aws

ユーザーガイド

EC2 Image Builder

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

EC2 Image Builder: ユーザーガイド

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標とトレードドレスは、Amazon 以外の製品またはサービスとの関連において、顧客に混乱を招いたり、Amazon の名誉または信用を毀損するような方法で使用することはできません。Amazon が所有しない他の商標はすべてそれぞれの所有者に帰属します。所有者は必ずしも Amazon との提携や関連があるわけではありません。また、Amazon の支援を受けているとはかぎりません。

Table of Contents

EC2 Image Builder とは何ですか

EC2 Image Builder はフルマネージド AWS のサービス 型で、カスタマイズ、セキュア、 up-to-date サーバーイメージの作成、管理、デプロイを自動化するのに役立ちます。 AWS Management Console、、または APIs を使用して AWS Command Line Interface、 でカスタムイメージを作成できます AWS アカウント。

Image Builder がアカウントで作成したカスタマイズされたイメージを所有しているのはお客様です。パイプラインを設定して、所有しているイメージの更新とシステムパッチを自動化できます。スタンドアロンコマンドを実行して、定義した設定リソースを使用してイメージを作成することもできます。

Image Builder パイプラインウィザードの指示に従って、カスタムイメージを作成します。手順は次のとおりです。

1. カスタマイズ用のベースイメージを選択します。
2. ベースイメージにソフトウェアを追加したり、ベースイメージからソフトウェアを削除したりします。
3. ビルドコンポーネントを使用して設定とスクリプトをカスタマイズします。
4. 選択したテストを実行するか、カスタムテストコンポーネントを作成します。
5. AMIs AWS リージョン と に配布します AWS アカウント。
6. Image Builder パイプラインがディストリビューション用のカスタム Amazon マシンイメージ (AMI) を作成する場合、他の AWS アカウント、組織、および OUs にアカウントからの起動を許可できます。所有者アカウントには、AMI に関連する料金が請求されます。

セクションの内容

EC2 Image Builder の機能

EC2 Image Builderは以下の機能を提供する：

生産性を向上させ、準拠した と up-to-date イメージを構築するためのオペレーションを減らす

Image Builder は、ビルドパイプラインを自動化することで、大規模なイメージの作成と管理にかかる作業量を削減します。ビルド実行スケジュールの設定を指定することで、ビルドを自動化できます。自動化により、最新のオペレーティングシステムパッチを適用してソフトウェアを保守する運用コストを削減できます。

サービスのアップタイムを増やす。

Image Builder では、デプロイ前にイメージをテストするために使用できるテストコンポーネントにアクセスできます。 AWS Task Orchestrator and Executor （ AWSTOE) を使用してカスタムテストコンポーネントを作成し、それらを使用することもできます。Image Builder は、設定されたテストがすべて成功した場合にのみイメージを配布します。

デプロイメントのセキュリティ・ バーを上げる。

Image Builder では、コンポーネントのセキュリティ上の脆弱性にさらされる不要なリスクを排除するイメージを作成できます。 AWS セキュリティ設定を適用して、業界および内部のセキュリティ基準を満たす安全なout-of-the-box イメージを作成できます。Image Builder には、規制対象の業界に属する企業向けの設定コレクションも用意されています。これらの設定を使用して、STIG 標準に準拠したイメージを迅速かつ簡単に構築できます。Image Builder で使用可能な STIG コンポーネントの完全なリストについては、「EC2 Image Builder用の Amazon managed STIG 強化コンポーネント」を参照してください。

一元管理と血統追跡。

Image Builder では AWS Organizations、 との組み込み統合を使用して、承認された AMIs からのみインスタンスを実行するようにアカウントを制限するポリシーを適用できます。

AWS アカウントリソース共有の簡素化

EC2 Image Builder は AWS Resource Access Manager （ AWS RAM) と統合されており、特定のリソースを AWS アカウント または を通じて共有できます AWS Organizations。共有できる EC2 Image Builder リソースは下記のとおりです。

- コンポーネント

- イメージ
- イメージのレシピ
- コンテナレシピ

詳細については、「EC2 Image Builder リソースを共有」を参照してください。

サポートされるオペレーティングシステム

Image Builderは、以下のオペレーティングシステムのバージョンをサポートしています：

オペレーティングシステム / ディストリビューション	サポートバージョン
Amazon Linux	2 と 2023 年
CentOS	7 と 8
CentOS Stream	8
Red Hat Enterprise Linux (RHEL)	7 と 8
SUSE Linux Enterprise Server (SUSE)	12 と 15
Ubuntu	18.04 LTS、20.04 LTS、22.04 LTS
Windows Server	2012 R2、2016、2019、2022年

対応イメージフォーマット

カスタム AMI イメージでは、開始点として既存の AMI を選択できます。Docker コンテナイメージの場合、 でホストされているパブリックイメージ DockerHub、Amazon ECR 内の既存のコンテナイメージ、または Amazon が管理するコンテナイメージから選択できます。

概念

以下の用語と概念は、EC2 Image Builderを理解し、使用する上で中心となるものです。

AMI

Amazon マシンイメージ (AMI) は Amazon EC2 のデプロイの基本単位であり、Image Builder で作成できるイメージの種類の 1 つです。AMI は、EC2 インスタンスをデプロイするためのオペレーティングシステム (OS) とプリインストールされたソフトウェアを含む事前設定された仮想マシンイメージです。詳細については、Amazon マシンイメージ (AMI)を参照。

イメージパイプライン

イメージパイプラインは、安全なAMIとコンテナイメージを構築するための自動化フレームワークを提供する AWS。Image Builder イメージパイプラインは、イメージビルドライフサイクルのビルド、検証、およびテストフェーズを定義するイメージ recipe に関連付けられています。

イメージパイプラインは、イメージの構築場所を定義するインフラストラクチャ構成に関連付けることができます。インスタンスタイプ、サブネット、セキュリティグループ、ログ記録、その他のインフラストラクチャ関連の構成などの属性を定義できます。また、イメージパイプラインをディストリビューション構成に関連付けて、イメージのデプロイ方法を定義することもできます。

マネージドイメージ

マネージドイメージは、AMI またはコンテナイメージに加えて、バージョンやプラットフォームなどのメタデータで構成される Image Builder のリソースです。管理対象イメージは、Image Builder パイプラインによってビルドに使用するベースイメージを決定するために使用されます。このガイドでは、管理対象イメージは「イメージ」と呼ばれることもありますが、イメージは AMI とは異なります。

イメージのレシピ

Image Builder イメージ recipe は、ベースイメージとベースイメージに適用するコンポーネントを定義し、必要な構成の出力イメージを生成するためのドキュメントです。イメージ recipe を使用して、ビルドを複製できます。Image Builder イメージレシピは、コンソールウィザード、、または API を使用して共有、分岐 AWS CLI、編集できます。バージョン管理ソフトウェアでイメージ recipe を使用して、バージョン管理された共有可能なイメージ recipe を維持できます。

コンテナレシピ

Image Builder コンテナレシピは、ベースイメージと、出力コンテナイメージに必要な構成を生成するためにベースイメージに適用されるコンポーネントを定義する文書です。コンテナレシピを使ってビルドを複製することができます。コンソールウィザード、 AWS CLI、または API を使用して、Image Builder イメージレシピを共有、分岐、編集できます。コンテナレシピをバージョン管理

ソフトウェアと一緒に使うことで、共有可能でバージョン管理されたコンテナレシピを維持することができます。

ベースイメージ

ベースイメージとは、イメージまたはコンテナレシピドキュメントで使用される、選択されたイメージとオペレーティングシステム、およびコンポーネントです。ベースイメージとコンポーネント定義を組み合わせることで、出力イメージに必要な設定が作成されます。

コンポーネント

コンポーネントは、イメージ作成前にインスタンスをカスタマイズする（ビルドコンポーネント）か、作成されたイメージから起動されたインスタンスをテストする（テストコンポーネント）ために必要な一連の手順を定義します。

コンポーネントは、パイプラインによって生成されたインスタンスをビルド、検証、またはテストするためのランタイム設定を記述した、宣言型のプレーンテキストの YAML または JSON ドキュメントから作成されます。コンポーネントは、コンポーネント管理アプリケーションを使用してインスタンス上で実行されます。コンポーネント管理アプリケーションはドキュメントを解析し、必要なステップを実行します。

作成後、イメージレシピまたはコンテナレシピを使用して 1 つ以上のコンポーネントをグループ化し、仮想マシンまたはコンテナイメージの構築とテストの計画を定義します。によって所有および管理されているパブリックコンポーネントを使用することも AWS、独自のコンポーネントを作成することもできます。コンポーネントの詳細については、「AWS Task Orchestrator and Executor コンポーネントマネージャー」を参照してください。

コンポーネント・ドキュメント

イメージに適用できるカスタマイズの設定を記述した、宣言型のプレーンテキストの YAML または JSON ドキュメント。このドキュメントは、ビルドまたはテストコンポーネントの作成に使用されます。

ランタイムステージ

EC2 Image Builderには2つのruntimeステージがある： buildとtestです。各ランタイムステージには、コンポーネントドキュメントで定義されている設定を含む 1 つ以上のフェーズがあります。

設定フェーズ

以下のリストは、buildとtestのステージで実行されるフェーズを示しています：

ビルドステージ

ビルドフェーズ

イメージパイプラインは、実行時にビルドステージのビルドフェーズから始まります。ベースイメージがダウンロードされ、コンポーネントのビルドフェーズに指定された構成がインスタンスのビルドと起動に適用されます。

検証フェーズ

Image Builder がインスタンスを起動し、ビルドフェーズのカスタマイズをすべて適用すると、検証フェーズが開始されます。このフェーズでは、Image Builder は、コンポーネントが検証フェーズで指定する構成に基づいて、すべてのカスタマイズが期待どおりに機能することを確認します。インスタンスの検証が成功すると、Image Builder はインスタンスを停止し、イメージを作成して、テスト段階に進みます。

テストステージ:

テストフェーズ

このフェーズでは、Image Builder は検証フェーズが正常に完了した後に作成したイメージからインスタンスを起動します。Image Builder は、このフェーズ中にテストコンポーネントを実行して、インスタンスが正常で期待どおりに機能していることを確認します。

コンテナホストテストフェーズ

Image Builder がコンテナレシピで選択したすべてのコンポーネントのテストフェーズを実行すると、Image Builder はコンテナワークフローに対してこのフェーズを実行します。コンテナホストのテストフェーズでは、コンテナ管理とカスタムランタイム設定を検証する追加のテストを実行できます。

ワークフロー

ワークフローは、Image Builder が新しいイメージを作成するときに実行する一連のステップを定義します。すべてのイメージには、ビルドおよびテストワークフローがあります。コンテナには配布用のワークフローが追加されています。

ワークフローの種類

`BUILD`

作成されたすべてのイメージのビルドステージ設定をカバーします。

TEST

作成されたすべてのイメージのビルドステージ設定をカバーします。

DISTRIBUTION

コンテナイメージの配布ワークフローについて説明します。

料金

カスタムEC2 Image Builder を使用してカスタム AMI またはコンテナイメージを作成してもコストはかかりません。ただし、このプロセスで使用される他のサービスには標準価格が適用されます。次のリストには、設定に応じてカスタム AMI またはコンテナイメージを作成、構築、保存、および配布するときにコストが発生する AWS のサービス 可能性のある の使用が含まれています。

- EC2 インスタンスの起動
- Amazon S3 にログを保存する
- Amazon Inspector によるイメージの検証
- AMI 用の Amazon EBS スナップショットの保存
- Amazon ECR へのコンテナイメージの保存
- Amazon ECR へのコンテナイメージのプッシュと Amazon ECR からのコンテナイメージのプッシュとプル
- Systems Manager アドバンスティアがオンになっていて、Amazon EC2 インスタンスがオンプレミスアクティベーションで実行されている場合、Systems Manager を通じてリソースの料金が請求されることがあります。

関連 AWS のサービス

EC2 Image Builder は、他の を使用してイメージを構築 AWS のサービス します。Image Builder のイメージレシピまたはコンテナレシピの設定によっては、次のサービスが使用される場合があります。

AWS License Manager

AWS License Manager では、アカウントライセンス設定ストアからライセンス設定を作成して適用できます。AMI ごとに、Image Builder を使用して、Image Builder ワークフローの一部として AWS アカウント がアクセスできる既存のライセンス設定にアタッチできます。ライセンス設定は AMI に

のみ適用できます。Image Builder は既存のライセンス構成のみを使用でき、ライセンス構成を直接作成または変更することはできません。License Manager の設定は、 (アジアパシフィック: 香港) リージョンと ap-east-1 (中東: バーレーン) me-south-1 リージョン間でなど、アカウントで有効に AWS リージョン する必要がある 間でレプリケートされません。

AWS Organizations

AWS Organizations では、組織内のアカウントにサービスコントロールポリシー (SCP) を適用できます。個々のポリシーを作成、管理、有効化、無効化できます。他のすべての AWS アーティファクトやサービスと同様に、Image Builder は で定義されているポリシーを尊重します AWS Organizations。承認された AMIs のみを使用してインスタンスを起動するようにメンバーアカウントに制約を適用するなど、一般的なシナリオのためにテンプレート SCPs AWS を提供します。

Amazon Inspector

Image Builder は Amazon Inspector をデフォルトの脆弱性スキャンエージェントとして使用し、Amazon Linux 2、Windows Server 2012、および Windows Server 2016 のセキュリティベースラインを確立します。詳しくは、Amazon Inspectorとはを参照してください。

AWS Resource Access Manager

AWS Resource Access Manager （ AWS RAM) を使用すると、 AWS アカウント または を介してリソースを共有できます AWS Organizations。が複数ある場合は AWS アカウント、リソースを一元的に作成し、 を使用してそれらのリソース AWS RAM を他の アカウントと共有できます。EC2 Image Builder では、コンポーネント、イメージ、イメージレシピなどのリソースを共有できます。の詳細については AWS RAM、「 AWS Resource Access Manager ユーザーガイド」を参照してください。Image Builder リソースの共有については、「EC2 Image Builder リソースを共有」を参照してください。

Amazon CloudWatch Logs

Amazon CloudWatch Logs を使用して、EC2 インスタンス、Amazon Route 53 AWS CloudTrail、およびその他のソースからログファイルをモニタリング、保存、およびアクセスできます。

Amazon Elastic Container Registry (Amazon ECR)

Amazon ECR は、安全でスケーラブル、信頼性の高いマネージド AWS コンテナイメージレジストリサービスです。Image Builder で作成したコンテナイメージは、ソースリージョン (ビルドが実行されるリージョン) と、コンテナイメージを配布するすべてのリージョンの Amazon ECR に保存されます。Amazon ECRの詳細については、Amazon Elastic Container Registry User Guideを参照してください。

EC2 Image Builderの仕組み

EC2 Image Builder パイプラインコンソールウィザードを使用してカスタムイメージを作成する場合、ウィザードの指示に従って次の手順を実行します。

1. パイプラインの詳細を指定する - 名前、説明、タグ、自動ビルドを実行するスケジュールなど、パイプラインに関する情報を入力します。手動ビルドを選択することもできます。
2. レシピを選択 - AMIをビルドするか、コンテナイメージをビルドするかを選択する。どちらのタイプの出力イメージでも、レシピの名前とバージョンを入力し、ベースイメージを選択し、ビルドとテスト用に追加するコンポーネントを選択します。また、自動バージョン管理を選択して、ベースイメージに使用可能な最新のオペレーティングシステム (OS) バージョンを常に使用できるようにすることもできます。コンテナレシピは、さらに Dockerfiles と、出力 Docker コンテナイメージのターゲット Amazon ECR リポジトリを定義します。

   > **Note**
   >
   > コンポーネントは、イメージレシピまたはコンテナレシピで使用されるビルディングブロックです。たとえば、インストール用パッケージ、セキュリティ強化手順、テストなどです。選択したベースイメージとコンポーネントがイメージレシピを構成します。

3. インフラ構成の定義 - Image Builderは、アカウント内のEC2インスタンスを起動し、イメージをカスタマイズして検証テストを実行します。インフラストラクチャ設定では、ビルドプロセス AWS アカウント 中に で実行されるインスタンスのインフラストラクチャの詳細を指定します。
4. 配布設定の定義 - ビルドが完了し、すべてのテストに合格した後、イメージを配布する AWS リージョンを選択します。パイプラインはビルドを実行するリージョンに自動的にイメージを配布します。他のリージョンにイメージ配布を追加することもできます。

カスタムベースイメージからビルドしたイメージは、 AWS アカウントにあります。ビルドスケジュールを入力することで、イメージの更新バージョンやパッチを適用したバージョンを生成するようにイメージパイプラインを設定できます。ビルドが完了すると、「Amazon 簡易通知サービス (SNS)」 を通じて通知を受け取ることができます。Image Builder コンソールウィザードは、最終イメージを生成するだけでなく、既存のバージョン管理システムや継続的インテグレーション/継続的デプロイ (CI/CD) パイプラインで使用できるレシピを生成し、繰り返し可能な自動化を実現します。レシピを共有したり、新しいバージョンを作成したりできます。

セクションの内容

AMI エレメント

Amazon マシンイメージ (AMI) は、EC2 インスタンスをデプロイするための OS とソフトウェアを含む事前設定済みの仮想マシン (VM) Image (VM) Image。

AMIには以下の要素が含まれます：

- VM のルートボリュームのテンプレート。Amazon EC2 VMを起動すると、ルートデバイスボリュームにインスタンスを起動するためのイメージが格納されます。インスタンスストアを使用する場合、ルートデバイスは、Amazon S3 のテンプレートから作成されるインスタンスストアボリュームになります。詳細については、Amazon EC2 Root Device Volumeを参照のこと。
- Amazon EBS を使用する場合、ルートデバイスは 「EBS スナップショット」から作成された EBS ボリュームです。
- AMI で VMs を起動 AWS アカウント できる を決定する起動許可。
- 起動後にインスタンスにアタッチするボリュームを指定する ブロックデバイスマッピング データ。
- 各リージョン、各アカウントの固有の「リソース識別子」。
- タグなどの「メタデータ」ペイロード、およびプロパティ (リージョン、オペレーティングシステム、アーキテクチャ、ルートデバイスタイプ、プロバイダー、起動権限、ルートデバイスのストレージ、署名ステータスなど)。
- 不正な改ざんから保護するための Windows イメージ用の AMI シグネチャ。詳細については、インスタンスアイデンティティドキュメントを参照してください。

デフォルトのクォータ

Image Builder のデフォルトクォータを確認するには、「Image Builder エンドポイントとクォータ」を参照してください。

AWS リージョンとエンドポイント

Image Builder のサービスエンドポイントを表示するには、「Image Builder エンドポイントとクォータ」を参照してください。

コンポーネント管理

EC2 Image Builder は、複雑なワークフローのオーケストレーション、システム設定の変更、YAML ベースのスクリプトコンポーネントによるシステムのテストに役立つコンポーネント管理アプリケーション AWS Task Orchestrator and Executor （ AWSTOE) を使用します。 AWSTOE はスタンドアロンアプリケーションであるため、追加のセットアップは必要ありません。どのクラウドインフラストラクチャーでもオンプレミスでも実行できます。スタンドアロンアプリケーション AWSTOE として の使用を開始するには、「」を参照してくださいの使用を開始する AWSTOE。

Image Builder は AWSTOE 、 を使用してすべてのインスタンス上のアクティビティを実行します。これには、スナップショットを作成する前のイメージの構築と検証、および最終的なイメージを作成する前にスナップショットが期待どおりに機能することを確認するためのテストが含まれます。Image Builder が AWSTOE を使用してコンポーネントを管理する方法の詳細については、「」を参照してくださいImage Builder によるコンポーネントの管理。 AWSTOEを使ったコンポーネントの作成については、AWS Task Orchestrator and Executor コンポーネントマネージャーを参照のこと。

イメージテスト

AWSTOE テストコンポーネントを使用してイメージを検証し、最終的なイメージを作成する前に、イメージが期待どおりに機能することを確認できます。

通常、各テストコンポーネントは、テストスクリプト、テストバイナリ、テストメタデータを含む YAML ドキュメントで構成されます。テストスクリプトにはテストバイナリを起動するためのオーケストレーションコマンドが含まれており、OS がサポートする任意の言語で記述できます。終了ステータスコードはテスト結果を示します。テストメタデータは、名前、説明、テストバイナリへのパス、予想される時間など、テストとその動作を記述します。

セマンティックバージョニング

Image Builder はセマンティックバージョニングを使用してリソースを整理し、それらに固有の ID が割り当てられるようにします。セマンティックバージョンには次の 4 つのノードがあります。

<major>。*<minor>*v*<patch>*/<build>

最初の 3 つの値を割り当てて、それらのすべてをフィルタリングできます。

セマンティックバージョニングは、各オブジェクトの Amazon リソースネーム (ARN) に、そのオブジェクトに適用されるレベルで次のように含まれています。

1. バージョンレス ARN と名前 ARN には、どのノードにも特定の値が含まれていません。ノードは完全に省略されるか、x.x.x のようにワイルドカードとして指定されます。
2. <major>バージョン ARN には最初の 3 つのノードしかありません。 <minor>。 <patch>
3. ビルドバージョン ARN には 4 つのノードすべてがあり、オブジェクトの特定のバージョンの特定のビルドを指します。

割り当て: 最初の3つのノードには、0を含む任意の正の整数値を割り当てることができ、各ノードの上限は2^30-1、つまり1073741823である。Image Builder は、4 番目のノードにビルド番号を自動的に割り当てます。

パターン: 割り当て可能なノードの割り当て要件に準拠する任意の数値パターンを使用できます。たとえば、1.0.0 などのソフトウェアバージョンパターン、または 2021.01.01 などの日付を選択できます。

選択: セマンティックバージョニングでは、レシピのベースイメージやコンポーネントを選択する際に、ワイルドカード(x)を使って最新のバージョンやノードを指定する柔軟性があります。任意のノードでワイルドカードを使用する場合、最初のワイルドカードの右側にあるすべてのノードもワイルドカードである必要があります。

たとえば、最近のバージョンが 2.2.4、1.7.8、1.6.8 の場合、ワイルドカードを使用してバージョンを選択すると次のような結果になります。

- `x.x.x` = 2.2.4
- `1.x.x` = 1.7.8
- `1.6.x` = 1.6.8
- `x.2.x` は無効で、エラーになっています。

- `1.x.8` は無効で、エラーになっています。

作成されたリソース

パイプラインを作成すると、以下の場合を除き、Image Builder の外部リソースは作成されません。

- パイプラインスケジュールに従ってイメージが作成された場合
- Image Builder コンソールの[アクション]メニューから[パイプラインを実行]を選択した場合
- API または からこれらのコマンドのいずれかを実行する場合 AWS CLI: StartImagePipelineExecutionまたは CreateImage

イメージビルドプロセス中に次のリソースが作成されます。

AMIイメージパイプライン

- EC2 インスタンス (一時的)
- EC2 インスタンス上のSystems Manager インベントリアソシエーション (`EnhancedImageMetadata` が有効になっている場合はSystems Manager ステートマネージャー経由)
- Amazon EC2 AMI
- Amazon EC2 AMI に関連付けられた Amazon EBS スナップショット

コンテナイメージパイプライン

- EC2インスタンス上で動作するDockerコンテナ(temporary)
- EC2インスタンスのSystems Managerインベントリ関連付け（Systems Manager State Manager経由）`EnhancedImageMetadata`が有効になっているます。
- Docker コンテナイメージ
- Dockerfile

イメージが作成されると、一時リソースはすべて削除されます。

ディストリビューション

EC2 Image Builder は、AMIs またはコンテナイメージを任意の AWS リージョンに配布できます。イメージは、イメージのビルドに使用したアカウントで指定した各リージョンにコピーされます。

AMI 出力イメージでは、AMI 起動許可を定義して、作成された AMI で EC2 インスタンスを起動 AWS アカウント することを許可する を制御できます。たとえば、イメージをプライベート、パブリック、または特定のアカウントと共有することができます。AMI を他のリージョンに配布し、他のアカウントの起動権限を定義すると、起動権限は AMI が配布されているすべてのリージョンの AMI に伝達されます。

AWS Organizations アカウントを使用して、承認された AMI と準拠した AMIs でのみインスタンスを起動するようにメンバーアカウントに制限を適用することもできます。詳細については、「組織 AWS アカウント での の管理」を参照してください。

Image Builder コンソールを使用してディストリビューション設定を更新するには、「新しいイメージレシピのバージョン (コンソール) を作成」または「コンソールで新しいコンテナ recipe の作成」のステップに従います。

リソースの共有

コンポーネント、レシピ、またはイメージを他の アカウントまたは 内で共有するには AWS Organizations、「」を参照してくださいEC2 Image Builder リソースを共有。

コンプライアンス

CIS については、EC2 Image Builder が Amazon Inspector を使用して、漏洩、脆弱性、ベストプラクティスやコンプライアンス標準からの逸脱がないか確認できます。たとえば、Image Builder は、意図しないネットワークアクセス、パッチが適用されていない CVE、公衆インターネット接続、リモートルートログインの有効化を評価します。Amazon Inspector はテストコンポーネントとして提供されており、イメージレシピに追加することを選択できます。Amazon Inspectorの詳細については、Amazon Inspectorユーザーガイドを参照してください。強化のため、EC2 Image Builder は STIG で検証します。Image Builder で使用可能な STIG コンポーネントの完全なリストについては、「EC2 Image Builder用の Amazon managed STIG 強化コンポーネント」を参照してください。詳細については、「インターネットセキュリティセンター (CIS) ベンチマークセンター」を参照してください。

EC2 Image Builder を使い始める

この章では、EC2 Image Builder の [Create Image Pipeline] コンソールウィザードを使用して、環境をセットアップし、自動イメージパイプラインまたはコンテナパイプラインを初めて作成する方法について説明します。

内容

前提条件

EC2 Image Builder でイメージパイプラインを作成するには、次の前提条件を確認してください。特に明記されていない限り、すべての種類のパイプラインに前提条件が必要です。

EC2 Image Builder サービスにリンクされたロールを使用する

EC2 Image Builder は、サービスにリンクされたロールを使用して、ユーザーに代わって他の AWS サービスにアクセス許可を付与します。サービスリンクロールを手動で作成する必要はありません。AWS マネジメントコンソール、、 AWS CLIまたは AWS API で最初の Image Builder リソースを作成すると、Image Builder によってサービスにリンクされたロールが作成されます。 が作成するサービスにリンクされたロールの詳細については、「EC2 Image Builder サービスにリンクされたロールを使用する」を参照してください。

での設定の要件

- Image Builder のサポート VPC インターフェイスエンドポイントの設定の詳細については、「EC2 Image Builder とインターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。
- Image Builder がコンテナイメージを構築するために使用するインスタンスには、Amazon S3 AWS CLI から をダウンロードし、該当する場合は Docker Hub リポジトリからベースイメージを

ダウンロードするためのインターネットアクセスが必要です。Image Builder は AWS CLI を使用してコンテナレシピから Dockerfile を取得し、データとして保存します。

- Image Builder がイメージの構築とテストの実行に使用するインスタンスには、Systems Manager サービスへのアクセス権が必要です。インストール要件はオペレーティングシステムによって異なります。

  ベースイメージのインストール要件を確認するには、ベースイメージのオペレーティングシステムに合ったタブを選択してください。

  Linux

  Amazon EC2 Linux インスタンスの場合、Image Builder はビルドインスタンスに Systems Manager エージェントがまだ存在しない場合はそれをインストールし、イメージを作成する前に削除します。

  Windows

  Image Builder は、Amazon EC2 Windows インスタンスに Systems Manager エージェントを自動ではインストールしません。基本イメージに Systems Manager エージェントがあらかじめインストールされていない場合は、ソースイメージからインスタンスを起動し、そのインスタンスに Systems Manager を手動でインストールして、インスタンスから新しい基本イメージを作成する必要があります。

  Amazon EC2 Windows ServerインスタンスにSystems Managerエージェントを手動でインストールするには、AWS Systems Manager ユーザーガイドのManually install Systems Manager Agent on EC2 instances for Windows Serverを参照してください。

コンテナリポジトリ (コンテナイメージパイプライン)

コンテナイメージパイプラインの場合、レシピはターゲットコンテナーリポジトリに生成され保存される Docker イメージの設定を定義します。Docker イメージのコンテナーレシピを作成する前に、ターゲットリポジトリを作成する必要があります。

Image Builder は Amazon ECR をコンテナイメージのターゲットリポジトリとして使用します。Amazon ECRリポジトリを作成するには、Amazon Elastic Container Registry User GuideのCreating a repositoryに記載されている手順に従います。

AWS Identity and Access Management (IAM)

インスタンスプロファイルに関連付ける IAM ロールには、イメージに含まれるビルドコンポーネントとテストコンポーネントを実行する権限が必要です。インスタンスプロファイルに関連付けられている IAM ロールに対し、次の IAM ロールポリシーをアタッチする必要があります。

- EC2InstanceProfileForImageBuilder
- EC2InstanceProfileForImageBuilderECRContainerBuilds
- AmazonSSMManagedInstanceCore

ロギングを設定する場合、インフラストラクチャ構成で指定されたインスタンスプロファイルは、ターゲットバケット(`arn:aws:s3:::`**`BucketName`**`/*`)に対して`s3:PutObject`の権限を持っている必要があります。例:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::bucket-name/*"
        }
    ]
}
```

ポリシーのアタッチ

以下の手順は、IAM ポリシーを IAM ロールにアタッチして前述のアクセス権限を付与するプロセスを示しています。

1. AWS マネジメントコンソールにサインインし、https://console.aws.amazon.com/iam/ で IAM コンソールを開きます。
2. 左のナビゲーションペインの [ポリシー] を選択します。
3. EC2InstanceProfileForImageBuilderでポリシーのリストをフィルタリングする。
4. ポリシーの横にあるbullet を選択し、[ポリシーアクション]ドロップダウンリストから[添付]を選択します。

5. ポリシーを添付する IAM ロールの名前を選択します。
6. Attach policy] (ポリシーのアタッチ) を選択します。
7. EC2InstanceProfileForImageBuilderECRContainerBuilds および AmazonSSMManagedInstanceCore ポリシーに対してステップ 3～6 を繰り返します。

> **Note**
>
> Image Builder で作成したイメージを別のアカウントにコピーする場合は、すべてのターゲットアカウントで `EC2ImageBuilderDistributionCrossAccountRole` ロールを作成し、「Ec2ImageBuilderCrossAccountDistributionAccess ポリシー」管理ポリシーをロールにアタッチする必要があります。詳細については、「EC2 Image Builder リソースを共有」を参照してください。

EC2 Image Builder へのアクセス

EC2 Image Builder は、以下のインターフェイスのいずれかから管理できます。

- EC2 Image Builderコンソールのランディングページ。EC2 Image Builder コンソールから。
- AWS Command Line Interface （ AWS CLI ）。 を使用して AWS API オペレーション AWS CLI にアクセスできます。詳細については、「 ユーザーガイド」の AWS 「 コマンドラインインターフェイスのインストール」を参照してください。 AWS Command Line Interface
- AWS SDKs用のツール。AWS SDKs とツールを使用して、お好みの言語で Image Builder にアクセスして管理できます。

EC2 Image Builder コンソールウィザードを使用してコンテナイメージパイプラインを作成します。

このチュートリアルでは、[Create image pipeline] コンソールウィザードを使用してカスタマイズされた EC2 Image Builder イメージを構築および管理するための自動パイプラインを作成する方法について説明します。ステップを効率的に進めるために、使用可能な場合はデフォルトの設定が使用され、オプションのセクションは省略されます。

イメージパイプラインワークフローを作成します。

- ステップ 1: パイプラインの詳細を指定する

ステップ 1: パイプラインの詳細を指定する

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. パイプラインの作成を開始するには、[イメージパイプラインの作成] を選択します。
3. ーセクションに、パイプライン名 (必須) を入力します。

> **Tip**
>
> 拡張メタデータの収集は、デフォルトで有効になっています。コンポーネントとベースイメージ間の互換性を確保するため、有効にしておいてください。

4. ビルドスケジュールセクションでは、スケジュールオプションはデフォルトのままでかまいません。デフォルトのスケジュールに表示される タイムゾーン は協定世界時 (UTC) であることに注意してください。UTC 時間の詳細とタイムゾーンのオフセットについては、「タイムゾーンの略語 — ワールドワイドリスト」を参照してください。

   依存関係の更新設定については、依存関係の更新がある場合はスケジュールされた時間にパイプラインを実行するオプションを選択します。この設定により、パイプラインはビルドを開始する前に更新をチェックします。更新がない場合は、スケジュールされたパイプラインビルドはスキップされます。

> **Note**
>
> パイプラインが依存関係の更新とビルドを想定どおりに認識できるようにするには、ベースイメージとコンポーネントにセマンティックバージョニング (x.x.x) を使用する必要があります。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。

5. 次へ を選択して次のステップに進みます。

ステップ 2: レシピを選択する

1. Image Builder は、デフォルトでレシピセクションの既存のレシピを使用に設定されています。初めて使用する場合は、「新規レシピを作成」オプションを選択します。
2. [イメージタイプ]セクションで、[Amazon マシンイメージ (AMI)]オプションを選択し、AMI を作成および配布するイメージパイプラインを作成します。
3. [キュー] セクションに、以下の情報を入力します。

   - 名前 — レシピ名
   - バージョン - レシピのバージョン (<major>.<minor>.<patch>のフォーマットで、メジャー、マイナー、パッチは整数値です) 。通常、新しいレシピは `1.0.0` で始まります。
4. ソースイメージセクションでは、イメージを選択、イメージオペレーティングシステム (OS)、およびイメージオリジンをデフォルト値のままにします。これにより、Amazon が管理する Amazon Linux 2 AMIコンテナイメージのリストが作成され、ベースイメージとして選択できます。

   a. [イメージ名] ドロップダウンから、イメージを選択します。

   b. 自動バージョニング管理オプション[はデフォルトのままにします (利用可能な最新の OS バージョンを使用)。

   > **Note**
   >
   > この設定により、パイプラインがベースイメージのセマンティックバージョニングを使用して、自動的にスケジュールされたジョブの依存関係の更新を検出できるようになります。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。
5. [インスタンス設定セクション] では、[Systems Manager エージェント] のデフォルト値をそのまま使用します。これにより、Image Builder はビルドとテストが完了した後も Systems Manager エージェントを保持し、新しいイメージに Systems Manager エージェントを含めることになります。

   このチュートリアルでは、[ユーザーデータ] は空白のままにしてください。構築インスタンスを起動するときにこのエリアを使用して、コマンドまたはコマンドスクリプトを提供で実行します。ただし、この値は、Systems Manager が確実にインストールされるようにするために Image Builder が追加されたコマンドを置き換えます。ユーザーデータを入力する際に、システ

ムマネージャーエージェントがベースイメージにあらかじめインストールされていますか、ユーザーデータにインストールを必ず含めてください。

6. コンポーネントセクションでは、少なくとも 1 つのビルドコンポーネントを選択する必要があります。

   「ビルドコンポーネント — Amazon Linux」パネルでは、ページに表示されているコンポーネントをブラウズできます。右上隅のページネーションコントロールを使用して、ベースイメージ OS で使用できるその他のコンポーネント間を移動できます。特定のコンポーネントを検索したり、Component Manager を使用して独自のビルドコンポーネントを作成したりすることもできます。

   このチュートリアルでは、Linux を最新のセキュリティアップデートで更新するコンポーネントを次のように選択します。

   a. パネルの上部にある検索バーに単語 `update` を入力して結果を絞り込みます。

   b. `update-linux` ビルドコンポーネントのチェックボックスをオンにします。

   c. 下にスクロールして、「選択したコンポーネント」リストの右上隅にある「すべて展開」を選択します。

   d. バージョニング管理 オプションはデフォルトのままにします([利用可能な最新のコンポーネントバージョンを使用])。

      > **Note**
      >
      > この設定により、パイプラインがベースイメージのセマンティックバージョニングを使用して、自動的にスケジュールされたジョブの依存関係の更新を検出できるようになります。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。

      入力パラメータのあるコンポーネントを選択した場合は、この領域にもパラメータが表示されます。パラメーターについては、このチュートリアルでは説明しません。コンポーネントでの入力パラメータの使用とレシピでの設定の詳細については、「EC2 Image Builder で AWSTOE コンポーネントパラメータを管理する」を参照してください。

   コンポーネントの順序変更 (オプション)

イメージに含めるコンポーネントを複数選択した場合は、 drag-and-drop アクションを使用して、ビルドプロセス中に実行する順序に再配置できます。

> **Note**
>
> CIS 強化コンポーネントは、Image Builder レシピの標準コンポーネント順序ルールに従っていません。CIS 強化コンポーネントは常に最後に実行され、ベンチマークテストが出力イメージに対して確実に実行されます。

1. スクロールして使用可能なコンポーネントのリストに戻ります。
2. `update-linux-kernel-mainline` ビルドコンポーネント (または任意の他のコンポーネント) のチェックボックスを選択します。
3. 「選択したコンポーネント」リストまでスクロールして、少なくとも 2 つの結果が表示されることを確認します。
4. 新しく追加されたコンポーネントは、バージョニング設定や入力パラメータ設定が拡張されていない可能性があります バージョニング管理オプション または [入力パラメーター] の設定を拡張するには、設定名の横にある矢印を選択します。選択したすべてのコンポーネントの設定をすべて展開するには、[すべて展開] スイッチのオンとオフを切り替えます。
5. コンポーネントの 1 つを選択し、それを上下にドラッグしてコンポーネントの実行順序を変更します。
6. `update-linux-kernel-mainline` コンポーネントを削除するには、X コンポーネントボックスの右上隅からを選択します。
7. 前のステップを繰り返して、追加した可能性のある他のコンポーネントをすべて削除し、その `update-linux` コンポーネントのみを選択したままにします。

7. 次へ を選択して次のステップに進みます。

ステップ 3: インフラストラクチャー設定を定義する (オプション)

Image Builder はアカウントで EC2 インスタンスを起動して、イメージをカスタマイズし、検証テストを実行します。インフラストラクチャ設定では、ビルドプロセス AWS アカウント 中に で実行されるインスタンスのインフラストラクチャの詳細を指定します。

[インフラストラクチャー設定]セクションでは、[設定]オプションのデフォルトは `Create infrastructure configuration using service defaults` です。これにより、ビルドイ

ンスタンスがコンテナイメージを設定するために使用する IAM ロールと関連するインスタンスプロファイルが作成されます。インフラストラクチャ設定の詳細については、EC2 Image Builder API リファレンスCreateInfrastructureConfigurationの「」を参照してください。

このチュートリアルでは、デフォルト設定を使用します。

> **Note**
>
> プライベート VPC に使用するサブネットを指定するには、独自のカスタムインフラストラクチャ設定を作成するか、すでに作成した設定を使用できます。

- 次へ を選択して次のステップに進みます。

ステップ 4: ディストリビューション設定を定義する (オプション)

ディストリビューション設定には、出力 AMI 名、暗号化用の特定のリージョン設定、起動許可、出力 AMI を起動できる AWS アカウント、組織、組織単位 (OUs)、ライセンス設定が含まれます。

[ディストリビューション設定]セクションの[設定オプション]はデフォルトで `Create distribution settings using service defaults` です。このオプションは出力 AMI を現在のリージョンに配信します。ディストリビューション設定の詳細については、「EC2 Image Builder ディストリビューション設定の管理」を参照してください。

このチュートリアルでは、デフォルト設定を使用します。

- 次へ を選択して次のステップに進みます。

ステップ 5: 確認

レビューセクションには、設定したすべての設定が表示されます。特定のセクションの情報を編集するには、ステップセクションの右上隅にある「編集」ボタンを選択します。たとえば、パイプライン名を変更する場合は、「ステップ 1: パイプラインの詳細」セクションの右上隅にある「編集」ボタンを選択します。

1. 設定を確認したら、[パイプラインを作成] を選択してパイプラインを作成します。
2. ディストリビューション設定、インフラストラクチャー設定、新しいレシピ、パイプライン用のリソースが作成されると、ページ上部に成功または失敗のメッセージが表示されます。リソース識別子を含むリソースの詳細を表示するには、[詳細を表示] を選択します。

3. リソースの詳細を確認したら、ナビゲーションペインからリソースタイプを選択すると、他のリソースの詳細を表示できます。たとえば、新しいパイプラインの詳細を表示するには、ナビゲーションペインから [Image pipelines] を選択します。ビルドが成功すると、新しいパイプラインが [イメージパイプライン] リストに表示されます。

ステップ 6: クリーンアップする

Image Builder 環境は、ご自宅と同様、必要なものを見つけて散らかることなくタスクを完了できるように、定期的なメンテナンスが必要です。テスト用に作成した一時リソースは定期的にクリーンアップしてください。そうしないと、それらのリソースのことを忘れてしまい、後でそのリソースが何に使用されたかを思い出せなくなる可能性があります。その時までには、それらを安全に取り除くことができるかどうかがはっきりしないかもしれません。

> **Tip**
>
> リソースを削除するときの依存関係エラーを防ぐため、必ず次の順序でリソースを削除してください。
>
> 1. イメージパイプライン
> 2. イメージのレシピ
> 3. 残っているすべてのリソース

このチュートリアルのために作成したリソースをクリーンアップするには、以下の手順に従ってください：

パイプラインを削除します。

1. アカウントで作成されたビルドパイプラインのリストを表示するには、ナビゲーションペインから [Image pipelines] を選択します。
2. パイプライン名 の横にあるチェックボックスをオンにして、削除するパイプラインを選択します。
3. イメージパイプラインパネルの上部にある「アクション」メニューで、「削除」を選択します。
4. `Delete` と入力して削除を確認し、削除 を選択します。

レシピを削除します。

1. アカウントで作成されたレシピのリストを見るには、ナビゲーションペインからイメージレシピを選択します。
2. レシピ名の横にあるチェックボックスを選択して、削除するレシピを選択します。
3. イメージレシピパネルの上部にある「アクション」メニューで、「レシピを削除」を選択します。
4. Delete と入力して削除を確認し、削除 を選択します。

インフラストラクチャ設定を削除します。

1. アカウントのインフラストラクチャー設定リソースのリストを表示するには、ナビゲーションペインから [インフラストラクチャー設定] を選択します。
2. 構成名の横にあるチェックボックスを選択して、削除するインフラストラクチャー構成を選択します。
3. 「インフラストラクチャー設定」パネルの上部にある「削除」を選択します。
4. Delete と入力して削除を確認し、削除 を選択します。

ディストリビューション設定

1. アカウントで作成された配布設定のリストを表示するには、ナビゲーションペインから [配布設定] を選択します。
2. 設定名の横にあるチェックボックスをオンにして、このチュートリアル用に作成したディストリビューション設定を選択します。
3. ディストリビューション設定パネルの上部で、「削除」を選択します。
4. Delete と入力して削除を確認し、削除 を選択します。

IPAM を削除します。

以下の手順に従って、チュートリアルパイプラインから作成されたイメージをすべて削除したことを確認します。このチュートリアルでは、ビルドスケジュールに従って実行するパイプラインを作成して十分な時間が経過しない限り、イメージは作成されない可能性があります。

1. 自分のアカウントで作成されたイメージの一覧を表示するには、ナビゲーションペインからイメージを選択します。

2. 削除するイメージのバージョンを選択します。これにより、「イメージビルドバージョン」ページが開きます。
3. 削除したいイメージのバージョンの横にあるチェックボックスを選択します。一度に複数のイメージバージョンを選択することもできます。
4. 「イメージビルドバージョン」パネルの上部にある「バージョンを削除」を選択します。
5. `Delete` と入力して削除を確認し、削除 を選択します。

EC2 Image Builder コンソールウィザードを使用してコンテナイメージパイプラインを作成します。

このチュートリアルでは、イメージパイプラインを作成コンソールウィザードを使用してカスタマイズされた EC2 Image Builder Docker イメージを構築および管理するための自動パイプラインを作成する方法について説明します。ステップを効率的に進めるために、使用可能な場合はデフォルトの設定が使用され、オプションのセクションは省略されます。

イメージパイプラインワークフローを作成します。

- ステップ 1: パイプラインの詳細を指定する
- ステップ 2: レシピを選択する
- ステップ 3: インフラストラクチャー設定を定義する (オプション)
- ステップ 4: ディストリビューション設定を定義する (オプション)
- ステップ 5: 確認
- ステップ 6: クリーンアップする

ステップ 1: パイプラインの詳細を指定する

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. パイプラインの作成を開始するには、[イメージパイプラインの作成] を選択します。
3. ーセクションに、パイプライン名 (必須) を入力します。
4. ビルドスケジュールセクションでは、スケジュールオプションはデフォルトのままでかまいません。デフォルトのスケジュールに表示される タイムゾーン は協定世界時 (UTC) であることに注意してください。UTC 時間の詳細とタイムゾーンのオフセットについては、「タイムゾーンの略語 — ワールドワイドリスト」を参照してください。

依存関係の更新設定については、依存関係の更新がある場合はスケジュールされた時間にパイプラインを実行するオプションを選択します。この設定により、パイプラインはビルドを開始する前に更新をチェックします。更新がない場合は、スケジュールされたパイプラインビルドはスキップされます。

> **Note**
>
> パイプラインが依存関係の更新とビルドを想定どおりに認識できるようにするには、ベースイメージとコンポーネントにセマンティックバージョニング (x.x.x) を使用する必要があります。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。

5. 次へ を選択して次のステップに進みます。

ステップ 2: レシピを選択する

1. Image Builder は、デフォルトでレシピセクションの既存のレシピを使用に設定されています。初めて使用する場合は、「新規レシピを作成」オプションを選択します。
2. イメージタイプ セクションで Docker イメージ オプションを選択し、Docker イメージを生成してターゲットリージョンの Amazon ECR リポジトリに配布するコンテナパイプラインを作成します。
3. [キュー] セクションに、以下の情報を入力します。
   - 名前 — レシピ名
   - バージョン - レシピのバージョン (<major>.<minor>.<patch>のフォーマットで、メジャー、マイナー、パッチは整数値です) 。通常、新しいレシピは `1.0.0` で始まります。
4. ソースイメージセクションでは、イメージを選択、イメージオペレーティングシステム (OS)、およびイメージオリジンをデフォルト値のままにします。これにより、Amazon が管理する Amazon Linux 2 コンテナイメージのリストが作成され、ベースイメージとして選択できます。
   a. [イメージ名] ドロップダウンから、イメージを選択します。
   b. 自動バージョニング管理オプション[はデフォルトのままにします (利用可能な最新の OS バージョンを使用)。

> **Note**
>
> この設定により、パイプラインがベースイメージのセマンティックバージョニングを使用して、自動的にスケジュールされたジョブの依存関係の更新を検出できるようになります。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。

5. コンポーネントセクションでは、少なくとも 1 つのビルドコンポーネントを選択する必要があります。

   「ビルドコンポーネント — Amazon Linux」パネルでは、ページに表示されているコンポーネントをブラウズできます。右上隅のページネーションコントロールを使用して、ベースイメージ OS で使用できるその他のコンポーネント間を移動できます。特定のコンポーネントを検索したり、Component Manager を使用して独自のビルドコンポーネントを作成したりすることもできます。

   このチュートリアルでは、Linux を最新のセキュリティアップデートで更新するコンポーネントを次のように選択します。

   a. パネルの上部にある検索バーに単語 `update` を入力して結果を絞り込みます。

   b. `update-linux` ビルドコンポーネントのチェックボックスをオンにします。

   c. 下にスクロールして、「選択したコンポーネント」リストの右上隅にある「すべて展開」を選択します。

   d. バージョニング管理 オプションはデフォルトのままにします([利用可能な最新のコンポーネントバージョンを使用])。

   > **Note**
   >
   > この設定により、パイプラインがベースイメージのセマンティックバージョニングを使用して、自動的にスケジュールされたジョブの依存関係の更新を検出できるようになります。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。

   入力パラメータのあるコンポーネントを選択した場合は、この領域にもパラメータが表示されます。パラメーターについては、このチュートリアルでは説明しません。コンポーネン

トでの入力パラメータの使用とレシピでの設定の詳細については、「EC2 Image Builder で AWSTOE コンポーネントパラメータを管理する」を参照してください。

コンポーネントの順序変更 (オプション)

イメージに含めるコンポーネントを複数選択した場合は、 drag-and-drop アクションを使用して、ビルドプロセス中に実行する順序に再配置できます。

> **Note**
>
> CIS 強化コンポーネントは、Image Builder レシピの標準コンポーネント順序ルールに従っていません。CIS 強化コンポーネントは常に最後に実行され、ベンチマークテストが出力イメージに対して確実に実行されます。

1. スクロールして使用可能なコンポーネントのリストに戻ります。
2. `update-linux-kernel-mainline` ビルドコンポーネント (または任意の他のコンポーネント) のチェックボックスを選択します。
3. 「選択したコンポーネント」リストまでスクロールして、少なくとも 2 つの結果が表示されることを確認します。
4. 新しく追加されたコンポーネントのバージョニング管理が拡張されていない可能性があります。バージョニング管理オプションを拡張するには、バージョニング管理オプションの横にある矢印を選択するか、すべて展開スイッチをオフにしてからオンに切り替えて、選択したすべてのコンポーネントのバージョニング管理を拡張できます。
5. コンポーネントの 1 つを選択し、それを上下にドラッグしてコンポーネントの実行順序を変更します。
6. `update-linux-kernel-mainline` コンポーネントを削除するには、X コンポーネントボックスの右上隅からを選択します。
7. 前のステップを繰り返して、追加した可能性のある他のコンポーネントをすべて削除し、その `update-linux` コンポーネントのみを選択したままにします。

6. Dockerfile テンプレート セクションで、サンプルを使用するオプションを選択します。コンテンツパネルで、Image Builder がコンテナイメージのレシピに基づいてビルド情報やスクリプトを配置するコンテキスト変数に注目してください。

   デフォルトでは、Image Builder は Dockerfile で次のコンテキスト変数を使用します。

parentImage (必須)

ビルド時に、この変数は recipe のベースイメージに解決されます。

例：

