン AMI を再作成し、それを使用してまったく新しいアプリケーションインスタンスをロールアウトします。

CodeDeploy エージェントは、すべての AMS AMIs。サポートされている AMIs。

- Amazon Linux (バージョン 1)
- Amazon Linux 2
- RedHat 7
- CentOS 7

すべての CT オプションの IDs「変更タイプリファレンス」に記載されています。

> **Note**
>
> 現在、このソリューションでは Amazon S3 ストレージを使用する必要があります。

1. Amazon S3 ストレージバケットを作成します。CT: ct-1a68ck03fn98r。S3 バケットではバージョニングが有効になっている必要があります (これを行う方法については、「バケットバージョニングの有効化」を参照してください)。
2. バンドルされたアプリケーションアーティファクトを配置します (アプリケーションの起動と動作に必要なすべて)。これは、AMS 経由でアクセスをAmazon S3 コンソールで行うことができます。または、このコマンドのバリエーションを使用します。

   ```
   aws s3 cp ZIP_FILEPATH_AND_NAME s3://S3BUCKET_NAME/
   ```

3. AMS AMI `customer-` を見つけます。次のいずれかを使用します。
   - AMS コンソール: 関連する VPC の VPC 詳細ページ
   - AMS API AMS SKMS API リファレンスについては、AWS Artifact コンソールのレポートタブを参照してください。または CLI: `aws amsskms list-amis`
4. その AMI を使用して EC2 インスタンスを作成します。CT: ct-14027q0sjyt1h。AMS AMI を指定`Key=backup, Value=true`し、タグを設定し、 `customer-mc-ec2-instance-profile`に を指定しますInstanceProfile。返されるインスタンス ID を書き留めます。
5. インスタンスへの管理者アクセスをリクエストします。CT: ct-1dmlg9g1l91h6。アカウントには FQDN が必要です。FQDN が不明な場合は、次の方法で確認できます。

- ディレクトリサービス用の AWS マネジメントコンソールの使用 (セキュリティとアイデンティティの下) ディレクトリ名タブ。
- これらのコマンド (戻りディレクトリクラス、DC+DC+DC=FQDN): Windows: `whoami / fqdn`または Linux: のいずれかを実行します`hostname --fqdn`。

6. インスタンスにログインします。「AMS ユーザーガイド」の「インスタンスへのアクセス」を参照してください。
7. S3 バケットからバンドルされたアプリケーションファイルをインスタンスにダウンロードします。起動時に完全に機能するアプリケーションを自己デプロイするようにインスタンスを設定します。
8. インスタンスにゴールデン AMI を作成します。CT: ct-3rqqu43krekby。詳細については、「AMI | Create」を参照してください。
9. Auto Scaling グループを設定して、その AMI を使用して新しいインスタンスを作成します。CT: ct-2tylseo8rxfsc。アプリケーションを更新する必要がある場合は、この手順に従い、新しいゴールデン AMI を使用するように ASG を更新するように AMS にリクエストします。これには Management | Other | Other | Update CT を使用します。

更新戦略

AMS マネージド環境のアプリケーションまたはインスタンスを更新するために使用できる方法はいくつかあります。

- スケジュールされたダウンタイム: このシンプルな戦略では、アプリケーションをオフラインにして手動で更新する時間をスケジュールします。これを行うには、管理 | その他 | その他 | CT (ct-0xdawir96cy7k) リクエストを送信して、必要なインスタンスを停止します。必要な更新を行い、別の Management | Other | Other | Update CT (ct-0xdawir96cy7k) リクエストを送信してインスタンスを起動します。

- Blue/Green: この戦略では、冗長な環境 (2 つの完全に機能する環境) があり、ドメインネームシステム (DNS) またはウェブファイアウォール (WAF) の更新を使用して 1 つの環境をオフラインにしてトラフィックをリダイレクトする必要があります。1 つの環境を更新し、再度リダイレクトして他の環境を更新します。

  詳細については、AWS CodeDeploy がブルー/グリーンデプロイを導入する」を参照してください。

- 新しい AMI を使用したローリング更新: ここでは、新しい AMI をカスタマイズし (「AMI の作成」を参照)、AMS に Auto Scaling グループにデプロイするようにリクエストします。これを行うには、管理 | その他 | その他 | CT の更新 (ct-0xdawir96cy7k) を使用します。

AWS Managed Services リソーススケジューラ

AWS Managed Services (AMS) リソーススケジューラを使用して、アカウント内の AutoScaling グループ、Amazon EC2 インスタンス、および RDS インスタンスの自動起動と停止をスケジュールします。これにより、リソースが 24 時間 365 日稼働することを意図していないインフラストラクチャコストを削減できます。このソリューションは、 上の Instance Scheduler AWS 上に構築されていますが、AMS のニーズに固有の追加機能とカスタマイズが含まれています。

> **Note**
>
> デフォルトでは、AMS Resource Scheduler は AWS CloudFormation スタックに含まれていないリソースとやり取りしません。リソースは、「スタック」、「sc-」、または「SC-」で始まるスタックの一部である必要があります。CloudFormation スタックの一部ではないリソースをスケジュールするには、Resource Scheduler スタックパラメータを `ScheduleNonStackResources`に更新しますYes。

AMS Resource Scheduler は期間とスケジュールを使用します。

- 期間は、開始時刻、終了時刻、曜日など、Resource Scheduler が実行される時間を定義します。
- スケジュールには、定義された期間と、SSM メンテナンスウィンドウ、タイムゾーン、休止設定などの追加設定が含まれており、設定された期間ルールに基づいてリソースを実行するタイミングを指定します。

これらの期間とスケジュールは、AMS Resource Scheduler の自動変更タイプ (CTs) を使用して設定できます。

AMS Resource Scheduler で使用できる設定の詳細については、「ソリューションコンポーネント」の対応する AWS Instance Scheduler ドキュメントを参照してください。ソリューションのアーキテクチャビューについては、「アーキテクチャの概要.html」の対応する AWS Instance Scheduler ドキュメントを参照してください。

AMS リソーススケジューラのデプロイ

AMS リソーススケジューラをデプロイするには、自動変更タイプ (CT) : デプロイ | AMS リソーススケジューラ | ソリューション | デプロイ (ct-0ywnhc8e5k9z5) を使用して、ソリューションをアカウントにデプロイする RFC を生成します。RFC が実行されると、デフォルト設定の AMS Resource Scheduler リソースを含む CloudFormation スタックがアカウントに自動的にプロビジョニングされます。Resource Scheduler の変更タイプの詳細については、「AMS Resource Scheduler」を参照してください。

> **Note**
>
> AMS Resource Scheduler がアカウントに既にデプロイされているかどうかを確認するには、そのアカウントの AWS Lambda コンソールをチェックし、AMSResourceScheduler 関数を探します。

AMS リソーススケジューラがアカウントでプロビジョニングされたら、デフォルト設定を確認し、必要に応じてタグキー、タイムゾーン、スケジュールされたサービスなどの設定を好みに応じてカスタマイズすることをお勧めします。推奨されるカスタマイズの詳細については、次のAMS リソーススケジューラのカスタマイズ「」を参照してください。

カスタム設定を行うか、Resource Scheduler の設定を確認するだけです。

AMS リソーススケジューラのカスタマイズ

AMS Resource Scheduler の更新変更タイプを使用して、AMS Resource Scheduler の次のプロパティをカスタマイズすることをお勧めします。「AMS Resource Scheduler」を参照してください。

- タグ名: Resource Scheduler がインスタンススケジュールをリソースに関連付けるために使用されるタグの名前。デフォルト値は Schedule です。
- スケジュールされたサービス: Resource Scheduler が管理できるサービスのカンマ区切りリスト。デフォルト値は「ec2,rds,autoscaling」です。有効な値は「ec2」、「rds」、「autoscaling」です
- デフォルトのタイムゾーン: Resource Scheduler で使用するデフォルトのタイムゾーンを指定します。デフォルト値は UTC です。
- CMK を使用する: Resource Scheduler にアクセス許可を付与できる Amazon KMS カスタマーマネージドキー (CMK) ARNs のカンマ区切りリスト。
- LicenseManager を使用する: その Resource Scheduler AWS に対する Licence Manager ARNs のカンマ区切りリストには、アクセス許可を付与できます。

> **Note**
>
> AMS は、アカウントで AMS Resource Scheduler を最新の状態に保つために、機能や修正をリリースすることがあります。この場合、AMS Resource Scheduler に対して行ったカスタマイズは保持されます。

AMS リソーススケジューラの使用

ソリューションのデプロイ後に AMS Resource Scheduler を設定するには、自動化された Resource Scheduler CTs を使用して、AMS Resource Scheduler の期間 (Resource Scheduler が実行される時間) とスケジュール (設定された期間およびその他のオプション) を作成、削除、更新、記述 (詳細を取得) します。AMS Resource Scheduler の変更タイプを使用する例については、「AMS Resource Scheduler」を参照してください。

AMS Resource Scheduler によって管理されるリソースを選択するには、デプロイとスケジュールの作成後に、AMS Tag Create CTsを使用して、デプロイ時に指定したタグキーと定義されたスケジュールをタグ値として Auto Scaling グループ、Amazon RDS スタック、Amazon EC2 リソースにタグ付けします。リソースにタグを付けると、リソースは定義された Resource Scheduler スケジュールに従って開始または停止するようにスケジュールされます。

AMS Resource Scheduler の使用には追加料金はかかりません。ただし、このソリューションでは複数の が使用され AWS のサービス ており、これらのリソースの使用時に課金されます。詳細については、「アーキテクチャの概要」を参照してください。

AMS Resource Scheduler をオプトアウトするには :

- 一時的なオプトアウトまたは無効化の場合: 自動管理 | AMS リーススケジューラ | 状態 | 変更タイプを無効にする (ct-14v49adibs4db) を使用して RFC を送信する
- 永続的な削除の場合: 管理を送信する | その他 | その他 | 更新 (レビューが必要) (ct-0xdawir96cy7k) RFC が Resource Scheduler リリースオートメーションシステムからの削除をリクエストする

AMS Resource Scheduler のコスト見積りツール

コスト削減を追跡するために、AMS Resource Scheduler には、スケジューラによって管理される Amazon EC2 および RDS リソースの推定コスト削減を時間単位で計算するコンポーネントがあります。このコスト削減データは、追跡に役立つ CloudWatch メトリクス (`AMS/ResourceScheduler`)

として公開されます。コスト削減推定器は、インスタンスの実行時間の節約のみを推定します。リソースに関連するデータ転送コストなど、他のコストは考慮されません。

Resource Scheduler では、コスト削減推定器が有効になっています。1 時間ごとに実行され、 からコストと使用状況データを取得します AWS Cost Explorer。そのデータから、各インスタンスタイプの 1 時間あたりの平均コストを計算し、スケジュールなしで実行されていた場合は 1 日のコストを予測します。コスト削減は、特定の日における Cost Explorer からの予測コストと実際に報告されたコストの差です。