```
FROM
{{{ imagebuilder:parentImage }}}
```

環境 (コンポーネントが指定されている場合は必須)

この変数は、コンポーネントを実行するスクリプトに解決されます。

例：

```
{{{ imagebuilder:environments }}}
```

コンポーネント (オプション)

Image Builder は、コンテナレシピに含まれるコンポーネントのビルドコンポーネントスクリプトとテストコンポーネントスクリプトを解決します。この変数は、Dockerfile の環境変数の後に配置できます。

例：

```
{{{ imagebuilder:components }}}
```

7. ターゲットリポジトリ セクションで、このチュートリアルの前提条件として作成した Amazon ECR リポジトリの名前を指定します。このリポジトリは、パイプラインが実行されるリージョン (リージョン 1) のディストリビューション設定のデフォルト設定として使用されます。

> **Note**
>
> ターゲットリポジトリは、配信前にすべてのターゲットリージョンの Amazon ECR に存在している必要があります。

8. 次へ を選択して次のステップに進みます。

ステップ 3: インフラストラクチャー設定を定義する (オプション)

Image Builder はアカウントで EC2 インスタンスを起動して、イメージをカスタマイズし、検証テストを実行します。インフラストラクチャ設定では、ビルドプロセス AWS アカウント 中に で実行されるインスタンスのインフラストラクチャの詳細を指定します。

[インフラストラクチャー設定]セクションでは、[設定]オプションのデフォルトは `Create infrastructure configuration using service defaults` です。これにより、ビルドインスタンスがコンテナイメージを設定するために使用する IAM ロールと関連するインスタンスプロファイルが作成されます。独自のカスタムインフラストラクチャ設定を作成したり、すでに作成した設定を使用することもできます。インフラストラクチャ設定の詳細については、EC2 Image Builder API リファレンスCreateInfrastructureConfigurationの「」を参照してください。

このチュートリアルでは、デフォルト設定を使用します。

- 次へ を選択して次のステップに進みます。

ステップ 4: ディストリビューション設定を定義する (オプション)

ディストリビューション設定は、ターゲットリージョンとターゲット Amazon ECR リポジトリ名で構成されます。出力 Docker イメージは、各リージョンの指定された Amazon ECR リポジトリにデプロイされます。

[ディストリビューション設定]セクションの[設定オプション]はデフォルトで `Create distribution settings using service defaults` です。このオプションでは、パイプラインが稼働するリージョン (リージョン 1) のコンテナレシピで指定されている Amazon ECR リポジトリに出力 Docker イメージを配信します。`Create new distribution settings` を選択した場合、現在のリージョンの ECR リポジトリをオーバーライドして、配信するリージョンをさらに追加できます。

このチュートリアルでは、デフォルト設定を使用します。

- 次へ を選択して次のステップに進みます。

ステップ 5: 確認

レビューセクションには、設定したすべての設定が表示されます。特定のセクションの情報を編集するには、ステップセクションの右上隅にある「編集」ボタンを選択します。たとえば、パイプライン

名を変更する場合は、「ステップ 1: パイプラインの詳細」セクションの右上隅にある「編集」ボタンを選択します。

1. 設定を確認したら、[パイプラインを作成] を選択してパイプラインを作成します。
2. ディストリビューション設定、インフラストラクチャー設定、新しいレシピ、パイプライン用のリソースが作成されると、ページ上部に成功または失敗のメッセージが表示されます。リソース識別子を含むリソースの詳細を表示するには、[詳細を表示] を選択します。
3. リソースの詳細を確認したら、ナビゲーションペインからリソースタイプを選択すると、他のリソースの詳細を表示できます。たとえば、新しいパイプラインの詳細を表示するには、ナビゲーションペインから [Image pipelines] を選択します。ビルドが成功すると、新しいパイプラインが [イメージパイプライン] リストに表示されます。

ステップ 6: クリーンアップする

Image Builder 環境は、ご自宅と同様、必要なものを見つけて散らかることなくタスクを完了できるように、定期的なメンテナンスが必要です。テスト用に作成した一時リソースは定期的にクリーンアップしてください。そうしないと、それらのリソースのことを忘れてしまい、後でそのリソースが何に使用されたかを思い出せなくなる可能性があります。その時までには、それらを安全に取り除くことができるかどうかがはっきりしないかもしれません。

> **Tip**
>
> リソースを削除するときの依存関係エラーを防ぐため、必ず次の順序でリソースを削除してください。
>
> 1. イメージパイプライン
> 2. イメージのレシピ
> 3. 残っているすべてのリソース

このチュートリアルのために作成したリソースをクリーンアップするには、以下の手順に従ってください：

パイプラインを削除します。

1. アカウントで作成されたビルドパイプラインのリストを表示するには、ナビゲーションペインから [Image pipelines] を選択します。

2. パイプライン名 の横にあるチェックボックスをオンにして、削除するパイプラインを選択します。
3. イメージパイプラインパネルの上部にある「アクション」メニューで、「削除」を選択します。
4. Delete と入力して削除を確認し、削除 を選択します。

コンテナレシピを削除します。

1. アカウントで作成されたコンテナレシピのリストを表示するには、ナビゲーションペインからコンテナレシピ を選択します。
2. レシピ名の横にあるチェックボックスを選択して、削除するレシピを選択します。
3. コンテナレシピ パネルの上部にあるアクションメニューで、レシピを削除を選択します。
4. Delete と入力して削除を確認し、削除 を選択します。

インフラストラクチャ設定を削除します。

1. アカウントのインフラストラクチャー設定リソースのリストを表示するには、ナビゲーションペインから [インフラストラクチャー設定] を選択します。
2. 構成名の横にあるチェックボックスを選択して、削除するインフラストラクチャー構成を選択します。
3. 「インフラストラクチャー設定」パネルの上部にある「削除」を選択します。
4. Delete と入力して削除を確認し、削除 を選択します。

ディストリビューション設定

1. アカウントで作成された配布設定のリストを表示するには、ナビゲーションペインから [配布設定] を選択します。
2. 設定名の横にあるチェックボックスをオンにして、このチュートリアル用に作成したディストリビューション設定を選択します。
3. ディストリビューション設定パネルの上部で、「削除」を選択します。
4. Delete と入力して削除を確認し、削除 を選択します。

IPAM を削除します。

以下の手順に従って、チュートリアルパイプラインから作成されたイメージをすべて削除したことを確認します。このチュートリアルでは、ビルドスケジュールに従って実行するパイプラインを作成して十分な時間が経過しない限り、イメージは作成されない可能性があります。

1. 自分のアカウントで作成されたイメージの一覧を表示するには、ナビゲーションペインからイメージを選択します。
2. 削除するイメージのバージョンを選択します。これにより、「イメージビルドバージョン」ページが開きます。
3. 削除したいイメージのバージョンの横にあるチェックボックスを選択します。一度に複数のイメージバージョンを選択することもできます。
4. 「イメージビルドバージョン」パネルの上部にある「バージョンを削除」を選択します。
5. `Delete` と入力して削除を確認し、削除 を選択します。

AWS Task Orchestrator and Executor コンポーネントマネージャー

EC2 Image Builder は AWS Task Orchestrator and Executor 、 (AWSTOE) アプリケーションを使用して、複雑なワークフローをオーケストレーションし、システム設定を変更し、コードを記述せずにシステムをテストします。このアプリケーションは、宣言型ドキュメントスキーマを実装するコンポーネントを管理および実行します。

スタンドアロンアプリケーションなので、追加のサーバー設定は不要です。どのクラウドインフラストラクチャーでもオンプレミスでも実行できます。

内容

AWSTOE ダウンロード

をインストールするには AWSTOE、アーキテクチャとプラットフォームのダウンロードリンクを選択します。サービスの VPC エンドポイント (Image Builder など) にアタッチする場合は、 AWSTOE ダウンロード用の S3 バケットへのアクセスを含むカスタムエンドポイントポリシーがアタッチされている必要があります。そうしないと、ビルドインスタンスとテストインスタンスはブートストラップスクリプト (`bootstrap.sh`) をダウンロードして AWSTOE アプリケーションをインストールできなくなります。詳細については、Image Builder用VPCエンドポイントポリシーの作成を参照してください。

> **Important**
>
> AWS は、TLS バージョン 1.0 および 1.1 のサポートを段階的に廃止しています。 AWSTOE ダウンロードのために S3 バケットにアクセスするには、クライアントソフトウェアで TLS バージョン 1.2 以降を使用する必要があります。詳細については、AWS セキュリティのブログ記事を参照してください。

アーキテクチャ	プラットフォーム	ダウンロードリンク	例
386	AL 2と2023年 RHEL 7 および 8 Ubuntu 16.04、18.04、20.04、22.04 CentOS 7 および 8 スーズ 12 と 15	`https://awstoe-<region>.s3.com/latest/linux/386/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/386/awstoe
AMD64	Windows Server 2012 R2、2016、2019、2022年	`https://awstoe-<region>.s3.com/latest/windows/amd64/awstoe.exe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/windows/amd64/awstoe.exe
AMD64	AL 2と2023年 RHEL 7 および 8 Ubuntu 16.04、18.04、20.04、22.04 CentOS 7 および 8 CentOS Stream 8 スーズ 12 と 15	`https://awstoe-<region>.s3.com/latest/linux/amd64/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/amd64/awstoe

アーキテクチャ	プラットフォーム	ダウンロードリンク	例
ARM64	AL 2と2023年 RHEL 7 および 8 Ubuntu 16.04、18.04、20.04、22.04 CentOS 7 および 8 CentOS Stream 8 スーズ 12 と 15	`https://awstoe- <region>.s3.com/latest/linux/arm64/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/arm64/awstoe

サポートされるリージョン

AWSTOE は、以下のリージョンでスタンドアロンアプリケーションとしてサポートされています。

AWS リージョン 名前	AWS リージョン
米国東部 (オハイオ)	us-east-2
米国東部 (バージニア北部)	us-east-1
AWS GovCloud （米国東部）	us-gov-east-1
AWS GovCloud （米国西部）	us-gov-west-1
米国西部 (北カリフォルニア)	us-west-1
米国西部 (オレゴン)	us-west-2
アフリカ (ケープタウン)	af-south-1
アジアパシフィック (香港)	ap-east-1
アジアパシフィック (大阪)	ap-northeast-3
アジアパシフィック (ソウル)	ap-northeast-2

AWS リージョン 名前	AWS リージョン
アジアパシフィック (ムンバイ)	ap-south-1
アジアパシフィック (ハイデラバード)	ap-south-2
アジアパシフィック (シンガポール)	ap-southeast-1
アジアパシフィック (シドニー)	ap-southeast-2
アジアパシフィック (ジャカルタ)	ap-southeast-3
アジアパシフィック (東京)	ap-northeast-1
カナダ (中部)	ca-central-1
欧州 (フランクフルト)	eu-central-1
欧州 (チューリッヒ)	eu-central-2
欧州 (ストックホルム)	eu-north-1
欧州 (ミラノ)	eu-south-1
欧州 (スペイン)	eu-south-2
欧州 (アイルランド)	eu-west-1
欧州 (ロンドン)	eu-west-2
欧州 (パリ)	eu-west-3
イスラエル (テルアビブ)	il-central-1
中東 (アラブ首長国連邦)	me-central-1
中東 (バーレーン)	me-south-1
南米 (サンパウロ)	sa-east-1
中国 (北京)	cn-north-1

AWS リージョン 名前	AWS リージョン
中国 (寧夏)	cn-northwest-1

の使用を開始する AWSTOE

AWS Task Orchestrator and Executor （ AWSTOE) アプリケーションは、コンポーネント定義フレームワーク内でコマンドを作成、検証、実行するスタンドアロンアプリケーションです。 AWS のサービスは、 AWSTOE を使用してワークフローのオーケストレーション、ソフトウェアのインストール、システム設定の変更、イメージビルドのテストを行うことができます。

AWSTOE アプリケーションをインストールして初めて使用するには、次の手順に従います。

AWSTOE インストールダウンロードの署名を確認する

このセクションでは、Linux および Windows ベースのオペレーティングシステム AWSTOE での のインストールダウンロードの有効性を検証するための推奨プロセスについて説明します。

トピック

- Linux での AWSTOE インストールダウンロードの署名の検証
- Windows での AWSTOE インストールダウンロードの署名の検証

Linux での AWSTOE インストールダウンロードの署名の検証

このトピックでは、Linux ベースのオペレーティングシステム AWSTOE での のインストールダウンロードの有効性を検証するための推奨プロセスについて説明します。

インターネットからアプリケーションをダウンロードする際は、必ずソフトウェアパブリッシャーの身元を認証することをお勧めします。また、アプリケーションが公開されてから変更または破損していないことを確認してください。これにより、ウイルスやマルウェアに感染したバージョンのアプリケーションをインストールせずに済みます。

このトピックのステップを実行した後に AWSTOE のソフトウェアが変更または破損していることが判明した場合は、インストールファイルを実行しないでください。代わりに、 AWS Support にお問い合わせください。サポートオプションの詳細については、「」を参照してくださいAWS Support。

AWSTOE Linux ベースのオペレーティングシステム用の ファイルはGnuPG、安全なデジタル署名のための Pretty Good Privacy (OpenPGP) 標準のオープンソース実装である を使用して署名されます。 GnuPG (とも呼ばれますGPG) は、デジタル署名を通じて認証と整合性チェックを提供します。Amazon EC2 は、ダウンロードした Amazon EC2 CLI ツールの検証に使用できるパブリックキーと署名を公開します。PGP と GnuPG (GPG) の詳細については、http://www.gnupg.org を参照してください。

まず、ソフトウェア発行元との信頼を確立します。ソフトウェア発行元のパブリックキーをダウンロードし、キー所有者が一致していることを確認してから、キーリングに追加します。キーリングとは、既知のパブリックキーの集合です。真正性が確立されたパブリック キーは、アプリケーションの署名を確認するために使用できます。

トピック

- GPG ツールのインストール
- パブリック キーの認証とインポート
- パッケージの署名の確認

GPG ツールのインストール

お使いのオペレーティングシステムが Linux または Unix の場合、GPG ツールが既にインストールされている場合があります。システムにツールがインストール済みかどうかをテストするには、コマンドラインプロンプトで gpg を入力します。GPG ツールがインストールされている場合、GPG のコマンドプロンプトが表示されます。GPG ツールがインストールされていない場合、コマンドが見つからないというエラーが表示されます。GnuPG パッケージはリポジトリからインストールできます。

Debian ベースの Linux に GPG ツールをインストールするには

- ターミナルから、次のコマンド apt-get install gnupg を実行します。

Red Hat ベースの Linux に GPG ツールをインストールするには

- ターミナルから、次のコマンド yum install gnupg を実行します。

パブリック キーの認証とインポート

プロセスの次のステップでは、 AWSTOE パブリックキーを認証し、信頼されたキーとしてGPGキーリングに追加します。

AWSTOE パブリックキーを認証してインポートするには

1. 次のいずれかを実行してパブリック GPG ビルドキーのコピーを取得します。

   - https://awstoe-**<region>**.s3.**<region>**.amazonaws.com/assets/awstoe.gpgからキーをダウンロードしてください。例えば https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/assets/awstoe.gpg です。
   - 次のテキストからキーをコピーし、[awstoe.gpg] という名前のファイルに貼り付けます。必ず次のすべてが含まれるようにしてください。

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2

mQENBF8UqwsBCACdiRF2bkZYaFSDPFC+LIkWLwFvtUCRwAHtD8KIwTJ6LVn3fHAU
GhuK0ZH9mRrqRT2bq/xJjGsnF9VqTj2AJqndGJdDjz75YCZYM+ocZ+r5HSJaeW9i
S5dykHj7Txti2zHe0G5+W0v7v5bPi2sPHsN7XWQ7+G2AMEPTz8PjxY//I0DvMQns
Sle3l9hz6wCClz1l9LbBzTyHfSm5ucTXvNe88XX5Gmt37OCDM7vfli0Ctv8WFoLN
6jbxuA/sV71yIkPm9IYp3+GvaKeT870+sn8/JOOKE/U4sJV1ppbqmuUzDfhrZUaw
8eW8IN9A1FTIuWiZED/5L83UZuQs1S7s2PjlABEBAAG0GkFXU1RPRSA8YXdzdG9l
QGFtYXpvbi5jb20+iQE5BBMBCAAjBQJfFKsLAhsDBwsJCAcDAgEGFQgCCQoLBBYC
AwECHgECF4AACgkQ3r3BVvWuvFJGiwf9EVmrBR77+Qe/DUeXZJYoaFr7If/fVDZl
6V3TC6p0J0Veme7uXleRUTFOjzbh+7e5sDX19HrnPquzCnzfMiqbp4lSoeUuNdOf
FcpuTCQH+M+sIEIgPno4PLl0Uj2uE1o++mxmonBl/Krk+hly8hB2L/9n/vW3L7BN
OMb1Ll9PmgGPbWipcT8KRdz4SUex9TXGYzjlWb3jU3uXetdaQY1M3kVKE1siRsRN
YYDtpcjmwbhjpu4xm19aFqNoAHCDctEsXJA/mkU3erwIRocPyjAZE2dnlkL9ZkFZ
z9DQkcIarbCnybDM5lemBbdhXJ6hezJE/b17VA0t1fY04MoEkn6oJg==
=oyze
-----END PGP PUBLIC KEY BLOCK-----
```

2. awstoe.gpg を保存したディレクトリのコマンドプロンプトで、次のコマンドを使用して AWSTOE パブリックキーをキーリングにインポートします。

```
gpg --import awstoe.gpg
```

   コマンドで次のような結果が返されます。

```
gpg: key F5AEBC52: public key "AWSTOE <awstoe@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

次のステップで必要になるため、キーの値を書きとめておきます。前述の例では、キーの値は F5AEBC52 です。

3. 次のコマンドを使用してフィンガープリントを確認し、キー値を前述の手順の値と置き換えます。

```
gpg --fingerprint key-value
```

このコマンドで次のような結果が返されます。

```
pub   2048R/F5AEBC52 2020-07-19
      Key fingerprint = F6DD E01C 869F D639 15E5  5742 DEBD C156 F5AE BC52
uid       [ unknown] AWSTOE <awstoe@amazon.com>
```

さらに、前述の例のように、フィンガープリント文字列は「F6DD E01C 869F D639 15E5 5742 DEBD C156 F5AE BC52」になります。返されたキー フィンガープリントをこのページで公開されているものと比較します。これらは一致するはずです。一致しない場合は、 AWSTOE インストールスクリプトをインストールせず、 にお問い合わせください AWS Support。

パッケージの署名の確認

GPG ツールをインストール後、 AWSTOE パブリックキーを認証してインポートし、そのパブリックキーが信頼済みであることを確認すると、インストールスクリプトの署名を確認できるようになります。

インストールスクリプトの署名を確認するには

1. コマンドプロンプトで次のコマンドを実行し、アプリケーションバイナリをダウンロードします。

```
curl -O https://awstoe-<region>.s3.<region>.amazonaws.com/latest/
linux/<architecture>/awstoe
```

例:

```
curl -O https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/amd64/
awstoe
```

architectureでサポートされる値は amd64、 386、および arm64 です。

2. コマンドプロンプトで次のコマンドを実行し、同じ S3 key prefix パスから対応するアプリケーションバイナリの署名ファイルをダウンロードします。

```
curl -O https://awstoe-<region>.s3.<region>.amazonaws.com/latest/
linux/<architecture>/awstoe.sig
```

例:

```
curl -O https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/amd64/
awstoe.sig
```

architectureでサポートされる値は amd64、 386、および arm64 です。

3. 保存したディレクトリのコマンドプロンプトawstoe.sigと AWSTOE インストールファイルで次のコマンドを実行して、署名を検証します。ファイルが2つとも存在している必要があります。

```
gpg --verify ./awstoe.sig ~/awstoe
```

出力は次のようになります。

```
gpg: Signature made Mon 20 Jul 2020 08:54:55 AM IST using RSA key ID F5AEBC52
gpg: Good signature from "AWSTOE awstoe@amazon.com" [unknown]
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: F6DD E01C 869F D639 15E5 5742 DEBD C156 F5AE BC52
```

出力に「Good signature from "AWSTOE <awstoe@amazon.com>"」という句が含まれる場合は、署名が正常に確認されており、 AWSTOE のインストールスクリプトを実行できることを意味しています。

出力結果に「BAD signature」という句が含まれる場合、手順が正しいことをもう一度確認してください。この応答が続く場合は、 AWS Supportに連絡してください。以前にダウンロードしたインストール ファイルを実行しないでください。

以下は、表示される可能性のある警告の詳細です。

- 警告: このキーは、信頼済みの署名で認定されていません! 署名が所有者に属していることが確認できません。理想的には、 AWS オフィスを訪問し、直接キーを受け取ることになります。しかし、キーは多くの場合ウェブサイトからダウンロードされます。この場合、ウェブサイトは AWS ウェブサイトです。
- gpg: 最終的に信用されたキーが見つかりません。これは、特定のキーがユーザー (またはユーザーが信頼する他のユーザー) によって「最終的に信頼された」キーでないことを意味します。

詳細については、「http://www.gnupg.org」を参照してください。

Windows での AWSTOE インストールダウンロードの署名の検証

このトピックでは、Windows ベースのオペレーティングシステム上の AWS Task Orchestrator and Executor アプリケーションのインストールファイルの有効性を検証するための推奨プロセスについて説明します。

インターネットからアプリケーションをダウンロードする場合は、常にソフトウェア発行元のアイデンティティを認証し、アプリケーションの発行後に改ざん、あるいは破損がないか確認することをお勧めします。これにより、ウイルスやマルウェアに感染したバージョンのアプリケーションをインストールせずに済みます。

このトピックのステップを実行した後に AWSTOE のソフトウェアが変更または破損していることが判明した場合は、インストールファイルを実行しないでください。代わりに、 にお問い合わせください AWS Support。

Windowsベースのオペレーティングシステムでダウンロードしたawstoeバイナリの有効性を確認するには、Amazon Services LLCの署名者証明書のサムプリントがこの値と等しいことを確認してください：

F8 83 11 EE F0 4A A2 91 E3 79 21 BA 6B FC AF F8 19 92 12 D7

> **Note**
>
> 新しいバイナリのロールアウトウィンドウ中に、署名者証明書が新しいサムプリントと一致しない場合があります。署名者証明書が一致しない場合は、サムプリントの値が次のものであることを確認します。
> 5B 77 F4 F0 C3 7A 8B 89 D9 A7 8F 54 B6 85 11 CE 9E A3 BF 17

この値を検証するには、以下の手順を実行します。

1. ダウンロードした `awstoe.exe` を右クリックして、プロパティウィンドウを開きます。
2. デジタル署名タブを選択します。
3. [Signature List] で [Amazon Services LLC] を選択し、[Details] をクリックします。
4. すでに選択していない場合は 一般タブにアクセスし、証明書の表示 を選びます。
5. 詳細 タブを選択し、まだの場合は すべてを表示 のドロップダウンリストで選択します。
6. 拇印 フィールドが表示されるまでスクロールして、拇印 を選択します。下のウィンドウにサムプリントの値全体が表示されます。

   - 下のウィンドウのサムプリントの値が次の値と等しい場合、

     F8 83 11 EE F0 4A A2 91 E3 79 21 BA 6B FC AF F8 19 92 12 D7

     ダウンロードした AWSTOE バイナリは本物であり、安全にインストールできます。

     > **Note**
     >
     > 新しいバイナリのロールアウトウィンドウ中に、署名者証明書が新しいサムプリントと一致しない場合があります。署名者証明書が一致しない場合は、サムプリントの値が次のものであることを確認します。
     > 5B 77 F4 F0 C3 7A 8B 89 D9 A7 8F 54 B6 85 11 CE 9E A3 BF 17

   - 下部の詳細ウィンドウのサムプリントの値が上記の値と等しくない場合には、`awstoe.exe` を実行しないでください。

使用を開始する手順

- ステップ 1: をインストールする AWSTOE
- ステップ 2: AWS 認証情報を設定する
- ステップ 3: コンポーネントドキュメントをローカルで開発する
- ステップ 4: AWSTOE コンポーネントを検証する
- ステップ 5: AWSTOE コンポーネントを実行する

ステップ 1: をインストールする AWSTOE

コンポーネントをローカルで開発するには、 AWSTOE アプリケーションをダウンロードしてインストールします。

1. AWSTOE アプリケーションをダウンロードする

   をインストールするには AWSTOE、アーキテクチャとプラットフォームに適したダウンロードリンクを選択します。アプリケーションのダウンロードリンクの詳細なリストについては、「AWSTOE ダウンロード」を参照してください。

   > **Important**
   >
   > AWS は、TLS バージョン 1.0 および 1.1 のサポートを段階的に廃止しています。AWSTOE ダウンロードのために S3 バケットにアクセスするには、クライアントソフトウェアで TLS バージョン 1.2 以降を使用する必要があります。詳細については、AWS セキュリティのブログ記事を参照してください。

2. 署名を検証する

   ダウンロードを確認する手順は、インストール後に AWSTOE アプリケーションを実行するサーバープラットフォームによって異なります。Linux サーバでダウンロードを確認する方法については、「Linux で署名を検証する」を参照してください。Windows サーバでダウンロードを確認する方法については、「Windows で署名を検証する」を参照してください。

> **Important**
>
> AWSTOE は、ダウンロード場所から直接呼び出されます。別途インストールを行う必要はありません。つまり、 はローカル環境を変更 AWSTOE することもできます。コンポーネント開発中に変更を確実に分離するには、EC2 インスタンスを使用して AWSTOE コンポーネントを開発およびテストすることをお勧めします。

ステップ 2: AWS 認証情報を設定する

AWSTOE では AWS のサービス、次のようなタスクを実行するときに CloudWatch、Amazon S3 や Amazon などの他の に接続するための AWS 認証情報が必要です。

- ユーザー提供の Amazon S3 パスから AWSTOE ドキュメントをダウンロードする。
- `S3Download` または `S3Upload` アクションモジュールを実行する。
- 有効にすると CloudWatch、 にログをストリーミングします。

EC2 インスタンス AWSTOE で を実行している場合、 を実行すると EC2 インスタンスにアタッチされた IAM ロールと同じアクセス許可 AWSTOE が使用されます。

EC2 の IAM ロールの詳細については、「Amazon EC2 向けの IAM ロール」を参照してください。

次の例は、 AWS_ACCESS_KEY_IDおよび AWS_SECRET_ACCESS_KEY環境変数を使用して AWS 認証情報を設定する方法を示しています。

これらの変数を Linux、macOS、または Unix で設定するには、export を使用します。

```
$ export AWS_ACCESS_KEY_ID=your_access_key_id
```

```
$ export AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

を使用して Windows でこれらの変数を設定するには PowerShell、 を使用します$env。

```
C:\> $env:AWS_ACCESS_KEY_ID=your_access_key_id
```

```
C:\> $env:AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

Windowsでコマンドプロンプトを使ってこれらの変数を設定するには、setを使う。

```
C:\> set AWS_ACCESS_KEY_ID=your_access_key_id
```

```
C:\> set AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

ステップ 3: コンポーネントドキュメントをローカルで開発する

AWSTOE コンポーネントはプレーンテキストの YAML ドキュメントで作成されます。ドキュメントの構文についてはでコンポーネントドキュメントを使用する AWSTOEを参照。

以下は、ドキュメントをローカルで開発する際に使用できる Hello World コンポーネントドキュメントの例です。

hello-world-windows.yml.

```
name: Hello World
description: This is Hello World testing document for Windows.
```

```
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host 'Hello World from the build phase.'
  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host 'Hello World from the validate phase.'
  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host 'Hello World from the test phase.'
```

`hello-world-linux.yml`.

```
name: Hello World
description: This is hello world testing document for Linux.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the build phase.'
  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
```

```
            - echo 'Hello World from the validate phase.'
  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo 'Hello World from the test phase.'
```

ステップ 4: AWSTOE コンポーネントを検証する

AWSTOE コンポーネントの構文は、アプリケーションでローカルで AWSTOE 検証できます。次の例は、コンポーネントを実行せずに構文を検証するための AWSTOE アプリケーションvalidateコマンドを示しています。

> **Note**
>
> AWSTOE アプリケーションは、現在のオペレーティングシステムのコンポーネント構文のみを検証できます。たとえば、awstoe.exe を Windows で実行している場合、ExecuteBash アクションモジュールを使用する Linux ドキュメントの構文は検証できません。

Windows

```
C:\> awstoe.exe validate --documents C:\Users\user\Documents\hello-world.yml
```

Linux

```
$ awstoe validate --documents /home/user/hello-world.yml
```

ステップ 5: AWSTOE コンポーネントを実行する

AWSTOE アプリケーションは、--phasesコマンドライン引数を使用して、指定されたドキュメントの 1 つ以上のフェーズを実行できます。--phasesでサポートされている値はbuild、validate、testです。複数のフェーズ値をカンマで区切って入力できます。

フェーズのリストを指定すると、 AWSTOE アプリケーションは各ドキュメントの指定されたフェーズを順番に実行します。例えば、 は の buildおよび validateフェーズ AWSTOE

を実行しdocument1.yaml、その後に の buildおよび validateフェーズを実行しますdocument2.yaml。

ログを安全に保存し、トラブルシューティングのために保持するには、Amazon S3 にログストレージを設定することをお勧めします。Image Builder では、ログを発行するための Amazon S3 の場所はインフラストラクチャ設定で指定されます。インフラ構成の詳細については、EC2 Image Builder インフラストラクチャ設定の管理を参照。

フェーズのリストが指定されていない場合、 AWSTOE アプリケーションは YAML ドキュメントに記載されている順序ですべてのフェーズを実行します。

1 つまたは複数のドキュメントで特定のフェーズを実行するには、以下のコマンドを使用します。

単一フェーズ

```
awstoe run --documents hello-world.yml --phases build
```

複数フェーズ

```
awstoe run --documents hello-world.yml --phases build,test
```

ドキュメント実行

1 つのドキュメントですべてのフェーズを実行

```
awstoe run --documents documentName.yaml
```

複数のドキュメントで全フェーズを実行

```
awstoe run --documents documentName1.yaml,documentName2.yaml
```

Amazon S3 情報を入力して、ユーザー定義のローカルパスから AWSTOE ログをアップロードする (推奨)

```
awstoe run --documents documentName.yaml --log-s3-bucket-name <S3Bucket> --log-s3-key-
prefix <S3KeyPrefix> --log-s3-bucket-owner <S3BucketOwner> --log-directory <local_path>
```

1 つのドキュメントですべてのフェーズを実行し、すべてのログをコンソールに表示します。

```
awstoe run --documents documentName.yaml --trace
```

コマンドの例

```
awstoe run --documents s3://bucket/key/doc.yaml --phases build,validate
```

一意の ID でドキュメントを実行

```
awstoe run --documents <documentName>.yaml --execution-id <user provided id> --phases
 <comma separated list of phases>
```

に関するヘルプを入手する AWSTOE

```
awstoe --help
```

でコンポーネントドキュメントを使用する AWSTOE

AWS Task Orchestrator and Executor （ AWSTOE) を使用してコンポーネントを構築するには、作成したコンポーネントに適用されるフェーズとステップを表す YAML ベースのドキュメントを提供する必要があります。コンポーネントは、新しい Amazon マシンイメージ (AMI) またはコンテナイメージを作成するときに AWS のサービス 使用します。

トピック

コンポーネントドキュメントワークフロー

AWSTOE コンポーネントドキュメントでは、フェーズとステップを使用して関連タスクをグループ化し、それらのタスクをコンポーネントの論理ワークフローに整理します。

> **Tip**
>
> コンポーネントを使用してイメージを構築するサービスには、ビルドプロセスにどのフェーズを使用するか、またそれらのフェーズをいつ実行できるかについてのルールが実装されている場合があります。これはコンポーネントを設計する際に考慮すべき重要な点です。

phases

フェーズは、イメージビルドプロセスにおけるワークフローの進行状況を表します。たとえば、Image Builder サービスは、生成するイメージのビルド段階で`build`と`validate`のフェーズを使用します。テスト段階では`test`と`container-host-test`フェーズを使用して、イメージスナップショットまたはコンテナイメージが期待どおりの結果を生成することを確認してから、最終的な AMI を作成するか、コンテナイメージを配布します。

コンポーネントが実行されると、各フェーズの関連コマンドがコンポーネントドキュメントに表示されている順序で適用されます。

フェーズのルール

- フェーズ名はドキュメント内で一意である必要があります。
- 文書には多数のフェーズを定義できます。
- ドキュメントには、次のうち、少なくとも 1 つは指定が必要です。
  - ビルド — Image Builder の場合、このフェーズは通常、ビルド段階で使用されます。
  - 検証 — Image Builder の場合、このフェーズは通常、ビルド段階で使用されます。
  - テスト — Image Builder の場合、このフェーズは通常、テスト段階で使用されます。
- フェーズは、常にドキュメントで定義されている順序で実行されます。で AWSTOE AWS CLI コマンドに指定されている順序は効果がありません。

ステップ

ステップは、各フェーズ内のワークフローを定義する個別の作業単位です。ステップは順番に実行されます。ただし、あるステップのインプットまたはアウトプットを、インプットとして後続のステップに送ることもあります。これをロールの連鎖と呼びます。

ステップのルール

- その名前はボットに対して一意である必要があります。

- ステップでは、終了コードを返すサポートされているアクション (アクションモジュール) を使用する必要があります。

  サポートされているアクションモジュールの全リスト、その仕組み、入出力値、例については、AWSTOE コンポーネントマネージャーがサポートするアクションモジュールを参照してください。

コンポーネントロギング

AWSTOE は、コンポーネントが実行されるたびに、新しいイメージの構築とテストに使用される新しいログフォルダを EC2 インスタンスに作成します。コンテナイメージの場合、ログフォルダはコンテナに保存されます。

イメージ作成プロセス中に問題が発生した場合のトラブルシューティングを支援するために、コンポーネントの実行中に AWSTOE が作成する入力ドキュメントとすべての出力ファイルは ログフォルダに保存されます。

ログフォルダー名は次の部分で構成されています。

1. ログディレクトリ — サービスが AWSTOE コンポーネントを実行すると、コマンドの他の設定とともにログディレクトリに渡されます。以下の例では、Image Builder が使用するログファイル形式を示します。
   - Linux: `/var/lib/amazon/toe/`
   - Windows: `$env:ProgramFiles\Amazon\TaskOrchestratorAndExecutor\`
2. ファイルプレフィックス — これはすべてのコンポーネントに使用される標準のプレフィックスです:"TOE_」。
3. ランタイム — これは YYYY-MM-DD_HH-MM-SS_UTC-0 形式のタイムスタンプです。
4. 実行 ID – これは、 が 1 つ以上のコンポーネント AWSTOE を実行するときに割り当てられる GUID です。

例: */var/lib/amazon/toe/*TOE*_2021-07-01_12-34-56_UTC-0_a1bcd2e3-45f6-789a-bcde-0fa1b2c3def4*

AWSTOE は、次のコアファイルを ログフォルダに保存します。

入力ファイル

- document.yaml — コマンドの入力として使用されるドキュメント。コンポーネントが実行されると、このファイルはアーティファクトとして保存されます。

出力ファイル

- application.log — アプリケーションログには、コンポーネントの実行中に何が起こっているかを示す。 AWSTOE のタイムスタンプ付きのデバッグレベルの情報が含まれます
- detailedoutput.json — この JSON ファイルには、コンポーネントのランタイムに適用されるすべてのドキュメント、フェーズ、ステップの実行ステータス、入力、出力、失敗に関する詳細情報が含まれています。
- console.log – コンソールログには、コンポーネントの実行中にコンソールに AWSTOE 書き込まれるすべての標準出力 (stdout) および標準エラー (stderr) 情報が含まれます。
- chaining.json – この JSON ファイルは、連鎖式を解決するために AWSTOE に適用された最適化を表します。

> **Note**
>
> ログフォルダーには、ここで説明していない他の一時ファイルが含まれている場合もあります。

入力チェーンと出力連鎖

AWSTOE 設定管理アプリケーションは、以下の形式でリファレンスを書き込むことで、入出力を連鎖する機能を提供します。

```
{{ phase_name.step_name.inputs/outputs.variable }}
```

または

```
{{ phase_name.step_name.inputs/outputs[index].variable }}
```

チェーニング特徴量を使うと、コードをリサイクルして文書の保守性を向上させることができます。

チェーニングのルール

- チェーニング式は各ステップの入力セクションでのみ使用できます。

- 連鎖式を含むステートメントは、引用符で囲む必要があります。例:
  - 式が無効です:`echo {{ phase.step.inputs.variable }}`
  - 有効な表現:`"echo {{ phase.step.inputs.variable }}"`
  - 有効な表現:`'echo {{ phase.step.inputs.variable }}'`
- 連鎖式は同じドキュメント内の他のステップやフェーズの変数を参照できます。ただし、呼び出し元のサービスには、連鎖式を 1 つのステージのコンテキスト内でのみ動作させることを要求するルールがある場合があります。たとえば、Image Builder は各ステージを独立して実行するため、ビルドステージからテストステージへのチェーニングをサポートしていません。
- 連鎖式のインデックスは 0 から始まるインデックスに従います。インデックスは最初の要素を参照するゼロ (0) から始まります。

例

次のサンプルステップの 2 番目のエントリでソース変数を参照する場合、チェーンパターンは`{{ build.SampleS3Download.inputs[1].source }}`です。

```
phases:
  -
    name: 'build'
    steps:
      -
        name: SampleS3Download
        action: S3Download
        timeoutSeconds: 60
        onFailure: Abort
        maxAttempts: 3
        inputs:
          -
            source: 's3://sample-bucket/sample1.ps1'
            destination: 'C:\sample1.ps1'
          -
            source: 's3://sample-bucket/sample2.ps1'
            destination: 'C:\sample2.ps1'
```

次のサンプルステップの出力変数 (「Hello」と等しい) を参照する場合の連鎖パターンは`{{ build.SamplePowerShellStep.outputs.stdout }}`です。

```
phases:
  -
```

```
    name: 'build'
    steps:
      -
        name: SamplePowerShellStep
        action: ExecutePowerShell
        timeoutSeconds: 120
        onFailure: Abort
        maxAttempts: 3
        inputs:
          commands:
            - 'Write-Host "Hello"'
```

文書スキーマと定義

ドキュメントの YAML スキーマを次に示します。

```
name: (optional)
description: (optional)
schemaVersion: "string"

phases:
  - name: "string"
    steps:
        - name: "string"
          action: "string"
          timeoutSeconds: integer
          onFailure: "Abort|Continue|Ignore"
          maxAttempts: integer
          inputs:
```

ドキュメントのスキーマ定義は次のとおりです。

フィールド	説明	タイプ	必須
name	文書の名前	文字列	いいえ
説明	文書の説明	文字列	いいえ
schemaVersion	ドキュメントのスキーマバージョン。現在は 1.0。	文字列	はい

フィールド	説明	タイプ	必須
phases	フェーズとそのステップのリスト。	リスト	はい

フェーズのスキーマ定義は次のとおりです。

フィールド	説明	タイプ	必須
name	フェーズの名前	文字列	はい
steps	フェーズ内のステップのリスト。	リスト	はい

ステップのスキーマ定義は次のとおりです。

フィールド	説明	タイプ	必須	デフォルト値
name	ステップのユーザー定義名。	文字列		
アクション	ステップを実行するモジュールに関するキーワード。	文字列		
timeoutSeconds	ステップが失敗または再試行されるまでに実行される秒数。 また、タイムアウトが無限であることを示す -1 値もサポートします。0 やその	整数	いいえ	7,200 秒 (120 分)

フィールド	説明	タイプ	必須	デフォルト値
	他の負の値は使用できません。			

フィールド	説明	タイプ	必須	デフォルト値
onFailure	ステップが失敗した場合は実行する内容を指定します。有効な値は次のとおりです。 • 中止 — 最大回数試行するとステップが失敗し、実行を停止します。フェーズとドキュメントのステータスをFailedに設定します。 • 続行 — 最大回数試行するとステップが失敗し、残りのステップの実行を継続します。フェーズとドキュメントのステータスをFailedに設定します。 • 無視 — 試行に失敗した回数が最大回数を超えた後IgnoredFailure にス	文字列	いいえ	中止

フィールド	説明	タイプ	必須	デフォルト値
	テップを設定し、残りのステップを引き続き実行します。フェーズとドキュメントのステータスをSuccessWithIgnoredFailure に設定します。			
maxAttempts	ステップが失敗するまでに許可される最大試行回数。	整数	いいえ	1
入力	アクションモジュールがステップを実行するのに必要なパラメータが含まれます。	dict	はい	

文書例のスキーマ

以下は、使用可能なすべての Windows アップデートのインストール、設定スクリプトの実行、AMI の作成前の変更の検証、AMI の作成後に変更をテストするためのドキュメントスキーマの例です。

```
name: RunConfig_UpdateWindows
description: 'This document will install all available Windows updates and run a config
 script. It will then validate the changes before an AMI is created. Then after AMI
 creation, it will test all the changes.'
schemaVersion: 1.0
phases:
```

```
- name: build
  steps:
    - name: DownloadConfigScript
      action: S3Download
      timeoutSeconds: 60
      onFailure: Abort
      maxAttempts: 3
      inputs:
        - source: 's3://customer-bucket/config.ps1'
          destination: 'C:\config.ps1'

    - name: RunConfigScript
      action: ExecutePowerShell
      timeoutSeconds: 120
      onFailure: Abort
      maxAttempts: 3
      inputs:
        file: '{{build.DownloadConfigScript.inputs[0].destination}}'

    - name: Cleanup
      action: DeleteFile
      onFailure: Abort
      maxAttempts: 3
      inputs:
        - path: '{{build.DownloadConfigScript.inputs[0].destination}}'

    - name: RebootAfterConfigApplied
      action: Reboot
      inputs:
        delaySeconds: 60

    - name: InstallWindowsUpdates
      action: UpdateOS

- name: validate
  steps:
    - name: DownloadTestConfigScript
      action: S3Download
      timeoutSeconds: 60
      onFailure: Abort
      maxAttempts: 3
      inputs:
        - source: 's3://customer-bucket/testConfig.ps1'
          destination: 'C:\testConfig.ps1'
```

```
      - name: ValidateConfigScript
        action: ExecutePowerShell
        timeoutSeconds: 120
        onFailure: Abort
        maxAttempts: 3
        inputs:
          file: '{{validate.DownloadTestConfigScript.inputs[0].destination}}'

      - name: Cleanup
        action: DeleteFile
        onFailure: Abort
        maxAttempts: 3
        inputs:
          - path: '{{validate.DownloadTestConfigScript.inputs[0].destination}}'

  - name: test
    steps:
      - name: DownloadTestConfigScript
        action: S3Download
        timeoutSeconds: 60
        onFailure: Abort
        maxAttempts: 3
        inputs:
          - source: 's3://customer-bucket/testConfig.ps1'
            destination: 'C:\testConfig.ps1'

      - name: ValidateConfigScript
        action: ExecutePowerShell
        timeoutSeconds: 120
        onFailure: Abort
        maxAttempts: 3
        inputs:
          file: '{{test.DownloadTestConfigScript.inputs[0].destination}}'
```

以下は、カスタム Linux バイナリファイルをダウンロードして実行するためのドキュメントスキーマの例です。

```
name: LinuxBin
description: Download and run a custom Linux binary file.
schemaVersion: 1.0
phases:
  - name: build
```

```
    steps:
      - name: Download
        action: S3Download
        inputs:
          - source: s3://<replaceable>mybucket</replaceable>/
<replaceable>myapplication</replaceable>
            destination: /tmp/<replaceable>myapplication</replaceable>
      - name: Enable
        action: ExecuteBash
        onFailure: Continue
        inputs:
          commands:
            - 'chmod u+x {{ build.Download.inputs[0].destination }}'
      - name: Install
        action: ExecuteBinary
        onFailure: Continue
        inputs:
          path: '{{ build.Download.inputs[0].destination }}'
          arguments:
            - '--install'
      - name: Delete
        action: DeleteFile
        inputs:
          - path: '{{ build.Download.inputs[0].destination }}'
```

セットアップファイルを使用して Windows インスタンス AWS CLI に をインストールするドキュメントスキーマの例を次に示します。

```
name: InstallCLISetUp
description: Install &CLI; using the setup file
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download
        action: S3Download
        inputs:
          - source: s3://aws-cli/AWSCLISetup.exe
            destination: C:\Windows\temp\AWSCLISetup.exe
      - name: Install
        action: ExecuteBinary
        onFailure: Continue
        inputs:
```

```
          path: '{{ build.Download.inputs[0].destination }}'
          arguments:
            - '/install'
            - '/quiet'
            - '/norestart'
      - name: Delete
        action: DeleteFile
        inputs:
          - path: '{{ build.Download.inputs[0].destination }}'
```

MSI インストーラ AWS CLI を使用して をインストールするためのドキュメントスキーマの例を次に示します。

```
name: InstallCLIMSI
description: Install &CLI; using the MSI installer
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download
        action: S3Download
        inputs:
          - source: s3://aws-cli/AWSCLI64PY3.msi
            destination: C:\Windows\temp\AWSCLI64PY3.msi
      - name: Install
        action: ExecuteBinary
        onFailure: Continue
        inputs:
          path: 'C:\Windows\System32\msiexec.exe'
          arguments:
            - '/i'
            - '{{ build.Download.inputs[0].destination }}'
            - '/quiet'
            - '/norestart'
      - name: Delete
        action: DeleteFile
        inputs:
          - path: '{{ build.Download.inputs[0].destination }}'
```

で変数を定義して参照する AWSTOE

変数を使用すると、アプリケーション全体で使用できるわかりやすい名前でデータにラベルを付けることができます。複雑なワークフローでは、シンプルで読み取り可能な形式でカスタム変数を定義し、 AWSTOE コンポーネントの YAML アプリケーションコンポーネントドキュメントで参照できます。

このセクションでは、構文、名前の制約、例など、YAML アプリケーション AWSTOE コンポーネントドキュメントでコンポーネントの変数を定義するのに役立つ情報を提供します。

パラメータ

パラメータは、呼び出し元のアプリケーションがランタイムに提供できる設定を含む可変変数です。YAML ドキュメントの `Parameters` セクションでパラメーターを定義できます。

パラメータ命名規則

- 名前は3文字以上128文字以下でなければならない。
- 名前には、英数字 (a-z, A-Z, 0-9) 、ダッシュ (-) 、アンダースコア (_) のみを含めることができます。
- 名前は文書内で一意でなければならない。
- 名前は YAML 文字列として指定する必要があります。

構文

```
parameters:
  - <name>:
      type: <parameter type>
      default: <parameter value>
      description: <parameter description>
```

キー名	必要	説明
name	はい	パラメータの名前。ドキュメント内で一意である必要があります (他のパラメータ名や定数と同じであってはなりません)。

キー名	必要	説明
type	はい	パラメータのデータ型。対応するタイプは以下の通り：string.
default	いいえ	パラメータのデフォルト値。
description	いいえ	パラメータを記述します。

ドキュメント内の参照パラメータ値。

次のように、YAML ドキュメント内のステップ入力またはループ入力でパラメーターを参照できます。

- パラメータ参照は大文字と小文字が区別され、名前は完全に一致する必要があります。
- 名前は二重中括弧 で囲む必要があります{{*MyParameter*}}。
- 中括弧内にはスペースを入れてもかまいませんが、自動的に切り捨てられます。たとえば、以下のリファレンスはすべて有効です。

  {{ *MyParameter* }}, {{ *MyParameter*}}, {{*MyParameter* }}, {{*MyParameter*}}
- YAML ドキュメント内の参照は文字列 (一重引用符または二重引用符で囲む) として指定する必要があります。

  例えば、- {{ *MyParameter* }} は文字列として識別されないため無効です。

  ただし、次の参照先はどちらも有効です :- '{{ *MyParameter* }}' と - "{{ *MyParameter* }}" 。

例

次の例は YAML ドキュメントでのパラメータの使用方法を示しています。

- ステップ入力のパラメーターを参照してください。

```
name: Download AWS CLI version 2
schemaVersion: 1.0
parameters:
  - Source:
```

```
      type: string
      default: 'https://awscli.amazonaws.com/AWSCLIV2.msi'
      description: The AWS CLI installer source URL.
phases:
  - name: build
    steps:
      - name: Download
        action: WebDownload
        inputs:
          - source: '{{ Source }}'
            destination: 'C:\Windows\Temp\AWSCLIV2.msi'
```

- ループ入力のパラメータを参照する：

```
name: PingHosts
schemaVersion: 1.0
parameters:
  - Hosts:
      type: string
      default: 127.0.0.1,amazon.com
      description: A comma separated list of hosts to ping.
phases:
  - name: build
    steps:
      - name: Ping
        action: ExecuteBash
        loop:
          forEach:
            list: '{{ Hosts }}'
            delimiter: ','
        inputs:
          commands:
            - ping -c 4 {{ loop.value }}
```

ランタイムにパラメーターをオーバーライドする

キーと値のペア AWS CLI で `--parameters`のオプションを使用して、実行時にパラメータ値を設定できます。

- <name><value>パラメータのキーと値のペアを等号 (=) で区切って名前と値として指定します。
- 複数のパラメータはカンマで区切る必要があります。

- YAML コンポーネントドキュメントにないパラメータ名は無視されます。
- パラメータ名と値はどちらも必須です。

> **Important**
>
> コンポーネントパラメータはプレーンテキストの値で、 AWS CloudTrailに記録されます。シークレットを保存するには、 AWS Secrets Manager または AWS Systems Manager Parameter Store を使用することをお勧めします。Secrets Managerの詳細については、AWS Secrets Manager ユーザーガイドのSecrets Managerとはを参照してください。 AWS Systems Manager パラメータストアについては、AWS Systems Manager ユーザーガイドのAWS Systems Manager パラメータストアを参照。

構文

```
--parameters name1=value1,name2=value2...
```

CLI オプション:	必要	説明
--パラメータ# = #,...	いいえ	このオプションは、パラメータ名をキーとして、キーと値のペアのリストを取得します。

例

次の例は YAML ドキュメントでのパラメータの使用方法を示しています。

- この `--parameter` オプションで指定されたパラメータのキーと値のペアは無効です。

```
--parameters ntp-server=
```

- AWS CLIに`--parameter`オプションでパラメータのキーと値のペアを1つ設定します：

```
--parameters ntp-server=ntp-server-windows-qe.us-east1.amazon.com
```

- AWS CLIの`--parameter`オプションで複数のパラメータのキーと値のペアを設定します：

```
--parameters ntp-server=ntp-server.amazon.com,http-url=https://internal-us-
east1.amazon.com
```

定数

定数は不変の変数で、一度定義すると変更したりオーバーライドしたりすることはできません。定数は、 AWSTOE ドキュメントの constantsセクションの値を使用して定義できます。

定数命名規則

- 名前は3文字以上128文字以下でなければならない。
- 名前には、英数字 (a-z, A-Z, 0-9)、ダッシュ (-)、アンダースコア (_) のみを含めることができます。
- 名前は文書内で一意でなければならない。
- 名前は YAML 文字列として指定する必要があります。

[Syntax (構文)]

```
constants:
  - <name>:
      type: <constant type>
      value: <constant value>
```

キー名	必要	説明
name	はい	定数の名前。ドキュメント内で一意である必要があります (他のパラメータ名や定数と同じであってはなりません)。
value	はい	定数の値。
type	はい	定数のタイプ。対応タイプはstring。

文書内の参照定数値

次のように、YAML ドキュメント内のステップもしくはループ入力で定数を参照できます：

- 定数参照は大文字と小文字を区別し、名前は正確に一致しなければならない。
- 名前は二重中括弧 で囲む必要があります{{*MyConstant*}}。
- 中括弧内にはスペースを入れてもかまいませんが、自動的に切り捨てられます。たとえば、以下のリファレンスはすべて有効です。

  {{ *MyConstant* }}, {{ *MyConstant*}}, {{*MyConstant* }}, {{*MyConstant*}}
- YAML ドキュメント内の参照は文字列 (一重引用符または二重引用符で囲む) として指定する必要があります。

  例えば、- {{ *MyConstant* }} は文字列として識別されないため無効です。

  ただし、次の参照先はどちらも有効です :- '{{ *MyConstant* }}' と - "{{ *MyConstant* }}" 。

例

ステップ入力で参照される定数

```
name: Download AWS CLI version 2
schemaVersion: 1.0
constants:
  - Source:
      type: string
      value: https://awscli.amazonaws.com/AWSCLIV2.msi
phases:
  - name: build
    steps:
      - name: Download
        action: WebDownload
        inputs:
          - source: '{{ Source }}'
            destination: 'C:\Windows\Temp\AWSCLIV2.msi'
```

ループ入力で参照される定数

```
name: PingHosts
schemaVersion: 1.0
constants:
```

```
  - Hosts:
      type: string
      value: 127.0.0.1,amazon.com
phases:
  - name: build
    steps:
      - name: Ping
        action: ExecuteBash
        loop:
          forEach:
            list: '{{ Hosts }}'
            delimiter: ','
        inputs:
          commands:
            - ping -c 4 {{ loop.value }}
```

AWSTOEでループ構文を使用する

このセクションでは、 AWSTOEでループ構文を作成する際に役立つ情報を提供します。ループは、繰り返される命令シーケンスを定義する。 AWSTOEでは以下のタイプのループ構文を使用できます。

- for コンストラクト — 制限付きの整数のシーケンスを反復処理します。
- forEach コンストラクト
  - 入力リストによる forEach ループ — 有限数の文字列を反復処理します。
  - 区切りリストによる forEach ループ — 区切り文字で結合された有限の文字列のコレクションを反復処理します。

> **Note**
>
> ループ構文は文字列データ型のみをサポートします。

ループ構文のトピック

- イテレーション変数の参照
- ループ構文のタイプ
- ステップフィールド
- ステップとイテレーションの出力

イテレーション変数の参照

現在のイテレーション変数のインデックスと値を参照するには、ループ構文を含むステップの入力ボディ内で参照式 {{ loop.* }} を使用する必要があります。この式は、別のステップのループ構文のイテレーション変数を参照する場合には使用できません。

参照式は、次のメンバーで構成されます。

- {{ loop.index }} — 0 でインデックスが付けられていまる現在のイテレーションの序数位置。
- {{ loop.value }} — 現在のイテレーション変数に関連付けられた値。

ループ名

ループ構文にはすべて、識別用のオプションの名前フィールドがあります。ループ名を指定すると、そのループ名を使用してステップの入力ボディ内のイテレーション変数を参照できます。名前付きループのイテレーションインデックスと値を参照するには、ステップの入力ボディで {{ <loop_name>.* }} と {{ loop.* }} を使用してください。この式は、他のステップの名前付きループ構成を参照するために使用することはできない。

参照式は、次のメンバーで構成されます。

- {{ <loop_name>.index }} — 0 でインデックスされる指定されたループの現在のイテレーションの序数位置。
- {{ <loop_name>.value }} — 指定したループの現在のイテレーション変数に関連付けられた値。

参照式を解決する

は参照式を次のように AWSTOE 解決します。

- {{ <loop_name>.* }} – 次の口ジックを使用してこの式を AWSTOE 解決します。
  - 現在実行中のステップのループが <loop_name> 値と一致すると、参照式は現在実行中のステップのループ構文に変換されます。
  - 現在実行中のステップ内に指定されたループ構文がある場合は、<loop_name> はそのループ構文に解決されます。
- {{ loop.* }} - 現在実行中のステップで定義されているループ構文を使用して式を AWSTOE 解決します。

ループを含まないステップ内で参照式が使用されている場合、 AWSTOE は式を解決せず、置き換えなしでステップに表示されます。

> **Note**
>
> YAML コンパイラーが参照式を正しく解釈するには、二重引用符で囲む必要があります。

ループ構文のタイプ

このセクションでは、 AWSTOEで使用できるループ構文タイプに関する情報と例を紹介します。

ループ構文のタイプ

- for ループ
- forEach ループ (入力リスト付き)
- 区切りリストによる forEach ループ

for ループ

for ループは、変数の先頭と末尾で囲まれた境界内で指定された整数の範囲で反復処理を行います。イテレーション値は [start, end] のセットに含まれており、境界値も含まれます。

AWSTOE は、start、、および updateByの値を検証してend、組み合わせによって無限ループが発生しないことを確認します。

for ループスキーマ

```
  - name: "StepName"
    action: "ActionModule"
    loop:
      name: "string"
      for:
        start: int
        end: int
        updateBy: int
inputs:
  ...
```

for ループ入力

フィールド	説明	タイプ	必須	デフォルト
name	ループの一意の名前。同じフェーズの他のループ名と比べると一意でなければなりません。	文字列	いいえ	""
start	イテレーションの開始値。連鎖式は受け付けません。	整数	はい	該当なし
end	反復の終了値。連鎖式は受け付けません。	整数	はい	該当なし
updateBy	加算によってイテレーション値が更新される場合の違い。負または正の 0 以外の値でなければなりません。連鎖式は受け付けません。	整数	はい	該当なし

for ループ入力の例

```
- name: "CalculateFileUploadLatencies"
  action: "ExecutePowerShell"
  loop:
    for:
      start: 100000
```

```
        end: 1000000
        updateBy: 100000
    inputs:
      commands:
        - |
          $f = new-object System.IO.FileStream c:\temp\test{{ loop.index }}.txt,
 Create, ReadWrite
          $f.SetLength({{ loop.value }}MB)
          $f.Close()
        - c:\users\administrator\downloads\latencyTest.exe --file c:\temp
\test{{ loop.index }}.txt
        - AWS s3 cp c:\users\administrator\downloads\latencyMetrics.json s3://bucket/
latencyMetrics.json
        - |
          Remove-Item -Path c:\temp\test{{ loop.index }}.txt
          Remove-Item -Path c:\users\administrator\downloads\latencyMetrics.json
```

forEach ループ (入力リスト付き)

forEach ループは明示的な値リスト (文字列でも連鎖式でもかまいません) を繰り返し処理します。

入力リストのスキーマを含む forEach ループ

```
  - name: "StepName"
    action: "ActionModule"
    loop:
      name: "string"
      forEach:
        - "string"
    inputs:
  ...
```

forEach ループ (入力リスト付き)

フィールド	説明	タイプ	必須	デフォルト
name	ループの一意の名前。同じフェーズの他のループ名と比べると一意でな	文字列	いいえ	""

フィールド	説明	タイプ	必須	デフォルト
	ければなりません。			
forEach ループの文字列のリスト	反復処理用の文字列のリスト。連鎖式をリスト内の文字列として受け入れます。YAMLコンパイラが正しく解釈するために、連鎖式は二重引用符で囲む必要があります。	文字列のリスト	はい	該当なし

入力リストによる forEach ループ (例 1)

```
- name: "ExecuteCustomScripts"
  action: "ExecuteBash"
  loop:
    name: BatchExecLoop
    forEach:
      - /tmp/script1.sh
      - /tmp/script2.sh
      - /tmp/script3.sh
  inputs:
    commands:
      - echo "Count {{ BatchExecLoop.index }}"
      - sh "{{ loop.value }}"
      - |
        retVal=$?
        if [ $retVal -ne 0 ]; then
          echo "Failed"
        else
          echo "Passed"
       fi
```

入力リストによる forEach ループ (例 2)

```
- name: "RunMSIWithDifferentArgs"
  action: "ExecuteBinary"
  loop:
    name: MultiArgLoop
    forEach:
      - "ARG1=C:\Users ARG2=1"
      - "ARG1=C:\Users"
      - "ARG1=C:\Users ARG3=C:\Users\Administrator\Documents\f1.txt"
  inputs:
    commands:
      path: "c:\users\administrator\downloads\runner.exe"
      args:
        - "{{ MultiArgLoop.value }}"
```

入力リストによる forEach ループ (例 3)

```
- name: "DownloadAllBinaries"
  action: "S3Download"
  loop:
    name: MultiArgLoop
    forEach:
      - "bin1.exe"
      - "bin10.exe"
      - "bin5.exe"
  inputs:
    - source: "s3://bucket/{{ loop.value }}"
      destination: "c:\temp\{{ loop.value }}"
```

区切りリストによる **forEach** ループ

ループは、区切り文字で区切られた値を含む文字列を繰り返し処理します。文字列の構成要素を反復処理するには、 は区切り文字 AWSTOE を使用して文字列を反復処理に適した配列に分割します。

区切りリストスキーマによる forEach ループ

```
- name: "StepName"
  action: "ActionModule"
  loop:
    name: "string"
    forEach:
      list: "string"
```

```
        delimiter: ".,;:\n\t -_"
    inputs:
  ...
```

区切りリスト入力による **forEach** ループ

フィールド	説明	タイプ	必須	デフォルト
name	ループに付けられた一意の名前。同じフェーズの他のループ名と比較した場合、ユニークでなければならない。	文字列	いいえ	""
list	構成文字列を共通の区切り文字で結合した文字列です。連鎖式も受け付けます。連鎖式の場合は、YAML コンパイラが正しく解釈できるように、必ず二重引用符で囲んでください。	文字列	はい	該当なし
delimiter	ブロック内の文字列を区切るために使用する文字。デフォルトはカンマ文字です。与えられたリストで使用できる区切り文字	文字列	いいえ	カンマ: ","

フィールド	説明	タイプ	必須	デフォルト
	は 1 つだけです。 • ドット: "." • カンマ: "," • セミコロン: ";" • コロン: ":" • 改行: "\n" • タブ: "\t" • スペース: " " • ハイフン: "-" • 下線: "_" 連鎖式は使用できません。			

> **Note**
>
> `list` の値は不変の文字列として扱われます。ランタイムに `list` のソースが変更されても、実行中には反映されません。

`forEach` 区切りリストによるループ 例1

この例では、次の連鎖式パターンを使用して、別のステップの出力 を参照します<phase_name>.<step_name>.[inputs | outputs].<var_name>。

```
- name: "RunMSIs"
  action: "ExecuteBinary"
  loop:
    forEach:
      list: "{{ build.GetAllMSIPathsForInstallation.outputs.stdout }}"
      delimiter: "\n"
  inputs:
    commands:
      path: "{{ loop.value }}"
```

forEach 区切りリストによるループ 例2

```
- name: "UploadMetricFiles"
  action: "S3Upload"
  loop:
    forEach:
      list: "/tmp/m1.txt,/tmp/m2.txt,/tmp/m3.txt,..."
  inputs:
    commands:
      - source: "{{ loop.value }}"
        destination: "s3://bucket/key/{{ loop.value }}"
```

ステップフィールド

ループはステップの一部です。ステップの実行に関連するフィールドは、個々の反復には適用されません。ステップフィールドは、以下のようにステップレベルでのみ適用されます。

- timeoutSeconds - ループのすべての反復は、このフィールドで指定された時間内に実行されなければならない。ループがタイムアウトすると、 AWSTOE はステップの再試行ポリシーを実行し、新しい試行ごとにタイムアウトパラメータをリセットします。最大再試行回数に達した後でループ実行がタイムアウト値を超えると、ステップの失敗メッセージにループ実行がタイムアウトになったことが示されます。
- onFailure - 以下のようにステップに失敗処理が適用される：
  - onFailure が に設定されている場合Abort、 はループ AWSTOE を終了し、再試行ポリシーに従ってステップを再試行します。最大再試行回数に達すると、 は現在のステップを失敗として AWSTOE マークし、プロセスの実行を停止します。

AWSTOE は、親フェーズとドキュメントのステータスコードを に設定しますFailed。

> **Note**
>
> 失敗したステップの後に、それ以上のステップは実行されません。

- onFailureがContinueに設定されている場合、 AWSTOE はループを抜け、リトライポリシーに従ってステップをリトライする。最大再試行回数に達すると、 は現在のステップを失敗として AWSTOE マークし、次のステップの実行を続行します。

  AWSTOE は、親フェーズとドキュメントのステータスコードを に設定しますFailed。

- onFailureがIgnoreに設定されている場合、 AWSTOE はループを抜け、リトライポリシーに従ってステップをリトライする。最大再試行回数に達すると、 は現在のステップを として AWSTOE マークしIgnoredFailure、次のステップの実行を続行します。

  AWSTOE は、親フェーズとドキュメントのステータスコードを に設定しますSuccessWithIgnoredFailure。

  > **Note**
  >
  > これでも実行は成功したと見なされますが、1 つ以上のステップが失敗して無視されたことを知らせる情報が含まれます。

- maxAttempts - 再試行ごとに、全ステップと全反復が最初から実行されます。
- status - ステップの実行の全体的なステータス。statusは個々の反復のステータスを表すものではない。ループを含むステップのステータスは次のように決定されます。
  - 1 回のイテレーションが実行に失敗した場合、ステップのステータスは失敗を示します。
  - すべてのイテレーションが成功すると、ステップのステータスは成功を示します。
- startTime - ステップ実行の全体的な開始時間。個々のイテレーションの開始時間を表すものではありません。
- endTime - ステップ実行の全体的な終了時間。個々の反復の終了時刻を表すものではない。
- failureMessage - タイムアウト以外のエラーの場合に失敗した反復インデックスを含みます。タイムアウトエラーの場合、メッセージにはループの実行が失敗したことが示されます。失敗メッセージのサイズを最小限に抑えるため、イテレーションごとに個別のエラーメッセージは表示されません。

ステップとイテレーションの出力

すべてのイテレーションには出力が含まれます。ループ実行の終了時に、 は成功したすべての反復出力を AWSTOE に統合しますdetailedOutput.json。統合出力は、アクションモジュールの出力スキーマで定義されている対応する出力キーに属する値を照合したものです。次の例では、出力の統合方法を示しています。

イテレーション 1 の **ExecuteBash** の出力

```
{
 "stdout":"Hello"
}
```

イテレーション 2 の **ExecuteBash** の出力

```
{
 "stdout":"World"
}
```

ステップ **ExecuteBash** の出力

```
{
 "stdout":"Hello\nWorld"
}
```

たとえば、ExecuteBash、ExecutePowerShell、および ExecuteBinary はアクションモジュール出力として STDOUT を返すアクションモジュールです。 STDOUT メッセージは改行文字で結合され、detailedOutput.json のステップの全体的な出力が生成されます。

AWSTOE は、失敗したイテレーションの出力を統合しません。

AWSTOE コンポーネントマネージャーがサポートするアクションモジュール

EC2 Image Builder などのイメージ構築サービスは、 AWSTOE アクションモジュールを使用して、カスタマイズされたマシンイメージの構築とテストに使用される EC2 インスタンスの設定を支援します。このセクションでは、一般的に使用される AWSTOE アクションモジュールの特徴と、それらの設定方法について説明します。例も示します。

AWSTOE コンポーネントはプレーンテキストの YAML ドキュメントで作成されます。ドキュメントの構文については[でコンポーネントドキュメントを使用する AWSTOE]を参照。

> **Note**
>
> すべてのアクションモジュールは、Linux の `root` と Windows の `NT Authority\SYSTEM` の両方で、実行時に Systems Manager エージェントと同じアカウントを使用します。

アクションモジュールのタイプ

- 汎用実行モジュール
- ファイルダウンロードとアップロードモジュール
- ファイルシステム操作モジュール
- ソフトウェアインストールアクション
- システムアクションモジュール

汎用実行モジュール

次のセクションには、一般的な実行コマンドと命令を実行するアクションモジュールの詳細が含まれています。

一般的な実行アクションモジュール

- ExecuteBash
- ExecuteBinary
- ExecuteDocument
- ExecutePowerShell

ExecuteBash

ExecuteBash アクションモジュールを使用すると、インラインシェルコード/コマンドを使用して bash スクリプトを実行できます。このモジュールは Linux をサポートします。

コマンドブロックで指定したすべてのコマンドと命令はファイル (例:`input.sh`) に変換され、bash シェルで実行されます。シェルファイルを実行した結果がステップの終了コードです。

スクリプトが終了コード で終了すると、ExecuteBashモジュールはシステムの再起動を処理します194。起動すると、アプリケーションは以下のいずれかのアクションを実行します。

- Systems Manager エージェントによって実行された場合、アプリケーションは終了コードを呼び出し側に渡します。Systems Manager エージェント はシステムの再起動を処理し、「スクリプトからのマネージドインスタンスの再起動」で説明されているように、再起動を開始したのと同じステップを実行します。
- アプリケーションは現在の `executionstate` を保存し、アプリケーションを再実行するための再起動トリガーを設定し、システムを再起動します。

システムの再起動後、アプリケーションは再起動を開始したのと同じステップを実行します。この機能が必要な場合は、同じシェルコマンドを複数回呼び出しても処理できる同等のスクリプトを記述する必要があります。

Input (入力)

プリミティブ型	説明	タイプ	必須
`commands`	bash 構文に従って実行する命令またはコマンドのリストが含まれています。複数行の YAML を使用できます。	リスト	はい

入力例:再起動前と再起動後

```
name: ExitCode194Example
description: This shows how the exit code can be used to restart a system with
 ExecuteBash
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: RestartTrigger
        action: ExecuteBash
        inputs:
          commands:
            - |
```

```
              REBOOT_INDICATOR=/var/tmp/reboot-indicator
              if [ -f "${REBOOT_INDICATOR}" ]; then
                echo 'The reboot file exists. Deleting it and exiting with success.'
                rm "${REBOOT_INDICATOR}"
                exit 0
              fi
              echo 'The reboot file does not exist. Creating it and triggering a
 restart.'
              touch "${REBOOT_INDICATOR}"
              exit 194
```

出力

フィールド	説明	[Type] (タイプ)
stdout	コマンド実行の標準出力。	string

再起動を開始し、194 アクションモジュールの一部として終了コードを返すと、再起動を開始したのと同じアクションモジュールステップでビルドが再開されます。終了コードなしで再起動を開始すると、ビルドプロセスが失敗する可能性があります。

出力例:再起動前 (初めてドキュメントから)

```
{
 “stdout”: “The reboot file does not exist. Creating it and triggering a restart."
}
```

出力例:再起動後 (2 回目にドキュメントを通過)

```
{
 “stdout”: “The reboot file exists. Deleting it and exiting with success."
}
```

ExecuteBinary

ExecuteBinary アクションモジュールを使用すると、コマンドライン引数のリストを含むバイナリファイルを実行できます。

バイナリファイルが (Linux) または 194 (3010Windows) の終了コードで終了すると、ExecuteBinaryモジュールはシステムの再起動を処理します。この場合、アプリケーションは以下のいずれかのアクションを実行する：

- Systems Manager エージェントによって実行された場合、アプリケーションは終了コードを呼び出し側に渡します。Systems Managerエージェントはシステムの再起動を処理し、スクリプトから管理対象インスタンスを再起動するで説明したように、再起動を開始したのと同じステップを実行します。
- アプリケーションは現在の `executionstate` を保存し、アプリケーションを再実行するための再起動トリガーを設定し、システムを再起動します。

システムの再起動後、アプリケーションは再起動を開始したのと同じステップを実行します。この機能が必要な場合は、同じシェルコマンドを複数回呼び出しても処理できる同等のスクリプトを記述する必要があります。

Input (入力)

プリミティブ型	説明	タイプ	必須
`path`	実行用のバイナリファイルへのパス。	文字列	はい
`arguments`	バイナリの実行時に使用するコマンドライン引数のリストが含まれています。	文字列リスト	いいえ

入力例:.NET のインストール

```
- name: "InstallDotnet"
  action: ExecuteBinary
  inputs:
    path: C:\PathTo\dotnet_installer.exe
    arguments:
      - /qb
      - /norestart
```

出力

フィールド	説明	[Type] (タイプ)
`stdout`	コマンド実行の標準出力。	string

出力例

```
{
 "stdout": "success"
}
```

ExecuteDocument

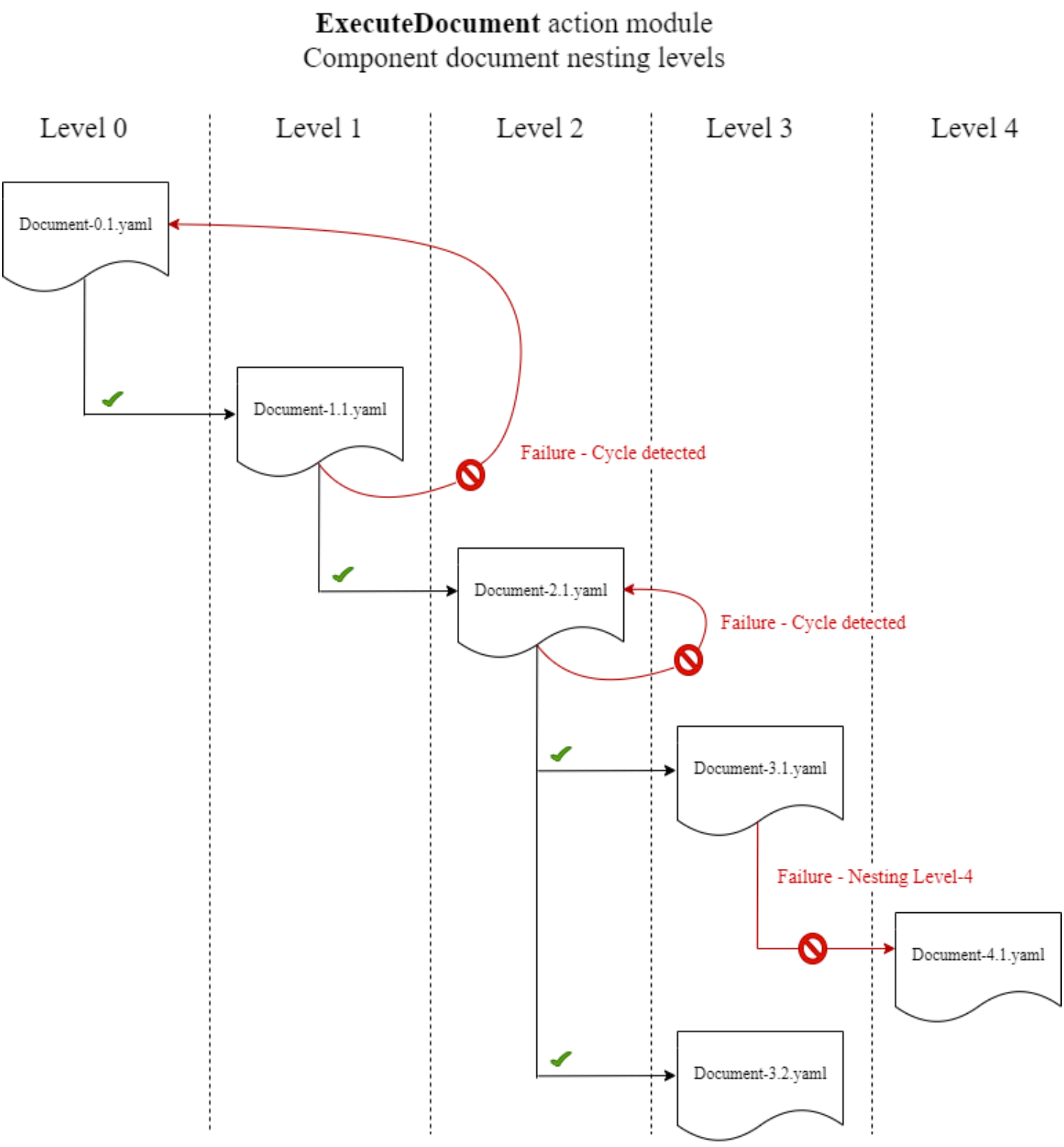
ExecuteDocument アクションモジュールは、ネストされたコンポーネントドキュメントのサポートを追加し、1 つのドキュメントから複数のコンポーネントドキュメントを実行します。 は、実行時に入力パラメータに渡されるドキュメント AWSTOE を検証します。

制限事項

- このアクションモジュールは 1 回だけ実行され、再試行はできません。また、タイムアウト制限を設定するオプションもありません。ExecuteDocument は、次のデフォルト値を設定し、変更しようとするとエラーを返します。
  - `timeoutSeconds`:-1
  - `maxAttempts`: 1

  > **Note**
  >
  > これらの値は空白のままにすると、 はデフォルト値 AWSTOE を使用します。

- 文書のネストは最大 3 レベルまで可能ですが、それ以上はできません。最上位レベルはネストされていないため、3 レベルのネストは 4 つのドキュメントレベルに相当します。このシナリオでは、最下位の文書は他の文書を呼んではならない。
- コンポーネントドキュメントを繰り返し実行することはできません。ループ構造の外部で自身を呼び出したり、現在の実行チェーンの上位にある別のドキュメントを呼び出したりするドキュメントは、サイクルを開始して無限ループに陥る可能性があります。 AWSTOE は周期的な実行を検出すると、実行を停止し、失敗を記録します。



コンポーネントドキュメント自体を実行しようとしたり、現在の実行チェーンで上位にあるコンポーネントドキュメントを実行しようとしたりすると、実行は失敗します。

Input (入力)

プリミティブ型	説明	タイプ	必須
document	コンポーネントドキュメントのパス。有効なオペレーションは以下のとおりです。 • ローカルファイルパス • S3 URI • EC2 Image Builder コンポーネントビルドバージョン ARN	文字列	はい
document-s3-bucket-owner	コンポーネントドキュメントが保存されている S3 バケットの S3 バケット所有者のアカウント ID。(コンポーネントドキュメントで S3 URI を使用している場合におすすめです)。	文字列	いいえ
phases	コンポーネントドキュメントで実行するフェーズ。カンマ区切りのリストで指定します。フェーズが指定されていない場合は、すべてのフェ	文字列	いいえ

プリミティブ型	説明	タイプ	必須
	ーズが実行されます。		
parameters	実行時にキーと値のペアとしてコンポーネントドキュメントに渡される入力パラメータ。	パラメータマップリスト	いいえ

パラメータマップ入力

プリミティブ型	説明	タイプ	必須
name	ExecuteDocument アクションモジュールが実行中のコンポーネントドキュメントに渡す入力パラメータの名前。	文字列	はい
value	入力パラメータの値。	文字列	はい

入力例

以下の例は、インストールパスによってコンポーネントドキュメントに入力される内容のバリエーションを示しています。

入力例:ローカルドキュメントパス

```
# main.yaml
schemaVersion: 1.0

phases:
```

```
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
        inputs:
          document: Sample-1.yaml
          phases: build
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

入力例:ドキュメントパスとしての S3 URI

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
        inputs:
          document: s3://my-bucket/Sample-1.yaml
          document-s3-bucket-owner: 123456789012
          phases: build,validate
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

入力例:ドキュメントパスとしてEC2 Image Builder コンポーネント ARN

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
```

```
        inputs:
          document: arn:aws:imagebuilder:us-west-2:aws:component/Sample-Test/1.0.0
          phases: test
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

ForEach ループを使用したドキュメントの実行

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
        loop:
          name: 'myForEachLoop'
          forEach:
            - Sample-1.yaml
            - Sample-2.yaml
        inputs:
          document: "{{myForEachLoop.value}}"
          phases: test
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

For ループを使ってドキュメントを実行する

```
# main.yaml
schemaVersion: 1.0

phases:
  - name: build
    steps:
      - name: ExecuteNestedDocument
        action: ExecuteDocument
```

```
        loop:
          name: 'myForLoop'
          for:
            start: 1
            end: 2
            updateBy: 1
        inputs:
          document: "Sample-{{myForLoop.value}}.yaml"
          phases: test
          parameters:
            - name: parameter-1
              value: value-1
            - name: parameter-2
              value: value-2
```

出力

AWSTOE は、 が実行されるdetailedoutput.jsonたびに という出力ファイルを作成します。このファイルには、実行中に呼び出される各コンポーネントドキュメントのすべてのフェーズとステップに関する詳細が含まれています。ExecuteDocument アクションモジュールの場合、outputsフィールドにランタイムの簡単な概要、および で実行されるフェーズ、ステップ、ドキュメントの詳細が表示されますdetailedOutput。

```
{
 \"executedStepCount\":1,\"executionId\":\"97054e22-06cc-11ec-9b14-acde48001122\",
\"failedStepCount\":0,\"failureMessage\":\"\",\"ignoredFailedStepCount\":0,\"logUrl\":
\"\",\"status\":\"success\"
}",
```

各コンポーネントドキュメントの出力サマリーオブジェクトには、次に示すように、以下の詳細とサンプル値が含まれています。

- executedStepCount「:1
- "executionId":"12345a67-89bc-01de-2f34-abcd56789012"
- failedStepCount「」:0
- "failureMessage":""
- ignoredFailedStep「カウント」:0
- "logUrl":""
- "status":"success"

出力例

次の例は、ネストされた実行が発生したときのExecuteDocumentアクションモジュールからの出力を示しています。この例では、main.yamlコンポーネントドキュメントは Sample-1.yaml コンポーネントドキュメントを正常に実行します。

```
{
    "executionId": "12345a67-89bc-01de-2f34-abcd56789012",
    "status": "success",
    "startTime": "2021-08-26T17:20:31-07:00",
    "endTime": "2021-08-26T17:20:31-07:00",
    "failureMessage": "",
    "documents": [
        {
            "name": "",
            "filePath": "main.yaml",
            "status": "success",
            "description": "",
            "startTime": "2021-08-26T17:20:31-07:00",
            "endTime": "2021-08-26T17:20:31-07:00",
            "failureMessage": "",
            "phases": [
                {
                    "name": "build",
                    "status": "success",
                    "startTime": "2021-08-26T17:20:31-07:00",
                    "endTime": "2021-08-26T17:20:31-07:00",
                    "failureMessage": "",
                    "steps": [
                        {
                            "name": "ExecuteNestedDocument",
                            "status": "success",
                            "failureMessage": "",
                            "timeoutSeconds": -1,
                            "onFailure": "Abort",
                            "maxAttempts": 1,
                            "action": "ExecuteDocument",
                            "startTime": "2021-08-26T17:20:31-07:00",
                            "endTime": "2021-08-26T17:20:31-07:00",
                            "inputs": "[{\"document\":\"Sample-1.yaml\",\"document-s3-
bucket-owner\":\"\",\"phases\":\"\",\"parameters\":null}]",
```

```
                            "outputs": "[{\"executedStepCount\":1,\"executionId\":
\"98765f43-21ed-09cb-8a76-fedc54321098\",\"failedStepCount\":0,\"failureMessage\":\"\",
\"ignoredFailedStepCount\":0,\"logUrl\":\"\",\"status\":\"success\"}]",
                            "loop": null,
                            "detailedOutput": [
                                {
                                    "executionId": "98765f43-21ed-09cb-8a76-
fedc54321098",
                                    "status": "success",
                                    "startTime": "2021-08-26T17:20:31-07:00",
                                    "endTime": "2021-08-26T17:20:31-07:00",
                                    "failureMessage": "",
                                    "documents": [
                                        {
                                            "name": "",
                                            "filePath": "Sample-1.yaml",
                                            "status": "success",
                                            "description": "",
                                            "startTime": "2021-08-26T17:20:31-07:00",
                                            "endTime": "2021-08-26T17:20:31-07:00",
                                            "failureMessage": "",
                                            "phases": [
                                                {
                                                    "name": "build",
                                                    "status": "success",
                                                    "startTime":
 "2021-08-26T17:20:31-07:00",
                                                    "endTime":
 "2021-08-26T17:20:31-07:00",
                                                    "failureMessage": "",
                                                    "steps": [
                                                        {
                                                            "name": "ExecuteBashStep",
                                                            "status": "success",
                                                            "failureMessage": "",
                                                            "timeoutSeconds": 7200,
                                                            "onFailure": "Abort",
                                                            "maxAttempts": 1,
                                                            "action": "ExecuteBash",
                                                            "startTime":
 "2021-08-26T17:20:31-07:00",
                                                            "endTime":
 "2021-08-26T17:20:31-07:00",
```

```
                                                    "inputs": "[{\"commands\":
[\"echo \\\"Hello World!\\\"\"]}]",
                                                    "outputs": "[{\"stdout\":
\"Hello World!\"}]",
                                                    "loop": null,
                                                    "detailedOutput": null
                                                }]
                                            }]
                                        }]
                                    }]
                                }]

                            }]
                        }]
}
```

ExecutePowerShell

ExecutePowerShell アクションモジュールを使用すると、インラインシェルコード/コマンドを使用して PowerShell スクリプトを実行できます。このモジュールは、Windows プラットフォームと Windows をサポートしています PowerShell。

コマンドブロックで指定されたすべてのコマンド/命令はスクリプトファイル (などinput.ps1) に変換され、Windows を使用して実行されますPowerShell。シェルファイルを実行した結果がステップの終了コードです。

シェルコマンドが終了コード で終了すると、ExecutePowerShellモジュールはシステムの再起動を処理します3010。起動すると、アプリケーションは以下のいずれかのアクションを実行します。

- Systems Manager エージェントによって実行された場合、アプリケーションは終了コードを呼び出し側に渡します。Systems Manager エージェント はシステムの再起動を処理し、「スクリプトからのマネージドインスタンスの再起動」で説明されているように、再起動を開始したのと同じステップを実行します。
- 現在の executionstate を保存し、アプリケーションを再実行するための再起動トリガーを設定し、システムを再起動します。

システムの再起動後、アプリケーションは再起動を開始したのと同じステップを実行します。この機能が必要な場合は、同じシェルコマンドを複数回呼び出しても処理できる同等のスクリプトを記述する必要があります。

Input (入力)

プリミティブ型	説明	タイプ	必須
commands	PowerShell 構文に従って実行する命令またはコマンドのリストが含まれます。複数行の YAML を使用できます。	文字列リスト	はい。両方ではなく、commandsまたは file を指定する必要があります。
file	PowerShell スクリプトファイルへのパスが含まれます。PowerShell は、-fileコマンドライン引数を使用してこのファイルに対して実行されます。パスは.ps1ファイルを指していなければなりません。	文字列	はい。両方ではなく、commandsまたは file を指定する必要があります。

入力例:再起動前と再起動後

```
name: ExitCode3010Example
description: This shows how the exit code can be used to restart a system with
 ExecutePowerShell
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: RestartTrigger
        action: ExecutePowerShell
        inputs:
          commands:
            - |
              $rebootIndicator = Join-Path -Path $env:SystemDrive -ChildPath 'reboot-
indicator'
```

```
            if (Test-Path -Path $rebootIndicator) {
              Write-Host 'The reboot file exists. Deleting it and exiting with
 success.'
              Remove-Item -Path $rebootIndicator -Force | Out-Null
              [System.Environment]::Exit(0)
            }
            Write-Host 'The reboot file does not exist. Creating it and triggering a
 restart.'
            New-Item -Path $rebootIndicator -ItemType File | Out-Null
            [System.Environment]::Exit(3010)
```

出力

フィールド	説明	[Type] (タイプ)
stdout	コマンド実行の標準出力。	string

アクションモジュールの一部として再起動を実行して終了コード 3010 を返した場合、ビルドは再起動を開始したのと同じアクションモジュールステップで再開されます。終了コードを指定せずに再起動を実行すると、ビルドプロセスが失敗する可能性があります。

出力例:再起動前 (初めてドキュメントから)

```
{
 “stdout”: “The reboot file does not exist. Creating it and triggering a restart."
}
```

出力例:再起動後 (2 回目にドキュメントを通過)

```
{
 “stdout”: “The reboot file exists. Deleting it and exiting with success."
}
```

ファイルダウンロードとアップロードモジュール

次のセクションでは、ダウンロードとアップロードのコマンドと指示を実行するアクションモジュールの詳細を説明します。

ダウンロードおよびアップロードアクションモジュール

- S3 ダウンロード
- S3: アップロード
- WebDownload

S3 ダウンロード

S3Download アクションモジュールを使用すると、Amazon S3 オブジェクトまたはオブジェクトのセットを、destination パスで指定したローカルファイルまたはフォルダにダウンロードできます。指定した場所に既にファイルが存在し、overwrite フラグが true に設定されている場合、S3Downloadはそのファイルは上書きされます。

source ロケーションは Amazon S3 内の特定のオブジェクトを指すこともできますし、アスタリスクワイルドカード (*) が付いたキー接頭辞 を使用して、キー接頭辞 パスと一致するオブジェクトのセットをダウンロードすることもできます。source ロケーションでキー接頭辞を指定すると、S3Download アクションモジュールはプレフィックスに一致するすべてのもの (ファイルとフォルダを含む) をダウンロードします。キー接頭辞 がフォワードスラッシュで終わり、その後にアスタリスク (/*) が続くことを確認してください。これにより、プレフィックスに一致するものをすべてダウンロードできるようになります。例: *s3://my-bucket/my-folder/**。

> **Note**
>
> ダウンロード先パス内のすべてのフォルダーは、ダウンロード前に存在している必要があります。そうでない場合、ダウンロードは失敗します。

ダウンロード中に指定されたキー接頭辞 S3Download アクションが失敗した場合、フォルダーの内容は失敗前の状態にロールバックされません。宛先フォルダーは、障害発生時のままです。

対応するユースケース

S3Download アクションモジュールは以下のユースケースをサポートしています。

- Amazon S3 オブジェクトは、ダウンロードパスで指定されたローカルフォルダにダウンロードされます。
- Amazon S3 オブジェクト (Amazon S3 ファイルパスにキー接頭辞 が付いている) は、指定されたローカルフォルダにダウンロードされます。このローカルフォルダは、キー接頭辞 と一致するすべての Amazon S3 オブジェクトをローカルフォルダに再帰的にコピーします。

IAMの要件

インスタンスプロファイルに関連付ける IAM ロールには、S3Download アクションモジュールを実行する権限が必要です。インスタンスプロファイルに関連付けられているIAMロールに、以下のIAMポリシーをアタッチする必要があります：

- 単一ファイル： s3:GetObjectバケツ/オブジェクトに対して(例えばarn:aws:s3:::**BucketName**/*)。
- 複数のファイル:s3:ListBucketバケット/オブジェクトに対するファイル(例:arn:aws:s3:::**BucketName**) s3:GetObject とバケット/オブジェクト(例:arn:aws:s3:::**BucketName**/*)。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト
source	ダウンロードのソースである Amazon S3 バケット。特定のオブジェクトへのパスを指定するか、またはキー接頭辞 (末尾がスラッシュ) で終わり、アスタリスクワイルドカード (/*) が続くものを使用して、キー接頭辞に一致するオブジェクトのセットをダウンロードできます。	文字列	はい	該当なし

プリミティブ型	説明	タイプ	必須	デフォルト
destination	Amazon S3 オブジェクトがダウンロードされるローカルパス。1 つのファイルをダウンロードするには、パスの一部としてファイル名を指定する必要があります。例えば */myfolder/package.zip* です。	文字列	はい	該当なし
expectedBucketOwner	source パスで指定されたバケットの必要な所有者アカウント ID。ソースで指定されている Amazon S3 バケットの所有権を確認することをお勧めします。	文字列	いいえ	該当なし

プリミティブ型	説明	タイプ	必須	デフォルト
overwrite	true に設定すると、指定したローカルパスの宛先フォルダに同じ名前のファイルがすでに存在する場合、ダウンロードファイルはローカルファイルを上書きします。false に設定すると、ローカルシステム上の既存のファイルは上書きされないよう保護され、アクションモジュールはダウンロードエラーで失敗します。 例えば、次のようになります: `Error: S3Download: File already exists and "overwrite" property for "destination" file`	ブール値	いいえ	true

プリミティブ型	説明	タイプ	必須	デフォルト
	is set to false. Cannot download.			

> **Note**
>
> 次の例では、Windows フォルダパスを Linux パスに置き換えることができます。たとえば、*C:\myfolder\package.zip* を */myfolder/package.zip* に置き換えることができます。

入力例:Amazon S3 オブジェクトをローカルファイルにコピーする

以下の例では、Amazon S3 オブジェクトをローカルファイルにコピーする方法を示します。