たとえば、インスタンス A が Resource Scheduler で午前 9 時から午後 5 時まで実行されるように設定されている場合、特定の日に 8 時間になります。Cost Explorer はコストを 1 USD、使用量を 8 と報告します。したがって、1 時間あたりの平均コストは 0.125 USD です。インスタンスが Resource Scheduler でスケジュールされていない場合、インスタンスはその日に 24 時間実行されます。この場合、コストは 24x0.125 = 3 USD になります。Resource Scheduler は、2 USD のコスト削減を実現しました。

Cost Explorer から Resource Scheduler によって管理されるリソースのコストと使用状況のみを取得するために、Resource Scheduler がリソースをターゲットにするために使用するタグキーを Billing Dashboard のコスト配分タグとしてアクティブ化する必要があります。アカウントが組織に属している場合、タグキーは組織の管理アカウントでアクティブ化する必要があります。これを行う方法については、「ユーザー定義のコスト配分タグとユーザー定義のコスト配分タグのアクティブ化」を参照してください。 https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/custom-tags.html

タグキーがコスト配分タグとしてアクティブ化されると、 AWS 請求は Resource Scheduler によって管理されるリソースのコストと使用状況の追跡を開始し、そのデータが利用可能になると、コスト削減推定器はコスト削減の計算を開始し、CloudWatch のAMS/ResourceSchedulerメトリクス名前空間にデータを発行します。

コスト見積もりのヒント

Cost Savings Estimator は、リザーブドインスタンスや Savings Plans などの割引を計算で考慮に入れません。推定器は Cost Explorer から使用コストを受け取り、リソースの 1 時間あたりの平均コストを計算します。詳細については、AWS「コストデータセットを理解する: チートシート」を参照してください。

Cost Explorer から Resource Scheduler によって管理されるリソースのコストと使用状況のみを取得するために、Resource Scheduler がリソースをターゲットにするために使用するタグキーを Billing Dashboard の Cost Allocation タグとしてアクティブ化する必要があります。アカウントが組織に属している場合、タグキーは組織の管理アカウントでアクティブ化する必要があります。これを行う

方法については、「ユーザー定義のコスト配分タグ」を参照してください。コスト配分タグが有効になっていない場合、推定器は、有効になっていても削減額を計算してメトリクスを発行することはできません。

AMS Resource Scheduler のベストプラクティス

Amazon EC2 インスタンスのスケジューリング

- インスタンスのシャットダウン動作は に設定しstop、 に設定しないでくださいterminate。これは、AMS Amazon EC2 自動変更タイプの作成 (ct-14027q0sjyt1h) で作成されたインスタンスstopの場合は に事前設定されており、 AWS CloudFormation InstanceInitiatedShutdownBehaviorプロパティを に設定することで、取り込みで作成された Amazon EC2 インスタンスに設定できますstop。インスタンスのシャットダウン動作が に設定されている場合terminate、Resource Scheduler が停止し、スケジューラが起動できなくなるとインスタンスは終了します。
- Auto Scaling グループの一部である Amazon EC2 インスタンスは、タグ付けされていても、AMS Resource Scheduler によって個別に処理されません。
- ターゲットインスタンスのルートボリュームが KMS カスタマーマスターキー (CMK) で暗号化されている場合は、スケジューラがそのようなインスタンスを開始できるように、Resource Scheduler IAM ロールに追加のkms:CreateGrantアクセス許可を追加する必要があります。このアクセス許可は、セキュリティを向上させるためにデフォルトではロールに追加されません。このアクセス許可が必要な場合は、 Management | AMS Resource Scheduler | Solution | Update change type を使用して RFC を送信し、KMS CMKs の ARNs のカンマ区切りリストを指定します。

Auto Scaling グループのスケジューリング

- AMS Resource Scheduler は、グループ内の個々のインスタンスではなく、Auto Scaling グループの自動スケーリングを開始または停止します。つまり、スケジューラは Auto Scaling グループのサイズを復元 (開始) するか、サイズを 0 (停止) に設定します。
- AutoScaling グループに、グループ内のインスタンスではなく、指定されたタグをタグ付けします。
- 停止中、AMS Resource Scheduler は Auto Scaling グループの最小容量、希望容量、最大容量の値を保存し、最小容量と希望容量を 0 に設定します。起動時に、スケジューラは Auto Scaling グループサイズを停止時と同じように復元します。したがって、Auto Scaling グループインスタンスは、インスタンスの終了と再起動が Auto Scaling グループで実行されているアプリケーションに影響を与えないように、適切な容量設定を使用する必要があります。

- 実行中に Auto Scaling グループが変更された場合 (最小容量または最大容量)、スケジューラは新しい Auto Scaling グループサイズを保存し、停止スケジュールの最後にグループを復元するときに使用します。

Amazon RDS インスタンスのスケジュール

- スケジューラは、RDS インスタンスを停止する前にスナップショットを作成できます (Aurora DB クラスターには適用されません)。この機能はデフォルトで有効になっており、RDS インスタンススナップショットの作成 CloudFormation テンプレートパラメータは true に設定されています。スナップショットは、次回 Amazon RDS インスタンスが停止して新しいスナップショットが作成されるまで保持されます。

  スケジューラは、クラスターまたは Amazon RDS Aurora データベースの一部である Amazon RDS インスタンス、またはマルチアベイラビリティーゾーン (マルチ AZ) 設定で起動/停止できます。ただし、スケジューラが Amazon RDS インスタンス、特にマルチ AZ インスタンスを停止できない場合は、Amazon RDS の制限を確認してください。Aurora クラスターの開始または停止をスケジュールするには、Schedule Aurora Clusters テンプレートパラメータを使用します (デフォルトは true)。Aurora クラスター (クラスター内の個々のインスタンスではなく) には、初期設定時に定義されたタグキーと、そのクラスターをスケジュールするためのタグ値としてスケジュール名をタグ付けする必要があります。