```
- name: DownloadMyFile
  action: S3Download
  inputs:
    - source: s3://mybucket/path/to/package.zip
      destination: C:\myfolder\package.zip
      expectedBucketOwner: 123456789022
      overwrite: false
    - source: s3://mybucket/path/to/package.zip
      destination: C:\myfolder\package.zip
      expectedBucketOwner: 123456789022
      overwrite: true
    - source: s3://mybucket/path/to/package.zip
      destination: C:\myfolder\package.zip
      expectedBucketOwner: 123456789022
```

入力例：キープレフィックスを持つAmazon S3バケット内のすべてのAmazon S3オブジェクトをローカルフォルダにコピーします。

次の例では、Amazon S3バケット内のすべてのAmazon S3オブジェクトを、キーのプレフィックス付きでローカルフォルダにコピーする方法を示しています。Amazon S3 にはフォルダという概念が

ないため、キー接頭辞 に一致するすべてのオブジェクトがコピーされます。ダウンロードできるオブジェクトの最大数は 1000 です。

```
- name: MyS3DownloadKeyprefix
  action: S3Download
  maxAttempts: 3
  inputs:
    - source: s3://mybucket/path/to/*
      destination: C:\myfolder\
      expectedBucketOwner: 123456789022
      overwrite: false
    - source: s3://mybucket/path/to/*
      destination: C:\myfolder\
      expectedBucketOwner: 123456789022
      overwrite: true
    - source: s3://mybucket/path/to/*
      destination: C:\myfolder\
      expectedBucketOwner: 123456789022
```

出力

なし。

S3: アップロード

S3Uploadアクションモジュールを使用すると、ソースファイルまたはフォルダからAmazon S3の場所にファイルをアップロードできます。ソースロケーションに指定されたパスにワイルドカード (*) を使用すると、ワイルドカードパターンに一致するパスを持つすべてのファイルをアップロードできます。

再帰的なS3Uploadアクションが失敗した場合、既にアップロードされたファイルは宛先のAmazon S3バケットに残ります。

対応するユースケース

- Amazon S3 オブジェクトへのローカルファイル。
- Amazon S3 キー接頭辞 へのフォルダ内のローカルファイル (ワイルドカード付き)。
- ローカルフォルダ (`recurse` を `true` に設定されている必要があります) を Amazon S3 キー接頭辞 にコピーします。

IAMの要件

インスタンスプロファイルに関連付ける IAM ロールには、S3Upload アクションモジュールを実行する権限が必要です。インスタンスプロファイルに関連付けられているIAMロールに、以下のIAMポリシーをアタッチする必要があります。ポリシーは、ターゲット Amazon S3 バケットに s3:PutObject アクセス権限を付与する必要があります。例えば、arn:aws:s3:::**BucketName**/* です。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト
source	ソースファイル/フォルダの生成元のローカルパス。source はアスタリスクワイルドカード (*) をサポートします。	文字列	はい	該当なし
destination	ソースファイル/フォルダがアップロードされる宛先 Amazon S3 バケットのパス。	文字列	はい	該当なし
recurse	true に設定すると、S3Uploadを再帰的に実行します。	文字列	いいえ	false
expectedBucketOwner	宛先パスで指定されている Amazon S3 バケットの予想所	文字列	いいえ	該当なし

プリミティブ型	説明	タイプ	必須	デフォルト
	有者アカウントID。宛先に指定されたAmazon S3バケットの所有権を確認することをお勧めします。			

入力例:Amazon S3 オブジェクトをローカルファイルにコピーする

次の例は、ローカルファイルをAmazon S3オブジェクトにコピーする方法を示しています。

```
- name: MyS3UploadFile
  action: S3Upload
  onFailure: Abort
  maxAttempts: 3
  inputs:
    - source: C:\myfolder\package.zip
      destination: s3://mybucket/path/to/package.zip
      expectedBucketOwner: 123456789022
```

入力例：ローカルフォルダ内のすべてのファイルを、キーの接頭辞を持つAmazon S3バケットにコピーします。

次の例では、ローカルフォルダ内のすべてのファイルを、キープレフィックスを持つAmazon S3バケットにコピーする方法を示しています。この例では、recurse が指定されていないためサブフォルダやその内容はコピーされません。デフォルトは false です。

```
- name: MyS3UploadMultipleFiles
  action: S3Upload
  onFailure: Abort
  maxAttempts: 3
  inputs:
    - source: C:\myfolder\*
      destination: s3://mybucket/path/to/
      expectedBucketOwner: 123456789022
```

入力例:ローカルフォルダから再帰的に Amazon S3 バケットにコピーする

次の例では、ローカルフォルダからAmazon S3バケットに、キープレフィックスを付けてすべてのファイルとフォルダを再帰的にコピーする方法を示しています。

```
- name: MyS3UploadFolder
  action: S3Upload
  onFailure: Abort
  maxAttempts: 3
  inputs:
    - source: C:\myfolder\*
      destination: s3://mybucket/path/to/
      recurse: true
      expectedBucketOwner: 123456789022
```

出力

なし。

WebDownload

WebDownload アクションモジュールを使用すると、HTTP/HTTPS プロトコル経由でリモートの場所からファイルとリソースをダウンロードできます (HTTPS が推奨されます) 。ダウンロードの数やサイズに制限はありません。このモジュールはリトライと指数バックオフロジックを処理します。

ユーザーの入力に応じて、各ダウンロード操作が成功するまでに最大 5 回の試行が割り当てられます。これらの試行は、アクションモジュールの障害に関連する `maxAttempts` の `steps` ドキュメントフィールドで指定されている試行とは異なります。

このアクションモジュールは暗黙的にリダイレクトを処理します。`200` を除く、すべての HTTP ステータスコードはエラーになります。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト
`source`	RFC 3986 標準に準拠した有効な HTTP/HTTPS URL (HTTPS を推奨)。式を連鎖	文字列	はい	該当なし

プリミティブ型	説明	タイプ	必須	デフォルト
	させることは許可されます。			
destination	ローカルシステム上のファイルまたはフォルダーの絶対パスまたは相対パス。フォルダーパスは / で終わる必要があります。末尾が / でなければ、ファイルパスとして扱われます。モジュールは、ダウンロードを成功させるために必要なファイルまたはフォルダを作成します。式を連鎖させることは許可されます。	文字列	はい	該当なし

プリミティブ型	説明	タイプ	必須	デフォルト
overwrite	有効にすると、ローカルシステム上の既存のファイルを、ダウンロードしたファイルまたはリソースで上書きします。有効にしないと、ローカルシステム上の既存のファイルは上書きされず、アクションモジュールはエラーで失敗します。上書きが有効で、チェックサムとアルゴリズムが指定されている場合、アクションモジュールは、既存のファイルのチェックサムとハッシュが一致しない場合にのみファイルをダウンロードします。	ブール値	いいえ	true

プリミティブ型	説明	タイプ	必須	デフォルト
checksum	チェックサムを指定すると、指定されたアルゴリズムで生成されたダウンロードファイルのハッシュと照合されます。ファイル検証を有効にするには、チェックサムとアルゴリズムの両方を指定する必要があります。式を連鎖させることは許可されます。	文字列	いいえ	該当なし

プリミティブ型	説明	タイプ	必須	デフォルト
algorithm	チェックサムの計算に使用するアルゴリズム。オプションには MD5、SHA1、SHA256 および SHA512 があります。ファイル検証を有効にするには、チェックサムとアルゴリズムの両方を指定する必要があります。式を連鎖させることは許可されます。	文字列	いいえ	該当なし
ignoreCertificateErrors	SSL 証明書の検証は有効になっていると無視されます。	ブール値	いいえ	false

出力

プリミティブ型	説明	[Type] (タイプ)
destination	ダウンロードしたファイルまたはリソースの保存先パスを指定する改行文字で	文字列

プリミティブ型	説明	[Type] (タイプ)				
	区切られた文字列。					

入力例:リモートファイルをローカルの宛先にダウンロードする

```
  - name: DownloadRemoteFile
    action: WebDownload
    maxAttempts: 3
    inputs:
      - source: https://testdomain/path/to/java14.zip
        destination: C:\testfolder\package.zip
```

出力:

```
{
 "destination": "C:\\testfolder\\package.zip"
}
```

入力例:複数のリモートファイルを複数のローカル宛先にダウンロードする

```
  - name: DownloadRemoteFiles
    action: WebDownload
    maxAttempts: 3
    inputs:
      - source: https://testdomain/path/to/java14.zip
        destination: /tmp/java14_renamed.zip
      - source: https://testdomain/path/to/java14.zip
        destination: /tmp/create_new_folder_and_add_java14_as_zip/
```

出力:

```
{
 "destination": "/tmp/create_new_folder/java14_renamed.zip\n/tmp/
create_new_folder_and_add_java14_as_zip/java14.zip"
}
```

入力例:ローカル宛先を上書きせずにリモートファイルを 1 つダウンロードし、ファイル検証を行って別のリモートファイルをダウンロードする

```
  - name: DownloadRemoteMultipleProperties
    action: WebDownload
    maxAttempts: 3
    inputs:
      - source: https://testdomain/path/to/java14.zip
        destination: C:\create_new_folder\java14_renamed.zip
        overwrite: false
      - source: https://testdomain/path/to/java14.zip
        destination: C:\create_new_folder_and_add_java14_as_zip\
        checksum: ac68bbf921d953d1cfab916cb6120864
        algorithm: MD5
        overwrite: true
```

出力:

```
{
 "destination": "C:\\create_new_folder\\java14_renamed.zip\nC:\
\create_new_folder_and_add_java14_as_zip\\java14.zip"
}
```

入力例:リモートファイルをダウンロードし、SSL 証明書の検証を無視

```
  - name: DownloadRemoteIgnoreValidation
    action: WebDownload
    maxAttempts: 3
    inputs:
      - source: https://www.bad-ssl.com/resource
        destination: /tmp/downloads/
        ignoreCertificateErrors: true
```

出力:

```
{
 "destination": "/tmp/downloads/resource"
}
```

ファイルシステム操作モジュール

次のセクションでは、ファイルシステム操作のコマンドや命令を実行するアクションモジュールの詳細を説明します。

ファイルシステム操作アクションモジュール

- AppendFile
- CopyFile
- CopyFolder
- CreateFile
- CreateFolder
- CreateSymlink
- DeleteFile
- DeleteFolder
- ListFiles
- MoveFile
- MoveFolder
- ReadFile
- SetFileEncoding
- SetFileOwner
- SetFolderOwner
- SetFilePermissions
- SetFolderPermissions

AppendFile

AppendFile アクションモジュールは、指定されたコンテンツをファイルの既存のコンテンツに追加します。

ファイルエンコーディング値がデフォルトの encoding (`utf-8`) 値と異なる場合は、`encoding` オプションを使用してファイルエンコーディング値を指定できます。デフォルトでは `utf-16` と `utf-32` リトルエンディアンエンディアンエンコーディングを使用すると想定されています。

アクションモジュールは、以下の場合にエラーを返します。

- 指定したファイルは実行時には存在しません。
- ファイルの内容を変更するための書き込み権限がない。
- ファイル操作中にモジュールにエラーが発生した。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	ファイルパス。	文字列	はい	該当なし	該当なし	はい
content	ファイルに追加するコンテンツ。	文字列	いいえ	空の文字列	該当なし	はい
encoding	エンコード形式です。	文字列	いいえ	utf8	utf8、utf-LE、utf-16LE 、utf16-BE、utf-16BE 、utf32、LE、utf-32LE 、utf32-BEおよびutf-32-BE 。エンコーディングオプションの値は大文字と小文	はい

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
					字を区別しません。	

入力例:エンコードせずにファイルを追加 (Linux)

```
- name: AppendingFileWithOutEncodingLinux
  action: AppendFile
  inputs:
    - path: ./Sample.txt
      content: "The string to be appended to the file"
```

入力例:エンコーディングなしでファイルを追加 (Windows)

```
- name: AppendingFileWithOutEncodingWindows
  action: AppendFile
  inputs:
    - path: C:\MyFolder\MyFile.txt
      content: "The string to be appended to the file"
```

入力例:エンコーディング付きファイルの追加 (Linux)

```
- name: AppendingFileWithEncodingLinux
  action: AppendFile
  inputs:
    - path: /FolderName/SampleFile.txt
      content: "The string to be appended to the file"
      encoding: UTF-32
```

入力例:エンコーディング付きファイルの追加 (Windows)

```
- name: AppendingFileWithEncodingWindows
  action: AppendFile
  inputs:
```

```
      - path: C:\MyFolderName\SampleFile.txt
        content: "The string to be appended to the file"
        encoding: UTF-32
```

入力例:空の文字列を含むファイルの追加 (Linux)

```
- name: AppendingEmptyStringLinux
  action: AppendFile
  inputs:
    - path: /FolderName/SampleFile.txt
```

入力例:空の文字列を含むファイルの追加 (Windows)

```
- name: AppendingEmptyStringWindows
  action: AppendFile
  inputs:
    - path: C:\MyFolderName\SampleFile.txt
```

出力

なし。

CopyFile

CopyFile アクションモジュールは、指定された送信元から指定された送信先にファイルをコピーします。デフォルトでは、実行時に宛先フォルダが存在しない場合、モジュールは再帰的に宛先フォルダを作成します。

指定された名前のファイルが指定したフォルダにすでに存在する場合、アクションモジュールはデフォルトで既存のファイルを上書きします。上書きオプションをfalseに設定することで、このデフォルトの動作を上書きすることができます。上書きオプションが false に設定されていて、指定した場所に指定した名前のファイルがすでに存在する場合、アクションモジュールはエラーを返します。このオプションは Linux cp のコマンドと同じように機能し、デフォルトでは上書きされます。

ソースファイル名にはワイルドカード (*) を含めることができます。ワイルドカード文字は、最後のファイルパスの区切り文字 (/ または \) の後でのみ使用できます。ソースファイル名にワイルドカード文字が含まれている場合、ワイルドカードに一致するすべてのファイルが宛先フォルダーにコピーされます。ワイルドカード文字を使用して複数のファイルを移動する場合は、destination オプションへの入力の末尾にファイルパスの区切り文字 (/ または \) を付ける必要があります。これは、移動先の入力がフォルダーであることを示します。

移動先のファイル名がソースファイル名と異なる場合は、destination オプションを使用して移動先のファイル名を指定できます。宛先ファイル名を指定しない場合、ソースファイルの名前を使用して宛先ファイルが作成されます。最後のファイルパス区切り文字 (/ または \) に続くテキストはすべてファイル名として扱われます。ソースファイルと同じファイル名を使用する場合は、destination オプションの入力がファイルパスの区切り文字 (/ または \) で終わる必要があります。

アクションモジュールは、以下の場合にエラーを返します。

- 指定されたフォルダにファイルを作成する権限がない。
- ソースファイルは実行時には存在しません。
- 指定したファイル名のフォルダが既に存在し、overwrite オプションは false に設定されています。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
source	ソースファイルのパス。	文字列	はい	該当なし	該当なし	はい
destination	デスティネーションファイルパス。	文字列	はい	該当なし	該当なし	はい
overwrite	false に設定すると、指定した場所に指定した名前のファイルが	ブール値	いいえ	true	該当なし	はい

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
	すでにある場合でも、宛先ファイルは置き換えられません。					

入力例:ファイルのコピー (Linux)

```
- name: CopyingAFileLinux
  action: CopyFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/destinationFile.txt
```

入力例:ファイルのコピー (Windows)

```
- name: CopyingAFileWindows
  action: CopyFile
  inputs:
    - source: C:\MyFolder\Sample.txt
      destination: C:\MyFolder\destinationFile.txt
```

入力例:ソースファイル名を使用してファイルをコピーする (Linux)

```
- name: CopyingFileWithSourceFileNameLinux
  action: CopyFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/
```

入力例:ソースファイル名を使用してファイルをコピーする (Windows)

```
- name: CopyingFileWithSourceFileNameWindows
  action: CopyFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder\
```

入力例:ワイルドカード文字を使用してファイルをコピーする (Linux)

```
- name: CopyingFilesWithWildCardLinux
  action: CopyFile
  inputs:
    - source: /Sample/MyFolder/Sample*
      destination: /MyFolder/
```

入力例:ワイルドカード文字を使用してファイルをコピーする (Windows)

```
- name: CopyingFilesWithWildCardWindows
  action: CopyFile
  inputs:
    - source: C:\Sample\MyFolder\Sample*
      destination: C:\MyFolder\
```

入力例:上書きせずにファイルをコピーする (Linux)

```
- name: CopyingFilesWithoutOverwriteLinux
  action: CopyFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/destinationFile.txt
      overwrite: false
```

入力例:上書きせずにファイルをコピーする (Windows)

```
- name: CopyingFilesWithoutOverwriteWindows
  action: CopyFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder\destinationFile.txt
      overwrite: false
```

出力

なし。

CopyFolder

CopyFolder アクションモジュールは、指定されたソースから指定された宛先にフォルダをコピーします。destinationオプションの入力はコピーするフォルダであり、sourceオプションの入力はソースフォルダの内容をコピーするフォルダです。デフォルトでは、実行時に宛先フォルダが存在しない場合、モジュールは再帰的に宛先フォルダを作成します。

指定されたフォルダー内に指定された名前のフォルダーが既に存在する場合、アクションモジュールは、デフォルトで、既存のフォルダーを上書きします。上書きオプションをfalseに設定することで、このデフォルトの動作を上書きすることができます。上書きオプションが false に設定されていて、指定した場所に指定した名前のフォルダが既に存在する場合、アクションモジュールはエラーを返します。

ソースフォルダ名にはワイルドカード (*) を含めることができます。ワイルドカード文字は、最後のファイルパスの区切り文字 (/ または \) の後でのみ使用できます。コピー元フォルダ名にワイルドカード文字が含まれている場合、ワイルドカードに一致するすべてのフォルダがコピー先フォルダにコピーされます。ワイルドカード文字を使用して複数のフォルダーをコピーする場合、destination オプションへの入力は、コピー先の入力がフォルダーであることを示すファイルパス区切り記号 (/ または \) で終わる必要があります。

コピー先フォルダー名がソースフォルダー名と異なる場合は、destination オプションを使用してコピー先フォルダー名を指定できます。宛先フォルダ名を指定しない場合、ソースフォルダの名前が宛先フォルダの作成に使用されます。最後のファイルパスの区切り文字 (/ または \) に続くテキストはすべてフォルダー名として扱われます。ソースフォルダーと同じフォルダー名を使用する場合は、destination オプションの入力がファイルパスの区切り文字 (/ または \) で終わる必要があります。

アクションモジュールは、以下の場合にエラーを返します。

- 指定されたフォルダにフォルダを作成する権限がありません。
- ソースフォルダは実行時には存在しません。
- 指定したフォルダー名のフォルダーが既に存在し、overwrite オプションは false に設定されています。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
source	ソースフォルダーのパス。	文字列	はい	該当なし	該当なし	はい
destination	宛先フォルダーのパス。	文字列	はい	該当なし	該当なし	はい
overwrite	falseに設定すると、指定された場所に指定された名前のフォルダがすでにある場合、保存先フォルダは置き換えられません。	ブール値	いいえ	true	該当なし	はい

入力例:フォルダーのコピー (Linux)

```
- name: CopyingAFolderLinux
  action: CopyFolder
  inputs:
    - source: /Sample/MyFolder/SampleFolder
      destination: /MyFolder/destinationFolder
```

入力例:フォルダーのコピー (Windows)

```
- name: CopyingAFolderWindows
  action: CopyFolder
  inputs:
    - source: C:\Sample\MyFolder\SampleFolder
      destination: C:\MyFolder\destinationFolder
```

入力例:ソースフォルダー名を使用してフォルダーをコピーする (Linux)

```
- name: CopyingFolderSourceFolderNameLinux
  action: CopyFolder
  inputs:
    - source: /Sample/MyFolder/SourceFolder
      destination: /MyFolder/
```

入力例:ソースフォルダー名を使用してフォルダーをコピーする (Windows)

```
- name: CopyingFolderSourceFolderNameWindows
  action: CopyFolder
  inputs:
    - source: C:\Sample\MyFolder\SampleFolder
      destination: C:\MyFolder\
```

入力例:ワイルドカード文字を使用してフォルダをコピーする (Linux)

```
- name: CopyingFoldersWithWildCardLinux
  action: CopyFolder
  inputs:
    - source: /Sample/MyFolder/Sample*
      destination: /MyFolder/
```

入力例:ワイルドカード文字を使用してフォルダをコピーする (Windows)

```
- name: CopyingFoldersWithWildCardWindows
  action: CopyFolder
  inputs:
    - source: C:\Sample\MyFolder\Sample*
      destination: C:\MyFolder\
```

入力例:フォルダを上書きせずにコピーする (Linux)

```
- name: CopyingFoldersWithoutOverwriteLinux
  action: CopyFolder
  inputs:
    - source: /Sample/MyFolder/SourceFolder
      destination: /MyFolder/destinationFolder
      overwrite: false
```

入力例:フォルダを上書きせずにコピーする (Windows)

```
- name: CopyingFoldersWithoutOverwrite
  action: CopyFolder
  inputs:
    - source: C:\Sample\MyFolder\SourceFolder
      destination: C:\MyFolder\destinationFolder
      overwrite: false
```

出力

なし。

CreateFile

CreateFile アクションモジュールは、指定された場所にファイルを作成します。デフォルトでは、モジュールは必要に応じて親フォルダも再帰的に作成します。

ファイルが指定されたフォルダーにすでに存在する場合、アクションモジュールはデフォルトで、既存のファイルを切り捨てるか上書きする。上書きオプションをfalseに設定することで、このデフォルトの動作を上書きすることができます。上書きオプションが `false` に設定されていて、指定した場所に指定した名前のファイルがすでに存在する場合、アクションモジュールはエラーを返します。

ファイルエンコーディング値がデフォルトの encoding (`utf-8`) 値と異なる場合は、`encoding` オプションを使用してファイルエンコーディング値を指定できます。デフォルトでは `utf-16` と `utf-32` リトルエンディアンエンディアンエンコーディングを使用すると想定されています。

`owner`、`group` および `permissions` はオプションの入力です。`permissions` の入力は文字列値でなければなりません。指定しない場合、ファイルはデフォルト値で作成されます。これらのオプションは Windows プラットフォームではサポートされていません。Windows プラットフォームで、`owner`、`group` と `permissions` オプションが使用されている場合、このアクションモジュールは検証してエラーを返します。

このアクションモジュールは、オペレーティングシステムのデフォルト umask 値で定義されたパーミッションを持つファイルを作成することができます。デフォルト値をオーバーライドする場合、umask 値を設定する必要があります。

アクションモジュールは、以下の場合にエラーを返します。

- 指定された親フォルダにファイルまたはフォルダを作成する権限がありません。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	ファイルパス。	文字列	はい	該当なし	該当なし	はい
content	ファイルのテキストコンテンツ。	文字列	いいえ	該当なし	該当なし	はい
encoding	エンコード形式です。	文字列	いいえ	utf8	utf8、utf-LE、utf-16 LE 、utf16-BE、utf-16 BE 、utf32、 LE、utf-32 LE 、utf32-BEおよび utf-32-BE 。エンコーディン	はい

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
					グオプションの値は大文字と小文字を区別しません。	
owner	ユーザー名またはID。	文字列	いいえ	該当なし	該当なし	Windowsではサポートされていません。
group	グループ名またはID。	文字列	いいえ	現在のユーザー。	該当なし	Windowsではサポートされていません。
permissions	ファイルのパーミッション。	文字列	いいえ	0666	該当なし	Windowsではサポートされていません。

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
overwrite	指定したファイルの名前が既に存在する場合、この値をに設定すると、falseデフォルトでファイルが切り捨てられたり上書きされたりするのを防ぐことができます。	ブール値	いいえ	true	該当なし	はい

入力例:上書きせずにファイルを作成 (Linux)

```
- name: CreatingFileWithoutOverwriteLinux
  action: CreateFile
  inputs:
    - path: /home/UserName/Sample.txt
      content: The text content of the sample file.
      overwrite: false
```

入力例:上書きせずにファイルを作成 (Windows)

```
- name: CreatingFileWithoutOverwriteWindows
  action: CreateFile
  inputs:
    - path: C:\Temp\Sample.txt
```

```
        content: The text content of the sample file.
        overwrite: false
```

入力例:ファイルプロパティを含むファイルの作成

```
- name: CreatingFileWithFileProperties
  action: CreateFile
  inputs:
    - path: SampleFolder/Sample.txt
      content: The text content of the sample file.
      encoding: UTF-16
      owner: Ubuntu
      group: UbuntuGroup
      permissions: 0777
   - path: SampleFolder/SampleFile.txt
      permissions: 755
    - path: SampleFolder/TextFile.txt
      encoding: UTF-16
      owner: root
      group: rootUserGroup
```

入力例：ファイルのプロパティなしでファイルを作成する

```
- name: CreatingFileWithoutFileProperties
  action: CreateFile
  inputs:
    - path: ./Sample.txt
    - path: Sample1.txt
```

入力例：Linuxクリーンアップスクリプトのセクションをスキップするために空のファイルを作成する

```
- name: CreateSkipCleanupfile
  action: CreateFile
  inputs:
    - path: <skip section file name>
```

詳細については、「Linux クリーンアップスクリプトをオーバーライドしてください。」を参照してください。

出力

なし。

CreateFolder

CreateFolder アクションモジュールは、指定された場所にフォルダを作成します。デフォルトでは、モジュールは必要に応じて親フォルダも再帰的に作成します。

指定されたフォルダーにすでにフォルダーが存在する場合、アクションモジュールはデフォルトで、既存のフォルダーを切り捨てるか上書きします。上書きオプションをfalseに設定することで、このデフォルトの動作を上書きすることができます。上書きオプションが false に設定されていて、指定した場所に指定した名前のフォルダが既に存在する場合、アクションモジュールはエラーを返します。

owner、group および permissions はオプションの入力です。permissions の入力は文字列値でなければなりません。これらのオプションは Windows プラットフォームではサポートされていません。Windows プラットフォームで、owner、group と permissions オプションが使用されている場合、このアクションモジュールは検証して エラーを返します。

このアクションモジュールは、umask オペレーティングシステムのデフォルト値で定義された権限を持つフォルダを作成できます。デフォルト値をオーバーライドする場合、umask 値を設定する必要があります。

アクションモジュールは、以下の場合にエラーを返します。

- 指定された場所にフォルダを作成する権限がありません。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	フォルダパス。	文字列	はい	該当なし	該当なし	はい

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
owner	ユーザー名またはID。	文字列	いいえ	現在のユーザー。	該当なし	Windowsではサポートされていません。
group	グループ名またはID。	文字列	いいえ	現在のユーザーのグループ。	該当なし	Windowsではサポートされていません。
permissions	フォルダのパーミッション。	文字列	いいえ	0777	該当なし	Windowsではサポートされていません。
overwrite	指定したファイルの名前が既に存在する場合、この値をに設定すると、falseデフォルトでファイルが切り捨てられたり上書きされたりするのを防ぐことができます。	ブール値	いいえ	true	該当なし	はい

入力例:フォルダーの作成 (Linux)

```
- name: CreatingFolderLinux
  action: CreateFolder
  inputs:
    - path: /Sample/MyFolder/
```

入力例:フォルダーの作成 (Windows)

```
- name: CreatingFolderWindows
  action: CreateFolder
  inputs:
    - path: C:\MyFolder
```

入力例:フォルダーのプロパティを指定してフォルダーの作成

```
- name: CreatingFolderWithFolderProperties
  action: CreateFolder
  inputs:
    - path: /Sample/MyFolder/Sample/
      owner: SampleOwnerName
      group: SampleGroupName
      permissions: 0777
    - path: /Sample/MyFolder/SampleFoler/
      permissions: 777
```

入力例:既存のフォルダー (ある場合) を上書きするフォルダーを作成します。

```
- name: CreatingFolderWithOverwrite
  action: CreateFolder
  inputs:
    - path: /Sample/MyFolder/Sample/
      overwrite: true
```

出力

なし。

CreateSymlink

CreateSymlink アクションモジュールは、シンボリックリンク、または別のファイルへの参照を含むファイルを作成します。このモジュールは Windows プラットフォームではサポートされていません。

path および target オプションの入力は、絶対パスでも相対パスでもかまいません。path オプションの入力が相対パスの場合は、リンクの作成時に絶対パスに置き換えられます。

デフォルトでは、指定された名前のリンクが指定されたフォルダにすでに存在する場合、アクションモジュールからエラーが返されます。forceオプションをtrueに設定することで、このデフォルト動作をオーバーライドすることができます。force オプションを true に設定すると、モジュールは既存のリンクを上書きします。

親フォルダが存在しない場合、アクションモジュールはデフォルトでそのフォルダを再帰的に作成します。

アクションモジュールは、以下の場合に エラーを返します。

- ターゲットファイルが実行時に存在しません。
- 指定された名前の非シンボリックリンクファイルがすでに存在する。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	ファイルパス。	文字列	はい	該当なし	該当なし	Windows ではサポートされていません。
target	シンボリックリンク	文字列	はい	該当なし	該当なし	Windows ではサポー

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
	が指すターゲットファイルのパス。					トされていません。
force	同じ名前のリンクがすでに存在する場合、リンクの作成を強制します。	ブール値	いいえ	false	該当なし	Windows ではサポートされていません。

入力例:リンクの作成を強制するシンボリックリンクの作成

```
- name: CreatingSymbolicLinkWithForce
  action: CreateSymlink
  inputs:
    - path: /Folder2/Symboliclink.txt
      target: /Folder/Sample.txt
      force: true
```

入力例:リンクの作成を強制しないシンボリックリンクの作成

```
- name: CreatingSymbolicLinkWithOutForce
  action: CreateSymlink
  inputs:
    - path: Symboliclink.txt
      target: /Folder/Sample.txt
```

出力

なし。

DeleteFile

DeleteFile アクションモジュールは、指定された場所にあるファイルを削除します。

pathの入力は、有効なファイルパスか、ファイル名にワイルドカード文字(*)を含むファイルパスでなければならない。ファイル名にワイルドカード文字を指定すると、ワイルドカードに一致する同じフォルダー内のすべてのファイルが削除されます。

アクションモジュールは、以下の場合にエラーを返します。

- あなたには削除操作を実行する権限がありません。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	ファイルパス。	文字列	はい	該当なし	該当なし	はい

入力例:1 つのファイルを削除する (Linux)

```
- name: DeletingSingleFileLinux
  action: DeleteFile
  inputs:
    - path: /SampleFolder/MyFolder/Sample.txt
```

入力例:1 つのファイルを削除する (Windows)

```
- name: DeletingSingleFileWindows
  action: DeleteFile
  inputs:
    - path: C:\SampleFolder\MyFolder\Sample.txt
```

入力例:「log」で終わるファイルを削除する (Linux)

```
- name: DeletingFileEndingWithLogLinux
  action: DeleteFile
  inputs:
    - path: /SampleFolder/MyFolder/*log
```

入力例:「log」で終わるファイルを削除する (Windows)

```
- name: DeletingFileEndingWithLogWindows
  action: DeleteFile
  inputs:
    - path: C:\SampleFolder\MyFolder\*log
```

入力例:指定したフォルダ内のファイルをすべて削除する (Linux)

```
- name: DeletingAllFilesInAFolderLinux
  action: DeleteFile
  inputs:
    - path: /SampleFolder/MyFolder/*
```

入力例:指定したフォルダ内のファイルをすべて削除する (Windows)

```
- name: DeletingAllFilesInAFolderWindows
  action: DeleteFile
  inputs:
    - path: C:\SampleFolder\MyFolder\*
```

出力

なし。

DeleteFolder

DeleteFolder アクションモジュールはフォルダを削除します。

フォルダが空でない場合は、 フォルダとその内容を削除するforceオプションを true に設定する必要があります。force オプションを true に設定せず、削除しようとしているフォルダが空でない場合、アクションモジュールはエラーを返します。forceオプションのデフォルト値はfalseです。

アクションモジュールは、以下の場合にエラーを返します。

- あなたには削除操作を実行する権限がありません。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	フォルダパス。	文字列	はい	該当なし	該当なし	はい
force	フォルダーが空であろうとなかろうと、フォルダーを削除します。	ブール値	いいえ	false	該当なし	はい

入力例: **force** オプションを使用して空でないフォルダーを削除する (Linux)

```
- name: DeletingFolderWithForceOptionLinux
  action: DeleteFolder
  inputs:
    - path: /Sample/MyFolder/Sample/
      force: true
```

入力例: **force** オプションを使用して空でないフォルダーを削除する (Windows)

```
- name: DeletingFolderWithForceOptionWindows
  action: DeleteFolder
  inputs:
    - path: C:\Sample\MyFolder\Sample\
      force: true
```

入力例:フォルダーの削除 (Linux)

```
- name: DeletingFolderWithOutForceLinux
  action: DeleteFolder
  inputs:
    - path: /Sample/MyFolder/Sample/
```

入力例:フォルダーの削除 (Windows)

```
- name: DeletingFolderWithOutForce
  action: DeleteFolder
  inputs:
    - path: C:\Sample\MyFolder\Sample\
```

出力

なし。

ListFiles

ListFiles アクションモジュールは、指定されたフォルダ内のファイルを一覧表示します。recursive オプションをに設定すると `true`、サブフォルダ内のファイルが一覧表示されます。このモジュールは、デフォルトではサブフォルダー内のファイルを一覧表示しません。

指定したパターンに一致する名前のファイルをすべて一覧表示するには、`fileNamePattern` オプションを使用してパターンを指定します。`fileNamePattern` オプションはワイルドカード (*) 値を受け入れます。`fileNamePattern` を指定すると、指定されたファイル名形式に一致するすべてのファイルが返されます。

アクションモジュールは、以下の場合にエラーを返します。

- 指定されたフォルダが実行時に存在しません。
- 指定された親フォルダにファイルまたはフォルダを作成する権限がありません。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	フォルダパス。	文字列	はい	該当なし	該当なし	はい
fileNamePattern	一致するパターンで、そのパターンに一致する名前を持つすべてのファイルを一覧表示します。	文字列	いいえ	該当なし	該当なし	はい
recursive	フォルダー内のファイルを再帰的に一覧表示します。	ブール値	いいえ	false	該当なし	はい

入力例:指定したフォルダ内のファイルを一覧表示する (Linux)

```
- name: ListingFilesInSampleFolderLinux
  action: ListFiles
  inputs:
    - path: /Sample/MyFolder/Sample
```

入力例:指定したフォルダ内のファイルを一覧表示する (Windows)

```
- name: ListingFilesInSampleFolderWindows
  action: ListFiles
```

```
    inputs:
      - path: C:\Sample\MyFolder\Sample
```

入力例:「log」で終わるファイルを一覧表示する (Linux)

```
- name: ListingFilesWithEndingWithLogLinux
  action: ListFiles
  inputs:
    - path: /Sample/MyFolder/
      fileNamePattern: *log
```

入力例:「log」で終わるファイルを一覧表示する (Windows)

```
- name: ListingFilesWithEndingWithLogWindows
  action: ListFiles
  inputs:
    - path: C:\Sample\MyFolder\
      fileNamePattern: *log
```

入力例:ファイルを再帰的に一覧表示する

```
- name: ListingFilesRecursively
  action: ListFiles
  inputs:
    - path: /Sample/MyFolder/
      recursive: true
```

出力

プリミティブ型	説明	[Type] (タイプ)				
files	ファイルのリストです。	文字列				

出力例

```
{
```

```
  "files": "/sample1.txt,/sample2.txt,/sample3.txt"
}
```

MoveFile

MoveFile アクションモジュールは、指定されたソースから指定された宛先にファイルを移動します。

指定された名前のファイルが指定したフォルダにすでに存在する場合、アクションモジュールはデフォルトで既存のファイルを上書きします。上書きオプションをfalseに設定することで、このデフォルトの動作を上書きすることができます。上書きオプションが false に設定されていて、指定した場所に指定した名前のファイルがすでに存在する場合、アクションモジュールはエラーを返します。このオプションは Linux mv のコマンドと同じように機能し、デフォルトでは上書きされます。

ソースファイル名にはワイルドカード (*) を含めることができます。ワイルドカード文字は、最後のファイルパスの区切り文字 (/ または \) の後でのみ使用できます。ソースファイル名にワイルドカード文字が含まれている場合、ワイルドカードに一致するすべてのファイルが宛先フォルダーにコピーされます。ワイルドカード文字を使用して複数のファイルを移動する場合は、destination オプションへの入力の末尾にファイルパスの区切り文字 (/ または \) を付ける必要があります。これは、移動先の入力がフォルダーであることを示します。

移動先のファイル名がソースファイル名と異なる場合は、destination オプションを使用して移動先のファイル名を指定できます。宛先ファイル名を指定しない場合、ソースファイルの名前を使用して宛先ファイルが作成されます。最後のファイルパス区切り文字 (/ または \) に続くテキストはすべてファイル名として扱われます。ソースファイルと同じファイル名を使用する場合は、destination オプションの入力がファイルパスの区切り文字 (/ または \) で終わる必要があります。

アクションモジュールは、以下の場合に エラーを返します。

- 指定されたフォルダにファイルを作成する権限がない。
- ソースファイルは実行時には存在しません。
- 指定したファイル名のフォルダが既に存在し、overwrite オプションは false に設定されています。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
source	ソースファイルのパス。	文字列	はい	該当なし	該当なし	はい
destination	デスティネーションファイルパス。	文字列	はい	該当なし	該当なし	はい
overwrite	false に設定すると、指定した場所に指定した名前のファイルがすでにある場合でも、宛先ファイルは置き換えられません。	ブール値	いいえ	true	該当なし	はい

入力例:ファイルを移動する (Linux)

```
- name: MovingAFileLinux
  action: MoveFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/destinationFile.txt
```

入力例:ファイルを移動する (Windows)

```
- name: MovingAFileWindows
  action: MoveFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder\destinationFile.txt
```

入力例:ソースファイル名を使用してファイルを移動する (Linux)

```
- name: MovingFileWithSourceFileNameLinux
  action: MoveFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/
```

入力例:ソースファイル名を使用してファイルを移動する (Windows)

```
- name: MovingFileWithSourceFileNameWindows
  action: MoveFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder
```

入力例:ワイルドカード文字を使用してファイルを移動する (Linux)

```
- name: MovingFilesWithWildCardLinux
  action: MoveFile
  inputs:
    - source: /Sample/MyFolder/Sample*
      destination: /MyFolder/
```

入力例:ワイルドカード文字を使用してファイルを移動する (Windows)

```
- name: MovingFilesWithWildCardWindows
  action: MoveFile
  inputs:
    - source: C:\Sample\MyFolder\Sample*
      destination: C:\MyFolder
```

入力例:ファイルを上書きせずに移動する (Linux)

```
- name: MovingFilesWithoutOverwriteLinux
  action: MoveFile
  inputs:
    - source: /Sample/MyFolder/Sample.txt
      destination: /MyFolder/destinationFile.txt
      overwrite: false
```

入力例:ファイルを上書きせずに移動する (Windows)

```
- name: MovingFilesWithoutOverwrite
  action: MoveFile
  inputs:
    - source: C:\Sample\MyFolder\Sample.txt
      destination: C:\MyFolder\destinationFile.txt
      overwrite: false
```

出力

なし。

MoveFolder

MoveFolder アクションモジュールは、フォルダを指定された送信元から指定された送信先に移動します。sourceオプションの入力は移動するフォルダであり、destinationオプションの入力は移動元フォルダのコンテンツを移動するフォルダです。

実行時に移動先の親フォルダーまたは destination オプションへの入力が存在しない場合、モジュールのデフォルト動作では、指定された宛先にフォルダーを再帰的に作成します。

ソースフォルダーと同じフォルダーが宛先フォルダーにすでに存在する場合、アクションモジュールはデフォルトで、既存のフォルダーを上書きします。上書きオプションをfalseに設定することで、このデフォルトの動作を上書きすることができます。上書きオプションが false に設定されていて、指定した場所に指定した名前のフォルダが既に存在する場合、アクションモジュールはエラーを返します。

ソースフォルダ名にはワイルドカード (*) を含めることができます。ワイルドカード文字は、最後のファイルパスの区切り文字 (/ または \) の後でのみ使用できます。コピー元フォルダ名にワイルドカード文字が含まれている場合、ワイルドカードに一致するすべてのフォルダがコピー先フォルダにコピーされます。ワイルドカード文字を使用して複数のフォルダーをコピーする場合、destination オプションへの入力は、コピー先の入力がフォルダーであることを示すファイルパス区切り記号 (/ または \) で終わる必要があります。

コピー先フォルダー名がソースフォルダー名と異なる場合は、destination オプションを使用してコピー先フォルダー名を指定できます。宛先フォルダ名を指定しない場合、ソースフォルダの名前が宛先フォルダの作成に使用されます。最後のファイルパスの区切り文字 (/ または \) に続くテキストはすべてフォルダー名として扱われます。ソースフォルダーと同じフォルダー名を使用する場合は、destination オプションの入力がファイルパスの区切り文字 (/ または \) で終わる必要があります。

アクションモジュールは、以下の場合にエラーを返します。

- 保存先フォルダにフォルダを作成する権限がありません。
- ソースフォルダは実行時には存在しません。
- 指定した名前のフォルダが既に存在し、overwrite オプションは false に設定されています。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
source	ソースフォルダーのパス。	文字列	はい	該当なし	該当なし	はい
destination	宛先フォルダーのパス。	文字列	はい	該当なし	該当なし	はい
overwrite	falseに設定すると、指定された場所に指定された名前のフォルダがすでにある場合、保	ブール値	いいえ	true	該当なし	はい

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
	存先フォルダは置き換えられません。					

入力例:フォルダーの移動 (Linux)

```
- name: MovingAFolderLinux
  action: MoveFolder
  inputs:
    - source: /Sample/MyFolder/SourceFolder
      destination: /MyFolder/destinationFolder
```

入力例:フォルダーの移動 (Windows)

```
- name: MovingAFolderWindows
  action: MoveFolder
  inputs:
    - source: C:\Sample\MyFolder\SourceFolder
      destination: C:\MyFolder\destinationFolder
```

入力例:ソースフォルダー名を使用してフォルダーを移動する (Linux)

```
- name: MovingFolderWithSourceFolderNameLinux
  action: MoveFolder
  inputs:
    - source: /Sample/MyFolder/SampleFolder
      destination: /MyFolder/
```

入力例:ソースフォルダー名を使用してフォルダーを移動する (Windows)

```
- name: MovingFolderWithSourceFolderNameWindows
  action: MoveFolder
```

```
    inputs:
      - source: C:\Sample\MyFolder\SampleFolder
        destination: C:\MyFolder\
```

入力例:ワイルドカード文字を使用してフォルダを移動 (Linux)

```
- name: MovingFoldersWithWildCardLinux
  action: MoveFolder
  inputs:
    - source: /Sample/MyFolder/Sample*
      destination: /MyFolder/
```

入力例:ワイルドカード文字を使用してフォルダを移動する (Windows)

```
- name: MovingFoldersWithWildCardWindows
  action: MoveFolder
  inputs:
    - source: C:\Sample\MyFolder\Sample*
      destination: C:\MyFolder\
```

入力例:フォルダを上書きせずに移動する (Linux)

```
- name: MovingFoldersWithoutOverwriteLinux
  action: MoveFolder
  inputs:
    - source: /Sample/MyFolder/SampleFolder
  destination: /MyFolder/destinationFolder
  overwrite: false
```

入力例:フォルダを上書きせずに移動する (Windows)

```
- name: MovingFoldersWithoutOverwriteWindows
  action: MoveFolder
  inputs:
    - source: C:\Sample\MyFolder\SampleFolder
      destination: C:\MyFolder\destinationFolder
      overwrite: false
```

出力

なし。

ReadFile

ReadFile アクションモジュールは、文字列型のテキストファイルの内容を読み取ります。このモジュールを使うと、ファイルの内容を読み取ってチェーニングによって後続のステップで使用したり、データをconsole.logファイルに読み込んだりできます。指定されたパスがシンボリックリンクの場合、このモジュールはターゲットファイルの内容を返します。このモジュールはテキストファイルのみをサポートします。

ファイルエンコーディング値がデフォルトの encoding (utf-8) 値と異なる場合は、encoding オプションを使用してファイルエンコーディング値を指定できます。デフォルトでは utf-16 と utf-32 リトルエンディアンエンディアンエンコーディングを使用すると想定されています。

デフォルトでは、このモジュールは console.log ファイルの内容をファイルに出力できません。printFileContent プロパティを true に設定すると、この設定を上書きできます。

このモジュールはファイルの内容のみを返すことができます。Excel や JSON ファイルなどのファイルを解析することはできません。

アクションモジュールは、以下の場合にエラーを返します。

- 実行時にファイルが存在しません。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	ファイルパス。	文字列	はい	該当なし	該当なし	はい
encoding	エンコード形式です。	文字列	いいえ	utf8	utf8、utf-LE、utf-16 LE 、utf16-BE、utf-16	はい tf-16、

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
					BE 、utf32、u LE、utf-32 LE 、utf32-BEおよび utf-32-BE 。エンコーディングオプションの値は大文字と小文字を区別しません。	32-
printFileContent	ファイルの内容を console.log ファイルに出力します。	ブール値	いいえ	false	該当なし	はい。

入力例:ファイルの読み取り (Linux)