  すべての Amazon RDS インスタンスには、システムの変更が適用される毎週のメンテナンスウィンドウがあります。メンテナンスウィンドウ中、Amazon RDS は 7 日以上停止したインスタンスを自動的に起動し、メンテナンスを適用します。メンテナンスイベントが完了すると、Amazon RDS はインスタンスを停止しないことに注意してください。

  スケジューラでは、Amazon RDS インスタンスの優先メンテナンスウィンドウをスケジュールの実行期間として追加するかどうかを指定できます。ソリューションでは、メンテナンスウィンドウの最初にインスタンスを起動しますが、他の実行期間でインスタンスを実行するように指定されておらず、メンテナンスイベントが完了した場合はメンテナンスウィンドウの終了時にインスタンスを停止します。

  メンテナンスウィンドウの終了までにメンテナンスイベントが完了しなかった場合、インスタンスはメンテナンスイベントが完了した後のスケジュール間隔まで実行されます。

> **Note**
>
> スケジューラは、リソースの開始または停止を検証しません。API コールを行い、次に進みます。API コールが失敗すると、調査のために エラーがログに記録されます。

アプリケーションセキュリティに関する考慮事項

アプリケーションセキュリティには、アプリケーションが実行する必要があるアクセス許可、ファイアウォールルール、アプリケーションへのアクセスを有効にする IAM ロールの検討が含まれます。

一般的な AWS セキュリティの詳細については、「セキュリティ、アイデンティティ、コンプライアンスのベストプラクティス」を参照してください。

設定管理のためのアクセス

AWS Managed Services (AMS) は、セキュリティの問題、パッチ適用の問題、バックアップの問題などについて心配する必要がないように、問題のないインフラストラクチャの提供を目指しています。そのために、AMS では、特定のグループまたはマスターサーバーのみを許可する最小限の IAM ロールを推奨しています。アプリケーションデプロイツールを使用している場合は、アプリケーションを実行しているインスタンスにアクセスします。

アプリケーションアクセスファイアウォールルール

オペレーティングシステム (OS) と同様に、すべてのアプリケーションアクセスは Active Directory (AD) グループを使用して管理する必要があります。Amazon Relational Database Service (Amazon RDS) を例として使用する場合、ミラー (レプリケーション) を破って新しいユーザーを追加する必要があります。最善の方法は、AD でグループを作成し、データベースの作成時にグループを追加することです。AMS AD にグループを含めると、アプリケーションアクセス用の CTsを作成できます。AD の公式グループ戦略の詳細については、「グループネスト戦略の使用 – グループ戦略の AD ベストプラクティス」を参照してください。

ドメインツリーと親/子ドメインの詳細については、「ドメインとフォレストの仕組み」を参照してください。

次のルールは、子ドメインにあるユーザーとのマルチドメインフォレストの信頼に適したソリューションを示しています。

Windows インスタンス

これらは、Windows の親ドメインコントローラーと子ドメインコントローラー用に設定するためのルールです。

親ドメインコントローラー、Windows

FROM: 親ドメインコントローラー TO: Windows スタックと共有サービスサブネット

ソースポート	発信先 ポート	プロトコル
88	49152 - 65535	TCP
389	49152 - 65535	UDP

FROM: 共有サービスを含むスタックサブネット TO: Windows フォレストルートドメインコントローラー

ソースポート	発信先 ポート	プロトコル
49152 - 65535	88	TCP
49152 - 65535	389	UDP

子ドメインコントローラー、Windows

FROM: 子ドメインコントローラー TO: Windows AWS ドメインコントローラー

ソースポート	発信先 ポート	プロトコル
49152 - 65535	53	TCP
49152 - 65535	88	TCP
49152 - 65535	389	UDP

FROM: 子ドメインコントローラー TO: Windows スタックと共有サービスサブネット

ソースポート	発信先 ポート	プロトコル
88	49152 - 65535	TCP
135	49152 - 65535	TCP

ソースポート	発信先 ポート	プロトコル
389	49152 - 65535	TCP
389	49152 - 65535	UDP
445	49152 - 65535	TCP
49152 - 65535	49152 - 65535	TCP

FROM: 共有サービスを含むスタックサブネット TO: Windows 子ドメインコントローラー

ソースポート	発信先 ポート	プロトコル
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 インスタンス

これらは、Linux の親ドメインコントローラーと子ドメインコントローラー用に設定するためのルールです。

すべてのテストは Amazon Linux を使用して実行されました。Windows の動的ポート範囲は 49152 ~ 65535 ですが、多くの Linux カーネルはポート範囲 32768 ~ 61000 を使用します。次のコマンドを実行して、IP ポート範囲を表示します。

```
cat /proc/sys/net/ipv4/ip_local_port_range
```

親ドメインコントローラー、Linux

FROM: 親ドメインコントローラー TO: Linux スタックと共有サービスサブネット

ソースポート	発信先 ポート	プロトコル
389	32768 ~ 61000	UDP
88	32768 ~ 61000	TCP

FROM: 共有サービスを含むスタックサブネット TO: Linux フォレストルートドメインコントローラー

ソースポート	発信先 ポート	プロトコル
32768 ~ 61000	88	TCP
32768 ~ 61000	389	UDP

子ドメインコントローラー、Linux

FROM: 子ドメインコントローラー TO: Linux AWS ドメインコントローラー

ソースポート	発信先 ポート	プロトコル
49152 - 65535	53	TCP
49152 - 65535	88	TCP
389	49152 - 65535	UDP
49152 - 65535	389	UDP

FROM: 子ドメインコントローラー TO: Linux スタックと共有サービスサブネット

ソースポート	発信先 ポート	プロトコル
88	32768 ~ 61000	TCP

ソースポート	発信先 ポート	プロトコル
389	32768 ~ 61000	UDP

FROM: 共有サービスを含むスタックサブネット TO: Linux 子ドメインコントローラー

ソースポート	発信先 ポート	プロトコル
32768 ~ 61000	88	TCP
32768 ~ 61000	389	UDP

AMS 出力トラフィック管理

デフォルトでは、AMS プライベートサブネットとカスタマーアプリケーションサブネットの送信先 CIDR が 0.0.0.0/0 のルートには、ターゲットとしてネットワークアドレス変換 (NAT) ゲートウェイがあります。AMS サービスである TrendMicro とパッチ適用は、AMS がそのサービスを提供し、TrendMicro とオペレーティングシステムが更新を取得できるように、インターネットへの出力アクセス権を持つ必要があるコンポーネントです。

AMS は、次の条件を満たす限り、カスタマー管理の Egress デバイスを介して Egress トラフィックをインターネットに転送することをサポートしています。

- 暗黙的な (透過的な) プロキシとして機能します。

  and

- AMS マネージドインフラストラクチャの継続的なパッチ適用とメンテナンスを可能にするために、AMS HTTP と HTTPS の依存関係 (このセクションに記載) を許可します。

いくつか例を挙げます。

- トランジットゲートウェイ (TGW) には、マルチアカウントランディングゾーンネットワークアカウントの AWS Direct Connect 接続を介したカスタマー管理のオンプレミスファイアウォールを指すデフォルトルートがあります。
- TGW には、AWS PrivateLink を利用するマルチアカウントランディングゾーン出力 VPC の AWS エンドポイントを指すデフォルトのルートがあり、別の AWS アカウントのカスタマーマネージドプロキシを指します。

- TGW には、別の AWS アカウントのカスタマーマネージドファイアウォールを指すデフォルトルートがあり、site-to-site接続をマルチアカウントランディングゾーン TGW へのアタッチメントとして使用します。

AMS は、対応する AMS HTTP と HTTPS の依存関係を特定し、これらの依存関係を継続的に開発および改良しています。「egressMgmt.zip」を参照してください。JSON ファイルに加えて、ZIP には README が含まれています。

> **Note**
>
> - この情報は包括的ではありません。必須の外部サイトの一部はここには記載されていません。
> - このリストを拒否リストまたはブロック戦略で使用しないでください。
> - このリストは、出力フィルタリングルールセットの開始点として意図されており、実際のトラフィックがリストのどこから逸脱するかを正確に判断するためにレポートツールが使用されることを期待しています。

送信トラフィックのフィルタリングに関する情報を要求するには、CSDM に E メールを送信します: ams-csdm@amazon.com。

セキュリティグループ

AWS VPCs では、AWS セキュリティグループは仮想ファイアウォールとして機能し、1 つ以上のスタック (インスタンスまたは一連のインスタンス) のトラフィックを制御します。スタックを起動すると、スタックは 1 つ以上のセキュリティグループに関連付けられ、スタックに到達できるトラフィックが決まります。

- パブリックサブネット内のスタックの場合、デフォルトのセキュリティグループはすべての場所 (インターネット) からの HTTP (80) および HTTPS (443) からのトラフィックを受け入れます。スタックは、企業ネットワークからの内部 SSH および RDP トラフィック、および AWS 踏み台も受け入れます。これらのスタックは、任意のポートを介してインターネットに出力できます。また、プライベートサブネットやパブリックサブネット内の他のスタックに出力することもできます。
- プライベートサブネット内のスタックは、プライベートサブネット内の他のスタックに出力でき、スタック内のインスタンスは、任意のプロトコルを介して相互に完全に通信できます。

> **Important**
>
> プライベートサブネット上のスタックのデフォルトのセキュリティグループでは、プライベートサブネット内のすべてのスタックが、そのプライベートサブネット内の他のスタックと通信できます。プライベートサブネット内のスタック間の通信を制限する場合は、制限を記述する新しいセキュリティグループを作成する必要があります。たとえば、そのプライベートサブネット内のスタックが特定のポート経由で特定のアプリケーションサーバーからのみ通信できるようにデータベースサーバーへの通信を制限する場合は、特別なセキュリティグループをリクエストします。これを行う方法については、このセクションで説明します。

デフォルトのセキュリティグループ

MALZ

次の表は、スタックのデフォルトのインバウンドセキュリティグループ (SG) 設定を示しています。SG はSentinelDefaultSecurityGroupPrivateOnly-vpc-ID」という名前で、*ID* は AMS マルチアカウントランディングゾーンアカウントの VPC ID です。すべてのトラフィックは、このセキュリティグループ経由でmc-initial-garden-SentinelDefaultSecurityGroupPrivateOnly」へのアウトバウンドが許可されます (スタックサブネット内のすべてのローカルトラフィックが許可されます) 。

すべてのトラフィックは、2 番目のセキュリティグループ SentinelDefaultSecurityGroupPrivateOnly」によって 0.0.0.0/0 へのアウトバウンドが許可されます。