```
- name: ReadingFileLinux
  action: ReadFile
  inputs:
    - path: /home/UserName/SampleFile.txt
```

入力例：ファイルの読み込み（Windows）

```
  - name: ReadingFileWindows
    action: ReadFile
    inputs:
      - path: C:\Windows\WindowsUpdate.log
```

入力例：ファイルを読み込んでエンコーディングの標準を指定する

```
  - name: ReadingFileWithFileEncoding
    action: ReadFile
    inputs:
      - path: /FolderName/SampleFile.txt
        encoding: UTF-32
```

入力例:**console.log**ファイルを読み込んでファイルに出力する

```
  - name: ReadingFileToConsole
    action: ReadFile
    inputs:
      - path: /home/UserName/SampleFile.txt
        printFileContent: true
```

出力

フィールド	説明	[Type] (タイプ)
content	ファイルの内容。	string

出力例

```
{
 "content" : "The file content"
}
```

SetFileEncoding

SetFileEncoding アクションモジュールは、既存のファイルのエンコーディングプロパティを変更します。このモジュールは、utf-8 ファイルのエンコーディングを指定されたエンコーディング標準に変換できます。デフォルトでは、utf-16 と utf-32 リトルエンディアンエンディアンエンコーディングと想定されています。

アクションモジュールは、以下の場合にエラーを返します。

- 指定された変更を実行するにはアクセス許可が必要です。
- 実行時にファイルが存在しません。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	ファイルパス。	文字列	はい	該当なし	該当なし	はい
encoding	エンコード形式です。	文字列	いいえ	utf8	utf8、utf-LE、utf-16LE 、utf16-BE、utf-16BE 、utf32、u LE、utf-32LE 、utf32-BEおよびutf-32-BE 。エンコーディングオプションの値は大文字と小文字を区別しません。	はい

入力例:ファイルエンコーディングプロパティの設定

```
- name: SettingFileEncodingProperty
  action: SetFileEncoding
  inputs:
    - path: /home/UserName/SampleFile.txt
      encoding: UTF-16
```

出力

なし。

SetFileOwner

SetFileOwner アクションモジュールは、既存のファイルの ownerプロパティとgroup所有者プロパティを変更します。指定されたファイルがシンボリックリンクの場合、owner モジュールはソースファイルのプロパティを変更します。このモジュールは Windows プラットフォームではサポートされていません。

このモジュールは、ユーザー名とグループ名を入力として受け入れます。グループ名が指定されていない場合、モジュールはファイルのグループ所有者をユーザが所属するグループに割り当てます。

アクションモジュールは、以下の場合にエラーを返します。

- 指定された変更を実行するにはアクセス許可が必要です。
- 指定したユーザー名またはグループ名は実行時には存在しません。
- 実行時にファイルが存在しません。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	ファイルパス。	文字列	はい	該当なし	該当なし	Windows ではサポー

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
						トされていません。
owner	ユーザー名。	文字列	はい	該当なし	該当なし	Windows ではサポートされていません。
group	ユーザーグループの名前。	文字列	いいえ	ユーザーが所属するグループ名。	該当なし	Windows ではサポートされていません。

入力例:ユーザーグループの名前を指定せずにファイル所有者プロパティを設定

```
- name: SettingFileOwnerPropertyNoGroup
  action: SetFileOwner
  inputs:
    - path: /home/UserName/SampleText.txt
      owner: LinuxUser
```

入力例:所有者とユーザーグループを指定してファイル所有者プロパティを設定

```
- name: SettingFileOwnerProperty
  action: SetFileOwner
  inputs:
    - path: /home/UserName/SampleText.txt
      owner: LinuxUser
      group: LinuxUserGroup
```

出力

なし。

SetFolderOwner

SetFolderOwner アクションモジュールは、既存のフォルダの ownerプロパティとgroup所有者プロパティを再帰的に変更します。デフォルトでは、モジュールはフォルダ内のすべてのコンテンツの所有権を変更できます。この動作をオーバーライドする recursive オプションを false に設定できます。このモジュールは Windows プラットフォームではサポートされていません。

このモジュールは、ユーザー名とグループ名を入力として受け入れます。グループ名が指定されていない場合、モジュールはファイルのグループ所有者をユーザが所属するグループに割り当てます。

アクションモジュールは、以下の場合にエラーを返します。

- 指定された変更を実行するにはアクセス許可が必要です。
- 指定したユーザー名またはグループ名は実行時には存在しません。
- 実行時にフォルダが存在しません。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	フォルダパス。	文字列	はい	該当なし	該当なし	Windows ではサポートされていません。
owner	ユーザー名。	文字列	はい	該当なし	該当なし	Windows ではサポートされていません。
group	ユーザーグループの名前。	文字列	いいえ	ユーザーが所属するグループ名。	該当なし	Windows ではサポー

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
						トされていません。
recursive	false に設定すると、フォルダのすべてのコンテンツの所有権を変更するというデフォルトの動作が上書きされます。	ブール値	いいえ	true	該当なし	Windows ではサポートされていません。

入力例:ユーザーグループの名前を指定せずにフォルダー所有者プロパティを設定する

```
- name: SettingFolderPropertyWithOutGroup
  action: SetFolderOwner
  inputs:
    - path: /SampleFolder/
      owner: LinuxUser
```

入力例:フォルダ内のすべてのコンテンツの所有権を変更せずにフォルダ所有者プロパティを設定する

```
- name: SettingFolderPropertyWithOutRecursively
  action: SetFolderOwner
  inputs:
    - path: /SampleFolder/
      owner: LinuxUser
```

```
        recursive: false
```

入力例:ユーザーグループの名前を指定してファイル所有権プロパティを設定します

```
- name: SettingFolderPropertyWithGroup
  action: SetFolderOwner
  inputs:
    - path: /SampleFolder/
      owner: LinuxUser
      group: LinuxUserGroup
```

出力

なし。

SetFilePermissions

SetFilePermissions アクションモジュールは、既存のファイルの `permissions` を変更します。このモジュールは Windows プラットフォームではサポートされていません。

`permissions` の入力は文字列値でなければなりません。

このアクションモジュールは、オペレーティングシステムのデフォルト 値で定義された権限を使用してファイルを作成できます。デフォルト値をオーバーライドする場合、`umask` 値を設定する必要があります。

アクションモジュールは、以下の場合にエラーを返します。

- 指定された変更を実行するにはアクセス許可が必要です。
- 実行時にファイルが存在しません。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	ファイルパス。	文字列	はい	該当なし	該当なし	Windows ではサポートされていません。
permissions	ファイルのパーミッション。	文字列	はい	該当なし	該当なし	Windows ではサポートされていません。

入力例:ファイル権限の変更

```
- name: ModifyingFilePermissions
  action: SetFilePermissions
  inputs:
    - path: /home/UserName/SampleFile.txt
      permissions: 766
```

出力

なし。

SetFolderPermissions

SetFolderPermissions アクションモジュールは、既存のフォルダpermissionsとそのすべてのサブファイルとサブフォルダの を再帰的に変更します。デフォルトでは、このモジュールは指定されたフォルダのすべてのコンテンツの許可を変更できます。この動作をオーバーライドする recursive オプションを false に設定できます。このモジュールは Windows プラットフォームではサポートされていません。

permissions の入力は文字列値でなければなりません。

このアクションモジュールは、オペレーティングシステムのデフォルト umask 値に従って権限を変更できます。デフォルト値をオーバーライドする場合、umask 値を設定する必要があります。

アクションモジュールは、以下の場合にエラーを返します。

- 指定された変更を実行するにはアクセス許可が必要です。
- 実行時にフォルダが存在しません。
- 操作の実行中にアクションモジュールがエラーに遭遇しました。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
path	フォルダパス。	文字列	はい	該当なし	該当なし	Windows ではサポートされていません。
permissions	フォルダのパーミッション。	文字列	はい	該当なし	該当なし	Windows ではサポートされていません。
recursive	false に設定すると、フォルダのすべての内容の権限を変更するというデフォルトの動作が上書	ブール値	いいえ	true	該当なし	Windows ではサポートされていません。

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値	すべてのプラットフォームでサポートされています。
	きされます。					

入力例:フォルダ権限の設定

```
- name: SettingFolderPermissions
  action: SetFolderPermissions
  inputs:
    - path: SampleFolder/
      permissions: 0777
```

入力例：フォルダーのすべてのコンテンツのパーミッションを変更せずに、フォルダーのパーミッションを設定する。

```
- name: SettingFolderPermissionsNoRecursive
  action: SetFolderPermissions
  inputs:
    - path: /home/UserName/SampleFolder/
      permissions: 777
      recursive: false
```

出力

なし。

ソフトウェアインストールアクション

このセクションでは、ソフトウェアインストールアクションのコマンドと指示を実行するアクションモジュールについて説明します。

IAMの要件

インストールダウンロードパスが S3 URI の場合、インスタンスプロファイルに関連付ける IAM ロールには S3Download アクションモジュールを実行する権限が必要です。必要なアクセス権限を

付与するには、インスタンスプロファイルに関連付けられている S3:GetObject IAM ロールに IAM ポリシーをアタッチし、バケットのパスを指定します。例えば、*arn:aws:s3:::**BucketName**/**** です。

複雑な MSI 入力

入力文字列に二重引用符 (") が含まれている場合は、以下のいずれかの方法を使用して正しく解釈されるようにする必要があります。

- 次の例のように、文字列の外側には一重引用符 (') を使用し、文字列の内側には二重引用符 (「) を使用できます。

```
properties:
  COMPANYNAME: '"Acme ""Widgets"" and ""Gizmos."""'
```

  この場合、文字列の中でアポストロフィを使用する必要がある場合は、そのアポストロフィをエスケープする必要があります。つまり、アポストロフィの前に一重引用符 (') をもう1つ使うということです。

- 文字列の外側には二重引用符 (「) を使用して格納できます。また、次の例のように、バックスラッシュ文字 (\) を使用して、文字列内の二重引用符をエスケープできます。

```
properties:
  COMPANYNAME: "\"Acme \"\"Widgets\"\" and \"\"Gizmos.\"\"\""
```

これらのメソッドはいずれも、COMPANYNAME="Acme ""Widgets"" and ""Gizmos."""値をmsiexecコマンドに渡します。

ソフトウェアのインストールアクションモジュール

- インストール/MSI
- MSI をアンインストールします。

インストール/MSI

InstallMSI アクションモジュールは MSI ファイルを使用して Windows アプリケーションをインストールします。ローカルパス、S3 オブジェクト URI、またはウェブ URL を使用して MSI ファイルを指定できます。再起動オプションはシステムの再起動動作を設定します。

AWSTOE は、アクションモジュールの入力パラメータに基づいてmsiexecコマンドを生成します。path (MSI ファイルの場所) と logFile (ログファイルの場所) の入力パラメータの値は、引用符 (「) で囲む必要があります。

次の MSI 終了コードは成功とみなされます。

- 0 - 成功
- 1614 (製品_アンインストールエラー)
- 1641 (再起動が開始されました)
- 3010 (再起動が必要です)

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
path	次のいずれかを使用して MSI ファイルの場所を指定します。 • ローカルファイルのパス。パスは絶対パスでも相対パスでもかまいません。 • 有効な S3 オブジェクト URI。 • RFC 3986 標準に従っ	文字列	はい	該当なし	該当なし

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
	た有効なウェブHTTP/HTTPS URL（HTTPSを推奨）。 チェーン式は許可されています。				

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
reboot	アクションモジュールが正常に実行された後のシステム再起動動作を設定します。 設定： • Force — msiexec コマンドが正常に実行されたら、システムの再起動を開始します。 • Allow — msiexec コマンドが再起動が必要であることを示す終了コードを返した場合、システムの再起動を開始します。 • Skip — 再起動がス	文字列	いいえ	Allow	Allow, Force, Skip

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
	キップされたことを示す情報メッセージを console.log ファイルに記録します。このオプションは、msiexec コマンドが再起動が必要であることを示す終了コードを返した場合でも、再起動を防止します。				

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
logOptions	MSI インストールロギングに使用するオプションを指定します。指定されたフラグは、ロギングを有効にする /L コマンドラインパラメーターとともに MSI インストーラーに渡されます。フラグが指定されていない場合、はデフォルト値 AWSTOE を使用します。 MSIのLogオプションの詳細については、Microsoft Windows Installer製品ドキュメントのコマンドラインオプションを参照してください。	文字列	いいえ	*VX	i,w,e,a,r,u,c,m,o,p,v,x,+,!,*

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
logFile	ログファイルの場所への絶対パスまたは相対パス。Logファイルのパスが存在しない場合は、作成される。ログファイルパスが指定されていない場合、AWSTOE は MSI インストールログを保存しません。	文字列	いいえ	該当なし	該当なし

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
properties	MSI ロギングプロパティのキーと値のペア。例: TARGETDIR: "C:\target\location" 注:以下のプロパティは変更できません。 • REBOOT="ReallySupress" • REINSTALLMODE="ecmus" • REINSTALL="ALL"	Map[String]String	いいえ	該当なし	該当なし

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
ignoreAuthenticodeSignatureErrors	path で指定されたインストーラーの authenticode 署名検証エラーを無視するフラグ。このGet-AuthenticodeSignatureコマンドはインストーラーを検証するために使用されます。 設定： • true — 検証エラーは無視され、インストーラーが実行されます。 • false — 検証エラーは無視されません。インストーラーは、検証が成功した場合にの	ブール値	いいえ	false	true, false

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
	み実行されます。これがデフォルトの動作です。				

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
allowUnsignedInstaller	パスに指定された署名なしインストーラーの実行を許可するフラグ。このGet-AuthenticodeSignatureコマンドはインストーラーを検証するために使用されます。 設定： • true - Get-AuthenticodeSignature コマンドが返す NotSigned ステータスを無視し、インストーラを実行する。 • false — インストーラーへの署	ブール値	いいえ	false	true, false

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
	名が必要です。署名のないインストーラーは実行されません。これがデフォルトの動作です。				

例

以下の例は、コンポーネントドキュメントの入力セクションのバリエーションを示しています。

入力例：ローカルドキュメントパスのインストール。

```
- name: local-path-install
  steps:
    - name: LocalPathInstaller
      action: InstallMSI
      inputs:
        path: C:\sample.msi
        logFile: C:\msilogs\local-path-install.log
        logOptions: '*VX'
        reboot: Allow
        properties:
          COMPANYNAME: '"Amazon Web Services"'
        ignoreAuthenticodeSignatureErrors: true
        allowUnsignedInstaller: true
```

入力例： Amazon S3パスのインストール。

```
- name: s3-path-install
  steps:
    - name: S3PathInstaller
      action: InstallMSI
      inputs:
```

```
        path: s3://<bucket-name>/sample.msi
        logFile: s3-path-install.log
        reboot: Force
        ignoreAuthenticodeSignatureErrors: false
        allowUnsignedInstaller: true
```

入力例：ウェブパスのインストール。

```
- name: web-path-install
  steps:
    - name: WebPathInstaller
      action: InstallMSI
      inputs:
        path: https://<some-path>/sample.msi
        logFile: web-path-install.log
        reboot: Skip
        ignoreAuthenticodeSignatureErrors: true
        allowUnsignedInstaller: false
```

出力

次は InstallMSI アクションモジュールの出力の例です。

```
{
 "logFile": "web-path-install.log",
 "msiExitCode": 0,
 "stdout": ""
}
```

MSI をアンインストールします。

UninstallMSIアクションモジュールでは、MSIファイルを使用してWindowsアプリケーションを削除することができます。ローカルファイルパス、S3オブジェクトURI、またはウェブURLを使用して、MSIファイルの場所を指定できます。再起動オプションはシステムの再起動動作を設定します。

AWSTOE は、アクションモジュールの入力パラメータに基づいてmsiexecコマンドを生成します。MSI ファイルの場所 (path) とログファイルの場所 (logFile) は、msiexecコマンドの生成時に二重引用符 (「) で明示的に囲まれます。

次の MSI 終了コードは成功とみなされます。

- 0 - 成功

- 1605 (製品不明エラー)
- 1614 (製品_アンインストールエラー)
- 1641 (再起動が開始されました)
- 3010 (再起動が必要です)

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
path	次のいずれかを使用して MSI ファイルの場所を指定します。 • ローカルファイルのパス。パスは絶対パスでも相対パスでもかまいません。 • 有効な S3 オブジェクト URI。 • RFC 3986 標準に従った有効なウェブHTTP/HTTPS URL (HTTPS を推奨) 。	文字列	はい	該当なし	該当なし

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
	チェーン式は許可されています。				

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
reboot	アクションモジュールが正常に実行された後のシステム再起動動作を設定します。 設定： • Force — msiexec コマンドが正常に実行されたら、システムの再起動を開始します。 • Allow — msiexec コマンドが再起動が必要であることを示す終了コードを返した場合、システムの再起動を開始します。 • Skip — 再起動がス	文字列	いいえ	Allow	Allow, Force, Skip

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
	キップされたことを示す情報メッセージを console.log ファイルに記録します。このオプションは、msiexec コマンドが再起動が必要であることを示す終了コードを返した場合でも、再起動を防止します。				

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
logOptions	MSI インストールロギングに使用するオプションを指定します。指定されたフラグは、ロギングを有効にする /L コマンドラインパラメーターとともに MSI インストーラーに渡されます。フラグが指定されていない場合、はデフォルト値 AWSTOE を使用します。 MSIのLogオプションの詳細については、Microsoft Windows Installer製品ドキュメントのコマンドラインオプションを参照してください。	文字列	いいえ	*VX	i,w,e,a,r,u,c,m,o,p,v,x,+,!,*

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
logFile	ログファイルの場所への絶対パスまたは相対パス。Logファイルのパスが存在しない場合は、作成される。ログファイルパスが指定されていない場合、AWSTOE は MSI インストールログを保存しません。	文字列	いいえ	該当なし	該当なし

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
properties	MSI ロギングプロパティのキーと値のペア。例: TARGETDIR: "C:\target\location" 注:以下のプロパティは変更できません。 • REBOOT="ReallySupress" • REINSTALLMODE="ecmus" • REINSTALL="ALL"	Map[String]String	いいえ	該当なし	該当なし

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
ignoreAuthenticodeSignatureErrors	path で指定されたインストーラーの authenticode 署名検証エラーを無視するフラグ。このGet-AuthenticodeSignatureコマンドはインストーラーを検証するために使用されます。 設定： • true — 検証エラーは無視され、インストーラーが実行されます。 • false — 検証エラーは無視されません。インストーラーは、検証が成功した場合にの	ブール値	いいえ	false	true, false

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
	み実行されます。これがデフォルトの動作です。				

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
allowUnsignedInstaller	パスに指定された署名なしインストーラーの実行を許可するフラグ。このGet-AuthenticodeSignatureコマンドはインストーラーを検証するために使用されます。 設定： • true - Get-AuthenticodeSignature コマンドが返す NotSigned ステータスを無視し、インストーラを実行する。 • false — インストーラーへの署	ブール値	いいえ	false	true, false

プリミティブ型	説明	タイプ	必須	デフォルト値	許容値
	名が必要です。署名のないインストーラーは実行されません。これがデフォルトの動作です。				

例

以下の例は、コンポーネントドキュメントの入力セクションのバリエーションを示しています。

入力例:ローカルドキュメントパスインストールの削除

```
- name: local-path-uninstall
  steps:
    - name: LocalPathUninstaller
      action: UninstallMSI
      inputs:
        path: C:\sample.msi
        logFile: C:\msilogs\local-path-uninstall.log
        logOptions: '*VX'
        reboot: Allow
        properties:
          COMPANYNAME: '"Amazon Web Services"'
        ignoreAuthenticodeSignatureErrors: true
        allowUnsignedInstaller: true
```

入力例:Amazon S3 パスインストールの削除

```
- name: s3-path-uninstall
  steps:
    - name: S3PathUninstaller
      action: UninstallMSI
      inputs:
```

```
        path: s3://<bucket-name>/sample.msi
        logFile: s3-path-uninstall.log
        reboot: Force
        ignoreAuthenticodeSignatureErrors: false
        allowUnsignedInstaller: true
```

入力例:ウェブパスのインストールを削除

```
- name: web-path-uninstall
  steps:
    - name: WebPathUninstaller
      action: UninstallMSI
      inputs:
        path: https://<some-path>/sample.msi
        logFile: web-path-uninstall.log
        reboot: Skip
        ignoreAuthenticodeSignatureErrors: true
        allowUnsignedInstaller: false
```

出力

次は UninstallMSI アクションモジュールの出力の例です。

```
{
 "logFile": "web-path-uninstall.log",
 "msiExitCode": 0,
 "stdout": ""
}
```

システムアクションモジュール

次のセクションでは、ファイルシステムのアクションコマンドと命令を実行するアクションモジュールについて説明します。

システムアクションモジュール

- 再起動
- SetRegistry
- UpdateOS

再起動

再起動アクションモジュールはインスタンスを再起動します。再起動の開始を遅らせる設定可能なオプションがあります。デフォルトでは、delaySecondsは0に設定されています。つまり、遅延はありません。ステップタイムアウトは、インスタンスの再起動時には適用されないため、再起動アクションモジュールではサポートされていません。

アプリケーションが Systems Manager エージェントによって呼び出されると、終了コード (Windows 3010 の場合、Linux 194 の場合) がSystems Manager エージェントに渡されます。システムマネージャエージェントは、スクリプトからマネージドインスタンスを再起動するの説明に従ってシステムの再起動を処理します。

アプリケーションがホスト上でスタンドアロンプロセスとして呼び出された場合、現在の実行状態を保存し、再起動後にアプリケーションを再実行するように再起動後の自動実行トリガーを構成し、システムを再起動します。

再起動後の自動実行トリガー:

- Windows。 AWSTOE は SystemStartup で自動的に実行されるトリガーを含む Windows タスクスケジューラエントリを作成します。
- Linux。 AWSTOE はシステム再起動後に自動的に実行されるジョブを crontab に追加します。

```
@reboot /download/path/awstoe run --document s3://bucket/key/doc.yaml
```

このトリガーはアプリケーションの起動時にクリーンアップされます。

> **再試行**
>
> デフォルトでは、再試行の最大回数は Systems Manager CommandRetryLimit に設定されています。再起動回数が再試行制限を超えると、自動化は失敗します。Systems Manager エージェント設定ファイル (Mds.CommandRetryLimit) を編集して制限を変更できます。Systems Manager エージェントオープンソースの「Runtime Configuration」を参照してください。

Rebootアクションモジュールを使用するには、rebootexitcodeを含むステップ (例えば3010) に対して、アプリケーションバイナリをsudo userとして実行する必要があります。

Input (入力)

プリミティブ型	説明	タイプ	必須	デフォルト
delaySeconds	再起動を開始する前に、特定の時間だけ遅延させます。	整数	いいえ	0

入力例:再起動ステップ

```
- name: RebootStep
  action: Reboot
  onFailure: Abort
  maxAttempts: 2
  inputs:
    delaySeconds: 60
```

出力

なし。

再起動モジュールが完了すると、Image Builder はビルドの次のステップに進みます。

SetRegistry

SetRegistry アクションモジュールは入力のリストを受け入れ、指定されたレジストリキーの値を設定できます。レジストリキーが存在しない場合、定義されたパスに作成されます。この機能はWindowsのみに適用されます。

Input (入力)

プリミティブ型	説明	タイプ	必須
path	レジストリキーのパス。	文字列	はい
name	レジストリキーの名前。	文字列	はい

プリミティブ型	説明	タイプ	必須
value	レジストリキーの値。	文字列/数値/配列	はい
type	レジストリキーの値の型。	文字列	はい

サポートされているパスプレフィックス

- HKEY_CLASSES_ROOT / HKCR:
- HKEY_USERS / HKU:
- HKEY_LOCAL_MACHINE / HKLM:
- HKEY_CURRENT_CONFIG / HKCC:
- HKEY_CURRENT_USER / HKCU:

サポートされている 型

- BINARY
- DWORD
- QWORD
- SZ
- EXPAND_SZ
- MULTI_SZ

入力例:レジストリキー値の設定

```
- name: SetRegistryKeyValues
  action: SetRegistry
  maxAttempts: 3
  inputs:
    - path: HKLM:\SOFTWARE\MySoftWare
      name: MyName
      value: FirstVersionSoftware
      type: SZ
    - path: HKEY_CURRENT_USER\Software\Test
```

```
        name: Version
        value: 1.1
        type: DWORD
```

出力

なし。

UpdateOS

UpdateOSアクションモジュールは、Windows と Linux のアップデートをインストールするためのサポートを追加します。使用可能なすべてのアップデートがデフォルトでインストールされます。あるいは、インストールするアクションモジュール用に 1 つ以上の特定のアップデートのリストを設定することもできます。インストールから除外するアップデートを指定することもできます。

「含める」リストと「除外」リストの両方を指定した場合、生成されるアップデートのリストには、「含む」リストにリストされているもののうち、「除外」リストには含まれていないものだけが含まれる可能性があります。

> **Note**
>
> UpdateOS は Amazon Linux 2023 (AL2023) をサポートしていません。ベース AMI をすべてのリリースに付属する新しいバージョンに更新することをお勧めします。他の方法については、Amazon Linux 2023 User GuideのControl updates received from major and minor releasesを参照してください。

- Windows アップデートは、ターゲットマシンに設定されているアップデートソースからインストールされます。
- Linux アプリケーションは Linux プラットフォームでサポートされているパッケージマネージャーを確認し、yum または apt-get パッケージマネージャーを使用します。どちらもサポートされていない場合はエラーが返ります。UpdateOSアクションモジュールを実行するためのsudo権限を持っている必要があります。sudo 権限がない場合は、error.Input が返されます。

Input (入力)

プリミティブ型	説明	タイプ	必須
include		文字列リスト	いいえ

プリミティブ型	説明	タイプ	必須
	Windowsの場合、以下のように指定できる： • インストールできるアップデートのリストに含める Microsoft Knowledge Base (KB) 記事 ID を 1 つ以上含めてください。有効な形式は、KB1234567 または 1234567 です。 • ワイルドカード値 (*) を使用したアップデート名。有効な形式は、Security* または *Security* です。 Linux では、インストールするアップデートのリストに含めるパッケージを 1 つ以上指定できます。		

プリミティブ型	説明	タイプ	必須
exclude	Windowsの場合、以下のように指定できる： • インストールから除外する更新プログラムのリストに含める 1 つ以上の Microsoft Knowledge Base (KB) の記事 ID。有効な形式は、KB1234567 または 1234567 です。 • ワイルドカード値 (*) を使用したアップデート名。有効な形式は、Security* または *Security* です。 Linuxの場合、インストールするアップデートのリストから除外するパッケージを1つ以上指定できます。	文字列リスト	いいえ

入力例:Linux アップデートのインストールのサポートを追加

```
- name: UpdateMyLinux
  action: UpdateOS
  onFailure: Abort
  maxAttempts: 3
  inputs:
    exclude:
      - ec2-hibinit-agent
```

入力例:Windows 更新プログラムのインストールサポートの追加

```
- name: UpdateWindowsOperatingSystem
  action: UpdateOS
  onFailure: Abort
  maxAttempts: 3
  inputs:
    include:
      - KB1234567
      - '*Security*'
```

出力

なし。

AWSTOE run コマンドの入力を設定する

コマンドのコマンドライン入力を AWSTOE run効率化するために、コマンドパラメータとオプションの設定を.jsonファイル拡張子付きの JSON 形式の入力設定ファイルに含めることができます。AWSTOE は、次のいずれかの場所からファイルを読み取ることができます。

- ローカルファイルパス (*. /config.json*)。
- S3 バケット (*s3://<bucket-path>/<bucket-name>/config.json*)。

run コマンドを入力すると、--config パラメータを使用して入力設定ファイルを指定できます。例:

```
awstoe run --config <file-path>/config.json
```

入力設定ファイル

入力設定 JSON ファイルには、run コマンドパラメータとオプションで直接指定できるすべての設定のキーと値のペアが含まれています。入力設定ファイルと run コマンドの両方の設定をパラメーターまたはオプションとして指定する場合、次の優先順位が適用されます。

優先順位のルール

1. パラメータまたはオプション AWS CLIを介して のrunコマンドに直接提供される設定は、同じ設定の入力設定ファイルで定義されている値を上書きします。
2. 入力設定ファイル内の設定は、コンポーネントのデフォルト値を上書きします。
3. コンポーネントドキュメントに他の設定が渡されない場合、デフォルト値があれば、そのデフォルト値を適用できます。

このルールには、ドキュメントとパラメータという 2 つの例外があります。これらの設定は、入力設定とコマンドパラメータでは動作が異なります。入力設定ファイルを使用する場合は、これらのパラメータを run コマンドに直接指定しないでください。そうするとエラーが発生する。

コンポーネント設定

入力設定ファイルには次の設定が含まれます。ファイルを効率化するために、不要なオプション設定は省略できます。特に明記されていない限り、すべての設定はオプションです。

- cwIgnoreFailures (ブール値) — CloudWatch ログからのログ記録の失敗を無視します。
- cwLogGroup (文字列) – CloudWatch ログ`LogGroup`の名前。
- cwLogRegion (文字列) – CloudWatch ログに適用される AWS リージョン。
- cwLogStream (文字列) - `console.log`ファイルをストリーミングする AWSTOE 場所を指定する CloudWatch ログ`LogStream`の名前。
- documentS3BucketOwner (文字列) – S3 URI ベースのドキュメントのバケット所有者のアカウント ID。
- documents (オブジェクトの配列、必須) — AWSTOE run コマンドが実行中の YAML コンポーネントドキュメントを表す JSON オブジェクトの配列。少なくとも1つのコンポーネント文書を指定しなければならない。

  各オブジェクトは以下のフィールドで構成される：

  - パス (文字列、必須) — YAML コンポーネントドキュメントのファイルの場所。これは、次のいずれかである必要があります。
    - ローカルファイルパス (*./component-doc-example.yaml)*。

- S3 URI (s3://*bucket/key*)。
- Image Builder コンポーネントのビルドバージョン ARN (arn:aws:imagebuilder:us-west-*2:123456789012*:component/*my-example-component*/2021.12.02/1)。

- パラメータ (オブジェクトの配列) — キーと値のペアオブジェクトの配列で、それぞれがコンポーネントドキュメントを実行するときに run コマンドが渡すコンポーネント固有のパラメーターを表します。コンポーネントではパラメータはオプションです。コンポーネントドキュメントにはパラメータが定義される場合とされない場合があります。

  各オブジェクトは以下のフィールドで構成される：

  - 名前 (文字列、必須) — コンポーネントパラメータの名前。
  - 値 (文字列、必須) — コンポーネントドキュメントに渡される名前付きパラメータの値。

  コンポーネントパラメータの詳細については、で変数を定義して参照する AWSTOE ページの [パラメータ] セクションを参照してください。

- ExecutOnID (文字列) — 現在の run コマンドの実行に適用される固有の ID。この ID は出力ファイル名とログファイル名に含まれ、これらのファイルを一意に識別し、現在のコマンド実行にリンクします。この設定を省略すると、 は GUID AWSTOE を生成します。
- logDirectory (文字列) – がこのコマンド実行のすべてのログファイル AWSTOE を保存する宛先ディレクトリ。デフォルトでは、このディレクトリは以下の親ディレクトリの中にある：`TOE_<DATETIME>_<EXECUTIONID>`. ログディレクトリを指定しない場合、 は現在の作業ディレクトリ () AWSTOE を使用します.。
- logS3BucketName (文字列) – コンポーネントログが Amazon S3 に保存されている場合 (推奨)、は AWSTOE コンポーネントアプリケーションログをこのパラメータで指定された S3 バケットにアップロードします。
- logS3BucketOwner (文字列) – コンポーネントログが Amazon S3 に保存されている場合 (推奨)、これは がログファイルを AWSTOE 書き込むバケットの所有者アカウント ID です。
- logS3KeyPrefix (文字列) – コンポーネントログが Amazon S3 に保存されている場合 (推奨)、これはバケット内のログの場所の S3 オブジェクトキープレフィックスです。
- パラメータ (オブジェクトの配列) — 現在の run コマンド実行に含まれるすべてのコンポーネントにグローバルに適用されるパラメータを表すキー値のペアオブジェクトの配列。
  - name (文字列、必須) - グローバルパラメータの名前。
  - 値 (文字列、必須) — すべてのコンポーネントドキュメントに渡される名前付きパラメータの値。

- フェーズ (文字列) — YAML コンポーネントドキュメントから実行するフェーズを指定するカンマ区切りのリスト。コンポーネントドキュメントに追加のフェーズが含まれている場合、そのフェーズは実行されません。
- StateDirectory (文字列) — ステートトラッキングファイルが保存されているファイルパス。
- トレース (ブール値) — コンソールへの冗長ロギングを有効にする。

例

次の例は、2つのコンポーネント文書に対してbuildとtestのフェーズを実行する入力設定ファイルを示している： sampledoc.yamlとconversation-intro.yamlです。各コンポーネントドキュメントには、それ自体にのみ適用されるパラメータがあり、どちらも 1 つの共有パラメータを使用します。この project パラメータは両方のコンポーネントドキュメントに適用されます。

```
{
   "documents": [
     {
       "path": "<file path>/awstoe/sampledoc.yaml>",
       "parameters": [
         {
           "name": "dayofweek",
           "value": "Monday"
         }
       ]
     },
     {
       "path": "<file path>/awstoe/conversation-intro.yaml>",
       "parameters": [
         {
           "name": "greeting",
           "value": "Hello, HAL."
         }
       ]
     }
   ],
   "phases": "build,test",
   "parameters": [
     {
       "name": "project",
       "value": "examples"
     }
   ],
```

```
  "cwLogGroup": "<log_group_name>",
  "cwLogStream": "<log_stream_name>",
  "documentS3BucketOwner": "<owner_aws_account_number>",
  "executionId": "<id_number>",
  "logDirectory": "<local_directory_path>",
  "logS3BucketName": "<bucket_name_for_log_files>",
  "logS3KeyPrefix": "<key_prefix_for_log_files>",
  "logS3BucketOwner": "<owner_aws_account_number>"
}
```

Windows 用ディストリビューターパッケージ管理コンポーネント

AWS Systems Manager Distributor は、ソフトウェアをパッケージ化して AWS Systems Manager マネージドノードに公開するのに役立ちます。独自のソフトウェアをパッケージ化して公開したり、Distributor を使用して AWSから提供されるエージェントソフトウェアパッケージを検索して公開することができます。Systems Manager Distributorの詳細については、AWS Systems Manager User GuideのAWS Systems Manager Distributorを参照してください。

Distributor の管理コンポーネント

次の Image Builder マネージドコンポーネントは、Distributor AWS Systems Manager を使用して Windows インスタンスにアプリケーションパッケージをインストールします。

- `distributor-package-windows` 管理コンポーネントは AWS Systems Manager Distributor を使用して、Windows イメージのビルドインスタンスで指定したアプリケーションパッケージをインストールします。このコンポーネントをレシピに含めるときにパラメータを設定するには、「スタンドアロンコンポーネントとして distributor-package-windows を設定します。」を参照してください。
- `aws-vss-components-windows` コンポーネントは Distributor AWS Systems Manager を使用して Windows イメージビルドインスタンスに `AwsVssComponents` パッケージをインストールします。このコンポーネントをレシピに含めるときにパラメータを設定するには、「スタンドアロンコンポーネントとして aws-vss-components-windows を設定します。」を参照してください。

Image Builder レシピで管理コンポーネントを使用する方法の詳細については、イメージレシピの新しいバージョンを作成します。 (イメージレシピ用) または 新しいイメージレシピのバージョンを作成 (コンテナレシピ用) を参照してください。`AwsVssComponents`パッケージの詳細については、Amazon EC2 User Guide for Windows InstancesのCreate a VSS application-consistent snapshotを参照してください。

前提条件

Systems Manager Distributor に依存する Image Builder コンポーネントを使用してアプリケーションパッケージをインストールする前に、次の前提条件が満たされていることを確認する必要があります。

- Systems Manager Distributor を使用してインスタンスにアプリケーションパッケージをインストールする Image Builder コンポーネントには、Systems Manager API を呼び出す権限が必要です。Image Builder レシピのコンポーネントを使用する前に、許可を付与する IAM ポリシーとロールを作成する必要があります。許可を設定するには、Systems Manager Distributor の権限設定 を参照してください。

> **Note**
>
> Image Builder は現在、インスタンスを再起動する Systems Manager Distributor パッケージをサポートしていません。たとえば、`AWSNVMe`、`AWSPVDrivers`、および `AwsEnaNetworkDriver` Distributor パッケージはインスタンスを再起動するため、許可されません。

Systems Manager Distributor の権限設定

`distributor-package-windows` コンポーネントとそれを使用する他のコンポーネント (`aws-vss-components-windows` など) を実行するには、ビルドインスタンスに対する追加の権限が必要です。ビルドインスタンスは Systems Manager API を呼び出して Distributor のインストールを開始し、結果をポーリングできる必要があります。

の手順に従って、Image Builder コンポーネント AWS Management Console がビルドインスタンスから Systems Manager Distributor パッケージをインストールするアクセス許可を付与するカスタム IAM ポリシーとロールを作成します。

ステップ 1: ポリシーの作成

Distributor 権限用の IAM ポリシーを作成します。

1. https://console.aws.amazon.com/iam/ で IAM コンソール を開きます。
2. ナビゲーションペインで ポリシーを選択してから ポリシーの作成を選択します。

3. 「ポリシーの作成」ページの [JSON] タブを選択し、デフォルトの内容を次の JSON ポリシーに置き換えます。必要に応じてパーティション、リージョン、アカウント ID に置き換えるか、ワイルドカードを使用します。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowDistributorSendCommand",
   "Effect": "Allow",
   "Action": [
    "ssm:SendCommand"
   ],
   "Resource": [
    "arn:${AWS::Partition}:ssm:${AWS::Region}::document/AWS-ConfigureAWSPackage",
    "arn:${AWS::Partition}:ec2:${AWS::Region}:${AWS::AccountId}:instance/*"
   ]
  },
  {
   "Sid": "AllowGetCommandInvocation",
   "Effect": "Allow",
   "Action": [
    "ssm:GetCommandInvocation"
   ],
   "Resource": [
    "*"
   ]
  }
 ]
}
```

4. [Review policy] (ポリシーの確認) を選択します。
5. [名前] に、ポリシーの識別名 (*InvokeDistributor* など) を入力するか、希望する別の名前を入力します。
6. (オプション) [説明] に、ロールの目的の説明を入力します。
7. [Create policy] を選択します。

ステップ 2: ロールの作成

Distributor 権限用の IAM ロールを作成します。

1. IAM コンソールのナビゲーションペインから、[ロール]、[ロールの作成] の順にクリックします。
2. [Select type of trusted entity] (信頼されたエンティティの種類を選択) で、[AWS のサービス] を選択します。
3. [このロールを使用するサービスを選択] のすぐ下で、[EC2]、[次へ: アクセス許可)] を選択します。
4. [ユースケースの選択] で、[EC2]、[次へ: アクセス許可)] の順に選択します。
5. ポリシーのリストで、AmazonSSMManagedInstanceCore の横にあるチェックボックスをオンにします。(リストを絞り込むには、検索ボックスに`SSM`と入力します)。
6. このポリシーのリストで、EC2InstanceProfileForImageBuilder の横にあるボックスを選択します。(リストを絞り込むには、検索ボックスに`ImageBuilder`と入力します)。
7. [次へ: タグ] を選択します。
8. (オプション) 1 つまたは複数のタグキー値のペアを追加して、このロールのアクセスを整理、追跡、または制御し、[次へ: 確認] () を選択します。
9. [ロール名] に、ロールの名前 (*`InvokeDistributor`* など) を入力するか、希望する別の名前を入力します。
10. (オプション) [ロールの説明] で、デフォルトのテキストをこのロールの目的の説明に置き換えます。
11. [ロールの作成] を選択します。ロールページが再度表示されます。

ステップ 3: ポリシーをロールにアタッチする

Distributor のアクセス許可を設定する最後のステップは、IAM ロールに IAM ポリシーをアタッチすることです。

1. IAM コンソールの [ロール] () ページで、作成したロールを選択します。ロールの 概要ページが表示されます。
2. [ポリシーのアタッチ] を選択します。
3. 前の手順で作成したポリシーを検索して、名前の隣にあるチェックボックスを選択します。
4. Attach policy] (ポリシーのアタッチ) を選択します。

Systems Manager Distributor を使用するコンポーネントを含むすべてのイメージの Image Builder インフラストラクチャー構成リソースでこのロールを使用してください。詳細については、「インフラストラクチャ構成を作成します。」を参照してください。

スタンドアロンコンポーネントとして distributor-package-windows を設定します。

レシピの distributor-package-windows コンポーネントを使用するには、インストールするパッケージを設定する以下のパラメータを設定します。

> **Note**
>
> distributor-package-windows コンポーネントをレシピで使用する前に、すべての 前提条件 が満たされていることを確認する必要があります。

- アクション (必須) — パッケージをインストール / アンインストールを指定します。有効な値は、Install および Uninstall です。デフォルト値は Install です。
- PackageName (必須) – インストールまたはアンインストールする Distributor パッケージの名前。有効なパッケージ名のリストについては、「Distributor パッケージの検索」を参照してください。
- PackageVersion (オプション) – インストールする Distributor パッケージのバージョン。PackageVersion デフォルトは推奨バージョンです。
- AdditionalArguments (オプション) – パッケージをインストール、アンインストール、または更新するためにスクリプトに提供する追加のパラメータを含む JSON 文字列。詳細については、[Systems Manager コマンドドキュメントプラグインリファレンス] ページの aws:configurePackage [入力] セクションにある additionalArguments を参照してください。

スタンドアロンコンポーネントとして aws-vss-components-windows を設定します。

aws-vss-components-windows コンポーネントをレシピで使用する場合、オプションで特定のバージョンの AwsVssComponents パッケージを使用するように PackageVersion パラメータを設定できます。このパラメータを省略すると、コンポーネントはデフォルトで推奨バージョンの AwsVssComponents パッケージを使用するようになります。

> **Note**
>
> aws-vss-components-windows コンポーネントをレシピで使用する前に、すべての 前提条件 が満たされていることを確認する必要があります。

Distributor パッケージの検索

Amazon とサードパーティは、Systems Manager Distributor でインストールできるパブリックパッケージを提供しています。

で利用可能なパッケージを表示するには AWS Management Console、AWS Systems Manager コンソールにログインし、ナビゲーションペインから Distributor を選択します。[Distributor] ページには、入手可能なすべてのパッケージが表示されます。で利用可能なパッケージのリスト化の詳細については AWS CLI、「 ユーザーガイド」の「パッケージの表示 (コマンドライン) AWS Systems Manager 」を参照してください。

独自のプライベート Systems Manager Distributor パッケージを作成することもできます。詳細については、AWS Systems Manager User GuideのCreate a packageを参照してください。

CIS Fardening のコンポーネント

Center for Internet Security (CIS) は、コミュニティ主導の非営利組織です。サイバーセキュリティの専門家が協力して、公的機関と民間組織をサイバー脅威から守る IT セキュリティガイドラインを策定しています。CIS Benchmarks と呼ばれる、世界的に認められた一連のベストプラクティスは、世界中の IT 組織がシステムを安全に構成できるよう支援しています。トレンド記事、ブログ記事、ポッドキャスト、ウェビナー、ホワイトペーパーについては、Center for Internet Securityウェブサイトの CIS Insightsをご覧ください。

CIS ベンチマーク

CIS は、オペレーティングシステム、クラウドプラットフォーム、アプリケーション、データベースなど、特定のテクノロジーに関する設定のベストプラクティスを提供する CIS ベンチマークと呼ばれる設定ガイドラインを作成および管理しています。CIS ベンチマークは、PCI DSS、HIPAA、DoD クラウドコンピューティング SRG、FISMA、DFARS、FEDRAMP などの組織や標準によって業界標準として認められています。詳細については、インターネットセキュリティセンタ ー Web サイトの「CIS ベンチマーク」を参照してください。

CIS Fardening のコンポーネント

で CIS 強化イメージをサブスクライブすると AWS Marketplace、設定に CIS Benchmarks Level 1 ガイドラインを適用するためのスクリプトを実行する、関連する強化コンポーネントにもアクセスできます。CIS 組織は CIS 強化コンポーネントを所有、保守し、最新のガイドラインに確実に反映されるようにしています。

> **Note**
>
> CIS 強化コンポーネントは、Image Builder レシピの標準コンポーネント順序ルールに従っていません。CIS 強化コンポーネントは常に最後に実行され、ベンチマークテストが出力イメージに対して確実に実行されます。

EC2 Image Builder用の Amazon managed STIG 強化コンポーネント

セキュリティ技術実装ガイド（STIGs）は、情報システムとソフトウェアを保護するために国防情報システム局（DISA）によって作成された構成強化基準です。システムを STIG 標準に準拠させるには、さまざまなセキュリティ設定をインストール、設定、およびテストする必要があります。

Image Builder には STIG 強化コンポーネントが用意されており、ベースラインとなる STIG 標準に準拠したイメージをより効率的に構築できます。これらのSTIGコンポーネントは、設定ミスをスキャンし、修正スクリプトを実行します。STIG準拠のコンポーネントを使用することによる追加料金は発生しません。

> **Important**
>
> いくつかの例外を除いて、STIG 強化コンポーネントはサードパーティのパッケージをインストールしません。サードパーティのパッケージがインスタンスにすでにインストールされていて、Image Builder がそのパッケージをサポートしている関連する STIG がある場合は、強化コンポーネントがそれらを適用します。

このページには、Image Builder がサポートするすべての STIG が一覧表示されており、新しいイメージをビルドしてテストするときに Image Builder が起動する EC2 インスタンスに適用されます。イメージに追加の STIG 設定を適用したい場合は、カスタムコンポーネントを作成して設定することができます。カスタムコンポーネントを作成する方法の詳細については、「Image Builder によるコンポーネントの管理」を参照してください。

イメージを作成すると、STIG 強化コンポーネントは、サポートされている STIG が適用されたかスキップされたかを記録します。STIG 強化コンポーネントを使用するイメージの Image Builder ログを確認することをお勧めします。Image Builder ログにアクセスして確認する方法については、「パイプラインビルドのトラブルシューティング」を参照してください。