> **Tip**
>
> EC2 create、OpenSearch create domain などの AMS 変更タイプのセキュリティグループを選択する場合は、ここで説明するデフォルトのセキュリティグループのいずれか、または作成したセキュリティグループを使用します。VPC あたりのセキュリティグループのリストは、AWS EC2 コンソールまたは VPC コンソールで確認できます。

内部 AMS 目的で使用される追加のデフォルトのセキュリティグループがあります。

AMS デフォルトセキュリティグループ (インバウンドトラフィック)

タイプ	プロトコル	ポート範囲	ソース
すべてのトラフィック	すべて	すべて	SentinelDefaultSecurityGroupPrivateOnly (同じセキュリティグループのメンバーへのアウトバウンドトラフィックを制限)
すべてのトラフィック	すべて	すべて	SentinelDefaultSecurityGroupPrivateOnlyEgressAll (アウトバウンドトラフィックを制限しません)
HTTP、HS、SSH、	TCP	80 / 443 (ソース 0.0.0.0/0) 踏み台からの SSH および RDP アクセスが許可されている	SentinelDefaultSecurityGroupPublic (アウトバウンドトラフィックを制限しません)
MALZ 踏み台：			
SSH	TCP	22	SharedServices VPC CIDR と DMZ VPC CIDR、およびお客様が用意したオンプレミス CIDRs
SSH	TCP	22	
RDP	TCP	3389	
RDP	TCP	3389	
SALZ 踏み台：			
SSH	TCP	22	mc-initial-garden-LinuxBastionSG
SSH	TCP	22	mc-initial-garden-LinuxBastionDMZSG
RDP	TCP	3389	mc-initial-garden-WindowsBastionSG

タイプ	プロトコル	ポート範囲	ソース
RDP	TCP	3389	mc-initial-garden-WindowsBastionDMZSG

SALZ

次の表は、スタックのデフォルトのインバウンドセキュリティグループ (SG) 設定を示しています。SG の名前はmc-initial-garden-SentinelDefaultSecurityGroupPrivateOnly-*ID*」で、*ID* は一意の識別子です。すべてのトラフィックは、このセキュリティグループ経由でmc-initial-garden-SentinelDefaultSecurityGroupPrivateOnly」へのアウトバウンドが許可されます (スタックサブネット内のすべてのローカルトラフィックが許可されます)。

すべてのトラフィックは、2 番目のセキュリティグループmc-initial-garden-SentinelDefaultSecurityGroupPrivateOnlyEgressAll-0 へのアウトバウンドが許可されます。

> **Tip**
>
> EC2 create、OpenSearch create domain などの AMS 変更タイプのセキュリティグループを選択する場合は、ここで説明するデフォルトのセキュリティグループのいずれか、または作成したセキュリティグループを使用します。VPC あたりのセキュリティグループのリストは、AWS EC2 コンソールまたは VPC コンソールで確認できます。

内部 AMS 目的で使用される追加のデフォルトのセキュリティグループがあります。

AMS デフォルトセキュリティグループ (インバウンドトラフィック)

タイプ	プロトコル	ポート範囲	ソース
すべてのトラフィック	すべて	すべて	SentinelDefaultSecurityGroupPrivateOnly (同じセキュリティグループのメンバーへのアウトバウンドトラフィックを制限)
すべてのトラ	すべて	すべて	SentinelDefaultSecurityGroupPrivateOnlyEgressAll (アウトバウンドトラフィックを制限しません)

タイプ	プロトコル	ポート範囲	ソース
フィック			
HTTP、HS、SSH、	TCP	80 / 443 (ソース 0.0.0.0/0) 踏み台からの SSH および RDP アクセスが許可されている	SentinelDefaultSecurityGroupPublic (アウトバウンドトラフィックを制限しません)
MALZ 踏み台：			
SSH	TCP	22	SharedServices VPC CIDR と DMZ VPC CIDR、およびお客様が用意したオンプレミス CIDRs
SSH	TCP	22	
RDP	TCP	3389	
RDP	TCP	3389	
SALZ 踏み台：			
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

セキュリティグループの作成、変更、または削除

カスタムセキュリティグループをリクエストできます。デフォルトのセキュリティグループがアプリケーションまたは組織のニーズを満たさない場合は、新しいセキュリティグループを変更または作成できます。このようなリクエストは承認が必要と見なされ、AMS オペレーションチームによってレビューされます。

スタックと VPCs の外部でセキュリティグループを作成するには、`Deployment | Advanced stack components | Security group | Create (review required)`変更タイプ (ct-1oxx2g2d7hc90) を使用して RFC を送信します。

Active Directory (AD) セキュリティグループの変更には、次の変更タイプを使用します。

- ユーザーを追加するには: 管理を使用して RFC を送信する | Directory Service | ユーザーとグループ | ユーザーをグループに追加する [ct-24pi85mjtza8k]
- ユーザーを削除するには: Management | Directory Service | Users and groups | Remove user from group [ct-2019s9y3nfml4] を使用して RFC を送信します。

> **Note**
>
> 「レビュー必須」CT を使用する場合、AMS では ASAP スケジューリングオプション (コンソールで ASAP を選択し、API/CLI で開始時刻と終了時刻を空白のままにしておく) を使用することをお勧めします。これらの CTs では、AMS オペレーターが RFC を調べる必要があり、承認して実行する前にお客様と通信する必要があるためです。これらの RFCsスケジュールする場合は、少なくとも 24 時間かかります。スケジュールされた開始時刻より前に承認が行われない場合、RFC は自動的に拒否されます。

セキュリティグループの検索

スタックまたはインスタンスにアタッチされているセキュリティグループを検索するには、EC2 コンソールを使用します。スタックまたはインスタンスを見つけたら、そのスタックまたはインスタンスにアタッチされているすべてのセキュリティグループを確認できます。

コマンドラインでセキュリティグループを検索し、出力をフィルタリングする方法については、「」を参照してくださいdescribe-security-groups。

付録: アプリケーションオンボーディングアンケート

このアンケートを使用してデプロイ要素と構造を記述し、AMS が必要なインフラストラクチャコンポーネントを特定できるようにします。Line-of-Business (LoB) アプリケーションのオンボーディング要件は製品アプリケーションと大きく異なるため、このアンケートは両方に対応するように設計されています。

トピック

デプロイの概要

デプロイの説明。例:

- このアカウントは、 (製品アプリケーションのデプロイではなく) Line-of-Business (LoB) アプリケーションのデプロイ用です。
- デプロイには、アカウントのパブリック/DMZ サブネット内の自動スケーリングされた ARP (認証されたリバースプロキシ) が含まれます。
- ウェブサーバーとアプリケーションサーバーは、アカウントのプライベートサブネット内にデプロイされます。
- Amazon RDS (Amazon Relational Database Service) インスタンスも、アカウントのプライベートサブネット内にデプロイされます。
- サーバー (ARP、ウェブ、アプリケーション、データベース、ロードバランサーなど) は、個別のセキュリティグループに分割されます。
- アカウントには、アベイラビリティーゾーン (AZs) にまたがる HA (高可用性) 設計、つまりマルチ AZ が必要です。

インフラストラクチャデプロイコンポーネント

アプリケーションをサポートするために を設定する必要があるさまざまなコンポーネントは何ですか？

- リージョン: どの AWS リージョン またはリージョンが必要ですか？
- 高可用性 (HA): どのアベイラビリティーゾーンが使用されますか？
- Virtual Private Cloud (VPC): VPC の CIDR ブロックとは何ですか？
- どのサーバーインスタンスが必要ですか？
  - 認証済みリバースプロキシ (ARP): OS、AMI、インスタンスタイプ、サブネット ID、セキュリティグループ、イングレスポート？
  - Application Deployment Tool サーバー: OS、AMI、インスタンスタイプ、サブネット ID、セキュリティグループ、イングレスポート (Chef、Puppet) またはエグレスポート (Ansible、Saltstack) ポート？
  - Amazon RDS with MySQL: DB バージョン、使用タイプ、インスタンスクラス、サブネット ID、セキュリティグループ、DB インスタンス ID、ストレージサイズ、マルチ AZ、認証タイプ、暗号化？
  - ストレージ: アプリはステートレスですか？ S3 バケットが必要ですか？ 永続的ストレージが必要ですか？ EBS ボリュームに保管時のデータの暗号化が必要ですか？ DB 暗号化が必要ですか？
  - 外部 (マネージドサービス VPC 向け) サーバーエンドポイント: SMTP? LDAP とは？
  - ネットワーク要件: ネットワークフィルタリング (セキュリティグループに基づくか?) ウェブトラフィック検査 (インバウンド？アウトバウンド?)
- タグ付け: リソースを論理コレクションにグループ化するには、どのタグを使用する必要がありますか？ たとえば、アプリケーションスタックのすべてのリソースです。バックアップを有効にするなど`backup=true`、ユースケースのタグを選択します。さらに、作成した EC2 `name=value` インスタンスがコンソールに名前を表示するには、 タグを使用する必要があります。
- セキュリティグループ：
  - どのようなセキュリティグループが必要ですか？
  - セキュリティグループの進入ルール
  - セキュリティグループの出力ルール

アプリケーションホスティングプラットフォーム

アプリケーションホスティングプラットフォームでは、以下の可能な要件を考慮してください。

- データベースは暗号化されていますか？
- 暗号化キーは誰によって管理されますか？
- 転送中および保管時のすべてのデータは暗号化されていますか？
- HTTPS 経由でシステムにアクセスするすべてのユーザー
- セキュリティオペレーションチームによって承認されたsystem-to-systemインタラクション

アプリケーションデプロイモデル

アプリケーションのデプロイを計画する方法に関する考慮事項。「運用モデルとは」を参照してください。

- 自動か手動か デプロイの自動化がないということは、Auto Scale がないことを意味します。アクセスをリクエストしてログインし、アプリケーションを手動で更新すると、更新は失敗します。AMS では、お客様を支援するために、更新をロールバックするか、サービスリクエストを通じてアラートを受け取ることを期待しています。
- 自動の場合、フレームワークとは何ですか？ スクリプト？ エージェントベース (puppet/chef) エージェントレス (SALT/Ansible) CodeDeploy？エージェントベースのデプロイツールとエージェントレスデプロイツールでは、ツールのマスターサーバーとして個別のインスタンスを作成してデプロイする必要があります。AMS では、アプリケーションデプロイツールを成功させるために必要なすべての要素について把握しておくことを期待していますが、関連するインフラストラクチャに関する質問にご協力いただけます。
- Line-of-Business アプリケーション (アプリケーションの作成と管理に使用するアプリケーション) にはパッチが必要ですか？

アプリケーション依存関係

Line-of-Business (LoB) アプリケーションのインスタンスが必要ですか？ 製品アプリケーションの場合

製品アプリケーションを適切に機能させるには、何が必要ですか？

- ネットワークレベルの依存関係: 例： Direct Connect

- パッケージの依存関係: 例: `pip`
- このアプリケーションが依存するアプリケーション: MySql など
- ファイアウォールの依存関係

LoB アプリケーションを適切に機能させるには、何が必要ですか？

- ネットワークレベルの依存関係: 例： Direct Connect
- パッケージの依存関係: Firefox ソースなど
- このアプリケーションが依存するアプリケーション: MySql など
- ファイアウォールの依存関係