コンプライアンスレベル

- 高 (カテゴリ I)

  最も深刻なリスクです。機密性、可用性、または整合性の損失につながる可能性のある脆弱性が含まれます。
- ミディアム (カテゴリ II)

  機密性、可用性、完全性が失われる可能性があるが、そのリスクを軽減できる脆弱性を含まれます。
- 低 (カテゴリー III)

  機密性、可用性、または整合性の喪失から保護するための対策を低下させる脆弱性。

トピック

Windows の STIG 強化コンポーネント

AWSTOE Windows STIG 強化コンポーネントはスタンドアロンサーバー用に設計されており、ローカルグループポリシーを適用します。STIG 準拠の強化コンポーネントは、国防総省 (DoD) InstallRoot から Windows インフラストラクチャにインストールされ、DoD 証明書をダウンロード、インストール、更新します。また、STIG への準拠を維持するために不要な証明書も削除します。現在、STIG ベースラインは 2012 R2、2016、2019、2022 の Windows サーバーのバージョンでサポートされています。

このセクションには、Windows STIG の各強化コンポーネントの現在の設定が一覧表示され、その後にバージョン履歴ログが続きます。

STIG-Build-Windows-Low バージョン2022.4.x

次のリストには、コンポーネント強化がインフラストラクチャーに適用される STIG 設定が含まれています。サポートされている設定がご使用のインフラストラクチャに当てはまらない場合、強化コン

ポーネントはその設定をスキップして次に進みます。例えば、一部の STIG 設定は、スタンドアロンサーバーには適用されない場合があります。組織固有のポリシーは、管理者が文書設定をレビューするための要件など、堅牢化コンポーネントが適用する設定にも影響します。

Windows 向け STIG の完全なリストについては、「STIG ドキュメントライブラリ」を参照してください。完全なリストを表示する方法の詳細については、「STIG 表示ツール」を参照してください。

- Windows Server 2022 STIG バージョン 1 リリース 1

  V-254335、V-254336、V-254337、V-254338、V-254351、V-254357、V-254363 および V-254481

- Windows Server 2019 STIG バージョン 2 リリース 5

  V-205691、V-205819、V-205858、V-205859、V-205860、V-205870、V-205871、および V-205923

- Windows Server 2016 STIG バージョン 2 リリース 5

  V-224916、V-224917、V-224918、V-224919、V-224931、V-224942、および V-225060

- Windows Server 2012 R2 MS STIG バージョン 3 リリース 5

  V-225537、V-225536、V-225526、V-225525、V-225514、V-225511、V-225490、V-225489、V-225488 および V-225250

- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 2

  Microsoft .NET フレームワークには、カテゴリ III の脆弱性に対応する STIG 設定は適用されません。

- Windows ファイアウォール STIG バージョン 2 リリース 1

  V-241994、V-241995、V-241996、V-241999、V-242000、V-242001、V-242006、V-242007 および V-242008

- Internet Explorer 11 STIG バージョン 2 リリース 3

  V-46477、V-46629、V-97527

- Microsoft Edge STIG バージョン 1 リリース 6 (Windows Server 2022 のみ)

  V-235727, V-235731, V-235751, V-235752, および V-235765

STIG-Build-Windows-Medium バージョン2022.4.x

次のリストには、コンポーネント強化がインフラストラクチャーに適用される STIG 設定が含まれています。サポートされている設定がご使用のインフラストラクチャに当てはまらない場合、強化コンポーネントはその設定をスキップして次に進みます。例えば、一部の STIG 設定は、スタンドアロンサーバーには適用されない場合があります。組織固有のポリシーは、管理者が文書設定をレビューするための要件など、堅牢化コンポーネントが適用する設定にも影響します。

Windows 向け STIG の完全なリストについては、「STIG ドキュメントライブラリ」を参照してください。完全なリストを表示する方法の詳細については、「STIG 表示ツール」を参照してください。

> **Note**
>
> STIG-Build-Windows-Medium 強化コンポーネントには、カテゴリ II の脆弱性専用にリストされている STIG 設定に加えて、STIG-Build-Windows-Low 強化コンポーネント AWSTOE に適用されるすべてのリストされた STIG 設定が含まれます。

- Windows Server 2022 STIG バージョン 1 リリース 1

  Category III (Low) の脆弱性に適用される、サポートされている STIG 設定に加えて、以下が含まれます。

  V-254247, V-254265, V-254269, V-254270, V-254271, V-254272, V-254273, V-254274, V-254276, V-254277, V-254278, V-254285, V-254286, V-254287, V-254288, V-254289, V-254290, V-254291, V-254292, V-254300, V-254301, V-254302, V-254303, V-254304, V-254305, V-254306, V-254307, V-254308, V-254309, V-254310, V-254311, V-254312, V-254313, V-254314, V-254315, V-254316, V-254317, V-254318, V-254319, V-254320, V-254321, V-254322, V-254323, V-254324, V-254325, V-254326, V-254327, V-254328, V-254329, V-254330, V-254331, V-254332, V-254333, V-254334, V-254339, V-254341, V-254342, V-254344, V-254345, V-254346, V-254347, V-254348, V-254349, V-254350, V-254355, V-254356, V-254358, V-254359, V-254360, V-254361, V-254362, V-254364, V-254365, V-254366, V-254367, V-254368, V-254369, V-254370, V-254371, V-254372, V-254373, V-254375, V-254376, V-254377, V-254379, V-254380, V-254382, V-254383, V-254431, V-254432, V-254433, V-254434, V-254435, V-254436, V-254438, V-254439, V-254442, V-254443, V-254444, V-254445, V-254449, V-254450, V-254451, V-254452, V-254453, V-254454, V-254455, V-254456, V-254459, V-254460, V-254461, V-254462, V-254463, V-254464, V-254468, V-254470, V-254471, V-254472, V-254473, V-254476, V-254477, V-254478, V-254479, V-254480, V-254482, V-254483, V-254484, V-254485, V-254486, V-254487, V-254488, V-254489, V-254490, V-254493, V-254494,

V-254495, V-254497, V-254499, V-254501, V-254502, V-254503, V-254504, V-254505, V-254507, V-254508, V-254509, V-254510, V-254511, および V-254512

- Windows Server 2019 STIG バージョン 2 リリース 5

  Category III (Low) の脆弱性に適用される、サポートされている STIG 設定に加えて、以下が含まれます。

  V-205625, V-205626, V-205627, V-205629, V-205630, V-205633, V-205634, V-205635, V-205636, V-205637, V-205638, V-205639, V-205643, V-205644, V-205648, V-205649, V-205650, V-205651, V-205652, V-205655, V-205656, V-205659, V-205660, V-205662, V-205671, V-205672, V-205673, V-205675, V-205676, V-205678, V-205679, V-205680, V-205681, V-205682, V-205683, V-205684, V-205685, V-205686, V-205687, V-205688, V-205689, V-205690, V-205692, V-205693, V-205694, V-205697, V-205698, V-205708, V-205709, V-205712, V-205714, V-205716, V-205717, V-205718, V-205719, V-205720, V-205722, V-205729, V-205730, V-205733, V-205747, V-205751, V-205752, V-205754, V-205756, V-205758, V-205759, V-205760, V-205761, V-205762, V-205764, V-205765, V-205766, V-205767, V-205768, V-205769, V-205770, V-205771, V-205772, V-205773, V-205774, V-205775, V-205776, V-205777, V-205778, V-205779, V-205780, V-205781, V-205782, V-205783, V-205784, V-205795, V-205796, V-205797, V-205798, V-205801, V-205808, V-205809, V-205810, V-205811, V-205812, V-205813, V-205814, V-205815, V-205816, V-205817, V-205821, V-205822, V-205823, V-205824, V-205825, V-205826, V-205827, V-205828, V-205830, V-205832, V-205833, V-205834, V-205835, V-205836, V-205837, V-205838, V-205839, V-205840, V-205841, V-205861, V-205863, V-205865, V-205866, V-205867, V-205868, V-205869, V-205872, V-205873, V-205874, V-205911, V-205912, V-205915, V-205916, V-205917, V-205918, V-205920, V-205921, V-205922, V-205924, V-205925, および V-236001

- Windows Server 2016 STIG バージョン 2 リリース 5

  Category III (Low) の脆弱性に適用される、サポートされている STIG 設定に加えて、以下が含まれます。

  V-224850, V-224852, V-224853, V-224854, V-224855, V-224856, V-224857, V-224858, V-224859, V-224866, V-224867, V-224868, V-224869, V-224870, V-224871, V-224872, V-224873, V-224881, V-224882, V-224883, V-224884, V-224885, V-224886, V-224887, V-224888, V-224889, V-224890, V-224891, V-224892, V-224893, V-224894, V-224895, V-224896, V-224897, V-224898, V-224899, V-224900, V-224901, V-224902, V-224903, V-224904, V-224905, V-224906, V-224907, V-224908, V-224909, V-224910, V-224911, V-224912, V-224913, V-224914, V-224915, V-224920, V-224922, V-224924, V-224925, V-224926, V-224927, V-224928, V-224929, V-224930, V-224935, V-224936, V-224937, V-224938, V-224939, V-224940, V-224941, V-224943, V-224944, V-224945, V-224946,

V-224947, V-224948, V-224949, V-224951, V-224952, V-224953, V-224955, V-224956, V-224957, V-224959, V-224960, V-224962, V-224963, V-225010, V-225013, V-225014, V-225015, V-225016, V-225017, V-225018, V-225019, V-225021, V-225022, V-225023, V-225024, V-225028, V-225029, V-225030, V-225031, V-225032, V-225033, V-225034, V-225035, V-225038, V-225039, V-225040, V-225041, V-225042, V-225043, V-225047, V-225049, V-225050, V-225051, V-225052, V-225055, V-225056, V-225057, V-225058, V-225061, V-225062, V-225063, V-225064, V-225065, V-225066, V-225067, V-225068, V-225069, V-225072, V-225073, V-225074, V-225076, V-225078, V-225080, V-225081, V-225082, V-225083, V-225084, V-225086, V-225087, V-225088, V-225089, V-225092, V-225093 および V-236000

- Windows Server 2012 R2 MS STIG バージョン 3 リリース 5

  Category III (Low) の脆弱性に適用される、サポートされている STIG 設定に加えて、以下が含まれます。

  V-225574、V-225573、V-225572、V-225571、V-225570、V-225569、V-225568、V-225567、V-225566 および V-225239

- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 2

  Category III (Low) の脆弱性に適用される、サポート対象のすべての STIG 設定に加えて、V-225238 が含まれます

- Windows ファイアウォール STIG バージョン 2 リリース 1

  Category III (Low) の脆弱性に適用される、サポートされている STIG 設定に加えて、以下が含まれます。

  V-241989, V-241990, V-241991, V-241993, V-241998, および V-242003

- Internet Explorer 11 STIG バージョン 2 リリース 3

  Category III (Low) の脆弱性に適用される、サポートされている STIG 設定に加えて、以下が含まれます。

  V-46473、V-46475、V-46481、V-46483、V-46501、V-46507、V-46509、V-46511、V-46513、V-46515、および V-75171

- Microsoft Edge STIG バージョン 1 リリース 6 (Windows Server 2022 のみ)

  Category III (Low) の脆弱性に適用される、サポートされている STIG 設定に加えて、以下が含まれます。

V-1072、V-1074、V-1076、V-1089、V-1112、V-1114、V-1115、V-1127、V-1145、V-2907、V-3289、V
および V-75915

- Defender STIG バージョン 2 リリース 4 (Windows Server 2022 のみ)

  Category III (Low) の脆弱性に適用される、サポートされている STIG 設定に加えて、以下が含まれます。

  V-213427, V-213429, V-213430, V-213431, V-213432, V-213433, V-213434, V-213435, V-213436, V-213437, V-213438, V-213439, V-213440, V-213441, V-213442, V-213443, V-213444, V-213445, V-213446, V-213447, V-213448, V-213449, V-213450, V-213451, V-213455, V-213464, V-213465, および V-213466

STIG-Build-Windows-High バージョン2022.4.x

次のリストには、コンポーネント強化がインフラストラクチャーに適用される STIG 設定が含まれています。サポートされている設定がご使用のインフラストラクチャに当てはまらない場合、強化コンポーネントはその設定をスキップして次に進みます。例えば、一部の STIG 設定は、スタンドアロンサーバーには適用されない場合があります。組織固有のポリシーは、管理者が文書設定をレビューするための要件など、堅牢化コンポーネントが適用する設定にも影響します。

Windows 向け STIG の完全なリストについては、「STIG ドキュメントライブラリ」を参照してください。完全なリストを表示する方法の詳細については、「STIG 表示ツール」を参照してください。

> **Note**
>
> STIG-Build-Windows-High 強化コンポーネントには、カテゴリ I の脆弱性専用にリストされている STIG 設定に加えて、STIG-Build-Windows-Low および STIG-Build-Windows-Medium 強化コンポーネント AWSTOE に適用されるすべてのリストされた STIG 設定が含まれます。

- Windows Server 2022 STIG バージョン 1 リリース 1

  Category II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定に加えて、以下が含まれます。

  V-254293, V-254352, V-254353, V-254354, V-254374, V-254378, V-254381, V-254446, V-254465, V-254466, V-254467, V-254469, V-254474, V-254475, および V-254500

- Windows Server 2019 STIG バージョン 2 リリース 5

  Category II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定に加えて、以下が含まれます。

  V-205653、V-205654、V-205711、V-205713、V-205724、V-205725、V-205757、V-205802、V-205804 および V-205919

- Windows Server 2016 STIG バージョン 2 リリース 5

  Category II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定に加えて、以下が含まれます。

  V-224874、V-224932、V-224933、V-224934、V-224954、V-224958、V-224961、V-225025、V-225044 および V-225079

- Windows Server 2012 R2 MS STIG バージョン 3 リリース 5

  Category II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定に加えて、以下が含まれます。

  V-225556、V-225552、V-225547、V-225507、V-225505、V-225498、V-225497、V-225496、V-225493 および V-225274

- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 2

  Microsoft .NET Framework のCategory II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定が含まれます。カテゴリ I の脆弱性に対して、追加的な STIG 設定は適用されません。

- Windows ファイアウォール STIG バージョン 2 リリース 1

  Category II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定に加えて、以下が含まれます。

  V-241992, V-241997, および V-242002

- Internet Explorer 11 STIG バージョン 2 リリース 3

  Internet Explorer 11 のCategory II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定が含まれます。カテゴリ I の脆弱性に対して、追加的な STIG 設定は適用されません。

- Microsoft Edge STIG バージョン 1 リリース 6 (Windows Server 2022 のみ)

Category II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定に加えて、以下が含まれます。

V-235758 と V-235759

- Defender STIG バージョン 2 リリース 4 (Windows Server 2022 のみ)

  Category II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定に加えて、以下が含まれます。

  V-213426, V-213452, および V-213453

Windows 用 STIG バージョン履歴ログ

このセクションには、四半期ごとの STIG アップデートの Windows 強化コンポーネントのバージョン履歴が記録されます。四半期ごとの変更点と公開されたバージョンを確認するには、タイトルを選択して情報を展開します。

2024 年Q1 四半期の変更 - 02/06/2024 (変更なし):

2024 年第 1 四半期リリースでは、Windows コンポーネント STIGS に変更はありませんでした。

2023 年Q4 四半期の変更 - 12/04/2023 (変更なし):

2023 年第 4 四半期リリースでは、Windows コンポーネント STIGS に変更はありませんでした。

2023 年第 3 四半期の変更 - 2023 年 10 月 4 日 (変更なし):

2023 年第 3 四半期リリースの Windows コンポーネント STIGS に変更はありませんでした。

2023 年第 2 四半期の変更 - 2023 年 5 月 3 日 (変更なし):

2023 年第 2 四半期リリースの Windows コンポーネント STIGS に変更はありませんでした。

2023 年第 1 四半期の変更 - 2023 年 3 月 27 日 (変更なし):

2023 年第 1 四半期リリースの Windows コンポーネント STIGS に変更はありませんでした。

2022 年第 4 四半期の変更 (2023 年 2 月 1 日):

STIGのバージョンを更新し、2022年第4四半期リリース用のSTIGSを適用。

STIG-Build-Windows-Low バージョン2022.4.x

- Windows Server 2022 STIG バージョン 1 リリース 1
- Windows Server 2019 STIG バージョン 2 リリース 5
- Windows Server 2016 STIG バージョン 2 リリース 5
- Windows Server 2012 R2 MS STIG バージョン 3 リリース 5
- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 2
- Windows Firewall STIG バージョン 2 リリース 1
- Internet Explorer 11 STIG バージョン 2 リリース 3
- Microsoft Edge STIG バージョン 1 リリース 6 (Windows Server 2022 のみ)

STIG-Build-Windows-Medium バージョン2022.4.x

- Windows Server 2022 STIG バージョン 1 リリース 1
- Windows Server 2019 STIG バージョン 2 リリース 5
- Windows Server 2016 STIG バージョン 2 リリース 5
- Windows Server 2012 R2 MS STIG バージョン 3 リリース 5
- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 2
- Windows Firewall STIG バージョン 2 リリース 1
- Internet Explorer 11 STIG バージョン 2 リリース 3
- Microsoft Edge STIG バージョン 1 リリース 6 (Windows Server 2022 のみ)
- Defender STIG バージョン 2 リリース 4 (Windows Server 2022 のみ)

STIG-Build-Windows-High バージョン2022.4.x

- Windows Server 2022 STIG バージョン 1 リリース 1
- Windows Server 2019 STIG バージョン 2 リリース 5
- Windows Server 2016 STIG バージョン 2 リリース 5
- Windows Server 2012 R2 MS STIG バージョン 3 リリース 5
- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 2
- Windows Firewall STIG バージョン 2 リリース 1
- Internet Explorer 11 STIG バージョン 2 リリース 3

- Microsoft Edge STIG バージョン 1 リリース 6 (Windows Server 2022 のみ)
- Defender STIG バージョン 2 リリース 4 (Windows Server 2022 のみ)

2022 年第 3 四半期の変更 - 2022 年 9 月 30 日 (変更なし):

2022 年第 3 四半期リリースの Windows コンポーネント STIGS に変更はありませんでした。

2022 年第 2 四半期の変更 - 2022 年 8 月 2 日:

STIGのバージョンを更新し、2022年第2四半期リリース用のSTIGSを適用。

STIG-Build-Windows-Low バージョン1.5.x

- Windows Server 2019 STIG バージョン 2 リリース 4
- Windows Server 2016 STIG バージョン 2 リリース 4
- Windows Server 2012 R2 MS STIG バージョン 3 リリース 3
- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 1
- Windows Firewall STIG バージョン 2 リリース 1
- Internet Explorer 11 STIG バージョン 1 リリース 19

STIG-Build-Windows-Medium バージョン1.5.x

- Windows Server 2019 STIG バージョン 2 リリース 4
- Windows Server 2016 STIG バージョン 2 リリース 4
- Windows Server 2012 R2 MS STIG バージョン 3 リリース 3
- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 1
- Windows Firewall STIG バージョン 2 リリース 1
- Internet Explorer 11 STIG バージョン 1 リリース 19

STIG-Build-Windows-High バージョン1.5.x

- Windows Server 2019 STIG バージョン 2 リリース 4
- Windows Server 2016 STIG バージョン 2 リリース 4
- Windows Server 2012 R2 MS STIG バージョン 3 リリース 3

- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 1
- Windows Firewall STIG バージョン 2 リリース 1
- Internet Explorer 11 STIG バージョン 1 リリース 19

2022 年第 1 四半期の変更 - 2022 年 8 月 2 日 (変更なし):

2022 年第 1 四半期リリースの Windows コンポーネント STIGS に変更はありませんでした。

2021 年第 4 四半期の変更 - 2021 年 12 月 20 日:

STIGのバージョンを更新し、2021年第4四半期リリース用のSTIGSを適用。

STIG-Build-Windows-Low バージョン1.5.x

- Windows Server 2019 STIG バージョン 2 リリース 3
- Windows Server 2016 STIG バージョン 2 リリース 3
- Windows Server 2012 R2 MS STIG バージョン 3 リリース 3
- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 1
- Windows Firewall STIG バージョン 2 リリース 1
- Internet Explorer 11 STIG バージョン 1 リリース 19

STIG-Build-Windows-Medium バージョン1.5.x

- Windows Server 2019 STIG バージョン 2 リリース 3
- Windows Server 2016 STIG バージョン 2 リリース 3
- Windows Server 2012 R2 MS STIG バージョン 3 リリース 3
- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 1
- Windows Firewall STIG バージョン 2 リリース 1
- Internet Explorer 11 STIG バージョン 1 リリース 19

STIG-Build-Windows-High バージョン1.5.x

- Windows Server 2019 STIG バージョン 2 リリース 3
- Windows Server 2016 STIG バージョン 2 リリース 3
- Windows Server 2012 R2 MS STIG バージョン 3 リリース 3

- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 1
- Windows Firewall STIG バージョン 2 リリース 1
- Internet Explorer 11 STIG バージョン 1 リリース 19

2021 年第 3 四半期の変更 - 2021 年 9 月 30 日:

STIGのバージョンを更新し、2021年第3四半期リリース用のSTIGSを適用。

STIG-Build-Windows-Low バージョン1.4.x

- Windows Server 2019 STIG バージョン 2 リリース 2
- Windows Server 2016 STIG バージョン 2 リリース 2
- Windows Server 2012 R2 MS STIG バージョン 3 リリース 2
- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 1
- Windows Firewall STIG バージョン 1 リリース 7
- Internet Explorer 11 STIG バージョン 1 リリース 19

STIG-Build-Windows-Medium バージョン1.4.x

- Windows Server 2019 STIG バージョン 2 リリース 2
- Windows Server 2016 STIG バージョン 2 リリース 2
- Windows Server 2012 R2 MS STIG バージョン 3 リリース 2
- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 1
- Windows Firewall STIG バージョン 1 リリース 7
- Internet Explorer 11 STIG バージョン 1 リリース 19

STIG-Build-Windows-High バージョン1.4.x

- Windows Server 2019 STIG バージョン 2 リリース 2
- Windows Server 2016 STIG バージョン 2 リリース 2
- Windows Server 2012 R2 MS STIG バージョン 3 リリース 2
- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 1
- Windows Firewall STIG バージョン 1 リリース 7

- Internet Explorer 11 STIG バージョン 1 リリース 19

Linux の STIG 強化コンポーネント

このセクションには Linux STIG 強化コンポーネントに関する情報が含まれ、その後にバージョン履歴ログが続きます。Linux ディストリビューションに独自の STIG 設定がない場合、強化コンポーネントは RHEL 設定を適用します。強化コンポーネントは、次のように Linux ディストリビューションに基づいてサポートされている STIG 設定を適用します。

Red Hat Enterprise Linux (RHEL) 7 STIG 設定

- RHEL 7
- CentOS 7
- Amazon Linux 2 (AL2)

RHEL 8 STIG 設定

- RHEL 8
- CentOS 8
- Amazon Linux 2023 (AL 2023)

STIG-Build-Linux-Low バージョン 2024.1.x

次のリストには、コンポーネント強化がインフラストラクチャーに適用される STIG 設定が含まれています。サポートされている設定がご使用のインフラストラクチャに当てはまらない場合、強化コンポーネントはその設定をスキップして次に進みます。例えば、一部の STIG 設定は、スタンドアロンサーバーには適用されない場合があります。組織固有のポリシーは、管理者が文書設定をレビューするための要件など、堅牢化コンポーネントが適用する設定にも影響します。

詳細なリストについては、「STIGs Document Library」を参照してください。完全なリストを表示する方法の詳細については、「STIG 表示ツール」を参照してください。

RHEL 7 STIG バージョン 3 リリース 14

- RHEL 7/CentOS 7

  V-204452、V-204576、および V-204605

- AL2

  V-204452、V-204576、および V-204605

RHEL 8 STIG バージョン 1 リリース 13

- RHEL 8/CentOS 8/AL 2023

  V-230241, V-244527, V-230269, V-230270, V-230285, V-230253, V-230346, V-230381, V-230395, V-230468, V-230469, V-230491, V-230485, V-230486, V-230494, V-230495, V-230496, V-230497, V-230498, V-230499、V-230281

Ubuntu 18.04 STIG バージョン 2 リリース 13

V-219172, V-219173, V-219174, V-219175, V-219210, V-219164, V-219165, V-219178, V-219180, V-219301, V-219163, V-219332, V-219327、および V-219333

Ubuntu 20.04 STIG バージョン 1 リリース 11

V-238202, V-238234, V-238235, V-238237, V-238323, V-238373, V-238221, V-238222, V-238223, V-238224, V-238226, V-238362, V-238357、および V-238308

STIG-Build-Linux-Medium バージョン 2024.1.x

次のリストには、コンポーネント強化がインフラストラクチャーに適用される STIG 設定が含まれています。サポートされている設定がご使用のインフラストラクチャに当てはまらない場合、強化コンポーネントはその設定をスキップして次に進みます。例えば、一部の STIG 設定は、スタンドアロンサーバーには適用されない場合があります。組織固有のポリシーは、管理者が文書設定をレビューするための要件など、堅牢化コンポーネントが適用する設定にも影響します。

詳細なリストについては、「STIGs Document Library」を参照してください。完全なリストを表示する方法の詳細については、「STIG 表示ツール」を参照してください。

> **Note**
>
> STIG-Build-Linux-Medium 強化コンポーネントには、カテゴリ II の脆弱性専用にリストされている STIG 設定に加えて、STIG-Build-Linux-Low 強化コンポーネント AWSTOE に適用されるすべてのリストされた STIG 設定が含まれます。

RHEL 7 STIG バージョン 3 リリース 14

この Linux ディストリビューションのCategory III (Low) の脆弱性に適用される、サポートされている STIG 設定に加えて、以下が含まれます。

- RHEL 7/CentOS 7

  V-204585、 V-204490、 V-204491、 V-255928、 V-204405、 V-204406、 V-204407、
  V-204408、 V-204409、 V-204410、 V-204411、 V-204412、 V-204413、 V-204414、
  V-204415、 V-204422、 V-204423、 V-204427、 V-204416、 V-204418、 V-204426、
  V-204431、 V-204457、 V-204466、 V-204417、 V-204434、 V-204435、 V-204587、
  V-204588、 V-204589、 V-204590、 V-204591、 V-204592、 V-204593、 V-204596、
  V-204597、 V-204598、 V-204599、 V-204600、 V-204601、 V-204602、 V-204622、
  V-233307、 V-255925、 V-204578、 V-204595、 V-204437、 V-204503、 V-204507、
  V-204508、 V-204510、 V-204511、 V-204512、 V-204514、 V-204515、 V-204516、
  V-204517、 V-204521、 V-204524、 V-204531、 V-204536、 V-204537、 V-204538、
  V-204539、 V-204540、 V-204541、 V-204542、 V-204543、 V-204544、 V-204545、
  V-204546、 V-204547、 V-204548、 V-204549、 V-204550、 V-204551、 V-204552、
  V-204553、 V-204554、 V-204555、 V-204556、 V-204557、 V-204558、 V-204559、
  V-204560、 V-204562、 V-204563、 V-204564、 V-204565、 V-204566、 V-204567、
  V-204568、 V-204572、 V-204584、 V-204609、 V-204610、 V-204611、 V-204612、
  V-204613、 V-204614、 V-204615、 V-204616、 V-204617、 V-204625、 V-204630、
  V-255927、 V-237634、 V-237635、 V-251703、 V-204449、 V-204450、 V-204451、
  V-204619、 V-204579、 V-204631、 V-204633、 および V-256970

- AL2:

  V-204585、 V-204490、 V-204491、 V-255928、 V-204405、 V-204406、 V-204407、
  V-204408、 V-204409、 V-204410、 V-204411、 V-204412、 V-204413、 V-204414、
  V-204415、 V-204422、 V-204423、 V-204427、 V-204416、 V-204418、 V-204426、
  V-204431、 V-204457、 V-204466、 V-204417、 V-204434、 V-204435、 V-204587、
  V-204588、 V-204589、 V-204590、 V-204591、 V-204592、 V-204593、 V-204596、
  V-204597、 V-204598、 V-204599、 V-204600、 V-204601、 V-204602、 V-204622、
  V-233307、 V-255925、 V-204578、 V-204595、 V-204437、 V-204503、 V-204507、
  V-204508、 V-204510、 V-204511、 V-204512、 V-204514、 V-204515、 V-204516、
  V-204517、 V-204521、 V-204524、 V-204531、 V-204536、 V-204537、 V-204538、
  V-204539、 V-204540、 V-204541、 V-204542、 V-204543、 V-204544、 V-204545、
  V-204546、 V-204547、 V-204548、 V-204549、 V-204550、 V-204551、 V-204552、
  V-204553、 V-204554、 V-204555、 V-204556、 V-204557、 V-204558、 V-204559、

V-204560、 V-204562、 V-204563、 V-204564、 V-204565、 V-204566、 V-204567、
V-204568、 V-204572、 V-204584、 V-204609、 V-204610、 V-204611、 V-204612、
V-204613、 V-204614、 V-204615、 V-204616、 V-204617、 V-204625、 V-204630、
V-255927、 V-237634、 V-237635、 V-251703、 V-204449、 V-204450、 V-204451、
V-204619、 V-204579、 V-204631、 V-204633、 および V-256970

RHEL 8 STIG バージョン 1 リリース 13

この Linux ディストリビューションのCategory III (Low) の脆弱性に適用される、サポートされている STIG 設定に加えて、以下が含まれます。

- RHEL 8/CentOS 8/AL 2023

V-230257、 V-230258、 V-230259、 V-230550、 V-230248、 V-230249、 V-230250、
V-230245、 V-230246、 V-230247、 V-230397、 V-230399、 V-230400、 V-230401、
V-230228、 V-230298、 V-230387、 V-230231、 V-230233、 V-230324、 V-230365、
V-230370、 V-230378、 V-230383、 V-230236、 V-230314、 V-230315、 V-244523、
V-230266、 V-230267、 V-230268、 V-230280、 V-230310、 V-230311、 V-230312、
V-230502、 V-230532、 V-230535、 V-230536、 V-230537、 V-230538、 V-230539、
V-230540、 V-230541、 V-230542、 V-230543、 V-230544、 V-230545、 V-230546、
V-230547、 V-230548、 V-230549、 V-244550、 V-244551、 V-244552、 V-244553、
V-244554、 V-250317、 V-251718、 V-230237、 V-230313、 V-230356、 V-230357、
V-230358、 V-230359、 V-230360、 V-230361、 V-230362、 V-230363、 V-230368、
V-230369、 V-230375、 V-230376、 V-230377、 V-244524、 V-244533、 V-251713、
V-251717、 V-251714、 V-251715、 V-251716、 V-230332、 V-230334、 V-230336、
V-230338、 V-230340、 V-230342、 V-230344、 V-230333、 V-230335、 V-230337、
V-230339、 V-230341、 V-230343、 V-230345、 V-230240、 V-230282、 V-250315、
V-250316、 V-230255、 V-230277、 V-230278、 V-230348、 V-230353、 V-230386、
V-230390、 V-230392、 V-230394、 V-230396、 V-230393、 V-230398、 V-230402、
V-230403、 V-230404、 V-230405、 V-230406、 V-230407、 V-230408、 V-230409、
V-230410、 V-230411、 V-230412、 V-230413、 V-230418、 V-230419、 V-230421、
V-230422、 V-230423、 V-230424、 V-230425、 V-230426、 V-230427、 V-230428、
V-230429、 V-230430、 V-230431、 V-230432、 V-230433、 V-230434、 V-230435、
V-230436、 V-230437、 V-230438、 V-230439、 V-230444、 V-230446、 V-230447、
V-230448、 V-230449、 V-230455、 V-230456、 V-230462、 V-230463、 V-230464、
V-230465、 V-230466、 V-230467、 V-230471、 V-230472、 V-230473、 V-230474、
V-230480、 V-230483、 V-244542、 V-230503、 V-230244、 V-230286、 V-230287、

V-230288、 V-230290、 V-230291、 V-230296、 V-230330、 V-230382、 V-230526、 V-230527、 V-230555、 V-230556、 V-244526、 V-244528、 V-237642、 V-237643、 V-251711、 V-230238、 V-230239、 V-230273、 V-230275、 V-230478、 V-230488、 V-230489、 V-230559、 V-230560、 V-230561、 V-237640、 および V-256974

Ubuntu 18.04 STIG バージョン 2 リリース 13

この Linux ディストリビューションのCategory III (Low) の脆弱性に適用される、サポートされている STIG 設定に加えて、以下が含まれます。

V-219188、 V-219190、 V-219191、 V-219198、 V-219199、 V-219200、 V-219201、 V-219202、 V-219203、 V-219204、 V-219205、 V-219206、 V-219207、 V-219208、 V-219209、 V-219303、 V-219326、 V-219328、 V-219330、 V-219342、 V-219189、 V-219192、 V-219193、 V-219194、 V-219315、 V-219195、 V-219196、 V-219197、 V-219213、 V-219214、 V-219215、 V-219216、 V-219217、 V-219218、 V-219219、 V-219220、 V-219221、 V-219222、 V-219223、 V-219224、 V-219227、 V-219228、 V-219229、 V-219230、 V-219231、 V-219232、 V-219233、 V-219234、 V-219235、 V-219236、 V-219238、 V-219239、 V-219240、 V-219241、 V-219242、 V-219243、 V-219244、 V-219250、 V-219254、 V-219257、 V-219263、 V-219264、 V-219265、 V-219266、 V-219267、 V-219268、 V-219269、 V-219270、 V-219271、 V-219272、 V-219273、 V-219274、 V-219275、 V-219276、 V-219277、 V-219279、 V-219281、 V-219287、 V-219291、 V-219297、 V-219298、 V-219299、 V-219300、 V-219309、 V-219310、 V-219311、 V-219312、 V-233779、 V-233780、 V-255906、 V-219336、 V-219338、 V-219344、 V-219181、 V-219184、 V-219186、 V-219155、 V-219156、 V-219160、 V-219306、 V-219149、 V-219166、 V-219176、 V-219339、 V-219331、 V-219337、 および V-219335

Ubuntu 20.04 STIG バージョン 1 リリース 11

この Linux ディストリビューションのCategory III (Low) の脆弱性に適用される、サポートされている STIG 設定に加えて、以下が含まれます。

V-238205、 V-238207、 V-238329、 V-238337、 V-238339、 V-238340、 V-238344、 V-238345、 V-238346、 V-238347、 V-238348、 V-238349、 V-238350、 V-238351、 V-238352、 V-238376、 V-238377、 V-238378、 V-238209、 V-238325、 V-238330、 V-238333、 V-238369、 V-238338、 V-238341、 V-238342、 V-238343、 V-238324、 V-238353、 V-238228、 V-238225、 V-238227、 V-238299、 V-238238、 V-238239、 V-238240、 V-238241、 V-238242、 V-238244、 V-238245、 V-238246、 V-238247、 V-238248、 V-238249、 V-238250、 V-238251、 V-238252、 V-238253、 V-238254、 V-238255、 V-238256、 V-238257、 V-238258、 V-238264、 V-238268、 V-238271、 V-238277、 V-238278、 V-238279、 V-238280、 V-238281、 V-238282、 V-238283、 V-238284、

V-238285、 V-238286、 V-238287、 V-238288、 V-238289、 V-238290、 V-238291、 V-238292、 V-238293、 V-238294、 V-238295、 V-238297、 V-238300、 V-238301、 V-238302、 V-238304、 V-238309、 V-238310、 V-238315、 V-238316、 V-238317、 V-238318、 V-238319、 V-238320、 V-251505、 V-238360、 V-238211、 V-238212、 V-238213、 V-238216、 V-238220、 V-255912、 V-238355、 V-238236、 V-238303、 V-238358、 V-238356、 V-238359、 V-238370、 および V-238334

STIG-Build-Linux-High バージョン 2024.1.x

次のリストには、コンポーネント強化がインフラストラクチャーに適用される STIG 設定が含まれています。サポートされている設定がご使用のインフラストラクチャに当てはまらない場合、強化コンポーネントはその設定をスキップして次に進みます。例えば、一部の STIG 設定は、スタンドアロンサーバーには適用されない場合があります。組織固有のポリシーは、管理者が文書設定をレビューするための要件など、堅牢化コンポーネントが適用する設定にも影響します。

詳細なリストについては、「STIGs Document Library」を参照してください。完全なリストを表示する方法の詳細については、「STIG 表示ツール」を参照してください。

> **Note**
>
> STIG-Build-Linux-High 強化コンポーネントには、カテゴリ I の脆弱性に特に AWSTOE 適用されるリストされた STIG 設定に加えて、STIG-Build-Linux-Low および STIG-Build-Linux-Medium 強化コンポーネントに適用されるすべてのリストされた STIG 設定が含まれます。

RHEL 7 STIG バージョン 3 リリース 14

この Linux ディストリビューションのCategory II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定に加えて、以下が含まれます。

- RHEL 7/CentOS 7

  V-204425, V-204594, V-204455, V-204424, V-204442, V-204443, V-204447, V-204448, V-204502, V-204620、および V-204621

- AL2:

  V-204425, V-204594, V-204455, V-204424, V-204442, V-204443, V-204447, V-204448, V-204502, V-204620、および V-204621

RHEL 8 STIG バージョン 1 リリース 13

この Linux ディストリビューションのCategory II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定に加えて、以下が含まれます。

- RHEL 8/CentOS 8/AL 2023

  V-230265, V-230529, V-230531, V-230264, V-230487, V-230492, V-230533、および V-230558

Ubuntu 18.04 STIG バージョン 2 リリース 13

この Linux ディストリビューションのCategory II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定に加えて、以下が含まれます。

V-219157, V-219158, V-219177, V-219212 V-219308, V-219314, V-219316、および V-251507

Ubuntu 20.04 STIG バージョン 1 リリース 11

この Linux ディストリビューションのCategory II および III (Medium および Low) の脆弱性に適用されるすべての STIG 設定に加えて、以下が含まれます。

V-238218, V-238219, V-238201, V-238326, V-238327, V-238380、V-251504

Linux 用 STIG バージョン履歴ログ

このセクションには Linux コンポーネントのバージョン履歴が記録されます。四半期ごとの変更点と公開されたバージョンを確認するには、タイトルを選択して情報を展開してください。

2024 年Q1 四半期の変更 - 02/06/2024:

2024 年第 1 四半期リリースの STIG バージョンを更新し、STIGS を次のように適用しました。

STIG-Build-Linux-Low バージョン 2024.1.x

- RHEL 7 STIG バージョン 3 リリース 14
- RHEL 8 STIG バージョン 1 リリース 13
- Ubuntu 18.04 STIG バージョン 2 リリース 13
- Ubuntu 20.04 STIG バージョン 1 リリース 11

STIG-Build-Linux-Medium バージョン 2024.1.x

- RHEL 7 STIG バージョン 3 リリース 14
- RHEL 8 STIG バージョン 1 リリース 13
- Ubuntu 18.04 STIG バージョン 2 リリース 13
- Ubuntu 20.04 STIG バージョン 1 リリース 11

STIG-Build-Linux-High バージョン 2024.1.x

- RHEL 7 STIG バージョン 3 リリース 14
- RHEL 8 STIG バージョン 1 リリース 13
- Ubuntu 18.04 STIG バージョン 2 リリース 13
- Ubuntu 20.04 STIG バージョン 1 リリース 11

2023 年Q4 四半期の変更 - 12/07/2023:

2023 年第 4 四半期リリースの STIG バージョンを更新し、STIGS を次のように適用しました。

STIG-Build-Linux-Low バージョン 2023.4.x

- RHEL 7 STIG バージョン 3 リリース 13
- RHEL 8 STIG バージョン 1 リリース 12
- Ubuntu 18.04 STIG バージョン 2 リリース 12
- Ubuntu 20.04 STIG バージョン 1 リリース 10

STIG-Build-Linux-Medium バージョン 2023.4.x

- RHEL 7 STIG バージョン 3 リリース 13
- RHEL 8 STIG バージョン 1 リリース 12
- Ubuntu 18.04 STIG バージョン 2 リリース 12
- Ubuntu 20.04 STIG バージョン 1 リリース 10

STIG-Build-Linux-High バージョン 2023.4.x

- RHEL 7 STIG バージョン 3 リリース 13

- RHEL 8 STIG バージョン 1 リリース 12
- Ubuntu 18.04 STIG バージョン 2 リリース 12
- Ubuntu 20.04 STIG バージョン 1 リリース 10

2023 年第 3 四半期の変更 - 2023 年 10 月 4 日:

2023 年第 3 四半期リリースに向けて STIG バージョンを更新し、STIG を次のように適用しました。

STIG-Build-Linux-Low バージョン2023.3.x

- RHEL 7 STIG バージョン 3 リリース 12
- RHEL 8 STIG バージョン 1 リリース 11
- Ubuntu 18.04 STIG バージョン 2 リリース 11
- Ubuntu 20.04 STIG バージョン 1 リリース 9

STIG-Build-Linux-Medium バージョン2023.3.x

- RHEL 7 STIG バージョン 3 リリース 12
- RHEL 8 STIG バージョン 1 リリース 11
- Ubuntu 18.04 STIG バージョン 2 リリース 11
- Ubuntu 20.04 STIG バージョン 1 リリース 9

STIG-Build-Linux-High バージョン2023.3.x

- RHEL 7 STIG バージョン 3 リリース 12
- RHEL 8 STIG バージョン 1 リリース 11
- Ubuntu 18.04 STIG バージョン 2 リリース 11
- Ubuntu 20.04 STIG バージョン 1 リリース 9

2023 年第 2 四半期の変更 - 2023 年 5 月 3 日:

2023 年第 2 四半期リリースに向けて STIG バージョンを更新し、STIG を次のように適用しました。

STIG-Build-Linux-Low バージョン2023.2.x

- RHEL 7 STIG バージョン 3 リリース 11
- RHEL 8 STIG バージョン 1 リリース 10
- Ubuntu 18.04 STIG バージョン 2 リリース 11
- Ubuntu 20.04 STIG バージョン 1 リリース 8

STIG-Build-Linux-Medium バージョン2023.2.x

- RHEL 7 STIG バージョン 3 リリース 11
- RHEL 8 STIG バージョン 1 リリース 10
- Ubuntu 18.04 STIG バージョン 2 リリース 11
- Ubuntu 20.04 STIG バージョン 1 リリース 8

STIG-Build-Linux-High バージョン2023.2.x

- RHEL 7 STIG バージョン 3 リリース 11
- RHEL 8 STIG バージョン 1 リリース 10
- Ubuntu 18.04 STIG バージョン 2 リリース 11
- Ubuntu 20.04 STIG バージョン 1 リリース 8

2023 年第 1 四半期の変更 - 2023 年 3 月 27 日:

2023 年第 1 四半期リリースに向けて STIG バージョンを更新し、STIG を次のように適用しました。

STIG-Build-Linux-Low バージョン2023.1.x

- RHEL 7 STIG バージョン 3 リリース 10
- RHEL 8 STIG バージョン 1 リリース 9
- Ubuntu 18.04 STIG バージョン 2 リリース 10
- Ubuntu 20.04 STIG バージョン 1 リリース 7

STIG-Build-Linux-Medium バージョン2023.1.x

- RHEL 7 STIG バージョン 3 リリース 10
- RHEL 8 STIG バージョン 1 リリース 9
- Ubuntu 18.04 STIG バージョン 2 リリース 10
- Ubuntu 20.04 STIG バージョン 1 リリース 7

STIG-Build-Linux-High バージョン2023.1.x

- RHEL 7 STIG バージョン 3 リリース 10
- RHEL 8 STIG バージョン 1 リリース 9
- Ubuntu 18.04 STIG バージョン 2 リリース 10
- Ubuntu 20.04 STIG バージョン 1 リリース 7

2022 年第 4 四半期の変更 (2023 年 2 月 1 日):

2022 年第 4 四半期リリースに向けて STIG バージョンを更新し、STIG を次のように適用しました。

STIG-Build-Linux-Low バージョン2022.4.x

- RHEL 7 STIG バージョン 3 リリース 9
- RHEL 8 STIG バージョン 1 リリース 8
- Ubuntu 18.04 STIG バージョン 2 リリース 9
- Ubuntu 20.04 STIG バージョン 1 リリース 6

STIG-Build-Linux-Medium バージョン2022.4.x

- RHEL 7 STIG バージョン 3 リリース 9
- RHEL 8 STIG バージョン 1 リリース 8
- Ubuntu 18.04 STIG バージョン 2 リリース 9
- Ubuntu 20.04 STIG バージョン 1 リリース 6

STIG-Build-Linux-High バージョン2022.4.x

- RHEL 7 STIG バージョン 3 リリース 9
- RHEL 8 STIG バージョン 1 リリース 8
- Ubuntu 18.04 STIG バージョン 2 リリース 9
- Ubuntu 20.04 STIG バージョン 1 リリース 6

2022 年第 3 四半期の変更 - 2022 年 9 月 30 日 (変更なし):

2022 年第 3 四半期リリースの Linux コンポーネント STIGS に変更はありませんでした。

2022 年第 2 四半期の変更 - 2022 年 8 月 2 日:

Ubuntu サポートの導入、STIG バージョンの更新、および 2022 年第 2 四半期リリース向けの STIG の適用を以下のとおり実施しました。

STIG-Build-Linux-Low バージョン2022.2.x

- RHEL 7 STIG バージョン 3 リリース 7
- RHEL 8 STIG バージョン 1 リリース 6
- Ubuntu 18.04 STIG バージョン 2 リリース 6 (新規)
- Ubuntu 20.04 STIG バージョン 1 リリース 4 (新規)

STIG-Build-Linux-Medium バージョン2022.2.x

- RHEL 7 STIG バージョン 3 リリース 7
- RHEL 8 STIG バージョン 1 リリース 6
- Ubuntu 18.04 STIG バージョン 2 リリース 6 (新規)
- Ubuntu 20.04 STIG バージョン 1 リリース 4 (新規)

STIG-Build-Linux-High バージョン2022.2.x

- RHEL 7 STIG バージョン 3 リリース 7
- RHEL 8 STIG バージョン 1 リリース 6
- Ubuntu 18.04 STIG バージョン 2 リリース 6 (新規)

- Ubuntu 20.04 STIG バージョン 1 リリース 4 (新規)

2022 年第 1 四半期の変更 - 2022 年 4 月 26 日:

コンテナのサポートを強化するようにリファクタリングされました。以前の AL2 スクリプトと RHEL 7 を組み合わせました。2022 年第 1 四半期リリースに向けて STIG バージョンを更新し、STIG を次のように適用しました。