製品アプリケーションの SSL 証明書

アプリケーション (LoB と製品) が実行とアクセスに必要なすべてに到達できるように、サーバーに必要な SSL 証明書は何ですか？

- Auto Scaling グループ
- データベース (Amazon RDS)
- Load Balancer?
- デプロイツールサーバー？
- ウェブアプリケーションファイアウォール (AWS WAF)
- その他のインスタンス

例えば、上記の各インスタンスには、次の証明書が必要になる場合があります。

WAF (証明書 1) - > ELB-Ext (証明書 2) - > ARP (証明書 3) - > ELB-Int (証明書 4) -> ウェブサイト (証明書 5) - > ELB-Int (証明書 6) -> ウェブサービス (証明書 7)。

ドキュメント履歴

次の表に、この AMS リリースのドキュメントを示します。

- API バージョン： 2019-05-21
- 最新のドキュメント更新: 2023 年 2 月 16 日

変更	説明	リンク
TOC リンクの削除	TOC AWS 用語集リンクを削除しました。	2025 年 8 月 8 日
更新された内容: ワークロードの移行: Windows の取り込み前検証	WIGs 前の検証スクリプトを使用して、Windows インスタンスが AMS アカウントに取り込む準備ができていることを検証するための詳細な手順を追加しました。	ワークロードの移行: Windows の取り込み前検証
更新されたコンテンツ、DMS 設定	必要なロール dms-vpc-role に関する重要な注意事項。	1: AWS DMS レプリケーションサブネットグループ: 作成
更新されたコンテンツ、CFN Ingest でサポートされているリソース	OpenSearch を追加しました。	サポートされているリソース
更新されたコンテンツ、ワークロードの移行	取り込み前検証の手順を更新しました。	ワークロードの移行: Windows の取り込み前検証
コンテンツ「CFN Ingest」を更新しました。	CFN 取り込みコンテンツから制限された「サポートされているリソース」を削除しました。	CloudFormation Ingest スタック: サ

変更	説明	リンク
		ポートされているリソース
サポートされている Windows バージョンの更新	Windows Server 2022 のサポートが追加されました。	AMS Amazon マシンイメージ (AMIs)、ワークロードの移行: Linux と Windows の前提条件、および ワークロードの移行: Windows の取り込み前検証
更新されたコンテンツ、Resource Scheduler。	SALZ と MALZ の両方に適用される専用デプロイ CT、ct-0ywnhc8e5k9z5 を使用するように手順を更新しました。	AMS Resource Scheduler クイックスタート
コンテンツ「ワークロード取り込み」を更新しました。	サポートされている SUSE Linux バージョンを更新しました。	ワークロードの移行: Linux と Windows の前提条件
コンテンツ「 Database Migration Service」を更新しました。	前提条件に を追加し、有用性と使いやすさのためにいくつかの変更を加えました。	AWS Database Migration Service (AWS DMS)

変更	説明	リンク
コンテンツ「ワークロード取り込み」を更新しました。	Linux Pre-WIGS Validation Zip が更新されました。	ワークロードの移行: Linux と Windows の前提条件
コンテンツを更新しました。	Linux の WIGS 検証前 zip を更新しました。また、サポートされているオペレーティングシステムとして Windows Server 2008 R2 を追加しました。	ワークロードの移行: Linux と Windows の前提条件
新しいコンテンツ	クイックスタートとチュートリアルは、廃止された AMS Advanced Change Management Guide からここに移動されました。	クイックスタート, チュートリアル.
更新された内容	デプロイ | 高度なスタックコンポーネント | Database Migration Service (DMS) | レプリケーションタスクの開始 (ct-1yq7hhqse71yg) DocumentName と Region が必須パラメータであることを示すように更新されました。以前は、オプションとして誤ってリストされていました。	Database Migration Service (DMS) | レプリケーションタスクの開始
更新された内容	CloudFormation の取り込み AWS::Route53Resolver::ResolverRuleAssociation と AWS::Route53Resolver::ResolverRule の 2 つの新しいサポートされているリソースを示すように更新されました。	サポートされているリソース

変更	説明	リンク
更新された内容	ワークロードの移行: Windows の取り込み前検証	Sysprep 情報をより詳細に更新しました。 ワークロードの移行: Windows の取り込み前検証
更新された内容	管理 | カスタムスタック | CloudFormation テンプレートからのスタック | 変更セットと更新の承認 (ct-1404e21baa2ox) ChangeSetName パラメータの CT ウォークスルーの説明が追加情報で更新されました。	CloudFormation テンプレートからのスタック | 変更セットと更新の承認
	Ubuntu 18.04 および Oracle Linux 8.3 が利用可能	ワークロードの移行: Linux と Windows の前提条件
新しいコンテンツ：	CFN Ingest および Stack Update CTs による IAM デプロイ。	2022 年 2 月 10 日
Database Migration Service (DMS) レプリケーションタスク	正規表現がハイフンを含むタスク ARNs を許可するように更新された変更タイプ。 AWS DMS レプリケーションタスクを開始するおよび Database Migration Service (DMS) | Stop Replication Task。	2022 年 1 月 13 日
Linux WIGS の取り込み前検証	zip ファイルが更新されました。 ワークロードの移行: Linux の取り込み前検証	2022 年 1 月 13 日

変更	説明	リンク
修正されたリンク	データベース (DB) の AMS SQL RDS へのインポート -> セットアップセクションに不正なリンクがありました。	2022 年 1 月 13 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。