STIG-Build-Linux-Low バージョン3.6.x

- RHEL 7 STIG バージョン 3 リリース 6
- RHEL 8 STIG バージョン 1 リリース 5

STIG-Build-Linux-Medium バージョン3.6.x

- RHEL 7 STIG バージョン 3 リリース 6
- RHEL 8 STIG バージョン 1 リリース 5

STIG-Build-Linux-High バージョン3.6.x

- RHEL 7 STIG バージョン 3 リリース 6
- RHEL 8 STIG バージョン 1 リリース 5

2021 年第 4 四半期の変更 - 2021 年 12 月 20 日:

STIG バージョンを更新し、2021 年第 4 四半期リリースに向けて STIG を次のように適用しました。

STIG-Build-Linux-Low バージョン3.5.x

- RHEL 7 STIG バージョン 3 リリース 5
- RHEL 8 STIG バージョン 1 リリース 4

STIG-Build-Linux-Medium バージョン3.5.x

- RHEL 7 STIG バージョン 3 リリース 5
- RHEL 8 STIG バージョン 1 リリース 4

STIG-Build-Linux-High バージョン3.5.x

- RHEL 7 STIG バージョン 3 リリース 5
- RHEL 8 STIG バージョン 1 リリース 4

2021 年第 3 四半期の変更 - 2021 年 9 月 30 日:

STIG バージョンを更新し、2021 年第 3 四半期リリースに向けて STIG を次のように適用しました。

STIG-Build-Linux-Low バージョン3.4.x

- RHEL 7 STIG バージョン 3 リリース 4
- RHEL 8 STIG バージョン 1 リリース 3

STIG-Build-Linux-Medium バージョン3.4.x

- RHEL 7 STIG バージョン 3 リリース 4
- RHEL 8 STIG バージョン 1 リリース 3

STIG-Build-Linux-High バージョン3.4.x

- RHEL 7 STIG バージョン 3 リリース 4
- RHEL 8 STIG バージョン 1 リリース 3

SCAP コンプライアンス検証ツール (コンポーネント)

セキュリティコンテンツ自動化プロトコル (SCAP) は、IT プロフェッショナルがアプリケーションのセキュリティ脆弱性を特定してコンプライアンスを確保するために使用できる一連の標準です。The SCAP Compliance Checker (SCC) は、Naval Information Warfare Center (NIWC) Atlantic がリリースした SCAP 検証済みのスキャンツールです。詳細については、NIWC Atlantic Web サイトの「Security Content Automation Protocol (SCAP) Compliance Checker (SCC)」を参照してください。

および AWSTOE `scap-compliance-checker-windows`コンポーネントは、パイプラインビルドインスタンスとテストインスタンスに SCC スキャナーをダウンロードしてインストールします。スキャナーを実行すると、DISA SCAP Benchmarks を使用して

認証された設定スキャンが実行され、以下の情報を含むレポートが提供されます。 AWSTOE また、は、情報をアプリケーションログに書き込みます。

- インスタンスに適用される STIG 設定。
- インスタンスの全体的なコンプライアンススコア。

正確なコンプライアンス検証結果を報告できるよう、ビルドプロセスの最終ステップとして SCAP 検証を実行することをお勧めします。

> **Note**
>
> レポートはいずれかの STIG 表示ツールで確認できます。これらのツールは、DoD サイバーエクスチェンジを通じてオンラインで入手できます。

以下のセクションでは、SCAP 検証コンポーネントに含まれるベンチマークについて説明します。

scap-compliance-checker-linux バージョン 2021.04.0

`scap-compliance-checker-linux` コンポーネントは、Image Builder パイプラインのビルドインスタンスとテストインスタンスで実行されます。SCC アプリケーションが生成するレポートとスコアの両方が AWSTOE ログに記録されます。

このコンポーネントは次のワークフローステップを実行します。

1. SCC アプリケーションをダウンロードしてインストールします。
2. コンプライアンスベンチマークをインポートします。
3. SCC アプリケーションを使用して検証を実行します。
4. コンプライアンスレポートとスコアをビルドインスタンスデスクトップにローカルに保存します。
5. ローカルレポートのコンプライアンススコアを AWSTOE アプリケーションログファイルに記録します。

> **Note**
>
> AWSTOE は現在、Windows Server 2012 R2、2016、および 2019 の SCAP コンプライアンス検証をサポートしています。

Windows 用の SCAP コンプライアンスチェッカーコンポーネントには、以下のベンチマークが含まれています。

SCC バージョン:5.4.2

2021 年第 4 四半期のベンチマーク:

- U_MSDotNet_Framework_4-0_V2R1_STIG_SCAP_1-2_Benchmark
- u_ms_IE11_v2r1_stig_scap_1-2_Benchmark
- u_ms_Windows_2012_and_2012_r2_ms_v3r2_stig_scap_1-2_Benchmark
- u_MS_Windows_Defender_AV_V2R2_Stig_Scap_1-2_Benchmark
- u_ms_Windows_Server_2016_v2r1_stig_scap_1-2_Benchmark
- u_MS_Windows_Server_2019_v2R1_Stig_scap_1-2_Benchmark
- u_ms_Windows_firewall_v2r1_stig_scap_1-2_Benchmark
- u_can_ubuntu_18-04_v2R4_Stig_scap_1-2_Benchmark
- u_rhel_7_v3r5_stig_scap_1-2_Benchmark
- u_rhel_8_v1r3_stig_scap_1-2_Benchmark

scap-compliance-checker-linux バージョン 2021.04.0

`scap-compliance-checker-linux` コンポーネントは、Image Builder パイプラインのビルドインスタンスとテストインスタンスで実行されます。SCC アプリケーションが生成するレポートとスコアの両方が AWSTOE ログに記録されます。

このコンポーネントは次のワークフローステップを実行します。

1. SCC アプリケーションをダウンロードしてインストールします。
2. コンプライアンスベンチマークをインポートします。
3. SCC アプリケーションを使用して検証を実行します。
4. コンプライアンスレポートとスコアをビルドインスタンス`/opt/scc/SCCResults`の次の場所にローカルに保存します。
5. ローカルレポートのコンプライアンススコアを AWSTOE アプリケーションログファイルに記録します。

> **Note**
>
> AWSTOE は現在、RHEL 7/8 および Ubuntu 18 の SCAP コンプライアンス検証をサポートしています。SCC アプリケーションは現在、検証用に x86 アーキテクチャをサポートしています。

Linux 用の SCAP コンプライアンスチェッカーコンポーネントには、以下のベンチマークが含まれています。

SCC バージョン:5.4.2

2021 年第 4 四半期のベンチマーク:

- u_can_ubuntu_18-04_v2R4_stig_scap_1-2_Benchmark
- u_rhel_7_v3r5_stig_scap_1-2_Benchmark
- u_rhel_8_v1r3_stig_scap_1-2_Benchmark
- U_MSDotNet_Framework_4-0_V2R1_STIG_SCAP_1-2_Benchmark
- u_ms_IE11_v2r1_stig_scap_1-2_Benchmark
- u_ms_Windows_2012_and_2012_r2_ms_v3r2_stig_scap_1-2_Benchmark
- u_MS_Windows_Defender_AV_V2R2_Stig_Scap_1-2_Benchmark
- u_ms_Windows_Server_2016_v2r1_stig_scap_1-2_Benchmark
- u_MS_Windows_Server_2019_v2R1_Stig_scap_1-2_Benchmark
- u_ms_Windows_firewall_v2r1_stig_scap_1-2_Benchmark

SCAP のバージョン履歴

次の表は、SCAP 環境と本書で説明されている設定に対する重要な変更について説明したものです。

変更	説明	日付
SCAP コンポーネントが追加されました。	次の SCAP コンポーネントが導入されました。 • scap-compliance-checker-linux バージョン 2021.04.0 を作成 (SCC バージョン: 5.4.2)	2021 年 12 月 20 日

変更	説明	日付
	• scap-compliance-checker-linux バージョン 2021.04.0 を作成 (SCC バージョン: 5.4.2)	

AWSTOE コマンドリファレンス

AWSTOE は、 で実行されるコンポーネント管理アプリケーションです AWS CLI。

> **Note**
>
> 一部の AWSTOE アクションモジュールでは、Linux サーバーで実行するために昇格されたアクセス許可が必要です。昇格された権限を使用するには、コマンド構文の前に sudo を付けるか、ログイン時に sudo su コマンドを 1 回実行してから、以下にリンクされているコマンドを実行します。 AWSTOE アクションモジュールの詳細については、「」を参照してくださいAWSTOE コンポーネントマネージャーがサポートするアクションモジュール。

run

run コマンドを使用して、1 つ以上のコンポーネントドキュメントの YAML ドキュメントスクリプトを実行します。

validate

1つ以上のコンポーネントドキュメントのYAMLドキュメント構文を検証するために、validateコマンドを実行する。

awstoe: コマンドを実行します。

このコマンドは、YAML コンポーネントドキュメントスクリプトを、`--config` パラメーターで指定された設定ファイル、または `--documents` パラメーターで指定されたコンポーネントドキュメントのリストに含まれている順序で実行します。

> **Note**
>
> 次のパラメータのうち、1 つのみを正確に指定する必要があります。両方は指定しないでください。

--config
ドキュメント

構文

```
awstoe run [--config <file path>] [--cw-ignore-failures <?>]
      [--cw-log-group <?>] [--cw-log-region us-west-2] [--cw-log-stream <?>]
      [--document-s3-bucket-owner <owner>] [--documents <file path,file path,...>]
      [--execution-id <?>] [--log-directory <file path>]
      [--log-s3-bucket-name <name>] [--log-s3-bucket-owner <owner>]
      [--log-s3-key-prefix <?>] [--parameters name1=value1,name2=value2...]
      [--phases <phase name>] [--state-directory <directory path>] [--version <?>]
      [--help] [--trace]
```

パラメータとオプション

パラメータ

--config*./config-example.json*

ショートフォーム:-c *./config-example.json*

設定ファイル(条件付き)。このパラメータには、このコマンドが実行しているコンポーネントの構成設定を含む JSON ファイルの場所が含まれます。設定ファイルで run コマンド設定を指定する場合、--documents パラメータは指定しないでください。入力設定の詳細については、AWSTOE run コマンドの入力を設定するを参照のこと。

有効な場所は以下の通り：

- ローカルファイルパス (*./config-example.json*)
- S3 URI (s3://*bucket/key*)

--cw-ignore-failures

ショートフォーム:N/A

CloudWatch ログからのログ記録の失敗を無視します。

--cw-log-group

ショートフォーム:N/A

CloudWatch ログLogGroupの名前。

--cw-log-region

ショートフォーム:N/A

CloudWatch ログに適用される AWS リージョン。

--cw-log-stream

ショートフォーム:N/A

console.log ファイルをストリーミングする AWSTOE 場所を指示する CloudWatch ログLogStreamの名前。

--document-s3-バケット所有者

ショートフォーム:N/A

S3 URIベースのドキュメントのバケット所有者のアカウントID。

ドキュメント*./doc-1.yaml,./doc-n.yaml*

ショートフォーム:-d *./doc-1.yaml*,*./doc-n*

コンポーネント文書 (条件付き)。このパラメータには、実行する YAML コンポーネントドキュメントのファイルロケーションのカンマ区切りリストが含まれます。--documents パラメーターを使用してrunコマンドにYAMLドキュメントを指定する場合、--configパラメーターを指定してはなりません。

有効な場所は以下の通り：

- ローカルファイルパス (*./component-doc-example.yaml)*。
- S3 URI (s3://*bucket/key*)。
- Image Builder コンポーネントのビルドバージョン ARNs (arn:aws:imagebuilder:us-west-*2:123456789012*:component/*my-example-component*/2021.12.02/1)。

> **Note**
>
> リスト内の項目間にはスペースがなく、カンマのみが入ります。

--実行-id

短縮形： -i

これは現在のrunコマンドの実行に適用される固有の ID です。この ID は出力ファイル名とログファイル名に含まれ、これらのファイルを一意に識別し、現在のコマンド実行にリンクします。この設定を省略すると、 は GUID AWSTOE を生成します。

ログディレクトリ

短縮形： -l

がこのコマンド実行 AWSTOE のすべてのログファイルを保存する送信先ディレクトリ。デフォルトでは、このディレクトリは以下の親ディレクトリの中にある： `TOE_<DATETIME>_<EXECUTIONID>`. ログディレクトリを指定しない場合、 は現在の作業ディレクトリ () AWSTOE を使用します`.`。

--log-s3-bucket-name

短縮形： -b

コンポーネントログが Amazon S3 に保存されている場合 (推奨)、 はコンポーネントアプリケーションログをこのパラメータで という名前 AWSTOE の S3 バケットにアップロードします。

--log-s3-バケット所有者

ショートフォーム:N/A

コンポーネントログが Amazon S3 に保存されている場合 (推奨)、これは がログファイルを AWSTOE 書き込むバケットの所有者アカウント ID です。

--log-s3-キープレフィックス

短縮形： -k

コンポーネントログがAmazon S3に保存されている場合（推奨）、これはバケット内のログの場所を示すS3オブジェクトキーのプレフィックスです。

--パラメータ名 ***## 1*** =***# 1*** ,***## 2*** =***# 2*** ...

ショートフォーム:N/A

パラメータは、コンポーネントドキュメントで定義される変更可能な変数であり、呼び出し元のアプリケーションが実行時に提供できる設定を持つ。

--phases

ショートフォーム:-p

YAML コンポーネントドキュメントから実行するフェーズを指定するカンマ区切りのリスト。コンポーネントドキュメントに追加のフェーズが含まれている場合、そのフェーズは実行されません。

--state-ディレクトリ

短縮形： -s

状態追跡ファイルが保存されているファイルパス。

--version

短縮形： -v

コンポーネントアプリケーションのバージョンを指定します。

オプション

--help

短縮形： -h

コンポーネント管理アプリケーションオプションを使用するためのヘルプマニュアルを表示します。

--trace

短縮形： -t

コンソールへの冗長ロギングを有効にする。

awstoe 検証コマンド

このコマンドを実行すると、--documents パラメータで指定された各コンポーネントドキュメントの YAML ドキュメント構文が検証されます。

構文

```
awstoe validate [--document-s3-bucket-owner <owner>]
      --documents <file path,file path,...> [--help] [--trace]
```

パラメータとオプション

パラメータ

--document-s3-バケット所有者

ショートフォーム:N/A

S3 URI ベースのドキュメントのソースアカウント ID が提供されました。

ドキュメント*./doc-1.yaml,./doc-n.yaml*

ショートフォーム:-d *./doc-1.yaml*,*./doc-n*

コンポーネントのドキュメント(必須)。このパラメータには、実行する YAML コンポーネントドキュメントのファイルロケーションのカンマ区切りリストが含まれます。有効な場所は以下の通り：

- ローカルファイルパス (*./component-doc-example.yaml*)
- S3 URI (s3://*bucket/key*)
- Image Builder コンポーネントのビルドバージョン ARNs (arn:aws:imagebuilder:us-west-*2:123456789012*:component/*my-example-component*/2021.12.02/1)

> **Note**
>
> リスト内の項目間にはスペースがなく、カンマのみが入ります。

オプション

--help

短縮形： -h

コンポーネント管理アプリケーションオプションを使用するためのヘルプマニュアルを表示します。

--trace

短縮形： -t

コンソールへの冗長ロギングを有効にする。

EC2 Image Builder リソースの管理

リソースは、イメージパイプラインを構成する構成要素であり、それらのパイプラインが生成するイメージでもあります。この章では、コンポーネント、レシピ、イメージなどの Image Builder リソースの作成、管理、共有、およびインフラストラクチャ設定と配布設定について説明します。

> **Note**
>
> Image Builder リソースを管理しやすくするために、タグ形式で各リソースに独自のメタデータを割り当てることができます。タグを使用して、 AWS リソースを目的、所有者、環境などさまざまな方法で分類します。これは、同じ種類のリソースが多い場合に役立ちます。リソースに割り当てたタグに基づいて、特定のリソースを簡単に識別できます。
> の Image Builder コマンドを使用してリソースにタグを付ける方法の詳細については AWS CLI、このガイドのリソースのタグ付け「」セクションを参照してください。

内容

Image Builder によるコンポーネントの管理

Image Builder は AWS Task Orchestrator and Executor 、 (AWSTOE) コンポーネント管理アプリケーションを使用して複雑なワークフローをオーケストレーションします。 AWSTOE アプリケーションと連携するビルドおよびテストコンポーネントは、イメージをカスタマイズまたはテストする

ためのスクリプトを定義する YAML ドキュメントに基づいています。AMI イメージの場合、Image Builder は Amazon EC2 ビルドインスタンスとテストインスタンスにコンポーネントと AWSTOE コンポーネント管理アプリケーションをインストールします。コンテナイメージの場合、コンポーネントと AWSTOE コンポーネント管理アプリケーションは実行中のコンテナ内にインストールされます。

Image Builder は AWSTOE 、 を使用してすべてのインスタンス上のアクティビティを実行します。Image Builder コマンドを実行したり、Image Builder コンソールを使用した AWSTOE りするときに、 を操作するための追加のセットアップは必要ありません。

> **Note**
>
> Amazon が管理するコンポーネントのサポート期間が終了すると、そのコンポーネントはメンテナンスされなくなります。この問題が発生する約 4 週間前に、コンポーネントを使用しているすべてのアカウントに通知が届き、そのアカウント内の影響を受けるレシピのリストが各 AWS Health Dashboardから届きます。の詳細については AWS Health、「AWS Health ユーザーガイド」を参照してください。

新しいイメージを構築するためのワークフローステージ

新しいイメージを構築するための Image Builder ワークフローには、次の 2 つの段階があります。

1. ビルドステージ (スナップショット前) — ビルドステージでは、ベースイメージを実行している Amazon EC2 ビルドインスタンスに変更を加え、新しいイメージのベースラインを作成します。例えば、レシピには、アプリケーションをインストールしたり、オペレーティングシステムのファイアウォール設定を変更したりするコンポーネントを含めることができます。

   ビルドステージでは以下のコンポーネントフェーズが実行されます。

   - buid (構築)
   - validate

   この段階が正常に完了すると、Image Builder はテスト段階以降に使用するスナップショットまたはコンテナイメージを作成します。
2. テストステージ (スナップショット後) — テストステージでは、AMI を作成するイメージとコンテナイメージにいくつかの違いがあります。AMI ワークフローでは、Image Builder はビルドステージの最終ステップとして作成したスナップショットから EC2 インスタンスを起動します。新しいインスタンスでテストを実行して設定を検証し、インスタンスが期待どおりに機能していること

を確認します。コンテナワークフローでは、テストはビルドに使用したのと同じインスタンスで実行されます。

テスト段階では、レシピに含まれるすべてのコンポーネントに対して次のコンポーネントフェーズが実行されます。

- test

このコンポーネントフェーズは、ビルドコンポーネントタイプとテストコンポーネントタイプの両方に適用されます。この段階が正常に完了すると、Image Builder はスナップショットまたはコンテナイメージから最終イメージを作成して配布できます。

> **Note**
>
> AWSTOE ではコンポーネントドキュメントで多くのフェーズを定義できますが、Image Builder には、実行するフェーズと、実行するステージに関する厳格なルールがあります。ビルド段階でコンポーネントを実行するには、コンポーネントドキュメントで少なくとも 1 つのフェーズ (`build` または `validate`) を定義する必要があります。テスト段階でコンポーネントを実行するには、コンポーネントドキュメントで `test` フェーズを定義する必要があり、他のフェーズを定義してはなりません。
> Image Builder はステージを独立して実行するため、コンポーネントドキュメント内の参照を連鎖させることはステージの境界を越えることはできません。ビルドステージで実行されるフェーズの値をテストステージで実行されるフェーズにチェーンすることはできません。ただし、目的のターゲットに入力パラメータを定義し、コマンドラインから値を渡すことはできます。Image Builder レシピのコンポーネントパラメータ設定の詳細については、「EC2 Image Builder で AWSTOE コンポーネントパラメータを管理する」を参照してください。

ビルドインスタンスまたはテストインスタンスのトラブルシューティングを支援するために、入力ドキュメントとログファイルを含むログフォルダ AWSTOE を作成し、コンポーネントが実行されるたびに何が起こっているかを追跡します。パイプライン設定で Amazon S3 バケットを設定した場合、ログもそこに書き込まれます。YAML ドキュメントとログ出力の詳細については、「でコンポーネントドキュメントを使用する AWSTOE」を参照してください。

> **Tip**
>
> 追跡用のコンポーネントが多い場合に、タグ付けを使用すると、割り当てたタグに基づいて特定のコンポーネントまたはバージョンを識別できます。の Image Builder コマンドを使用

してリソースにタグを付ける方法の詳細については AWS CLI、このガイドのリソースのタグ付け「」セクションを参照してください。

このセクションでは、Image Builder コンソールまたは AWS CLI内のコマンドを使用して、コンポーネントを一覧表示、表示、作成、インポートする方法について説明します。

内容

YAML コンポーネントドキュメントを作成します。

コンポーネントを構築するには、YAML アプリケーションコンポーネントドキュメントを提供します。これは、コンポーネントの作成に必要なフェーズとステップを表しています。

このセクションの例では、コンポーネント管理アプリケーションでUpdateOSアクションモジュールを呼び出すビルド AWSTOE コンポーネントを作成します。モジュールでは、オペレーティングシステムを更新します。UpdateOS アクションの使用方法の詳細については、UpdateOSを参照してください。 AWSTOE YAML アプリケーションコンポーネントドキュメントのフェーズ、ステップ、構文の詳細については、「 でドキュメント AWSTOEを使用する」を参照してください。

> **Note**
>
> Image Builder は、パイプラインワークフロー内のコンポーネントタイプを決定します。このワークフローは、ビルドプロセスのビルドステージとテストステージに対応しています。Image Builder は、コンポーネントタイプを次のように決定します。
>
> - ビルド — これはデフォルトのコンポーネントタイプです。テストコンポーネントとして分類されないものはすべてビルドコンポーネントです。このタイプのコンポーネントはビルドステージで実行されます。このビルドコンポーネントにtestフェーズが定義されている場合、そのフェーズはテストステージ中に実行されます。

- テスト — テストコンポーネントとして認定されるには、コンポーネントドキュメントにtestという名前のフェーズが 1 つだけ含まれている必要があります。ビルドコンポーネントの設定に関連するテストでは、スタンドアロンのテストコンポーネントを使用しないことをお勧めします。むしろ、関連するビルドコンポーネントのtestフェーズを使用してください。

Image Builder がステージとフェーズを使用してビルドプロセスのコンポーネントワークフローを管理する方法の詳細については、Image Builder によるコンポーネントの管理を参照してください。

サンプルアプリケーション用の YAML アプリケーションコンポーネントドキュメントを作成するには、使用しているイメージオペレーティングシステムに対応するタブの手順に従ってください。

Linux

YAML コンポーネントファイルを作成します。

ファイル編集ツールを使用して、*update-linux-os.yaml* という名前のファイルを作成します。次の内容を含めます:

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy, modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
name: update-linux-os
description: Updates Linux with the latest security updates.
```

```
schemaVersion: 1
phases:
  - name: build
    steps:
    - name: UpdateOS
      action: UpdateOS
# Document End
```

> **Tip**
>
> このオンライン YAML Validator のようなツールを使用するか、コード環境の YAML lint 拡張機能を使用して、YAML が正しい形式であることを確認してください。

Windows

YAML コンポーネントファイルを作成します。

ファイル編集ツールを使用して、*update-windows-os.yaml* という名前のファイルを作成します。次の内容を含めます:

```
# Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: MIT-0
#
# Permission is hereby granted, free of charge, to any person obtaining a copy of
 this
# software and associated documentation files (the "Software"), to deal in the
 Software
# without restriction, including without limitation the rights to use, copy, modify,
# merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
# permit persons to whom the Software is furnished to do so.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED,
# INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
# PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
# HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
# OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
# SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
name: update-windows-os
description: Updates Windows with the latest security updates.
schemaVersion: 1.0
```

```
phases:
  - name: build
    steps:
      - name: UpdateOS
        action: UpdateOS
# Document End
```

> **Tip**
>
> このオンライン YAML Validator のようなツールを使用するか、コード環境の YAML lint 拡張機能を使用して、YAML が正しい形式であることを確認してください。

EC2 Image Builder で AWSTOE コンポーネントパラメータを管理する

AWSTOE コンポーネントパラメータの作成と設定を含むコンポーネントは、EC2 Image Builder コンソールから直接、または AWS CLI コマンド、または Image Builder SDKs のいずれかを使用して管理できます。このセクションでは、コンポーネントでのパラメータの作成と使用、および Image Builder コンソールと AWS CLI コマンドを使用したコンポーネントパラメータの設定について説明します。

> **Important**
>
> コンポーネントパラメータはプレーンテキストの値で、 AWS CloudTrailに記録されます。シークレットを保存するには、 AWS Secrets Manager または AWS Systems Manager Parameter Store を使用することをお勧めします。Secrets Managerの詳細については、AWS Secrets Manager ユーザーガイドのSecrets Managerとはを参照してください。 AWS Systems Manager パラメータストアについては、AWS Systems Manager ユーザーガイドのAWS Systems Manager パラメータストアを参照。

YAML コンポーネントドキュメントでのパラメータの使用

コンポーネントを構築するには、YAML アプリケーションコンポーネントドキュメントを提供します。これは、コンポーネントの作成に必要なフェーズとステップを表しています。コンポーネントを参照するレシピでは、ランタイムにパラメーターを設定して値をカスタマイズできます。デフォルト値は、パラメーターが特定の値に設定されていない場合に有効になります。

入力パラメータを使用してコンポーネントドキュメントを作成します。

このセクションでは、YAML コンポーネントドキュメントで入力パラメータを定義し使用する方法を示します。

Image Builder のビルドインスタンスまたはテストインスタンスでパラメータを使用してコマンドを実行する YAML アプリケーションコンポーネントドキュメントを作成するには、ご使用のイメージオペレーティングシステムに対応する手順に従ってください。

Linux

YAML コンポーネントドキュメントを作成します。

ファイル編集ツールを使用して、*hello-world-test.yaml* という名前のファイルを作成します。次の内容を含めます:

```
# Document Start
#
name: "HelloWorldTestingDocument-Linux"
description: "Hello world document to demonstrate parameters."
schemaVersion: 1.0
parameters:
  - MyInputParameter:
      type: string
      default: "It's me!"
      description: This is an input parameter.
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Build phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Validate phase. My input parameter value is
 {{ MyInputParameter }}"
```

```
  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecuteBash
        inputs:
          commands:
            - echo "Hello World! Test phase. My input parameter value is
 {{ MyInputParameter }}"
# Document End
```

> **Tip**
>
> このオンライン YAML Validator のようなツールを使用するか、コード環境の YAML lint 拡張機能を使用して、YAML が正しい形式であることを確認してください。

Windows

YAML コンポーネントドキュメントを作成します。

ファイル編集ツールを使用して、*hello-world-test.yaml* という名前のファイルを作成します。次の内容を含めます:

```
# Document Start
#
name: "HelloWorldTestingDocument-Windows"
description: "Hello world document to demonstrate parameters."
schemaVersion: 1.0
parameters:
  - MyInputParameter:
      type: string
      default: "It's me!"
      description: This is an input parameter.
phases:
  - name: build
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host "Hello World! Build phase. My input parameter value is
 {{ MyInputParameter }}"
```

```
  - name: validate
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host "Hello World! Validate phase. My input parameter value is
 {{ MyInputParameter }}"

  - name: test
    steps:
      - name: HelloWorldStep
        action: ExecutePowerShell
        inputs:
          commands:
            - Write-Host "Hello World! Test phase. My input parameter value is
 {{ MyInputParameter }}"
# Document End
```

> **Tip**
>
> このオンライン YAML Validator のようなツールを使用するか、コード環境の YAML lint 拡張機能を使用して、YAML が正しい形式であることを確認してください。

AWSTOE YAML アプリケーションコンポーネントドキュメントのフェーズ、ステップ、構文の詳細については、「 でドキュメント AWSTOEを使用する」を参照してください。パラメータとその要件についての詳細は、変数の定義と参照 AWSTOEページのパラメータセクションを参照のこと。

YAML コンポーネントドキュメントからコンポーネントを作成する

AWSTOE コンポーネントの作成に使用する方法にかかわらず、YAML アプリケーションコンポーネントドキュメントは常にベースラインとして必要です。

- Image Builder コンソールを使用して YAML ドキュメントから直接コンポーネントを作成するには、「Image Builder コンソールを使用してコンポーネントを作成する」を参照してください。
- で Image Builder コマンドを使用してコンポーネント AWS CLI を作成するには、「」を参照してくださいを使用して Image Builder で AWSTOE コンポーネントを作成する AWS CLI。これらの例の YAML ドキュメント名を Hello World YAML ドキュメントの名前 (*hello-world-test.yaml*) に置き換えてください。

Image Builder レシピのコンポーネントパラメーターの設定 (コンソール)

コンポーネントパラメータの設定は、イメージレシピでもコンテナレシピでも同じように機能します。新しいレシピ、またはレシピの新しいバージョンを作成するときは、[ビルドコンポーネント]リストと[テストコンポーネント]リストから、どのコンポーネントを含めるかを選択します。コンポーネントリストには、イメージ用に選択したベースオペレーティングシステムに該当するコンポーネントが含まれています。

コンポーネントを選択すると、そのコンポーネントはコンポーネントリストのすぐ下の [選択したコンポーネント] セクションに表示されます。選択した各コンポーネントの設定オプションが表示されます。コンポーネントに入力パラメータが定義されている場合、[入力パラメータ]という展開可能なセクションとして表示されます。

コンポーネントに定義されている各パラメーターには、以下のパラメーター設定が表示されます。

- パラメータ名 (編集不可) - パラメータの名前。
- 説明 (編集不可) - パラメータの説明。
- 型 (編集不可) - パラメータ値のデータ型。
- 値 – パラメータの値。このレシピでこのコンポーネントを初めて使用し、入力パラメータにデフォルト値が定義されている場合、デフォルト値はグレーアウトされたテキストを持つ値ボックスに表示されます。他の値が入力されていない場合、Image Builder はデフォルト値を使用します。

コンポーネントの詳細を一覧表示して表示する

このセクションでは、EC2 Image Builder レシピで使用する AWS Task Orchestrator and Executor (AWSTOE) コンポーネントに関する情報を検索し、詳細を表示する方法について説明します。

コンポーネントの詳細

- AWSTOE コンポーネントを一覧表示する
- コンポーネントのビルドバージョン (AWS CLI) を一覧表示します。
- コンポーネントの詳細を取得 (AWS CLI)
- コンポーネントポリシーの詳細を取得 (AWS CLI)

AWSTOE コンポーネントを一覧表示する

次のいずれかの方法を使用して、 AWSTOE コンポーネントを一覧表示およびフィルタリングできます。

AWS Management Console

でコンポーネントのリストを表示するには AWS Management Console、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから コンポーネント を選択します。デフォルトでは、Image Builder にはアカウントが所有するコンポーネントのリストが表示されます。
3. オプションでコンポーネントの所有権でフィルタリングできます。自分が所有していないがアクセスできるコンポーネントを表示するには、所有者タイプドロップダウンリストを展開し、いずれかの値を選択します。所有者タイプリストは、検索バーの検索テキストボックスの横にあります。次の値から選択できます。

   - クイックスタート（Amazonマネージド） - Amazonが作成管理する一般公開されているコンポーネント。
   - Owned by me - あなたが作成したコンポーネント。デフォルトではこれが選択されています。
   - Shared with me - 他人が作成し、自分のアカウントからあなたと共有したコンポーネント。
   - サードパーティー管理 – でサブスクライブしているサードパーティーが所有するコンポーネント AWS Marketplace。

AWS CLI

次の例は、 list-components コマンドを使用して、アカウントが所有する AWSTOE コンポーネントのリストを返す方法を示しています。

```
aws imagebuilder list-components
```

オプションでコンポーネントの所有権でフィルタリングできます。owner 属性は、一覧表示するコンポーネントの所有者を定義します。デフォルトでは、このリクエストはアカウントが所有するコンポーネントのリストを返します。`--owner` コンポーネント所有者別に結果をフィルタリングするには、list-components コマンドを実行するときにパラメータで次の値のいずれかを指定します。

コンポーネントオーナーの値

- 自分

- Amazon
- ThirdParty
- Shared

以下の例では、list-componentsコマンドに--ownerパラメータを付けて結果をフィルタリングしている。

```
aws imagebuilder list-components --owner Self
{
    "requestId": "012a3456-b789-01cd-e234-fa5678b9012b",
    "componentVersionList": [
        {
            "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/sample-
component01/1.0.0",
            "name": "sample-component01",
            "version": "1.0.0",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "dateCreated": "2020-09-24T16:58:24.444Z"
        },
        {
            "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/sample-
component01/1.0.1",
            "name": "sample-component01",
            "version": "1.0.1",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "dateCreated": "2021-07-10T03:38:46.091Z"
        }
    ]
}
```

```
aws imagebuilder list-components --owner Amazon
```

```
aws imagebuilder list-components --owner Shared
```

```
aws imagebuilder list-components --owner ThirdParty
```

コンポーネントのビルドバージョン (AWS CLI) を一覧表示します。

次の例は、list-component-build-versionsコマンドを使用して特定のセマンティックバージョンを持つコンポーネントビルドバージョンを一覧表示する方法を示しています。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。

```
aws imagebuilder list-component-build-versions --component-version-arn
 arn:aws:imagebuilder:us-west-2:123456789012:component/example-component/1.0.1
{
    "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "componentSummaryList": [
        {
            "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/
examplecomponent/1.0.1/1",
            "name": "examplecomponent",
            "version": "1.0.1",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "description": "An example component that builds, validates and tests an
 image",
            "changeDescription": "Updated version.",
            "dateCreated": "2020-02-19T18:53:45.940Z",
            "tags": {
                "KeyName": "KeyValue"
            }
        }
    ]
}
```

コンポーネントの詳細を取得 (AWS CLI)

次の例は、コンポーネントの Amazon リソースネーム (ARN) を指定するときに、「get-component」コマンドを使用してコンポーネントの詳細を取得する方法を示しています。

```
aws imagebuilder get-component --component-build-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/example-component/1.0.1/1
   {
    "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11112",
    "component": {
```

```
        "arn": "arn:aws:imagebuilder:us-west-2:123456789012:component/
examplecomponent/1.0.1/1",
        "name": "examplecomponent",
        "version": "1.0.1",
        "type": "BUILD",
        "platform": "Linux",
        "owner": "123456789012",
        "data": "name: HelloWorldTestingDocument\ndescription: This is hello world
 testing document... etc.\"\n",
        "encrypted": true,
        "dateCreated": "2020-09-24T16:58:24.444Z",
        "tags": {}
    }
}
```

コンポーネントポリシーの詳細を取得 (AWS CLI)

次の例は、コンポーネントの Amazon リソースネーム (ARN) を指定するときに、get-component-policyコマンドを使用してコンポーネントの詳細を取得する方法を示しています。

```
aws imagebuilder get-component-policy --component-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/example-component/1.0.1
```

Image Builder コンソールを使用してコンポーネントを作成する

Image Builder コンソールから AWSTOE アプリケーションコンポーネントを作成するには、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから コンポーネント を選択します。次に コンポーネントを作成 を選択します。
3. 「コンポーネントの作成」ページの「コンポーネントの詳細」で、次のように入力します。

    a. イメージオペレーティングシステム (OS)。コンポーネントと互換性のあるオペレーティングシステムを指定します。

    b. コンポーネントカテゴリ。ドロップダウンから、作成するビルドコンポーネントまたはテストコンポーネントのタイプを選択します。

    c. コンポーネント名 コンポーネントの名前を入力します。

d. コンポーネントのバージョン コンポーネントのバージョン番号を入力します。

e. 説明。ジョブの説明を追加して、ジョブを識別することも可能です。

f. 説明の変更 このバージョンのコンポーネントに加えられた変更点を理解しやすいように、オプションで説明を入力します。

4. 「定義文書」セクションのデフォルトオプションは「文書コンテンツの定義」です。コンポーネントドキュメントは、イメージを作成するために Image Builder がビルドインスタンスとテストインスタンスで実行するアクションを定義します。

「コンテンツ」ボックスに、YAML コンポーネントドキュメントの内容を入力します。Linux 用の Hello World サンプルから始めるには、「サンプルを使用する」オプションを選択します。YAML コンポーネントドキュメントの作成方法や、そのページから UpdateOS サンプルをコピーして貼り付ける方法について詳しくは、YAML コンポーネントドキュメントを作成します。を参照してください。

5. コンポーネントの詳細を入力したら、「コンポーネントを作成」を選択します。

> **Note**
>
> レシピを作成または更新するときに新しいコンポーネントを表示するには、ビルドまたはテスト用のコンポーネントリストに Owned by me フィルターを適用します。フィルターは、コンポーネントリストの上部にある、検索ボックスの横にあります。

6. コンポーネントを削除するには、コンポーネント ページで、削除するコンポーネントの横にあるチェックボックスをオンにします。 Actions （ アクション ）ドロップダウンから Deploy API (デプロイAPI ）を選択します。

新しいコンポーネントバージョンを作成するには、次の手順に従います。

1. どこから始めるかによって:

- コンポーネントリストページから — コンポーネント名の横にあるチェックボックスを選択し、「アクション」メニューから「新規バージョンを作成」を選択します。
- コンポーネントの詳細ページから — 見出しの右上隅にある 新規バージョンを作成 ボタンをクリックします。

2. 「Create Component」ページが表示されると、コンポーネント情報には現在の値がすでに入力されています。「コンポーネントを作成」の手順に従って、コンポーネントを更新します。これにより、コンポーネントバージョンには固有のセマンティックバージョンを確実に入力できま

す。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。

を使用してコンポーネントを作成する AWS CLI

このセクションでは、Image Builder コマンドを使用して から AWS Task Orchestrator and Executor (AWSTOE) コンポーネントを作成する方法について説明します AWS Command Line Interface。コンポーネントを構築するには、YAML アプリケーションコンポーネントドキュメントを提供します。これは、コンポーネントの作成に必要なフェーズとステップを表しています。YAML コンポーネントドキュメントを新規作成するには、YAML コンポーネントドキュメントを作成します。を参照してください。

を使用して Image Builder で AWSTOE コンポーネントを作成する AWS CLI

このセクションでは、次のように、 で Image Builder コマンドをセットアップして使用 AWS CLI して AWSTOE アプリケーションコンポーネントを作成する方法について説明します。

- YAML コンポーネントドキュメントを S3 バケットにアップロードし、コマンドラインから参照できるようにします。
- create-component コマンドを使用して AWSTOE アプリケーションコンポーネントを作成します。
- list-componentsコマンドと名前フィルターを使用してコンポーネントのバージョンを一覧表示し、すでに存在するバージョンを確認します。この出力を使用して、更新に必要な次のバージョンを決定できます。

入力 YAML ドキュメントから AWSTOE アプリケーションコンポーネントを作成するには、イメージオペレーティングシステムプラットフォームに一致するステップに従います。

Linux

アプリケーションコンポーネントドキュメントを Amazon S3 に保存する

S3 バケットは、 AWSTOE アプリケーションコンポーネントのソースドキュメントのリポジトリとして使用できます。コンポーネントドキュメントを保存するには、次の手順に従います。

- ドキュメントを Amazon S3 にアップロードする

  ドキュメントが 64 KB 未満の場合は、このステップをスキップできます。64 KB とその以上のサイズのドキュメントは Amazon S3 に保存する必要があります。

  ```
  aws s3 cp update-linux-os.yaml s3://my-s3-bucket/my-path/update-linux-os.yaml
  ```

YAML ドキュメントからコンポーネントを作成します。

で使用するcreate-componentコマンドを効率化するには AWS CLI、コマンドに渡すすべてのコンポーネントパラメータを含む JSON ファイルを作成します。前のステップで作成した*update-linux-os.yaml*ドキュメントの場所を含めてください。uriキー値のペアには、ファイル参照が含まれます。

> **Note**
>
> JSON ファイル内のデータ値の命名規則は、Image Builder API アクションリクエストパラメータに指定されているパターンに従います。API コマンドリクエストパラメータを確認するには、EC2 Image Builder API リファレンスの CreateComponent コマンドを参照してください。
> データ値をコマンドラインパラメータとして指定するには、AWS CLI コマンドリファレンスで指定されているパラメータ名を参照してください。

1. CLI 入力 JSON ファイルの作成

   ファイル編集ツールを使用して、*create-update-linux-os-component.json* という名前のファイルを作成します。次の内容を含めます:

   ```
   {
    "name": "update-linux-os",
    "semanticVersion": "1.1.2",
    "description": "An example component that updates the Linux operating system",
    "changeDescription": "Initial version.",
    "platform": "Linux",
    "uri": "s3://my-s3-bucket/my-path/update-linux-os.yaml",
    "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/98765432-
   b123-456b-7f89-0123456f789c",
    "tags": {
   ```

```
  "MyTagKey-purpose": "security-updates"
 }
}
```

> **Note**
>
> - JSON ファイルパスの先頭にfile://ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

2. コンポーネントを作成する

   以下のコマンドを使用して、前のステップで作成した JSON ファイルのファイル名を参照してコンポーネントを作成します。

```
aws imagebuilder create-component --cli-input-json file://create-update-linux-os-component.json
```

> **Note**
>
> - JSON ファイルパスの先頭にfile://ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

Windows

アプリケーションコンポーネントドキュメントを Amazon S3 に保存する

S3 バケットは、 AWSTOE アプリケーションコンポーネントのソースドキュメントのリポジトリとして使用できます。コンポーネントドキュメントを保存するには、次の手順に従います。

- ドキュメントを Amazon S3 にアップロードする

  ドキュメントが 64 KB 未満の場合は、このステップをスキップできます。64 KB とその以上のサイズのドキュメントは Amazon S3 に保存する必要があります。

  ```
  aws s3 cp update-windows-os.yaml s3://my-s3-bucket/my-path/update-windows-os.yaml
  ```

YAML ドキュメントからコンポーネントを作成します。

で使用するcreate-componentコマンドを効率化するには AWS CLI、コマンドに渡すすべてのコンポーネントパラメータを含む JSON ファイルを作成します。前のステップで作成した*update-windows-os.yaml*ドキュメントの場所を含めてください。uriキー値のペアには、ファイル参照が含まれます。

> **Note**
>
> JSON ファイル内のデータ値の命名規則は、Image Builder API アクションリクエストパラメータに指定されているパターンに従います。API コマンドリクエストパラメータを確認するには、EC2 Image Builder API リファレンスの CreateComponent コマンドを参照してください。
> データ値をコマンドラインパラメータとして指定するには、AWS CLI コマンドリファレンスで指定されているパラメータ名を参照してください。

1. CLI 入力 JSON ファイルの作成

   ファイル編集ツールを使用して、*create-update-windows-os-component.json* という名前のファイルを作成します。次の内容を含めます:

   ```
   {
    "name": "update-windows-os",
    "semanticVersion": "1.1.2",
    "description": "An example component that updates the Windows operating system.",
    "changeDescription": "Initial version.",
    "platform": "Windows",
    "uri": "s3://my-s3-bucket/my-path/update-windows-os.yaml",
   ```

```
 "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/98765432-
b123-456b-7f89-0123456f789c",
 "tags": {
  "MyTagKey-purpose": "security-updates"
 }
}
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

2. コンポーネントを作成する

以下のコマンドを使用して、前のステップで作成した JSON ファイルのファイル名を参照してコンポーネントを作成します。

```
aws imagebuilder create-component --cli-input-json file://create-update-windows-
os-component.json
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

AWSTOE 更新のコンポーネントバージョニング (AWS CLI)

AWSTOE コンポーネント名とバージョンは、コンポーネントのプレフィックスの後にコンポーネントの Amazon リソースネーム (ARN) に埋め込まれます。コンポーネントの新しいバージョンにはそれぞれ固有の ARN があります。新しいバージョンを作成する手順は、そのコンポーネント名に対し

てセマンティックバージョンが一意である限り、新しいコンポーネントを作成する手順と完全に同じです。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。

次の論理的なバージョンを割り当てるために、まず変更したいコンポーネントの既存のバージョンのリストを取得してください。list-components コマンドを とともに使用し AWS CLI、名前をフィルタリングします。

この例では、以前の Linux の例で作成したコンポーネントの名前をフィルターします。作成したコンポーネントをリストアップするには、create-componentコマンドで使用したJSONファイルのnameパラメータの値を使用する。

```
aws imagebuilder list-components --filters name="name",values="update-linux-os"
{
    "requestId": "123a4567-b890-123c-45d6-ef789ab0cd1e",
    "componentVersionList": [
        {
            "arn": "arn:aws:imagebuilder:us-west-2:1234560087789012:component/update-
linux-os/1.0.0",
            "name": "update-linux-os",
            "version": "1.0.0",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "dateCreated": "2020-09-24T16:58:24.444Z"
        },
        {
            "arn": "arn:aws:imagebuilder:us-west-2:1234560087789012:component/update-
linux-os/1.0.1",
            "name": "update-linux-os",
            "version": "1.0.1",
            "platform": "Linux",
            "type": "BUILD",
            "owner": "123456789012",
            "dateCreated": "2021-07-10T03:38:46.091Z"
        }
    ]
}
```

結果に基づいて、次のバージョンを決定できます。

コンポーネントのインポート (AWS CLI)

シナリオによっては、既存のスクリプトから始めるほうが簡単な場合もあります。このシナリオでは、以下の例を使うことができます。

この例では、*import-component.json* (図のように) というファイルがあることを前提としています。ファイルは、既に にアップロードAdminConfig.ps1されている という PowerShell スクリプトを直接参照することに注意してください*my-s3-bucket*。現在、コンポーネントSHELLがサポートされているformat。

```
{
"name": "MyImportedComponent",
"semanticVersion": "1.0.0",
"description": "An example of how to import a component",
"changeDescription": "First commit message.",
"format": "SHELL",
"platform": "Windows",
"type": "BUILD",
"uri": "s3://my-s3-bucket/AdminConfig.ps1",
"kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/60763706-
b131-418b-8f85-3420912f020c"
}
```

コンポーネントをインポートするには、以下のコマンドを実行します。

```
aws imagebuilder import-component --cli-input-json file://import-component.json
```

リソースをクリーンアップする

予期しない料金が発生しないように、このガイドの例で作成したリソースとパイプラインは必ずクリーンアップしてください。Image Builder でのリソースの削除については、「EC2 Image Builder resourcesの削除」を参照してください。

レシピの管理

EC2 Image Builder レシピでは、新しいイメージを作成するための開始点として使用するベースイメージと、イメージをカスタマイズしてすべてが期待どおりに動作することを確認するために追加する一連のコンポーネントを定義します。Image Builder では、コンポーネントごとに自動的にバー

ジョンを選択できます。デフォルトでは、レシピに最大 20 個のコンポーネントを適用できます。これには、ビルドコンポーネントとテストコンポーネントの両方が含まれます。

レシピを作成後に変更または置換することはできません。レシピを作成した後でコンポーネントを更新するには、新しいレシピまたはレシピバージョンを作成する必要があります。既存のレシピにはいつでもタグを適用できます。の Image Builder コマンドを使用してリソースにタグを付ける方法の詳細については AWS CLI、このガイドのリソースのタグ付け「」セクションを参照してください。

> **Tip**
>
> レシピで Amazon マネージドコンポーネントを使用することも、 AWS Task Orchestrator and Executor （ AWSTOE) アプリケーションで独自のカスタムコンポーネントを開発することもできます。開始するには、 の使用を開始する AWSTOE を参照してください。

このセクションでは、レシピを一覧表示、表示、作成する方法について説明します。

内容

- イメージレシピの詳細を一覧と詳細表示
- コンテナレシピ詳細の一覧表示
- イメージレシピの新しいバージョンを作成します。
- 新しいイメージレシピのバージョンを作成
- リソースをクリーンアップする

イメージレシピの詳細を一覧と詳細表示

このセクションでは、EC2 Image Builder イメージレシピの情報を検索したり詳細を表示したりするさまざまな方法について説明します。

イメージレシピの詳細

- イメージレシピを一覧表示する (コンソール)
- イメージレシピを一覧表示する (AWS CLI)
- イメージレシピの詳細を表示する (コンソール)
- イメージレシピの詳細を取得 (AWS CLI)
- イメージレシピポリシーの詳細を取得 (AWS CLI)

イメージレシピを一覧表示する (コンソール)

アカウントで作成されたイメージレシピのリストを Image Builder コンソールに表示するには、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインからイメージレシピを選択します。これにより、アカウントで作成されたイメージレシピのリストが表示されます。
3. 詳細を表示したり、新しいレシピバージョンを作成したりするには、[レシピ名] リンクを選択します。レシピの詳細ビューが開きます。

> **Note**
>
> また、レシピ名の横にあるチェックボックスを選択し、詳細を見るを選択することもできます。

イメージレシピを一覧表示する (AWS CLI)

次の例は、 AWS CLIを使ってすべてのイメージレシピを一覧表示する方法を示しています。

```
aws imagebuilder list-image-recipes
```

イメージレシピの詳細を表示する (コンソール)

Image Builder コンソールを使用して特定のイメージレシピの詳細を表示するには、イメージレシピを一覧表示する (コンソール)で説明されている手順を使用して、レビューするイメージレシピを選択します。

レシピの詳細ページでは

- レシピを削除します。Image Builder でのリソースの削除については、「EC2 Image Builder resourcesの削除」を参照してください。
- 新しいバージョンの作成
- レシピからパイプラインを作成する。このレシピから パイプラインを作成 を選択すると、パイプラインウィザードが表示されます。パイプラインウィザードを使って Image Builder パイプラインを作成する詳しい方法については、「EC2 Image Builder コンソールウィザードを使用してコンテナイメージパイプラインを作成します。」を参照してください。

> **Note**
>
> 既存のレシピからパイプラインを作成する場合、新しいレシピを作成するオプションは使用できません。

イメージレシピの詳細を取得 (AWS CLI)

次の例は、imagebuilder CLI コマンドを使用して Amazon リソースネーム (ARN) を指定してイメージレシピの詳細を取得する方法を示しています。

```
aws imagebuilder get-image-recipe --image-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-recipe/my-example-recipe/2020.12.03
```

イメージレシピポリシーの詳細を取得 (AWS CLI)

次の例は、imagebuilder CLI コマンドを使用して ARN を指定してイメージレシピポリシーの詳細を取得する方法を示しています。

```
aws imagebuilder get-image-recipe-policy --image-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-recipe/my-example-recipe/2020.12.03
```

コンテナレシピ詳細の一覧表示

このセクションでは、EC2 Image Builder コンテナレシピの情報を検索したり詳細を表示したりする方法について説明します。

コンテナレシピの詳細

- コンソールにコンテナレシピを一覧表示する
- コンテナレシピを AWS CLIで一覧表示します。
- コンソールでのコンテナレシピ詳細の表示
- コンテナレシピの詳細は、 AWS CLIで取得できます。
- でコンテナレシピポリシーの詳細を取得する AWS CLI

コンソールにコンテナレシピを一覧表示する

アカウントで作成されたコンテナレシピのリストを Image Builder コンソールに表示するには、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから、[コンテナレシピ] を選択します。アカウントで作成されたコンテナレシピのリストが表示されます。
3. 詳細を表示したり、新しいレシピバージョンを作成したりするには、[レシピ名] リンクを選択します。レシピの詳細ビューが開きます。

> **Note**
>
> [レシピ名] の横にあるチェックボックスをオンにして、[詳細を表示] を選択することもできます。

コンテナレシピを AWS CLIで一覧表示します。

次の例は、を使ってすべてのコンテナレシピを AWS CLIで一覧表示する方法を示しています。

```
aws imagebuilder list-container-recipes
```

コンソールでのコンテナレシピ詳細の表示

Image Builder コンソールで特定のコンテナレシピの詳細を表示するには、確認するコンテナレシピを選択し、コンソールにコンテナレシピを一覧表示するで説明されている手順に従います。

レシピ詳細 ページでは、以下の操作ができます。

- レシピを削除します。Image Builder でリソースを削除する方法の詳細については、「EC2 Image Builder resourcesの削除」を参照してください。
- 新しいバージョンの作成
- レシピからパイプラインを作成する。[このレシピからパイプラインを作成] を選択すると、パイプラインウィザードが表示されます。パイプラインウィザードを使って Image Builder パイプラインを作成する方法の詳細は、EC2 Image Builder コンソールウィザードを使用してコンテナイメージパイプラインを作成します。 を参照してください。

> **Note**
>
> 既存のレシピからパイプラインを作成する場合、新しいレシピを作成するオプションは使用できません。

コンテナレシピの詳細は、 AWS CLIで取得できます。

次の例は、imagebuilder CLI コマンドを使用して ARN を指定してコンテナレシピの詳細を取得する方法を示しています。

```
aws imagebuilder get-container-recipe --container-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:container-recipe/my-example-recipe/2020.12.03
```

でコンテナレシピポリシーの詳細を取得する AWS CLI

次の例は、imagebuilder CLI コマンドを使用して ARN を指定することで、コンテナレシピの詳細を取得する方法を示しています。

```
aws imagebuilder get-container-recipe-policy --container-recipe-arn
 arn:aws:imagebuilder:us-west-2:123456789012:container-recipe/my-example-
recipe/2020.12.03
```

イメージレシピの新しいバージョンを作成します。

このセクションでは、イメージレシピの新しいバージョンを作成する方法について説明します。

内容

- 新しいイメージレシピのバージョン (コンソール) を作成
- でイメージレシピを作成する AWS CLI
- VM をベースイメージとしてコンソールにインポートします。

新しいイメージレシピのバージョン (コンソール) を作成

新しいレシピバージョンを作成することは、新しいレシピを作成することと実質的に同じです。違いは、ほとんどの場合、基本レシピに合わせて特定の詳細が事前に選択されていることです。以下のリ

ストは、新しいレシピを作成することと、既存のレシピの新しいバージョンを作成することの違いを説明しています。

新しいバージョンの基本レシピの詳細

- 名前 - 編集不可。
- バージョン – 必須 この基本情報には、現在のバージョンやシーケンスがあらかじめ入力されていません。作成したいバージョン番号を<major>.<minor>.<patch>の形式で入力する。そのバージョンが既に存在している場合は、エラーが発生します。
- イメージを選択 オプション — 事前に選択されていますが、編集できます。ベースイメージのソースの選択を変更すると、選択した元のオプションに依存するその他の詳細が失われる可能性があります。

  基本イメージの選択に関連する詳細を表示するには、選択内容と一致するタブを選択してください。

  Managed image

  - イメージオペレーティングシステム（OS） - 編集不可。
  - イメージ名 — 既存のレシピで選択した基本イメージの組み合わせに基づいて事前に選択されています。ただし、イメージを選択 オプションを変更すると、事前に選択した イメージ名 は失われます。
  - 自動バージョンアップオプション - 基本レシピと一致しません。このイメージオプションのデフォルトは選択した OS バージョンを使用です。

    > **Important**
    >
    > セマンティックバージョニングを使用してパイプラインのビルドを開始する場合は、この値を利用可能な最新の OS バージョンを使用するに変更してください。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。

  AWS Marketplace image

  - サブスクリプション – このタブは開いていて、 からサブスクライブされたイメージは、ベースレシピに合わせて事前に選択 AWS Marketplace されている必要があります。レシピがベースイメージとして使用するイメージを変更すると、選択した元のイメージに依存するその他の詳細が失われる可能性があります。

AWS Marketplace 製品の詳細については、「 AWS Marketplace 購入者ガイド」の「製品の購入」を参照してください。

Custom AMI

- AMI ID — 必須。ただし、この設定には元のエントリがあらかじめ入力されていません。ベースイメージの AMI ID を入力する必要があります。

- インスタンスの設定 — 設定は事前に選択されていますが、編集できます。
  - システムマネージャーエージェント — このチェックボックスをオンまたはオフにして、新しいイメージへのシステムマネージャーエージェントのインストールを制御できます。システムマネージャーエージェントを新しいイメージに含めるには、このチェックボックスはデフォルトでオフになっています。システムマネージャーエージェントを最終イメージから削除するには、エージェントが AMI に含まれないようにチェックボックスを選択します。
  - ユーザーデータ – 構築インスタンスを起動するときにこのエリアを使用して、コマンドまたはコマンドスクリプトを提供で実行します。ただし、この値は、Systems Manager が確実にインストールされるようにするために Image Builder が追加されたコマンドを置き換えます。これらのコマンドには、新しいイメージを作成する前に Image Builder が Linux イメージに対して通常実行するクリーンアップスクリプトが含まれます。

    > **Note**
    > - ユーザーデータを入力する際に、システムマネージャーエージェントがベースイメージにあらかじめインストールされていますか、ユーザーデータにインストールを必ず含めてください。
    > - Linux イメージの場合は、`perform_cleanup`ユーザーデータスクリプトで指定された空のファイルを作成するコマンドを含めて、クリーンアップ手順を実行するようにしてください。Image Builder はこのファイルを検出し、新しいイメージを作成する前にクリーンアップスクリプトを実行します。詳細とスクリプトのサンプルはEC2 Image Builder でのセキュリティのベストプラクティスを

- 作業ディレクトリ — 事前に選択されていますが、編集できます。
- コンポーネント — レシピに既に含まれているコンポーネントは、各コンポーネントリスト (ビルドとテスト) の最後にある 選択されたコンポーネント セクションに表示されます。ニーズに合わせられるように、選択したコンポーネントを削除または並べ替えることができます。

CIS 強化コンポーネントは、Image Builder レシピの標準コンポーネント順序ルールに従っていません。CIS 強化コンポーネントは常に最後に実行され、ベンチマークテストが出力イメージに対して確実に実行されます。

> **Note**
>
> ビルドコンポーネントリストとテストコンポーネントリストには、コンポーネント所有者のタイプに基づいて使用可能なコンポーネントが表示されます。レシピのコンポーネントを追加または更新するには、探しているコンポーネントの所有者タイプを選択します。例えば、 でサブスクライブしたベースイメージに関連付けられているコンポーネントを追加する場合は AWS Marketplace、検索バーの横にあるThird party managed所有者タイプリストから を選択します。

選択されたコンポーネントについては、次の設定を指定できます。

- バージョニングオプション — 事前に選択されていますが、変更できます。イメージビルドで常に最新バージョンのコンポーネントが使用されるように、使用可能な最新のコンポーネントバージョンを使用するオプションを選択することをおすすめします。レシピで特定のコンポーネントバージョンを使用する必要がある場合は、コンポーネントバージョンを指定 を選択し、表示される コンポーネントバージョン ボックスにバージョンを入力できます。
- 入力パラメーター — コンポーネントが受け付ける入力パラメーターを表示します。値 には、以前のバージョンのレシピの値があらかじめ入力されています。このレシピでこのコンポーネントを初めて使用し、入力パラメータにデフォルト値が定義されている場合、デフォルト値はグレーアウトされたテキストを持つ値ボックスに表示されます。他の値が入力されていない場合、Image Builder はデフォルト値を使用します。

  入力パラメータは必要ですが、コンポーネントでデフォルト値が定義されていない場合は、値を指定する必要があります。Image Builder は、必須パラメータが欠落していてデフォルト値が定義されていない場合、レシピバージョンを作成しません。

  > **Important**
  >
  > コンポーネントパラメータはプレーンテキストの値で、 AWS CloudTrailに記録されます。シークレットを保存するには、 AWS Secrets Manager または AWS Systems Manager Parameter Store を使用することをお勧めします。Secrets Managerの詳細については、AWS Secrets Manager ユーザーガイドのSecrets Managerとはを参照してくだ

さい。 AWS Systems Manager パラメータストアについては、AWS Systems Manager ユーザーガイドのAWS Systems Manager パラメータストアを参照。

バージョニング管理オプション または [入力パラメーター] の設定を拡張するには、設定名の横にある矢印を選択します。選択したすべてのコンポーネントの設定をすべて展開するには、[すべて展開] スイッチのオンとオフを切り替えます。

- ストレージ (ボリューム) — あらかじめ入力されています。ルートボリュームの デバイス名、スナップショット、及び IOPS の選択は編集できません。ただし、サイズ など、残りの設定はすべて変更できます。新しいボリュームを追加したり、新規または既存のボリュームを暗号化したりすることもできます。

  Image Builder がソースリージョン (ビルドが実行される地域) のアカウントで作成するイメージのボリュームを暗号化するには、イメージレシピでストレージボリューム暗号化を使用する必要があります。ビルドの配布フェーズで実行される暗号化は、他のアカウントまたはリージョンに配布されるイメージのみに適用されます。

  > **Note**
  >
  > ボリュームに暗号化を使用する場合は、キーがルートボリュームに使用されているものと同じであっても、ボリュームごとにキーを個別に選択する必要があります。

新しいイメージレシピバージョンを作成するには：

1. recipe の詳細ページの上部で、新しいバージョンを作成 を選択します。これにより、イメージレシピの作成 ページが表示されます。
2. 新しいバージョンを作成するには、変更を加え、イメージレシピの作成 を選択します。

イメージパイプラインを作成するときにイメージレシピを作成する方法の詳細については、本ガイドの はじめに セクションの「ステップ 2: レシピを選択する」を参照してください。

でイメージレシピを作成する AWS CLI

で Image Builder `create-image-recipe` コマンドを使用してイメージレシピを作成するには AWS CLI、次の手順に従います。

前提条件

このセクションの Image Builder コマンドを実行して AWS CLIからイメージレシピを作成する前に、レシピが使用するコンポーネントを作成する必要があります。次のステップのイメージレシピの例は、このガイドの を使用してコンポーネントを作成する AWS CLI セクションで作成したサンプルコンポーネントを参照しています。

コンポーネントを作成したら、または既存のコンポーネントを使用している場合は、recipe に含める ARN をメモしてください。

1. CLI 入力 JSON ファイルの作成

   create-image-recipe コマンドのすべての入力をインラインコマンドパラメータで指定できます。ただし、生成されるコマンドはかなり長くなる可能性があります。コマンドを効率化するために、代わりにすべてのレシピ設定を含む JSON ファイルを提供できます。

   > **Note**
   >
   > JSON ファイル内のデータ値の命名規則は、Image Builder API アクションリクエストパラメータに指定されているパターンに従います。API コマンドリクエストパラメータを確認するには、EC2 Image Builder API リファレンス」のCreateImageRecipe「コマンド」を参照してください。
   > データ値をコマンドラインパラメータとして指定するには、AWS CLI コマンドリファレンスで指定されているパラメータ名を参照してください。

   以下に、これらの例で指定するパラメータの概要を示します。

   - 名前 (文字列、必須) — イメージレシピの名前。
   - 説明 (文字列) — イメージレシピの説明。
   - parentImage (文字列、必須) — イメージレシピがカスタマイズしたイメージのベースとして使用するイメージ。値を入れられるのは、ベースイメージ ARN または AMI ID です。

     > **Note**
     >
     > Linux の例ではImage Builder AMI を使用し、Windows の例では ARN を使用しています。

   - semanticVersion (文字列、必須) - イメージレシピのセマンティックバージョンは以下のフォーマットで表されます： <major>.<minor>.<patch>。例えば、値は `1.0.0` であるかもし

れません。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。

- コンポーネント (配列、必須) — ComponentConfiguration オブジェクトの配列が含まれます。少なくとも 1 つのビルドコンポーネントを指定する必要があります。

> **Note**
>
> Image Builder は、レシピで指定した順序でコンポーネントをインストールします。しかし、CISのハードニング・コンポーネントは、ベンチマーク・テストが出力イメージに対して実行されるように、常に最後に実行

- コンポーネント ARN (文字列、必須) — コンポーネント ARN。

> **Tip**
>
> 例の 1 つを使用して独自のイメージレシピを作成するには、サンプル ARN をレシピに使用しているコンポーネントの ARN に置き換える必要があります。

- パラメータ (オブジェクトの配列) — ComponentParameter オブジェクトの配列が含まれます。入力パラメータは必要ですが、コンポーネントでデフォルト値が定義されていない場合は、値を指定する必要があります。Image Builder は、必須パラメータが欠落していてデフォルト値が定義されていない場合、レシピバージョンを作成しません。

> **Important**
>
> コンポーネントパラメータはプレーンテキストの値で、 AWS CloudTrailに記録されます。シークレットを保存するには、 AWS Secrets Manager または AWS Systems Manager Parameter Store を使用することをお勧めします。Secrets Managerの詳細については、AWS Secrets Manager ユーザーガイドのSecrets Managerとはを参照してください。 AWS Systems Manager パラメータストアについては、AWS Systems Manager ユーザーガイドのAWS Systems Manager パラメータストアを参照。

  - 名前 (文字列、必須) — 設定するコンポーネントパラメータの名前。

- 値 (文字列の配列、必須) — 指定されたコンポーネントパラメータの値を設定する文字列の配列を含みます。コンポーネントにデフォルト値が定義されていて、他の値が指定されていない場合、 はデフォルト値 AWSTOE を使用します。

- additionalInstanceConfiguration (オブジェクト) — ビルドインスタンスの追加設定と起動スクリプトを指定します。
  - systemsManagerAgent (オブジェクト) — ビルドインスタンスの Systems Manager エージェントの設定が含まれます。
    - uninstallAfterBuild (ブール値) — 新しい AMI を作成する前に、Systems Manager エージェントを最終的なビルドイメージから削除するかどうかを制御します。これが `true` に設定されている場合、エージェントは最終イメージから削除されます。オプションが`false`に設定されている場合、エージェントは新しいAMIに含まれるように残され デフォルト値は、`false`です。

      > **Note**
      >
      > `uninstallAfterBuild` 属性が JSON ファイルに含まれておらず、次の条件に当てはまる場合、Image Builder は最終イメージから Systems Manager エージェントを削除し、AMI で使用できないようにします。
      >
      > - `userDataOverride` は空であるか、JSON ファイルから省略されています。
      > - Image Builder は、ベースイメージにエージェントがプリインストールされていないオペレーティングシステムのビルドインスタンスに Systems Manager エージェントを自動的にインストールしました。

  - userDataOverride (文字列) — ビルドインスタンスの起動時に実行するコマンドまたはコマンドスクリプトを指定します。

    > **Note**
    >
    > ユーザーデータは常に Base 64 でエンコードされます。例えば、以下のコマンドは `IyEvYmluL2Jhc2gKbWtkaXIgLXAgL3Zhci9iYi8KdG91Y2ggL3Zhcg==` としてエンコードされます。
    >
    > ```
    > #!/bin/bash
    > mkdir -p /var/bb/
    > touch /var
    > ```

Linux の例では、このエンコードされた値を使用しています。

Linux

次の例のベースイメージ (parentImage プロパティ) は AMI です。AMI を使用するときは、その AMI にアクセスできる必要があり、AMI はソースリージョン (Image Builder がコマンドを実行するのと同じリージョン) にある必要があります。ファイルをcreate-image-recipe.jsonの名前で保存し、create-image-recipeコマンドで使用します。

```
{
"name": "BB Ubuntu Image recipe",
"description": "Hello World image recipe for Linux.",
"parentImage": "ami-0a01b234c5de6fabc",
"semanticVersion": "1.0.0",
"components": [
 {
  "componentArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/bb$"
 }
],
"additionalInstanceConfiguration": {
 "systemsManagerAgent": {
   "uninstallAfterBuild": true
 },
 "userDataOverride": "IyEvYmluL2Jhc2gKbWtkaXIgLXAgL3Zhci9iYi8KdG91Y2ggL3Zhcg=="
}
}
```

Windows

次の例は、Windows Server 2016 英語版フルベースイメージの最新バージョンを参照しています。この例の ARN は、指定したセマンティックバージョンフィルターに基づいて SKU 内の最新のイメージを参照します: arn:aws:imagebuilder:us-west-2:aws:image/windows-server-2016-english-full-base-x86/x.x.x。

```
{
"name": "MyBasicRecipe",
"description": "This example image recipe creates a Windows 2016 image.",
"parentImage": "arn:aws:imagebuilder:us-west-2:aws:image/windows-server-2016-
english-full-base-x86/x.x.x",
```

```
"semanticVersion": "1.0.0",
"components": [
 {
  "componentArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/my-example-component/2019.12.02/1"
 },
 {
  "componentArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/my-imported-component/1.0.0/1"
 }
]
}
```

> **Note**
>
> Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。

2. レシピを作成する

   レシピを作成するには以下のコマンドを使用します。前のステップで作成した JSON ファイルの名前を `--cli-input-json` パラメーターに入力します。

```
aws imagebuilder create-image-recipe --cli-input-json file://create-image-recipe.json
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

VM をベースイメージとしてコンソールにインポートします。

このセクションでは、仮想マシン (VM) をイメージレシピのベースイメージとしてインポートする方法に焦点を当てます。レシピやレシピバージョンの作成に関連する他の手順については、ここでは説

明しません。Image Builder コンソールのパイプライン作成ウィザードを使用して新しいイメージレシピを作成するその他の手順については、「イメージパイプラインを作成します。」を参照してください。新しいイメージレシピまたはレシピバージョンを作成するその他の手順については、「イメージレシピの新しいバージョンを作成します。」を参照してください。

Image Builder コンソールでイメージレシピのベースイメージとして VM をインポートするには、以下の手順とその他の必要な手順に従って、レシピまたはレシピバージョンを作成します。

1. ベースイメージの イメージの選択 セクションで、ベースイメージをインポート オプションを選択します。
2. 通常どおり、イメージオペレーティングシステム (OS) とOS バージョン を選択します。

VM インポート設定

VM を仮想化環境からエクスポートすると、そのプロセスによって VM 環境、設定、データのスナップショットとして機能する 1 つ以上のディスクコンテナファイルのセットが作成されます。これらのファイルを使用して、VM をイメージレシピのベースイメージとしてインポートできます。Image Builder での VM のインポートに関する詳細については、「VM イメージのインポートとエクスポート」を参照してください。

インポートソースの場所を指定するには、次の手順に従います。

インポートソース

ディスクコンテナ 1 セクションで、インポートする最初の VM イメージディスクコンテナまたはスナップショットのソースを指定します。

1. ソース - これはS3バケットまたはEBSスナップショットのいずれかになります。
2. ディスクのS3の場所を選択 - ディスクイメージが保存されているAmazon S3の場所を入力します。場所を参照するには、[S3 を参照] を選択します。
3. ディスクコンテナを追加するには、[ディスクコンテナを追加] を選択します。

IAM ロール

IAM ロールを VM インポート設定に関連付けるには、[IAM ロール] ドロップダウンリストからロールを選択するか、[新規ロールを作成] を選択して新しいロールを作成します。新しいロールを作成すると、[IAM ロール] コンソールページが別のタブで開きます。

詳細設定 - オプション

次のオプション設定はオプショナルです。これらの設定により、インポートによって作成されるベースイメージの暗号化、ライセンス、タグなどを設定できます。

全般

1. ベースイメージに固有の 名前 を指定します。値を入れない場合、ベースイメージで自動的に recipe 名が継承されます。
2. ベースイメージの バージョン を指定します。形式は以下のようになります: *<major>.<minor>.<patch>* 値を入力しない場合、ベースイメージはレシピバージョンを継承します。
3. ベースイメージの 説明 を入力することもできます。

ベースイメージアーキテクチャ

VM インポートソースのアーキテクチャを指定するには、アーキテクチャ リストから値を選択します。

暗号化

VM ディスクイメージが暗号化されている場合は、インポートプロセスに使用するキーを指定する必要があります。インポート AWS KMS key に を指定するには、暗号化 (KMS キー) リストから値を選択します。このリストには、現在のリージョンでアカウントがアクセスできる KMS キーが含まれています。

ライセンス管理

VM をインポートすると、インポートプロセスによって VM OS が自動的に検出され、適切なライセンスがベースイメージに適用されます。お使いの OS プラットフォームに応じて、ライセンスの種類は次のとおりです。

- License included - あなたのプラットフォームに適した AWS ライセンスがベースイメージに適用されます。
- Bring-Your-Own-License (BYOL) - VMのライセンスを保持します (該当する場合) 。

で作成されたライセンス設定をベースイメージ AWS License Manager にアタッチするには、ライセンス設定名リストから を選択します。License Manager の詳細については、「 の使用 AWS License Manager」を参照してください。

> **Note**
>
> - ライセンスコンフィギュレーションには、企業契約の条件に基づくライセンスルールが含まれています。
> - Linux は BYOL ライセンスのみをサポートします。

タグ (ベースイメージ)

タグはキーと値のペアを使用して、検索可能なテキストを Image Builder リソースに割り当てます。インポートしたベースイメージのタグを指定するには、キー ボックスと 値 ボックスにキーと値のペアを入力します。

タグを追加するには、タグの追加 を選択します。タグを削除するには、[タグの削除] を選択します。

新しいイメージレシピのバージョンを作成

このセクションでは、コンテナ recipe の新しいバージョンを作成する方法を示します。

内容

- コンソールで新しいコンテナ recipe の作成
- コンテナ recipe を作成します AWS CLI。

コンソールで新しいコンテナ recipe の作成

コンテナレシピの新しいバージョンを作成することは、新しいレシピを作成することと実質的に同じです。違いは、ほとんどの場合、基本レシピに合わせて特定の詳細が事前に選択されていることです。以下のリストは、新しいレシピを作成することと、既存のレシピの新しいバージョンを作成することの違いを説明しています。

レシピ詳細

- 名前 - 編集不可。
- バージョン – 必須 この基本情報には、現在のバージョンやシーケンスがあらかじめ入力されていません。作成したいバージョン番号をmajor.minor.patchの形式で入力します。そのバージョンが既に存在している場合は、エラーが発生します。

ベースイメージ

- イメージオプションを選択 — あらかじめ選択されていますが、編集可能です。ベースイメージのソースの選択を変更すると、選択した元のオプションに依存するその他の詳細が失われる可能性があります。

  基本イメージの選択に関連する詳細を表示するには、選択内容と一致するタブを選択してください。

  Managed images

  - イメージオペレーティングシステム（OS）- 編集不可。
  - イメージ名 — 既存のレシピで選択した基本イメージの組み合わせに基づいて事前に選択されています。ただし、イメージを選択 オプションを変更すると、事前に選択した イメージ名 は失われます。
  - 自動バージョンアップオプション - 基本レシピと一致しません。このイメージオプションのデフォルトは選択した OS バージョンを使用です。

    > **Important**
    >
    > セマンティックバージョニングを使用してパイプラインのビルドを開始する場合は、この値を利用可能な最新の OS バージョンを使用するに変更してください。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。

  ECR image

  - イメージオペレーティングシステム (OS) — 事前に選択されていますが、編集可能です。
  - OS バージョン — 事前に選択されていますが、編集可能です。
  - ECR イメージ ID — 事前入力されていますが、編集可能です。

  Docker Hub image

  - イメージオペレーティングシステム（OS）- 編集不可。
  - OS バージョン — 事前に選択されていますが、編集可能です。
  - Docker イメージ ID — 事前に入力されていますが、編集可能です。

インスタンス設定

- AMI ID — 事前入力されていますが、編集可能です。

- ストレージボリューム

  EBS ボリューム 1 (AMI ルート) — 事前に入力されています。ルートボリュームのデバイス名、スナップショット、または IOPS の選択内容は編集できません。ただし、サイズ など、残りの設定はすべて変更できます。新しいボリュームを追加することもできます。

  > **Note**
  >
  > 別のアカウントから共有した場合、ベース AMI を指定した場合、指定したすべての 2 次ボリュームのスナップショットも自分のアカウントと共有する必要があります。

作業ディレクトリパス

- 作業ディレクトリパス — 事前に入力されていますが、編集可能です。

コンポーネント

- コンポーネント — レシピに既に含まれているコンポーネントは、各コンポーネントリスト (ビルドとテスト) の最後にある 選択されたコンポーネント セクションに表示されます。ニーズに合わせられるように、選択したコンポーネントを削除または並べ替えることができます。

  CIS 強化コンポーネントは、Image Builder レシピの標準コンポーネント順序ルールに従っていません。CIS 強化コンポーネントは常に最後に実行され、ベンチマークテストが出力イメージに対して確実に実行されます。

  > **Note**
  >
  > ビルドコンポーネントリストとテストコンポーネントリストには、コンポーネント所有者のタイプに基づいて使用可能なコンポーネントが表示されます。レシピのコンポーネントを追加または更新するには、探しているコンポーネントの所有者タイプを選択します。例えば、 でサブスクライブしたベースイメージに関連付けられているコンポーネントを追加する場合は AWS Marketplace、検索バーの横にある`Third party managed`所有者タイプリストから を選択します。

  選択されたコンポーネントについては、次の設定を指定できます。

- バージョニングオプション — 事前に選択されていますが、変更できます。イメージビルドで常に最新バージョンのコンポーネントが使用されるように、使用可能な最新のコンポーネントバージョンを使用するオプションを選択することをおすすめします。レシピで特定のコンポーネントバージョンを使用する必要がある場合は、コンポーネントバージョンを指定 を選択し、表示される コンポーネントバージョン ボックスにバージョンを入力できます。
- 入力パラメーター — コンポーネントが受け付ける入力パラメーターを表示します。値 には、以前のバージョンのレシピの値があらかじめ入力されています。このレシピでこのコンポーネントを初めて使用していて、入力パラメータにデフォルト値が定義されている場合、デフォルト値はグレーアウトされたテキストを持つ値ボックスに表示されます。他の値が入力されていない場合、Image Builder はデフォルト値を使用します。

  入力パラメータは必要ですが、コンポーネントでデフォルト値が定義されていない場合は、値を指定する必要があります。Image Builder は、必須パラメータが欠落していてデフォルト値が定義されていない場合、レシピバージョンを作成しません。

  > **Important**
  >
  > コンポーネントパラメータはプレーンテキストの値で、 AWS CloudTrailに記録されます。シークレットを保存するには、 AWS Secrets Manager または AWS Systems Manager Parameter Store を使用することをお勧めします。Secrets Managerの詳細については、AWS Secrets Manager ユーザーガイドのSecrets Managerとはを参照してください。 AWS Systems Manager パラメータストアについては、AWS Systems Manager ユーザーガイドのAWS Systems Manager パラメータストアを参照。

バージョニング管理オプション または [入力パラメーター] の設定を拡張するには、設定名の横にある矢印を選択します。選択したすべてのコンポーネントの設定をすべて展開するには、[すべて展開] スイッチのオンとオフを切り替えます。

Dockerfile テンプレート

- Dockerfile テンプレート – 事前に入力されていますが、編集可能です。Image Builder が実行時にビルド情報に置き換える以下のコンテキスト変数のいずれかを指定できます。

  parentImage (必須)

  ビルド時に、この変数は recipe のベースイメージに解決されます。

例：

```
FROM
{{{ imagebuilder:parentImage }}}
```

環境 (コンポーネントが指定されている場合は必須)

この変数は、コンポーネントを実行するスクリプトに解決されます。

例：

```
{{{ imagebuilder:environments }}}
```

コンポーネント (オプション)

Image Builder は、コンテナレシピに含まれるコンポーネントのビルドコンポーネントスクリプトとテストコンポーネントスクリプトを解決します。この変数は、Dockerfile の環境変数の後に配置できます。

例：

```
{{{ imagebuilder:components }}}
```

ターゲットリポジトリ

- ターゲットリポジトリ名 — パイプラインが実行されるリージョン (リージョン 1) のパイプラインのディストリビューション設定で他にリポジトリが指定されていない場合に、出力イメージが保存される Amazon ECR リポジトリ。

新しいイメージレシピのバージョンを作成

1. recipe の詳細ページの上部で、新しいバージョンを作成 を選択します。コンテナレシピのレシピの作成ページが表示されます。
2. 新しいバージョンを作成するには、変更を加え、イメージレシピの作成 を選択します。

イメージパイプラインを作成するときにコンテナレシピを作成する方法の詳細については、本ガイドのはじめにセクションの「ステップ 2: レシピを選択する」を参照してください。

コンテナ recipe を作成します AWS CLI。

の `imagebuilder create-container-recipe` コマンドを使用して Image Builder コンテナレシピを作成するには AWS CLI、次の手順に従います。

前提条件

このセクションの Image Builder コマンドを実行して でコンテナレシピを作成する前に AWS CLI、レシピが使用するコンポーネントを作成する必要があります。次のステップのコンテナレシピの例は、このガイドの を使用してコンポーネントを作成する AWS CLI セクションで作成したサンプルコンポーネントを参照しています。

コンポーネントを作成したら、または既存のコンポーネントを使用している場合は、recipe に含める ARN をメモしてください。

1. CLI 入力 JSON ファイルの作成

   create-container-recipe コマンドのすべての入力をインラインコマンドパラメータで指定できます。ただし、生成されるコマンドはかなり長くなる可能性があります。コマンドを効率化するために、代わりにすべてのレシピ設定を含む JSON ファイルを提供できます。

   > **Note**
   >
   > JSON ファイル内のデータ値の命名規則は、Image Builder API アクションリクエストパラメータに指定されているパターンに従います。API コマンドリクエストパラメータを確認するには、EC2 Image Builder API リファレンス」のCreateContainerRecipe「 コマンド」を参照してください。
   > データ値をコマンドラインパラメータとして指定するには、AWS CLI コマンドリファレンスで指定されているパラメータ名を参照してください。

   以下に、これらの例で指定するパラメータの概要を示します。

   - コンポーネント (配列、必須) — `ComponentConfiguration` オブジェクトの配列が含まれます。少なくとも 1 つのビルドコンポーネントを指定する必要があります。

> **Note**
>
> Image Builder は、レシピで指定した順序でコンポーネントをインストールします。しかし、CISのハードニング・コンポーネントは、ベンチマーク・テストが出力イメージに対して実行されるように、常に最後に実行

- コンポーネント ARN (文字列、必須) — コンポーネント ARN。

  > **Tip**
  >
  > 例の 1 つを使用して独自のイメージレシピを作成するには、サンプル ARN をレシピに使用しているコンポーネントの ARN に置き換える必要があります。これらには AWS リージョン、それぞれの、名前、バージョン番号が含まれます。

- パラメータ (オブジェクトの配列) — `ComponentParameter` オブジェクトの配列が含まれます。入力パラメータは必要ですが、コンポーネントでデフォルト値が定義されていない場合は、値を指定する必要があります。Image Builder は、必須パラメータが欠落していてデフォルト値が定義されていない場合、レシピバージョンを作成しません。

  > **Important**
  >
  > コンポーネントパラメータはプレーンテキストの値で、 AWS CloudTrailに記録されます。シークレットを保存するには、 AWS Secrets Manager または AWS Systems Manager Parameter Store を使用することをお勧めします。Secrets Managerの詳細については、AWS Secrets Manager ユーザーガイドのSecrets Managerとはを参照してください。 AWS Systems Manager パラメータストアについては、AWS Systems Manager ユーザーガイドのAWS Systems Manager パラメータストアを参照。

  - 名前 (文字列、必須) — 設定するコンポーネントパラメータの名前。
  - 値 (文字列の配列、必須) — 指定されたコンポーネントパラメータの値を設定する文字列の配列を含みます。コンポーネントにデフォルト値が定義されていて、他の値が指定されていない場合、 はデフォルト値 AWSTOE を使用します。

- containerType (文字列、必須) — 作成するコンテナのタイプ。有効な値には次のものが含まれます。DOCKER。
- dockerfileTemplateData (文字列) – イメージの構築に使用される Dockerfile テンプレート。インラインデータ BLOB として表されます。
- 名前 (文字列、必須) — イメージレシピの名前。
- 説明 (文字列) — イメージレシピの説明。
- 親イメージ (文字列、必須) — コンテナレシピがカスタマイズ画像のベースとして使用する 値を入れられるのは、ベースイメージ ARN または AMI ID です。
- platformOverride (文字列) –カスタムベースイメージを使用する場合のオペレーティングシステムプラットフォームを指定します。
- semanticVersion 文字列、必須) - コンテナレシピのセマンティックバージョンを以下のフォーマットで指定します：<major>.<minor>.<patch>。文字列の例は 1.0.0 などです。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。
- タグ (文字列マップ)コンテナ recipe にアタッチされているタグ。
- instanceConfiguration (オブジェクト) —コンテナイメージの構築とテストを目的としてインスタンスを設定するために使用できるオプションのグループ。
  - image (文字列) – コンテナの構築およびテストインスタンスのベースイメージとして使用する AMI ID。この値を指定しない場合、Image Builder は適切な Amazon ECS 最適化 AMI をベースイメージとして使用します。
  - blockDeviceMappings (オブジェクトの配列) — imageパラメータで指定された Image Builder AMI からインスタンスを構築するためにアタッチするブロックデバイスを定義します。
    - DeviceName (文字列) — これらのマッピングが適用されるデバイス。
    - ebs (オブジェクト) – このマッピングの Amazon EBS 固有の構成を管理するために使用します。
      - deleteOnTermination (ブール値) — 関連付けられたデバイスの終了時に削除を設定するために使用されます。
      - 暗号化済み (ブール値) — デバイス暗号化の設定に使用されます。
      - volumeSize (整数) — デバイスのボリュームサイズを上書きするために使用します。
      - volumeType (文字列) — デバイスのボリュームサイズを上書きするために使用します。

- targetRepository (オブジェクト、必須) — パイプラインが稼働するリージョン (リージョン 1) のパイプラインのディストリビューション設定で他にリポジトリが指定されていない場合のコンテナイメージのデスティネーションリポジトリ。
  - repositoryName (文字列、必須) - 出力コンテナイメージを保存するコンテナリポジトリの名前。この名前の先頭には、リポジトリの場所が付きます。
  - サービス (文字列、必須) - このイメージが登録されたサービスを指定します。
- workingDirectory (文字列) - ビルドとテストのワークフローで使用する作業ディレクトリ。

```
{
 "components": [
   {
   "componentArn": "arn:aws:imagebuilder:us-east-1:123456789012:component/
helloworldal2/x.x.x"
   }
 ],
 "containerType": "DOCKER",
 "description": "My Linux Docker container image",
 "dockerfileTemplateData": "FROM
 {{{ imagebuilder:parentImage }}}\n{{{ imagebuilder:environments }}}\n{{{ imagebuilder:comp
 "name": "amazonlinux-container-recipe",
 "parentImage": "amazonlinux:latest",
 "platformOverride": "Linux",
 "semanticVersion": "1.0.2",
 "tags": {
   "sometag" : "Tag detail"
 },
 "instanceConfiguration": {
   "image": "ami-1234567890",
   "blockDeviceMappings": [
   {
   "deviceName": "/dev/xvda",
   "ebs": {
    "deleteOnTermination": true,
    "encrypted": false,
    "volumeSize": 8,
    "volumeType": "gp2"
    }
    }
   ]
 },
```

```
  "targetRepository": {
    "repositoryName": "myrepo",
    "service": "ECR"
  },
  "workingDirectory": "/tmp"
}
```

2. レシピを作成する

   レシピを作成するには以下のコマンドを使用します。前のステップで作成した JSON ファイルの名前を `--cli-input-json` パラメーターに入力します。

```
aws imagebuilder create-container-recipe --cli-input-json file://create-container-recipe.json
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

リソースをクリーンアップする

予期しない料金が発生しないように、このガイドの例で作成したリソースとパイプラインは必ずクリーンアップしてください。Image Builder でのリソースの削除については、「EC2 Image Builder resourcesの削除」を参照してください。

EC2 Image Builderイメージの管理

Image Builder で AMI またはコンテナイメージ用のイメージリソースを作成した後は、Image Builder コンソール、Image Builder API、または AWS CLIの imagebuilder コマンドを使用して管理できます。

> **Tip**
>
> 同じ型のリソースが複数にある場合に、割り当てたタグに基づいて特定のリソースをすばやく識別できます。の Image Builder コマンドを使用してリソースにタグを付ける方法の詳細については AWS CLI、このガイドのリソースのタグ付け「」セクションを参照してください。

このセクションでは、レシピを一覧表示、表示、作成する方法について説明します。イメージワークフローとその管理方法については、「EC2 Image Builder イメージのビルドワークフローとテストワークフローを管理する」を参照してください。

内容

イメージとビルドバージョンを一覧表示する

Image Builder コンソールの [イメージ] ページには、所有しているもの、共有されているもの、アクセス可能なすべての Image Builder イメージリソースのリストが表示されます。リストの結果には、それらのリソースに関する重要な詳細がいくつか含まれています。

ワークフローアクションが保留になっているアカウント内のイメージもすべて表示できます。

内容

イメージを一覧表示する

このセクションでは、イメージに関する情報を一覧表示するさまざまな方法について説明します。

以下の方法のいずれかを使用して、アクセスできる Image Builder イメージリソースを一覧表示できます。API アクションについては、EC2 Image Builder API リファレンスListImagesの「」を参照してください。関連する SDK リクエストについては、同じページの「関連項目」リンクを参照してください。

内容

- コンソールに画像を一覧表示します。
- AWS CLI コマンドを使用してイメージを一覧表示する

コンソールに画像を一覧表示します。

コンソールで [イメージ] リストページを開くには、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから [イメージ] を選択します。

コンソールの [イメージ] ページは、イメージの所有権または保留中のワークフローアクションに基づいて、複数のタブに分かれています。このセクションでは、自分が所有している、またはアクセスできるイメージを表示する最初の 3 つのタブについて説明します。

コンソールタブ: 自分で所有

[自分で所有] タブでは、以下のフィルターを使用してイメージリストの結果を効率化できます。

- 検索バーで名前の全部または一部を検索できます。
- オペレーティングシステムプラットフォーム (Windows または Linux) に基づいてイメージをフィルタリングできます。
- 生成される出力のタイプ (AMI またはコンテナイメージ) に基づいてイメージをフィルタリングできます。
- [フィルターソース] を使用すると、VMIE を使用して仮想マシンからインポートされたイメージを検索できます。

フィルターコントロールに従って、[自分で所有] タブには、作成した Image Builder イメージのリストと、リストされたリソースに関する以下の詳細が表示されます。

[名前 / バージョン]

Image Builder のイメージリソース名は、ビルド元のレシピ名とバージョンで始まります。リンクを選択すると、関連するすべてのイメージビルドバージョンが表示されます。

タイプ

このイメージリソース (AMI またはコンテナイメージ) に対して Image Builder が作成する出力イメージのタイプ。

プラットフォーム

「Windows」や「Linux」など、イメージリソースバージョンのオペレーティングシステムプラットフォーム。

[イメージソース]

Image Builder がこのイメージリソースの構築に使用したベースイメージのオリジン。これは主に、仮想マシン ([VMIE]) からインポートされたイメージの結果をフィルタリングするために使用されます。

作成時刻

Image Builder が現在のバージョンのイメージリソースを作成した日付と時刻。

ARN

イメージリソースの現在のバージョンの Amazon リソースネーム (ARN)。

コンソールタブ: 自分と共有

[自分と共有] タブでは、以下のフィルターを使用してイメージリストの結果を効率化できます。

- 検索バーで名前の全部または一部を検索できます。
- オペレーティングシステムプラットフォーム (Windows または Linux) に基づいてイメージをフィルタリングできます。
- 生成される出力のタイプ (AMI またはコンテナイメージ) に基づいてイメージをフィルタリングできます。
- [フィルターソース] を使用すると、VMIE を使用して仮想マシンからインポートされたイメージを検索できます。

フィルターコントロールに従って、[自分と共有] タブには、共有された Image Builder イメージのリストと、リストされたリソースに関する以下の詳細が表示されます。

[イメージ名]

共有されたイメージリソースの名前。レシピで共有画像を使用するには、[管理対象イメージを選択する] オプションを選択し、[イメージのオリジン] を [ユーザーと共有されたイメージ] に変更します。

タイプ

このイメージリソース (AMI またはコンテナイメージ) に対して Image Builder が作成する出力イメージのタイプ。

バージョン

「Windows」や「Linux」など、イメージリソースバージョンのオペレーティングシステムプラットフォーム。

[イメージソース]

Image Builder がこのイメージリソースの構築に使用したベースイメージのオリジン (該当する場合)。これは主に、仮想マシン ([VMIE]) からインポートされたイメージの結果をフィルタリングするために使用されます。

プラットフォーム

「Windows」や「Linux」など、イメージリソースバージョンのオペレーティングシステムプラットフォーム。

作成時刻

Image Builder がユーザーと共有していたバージョンのイメージリソースを作成した日付と時刻。

[所有者]

共有されたイメージリソースの所有者。

ARN

共有されたイメージリソースバージョンの Amazon リソースネーム (ARN)。

コンソールタブ: Amazon が管理

[Amazon が管理] タブでは、以下のフィルターを使用してイメージリストの結果を効率化できます。

- 検索バーで名前の全部または一部を検索できます。

- オペレーティングシステムプラットフォーム (Windows または Linux) に基づいてイメージをフィルタリングできます。
- 生成される出力のタイプ (AMI またはコンテナイメージ) に基づいてイメージをフィルタリングできます。
- [フィルターソース] を使用すると、VMIE を使用して仮想マシンからインポートされたイメージを検索できます。

フィルターコントロールに従い、[Amazon が管理] タブには、レシピのベースイメージとして使用できる Amazon が管理する Image Builder イメージのリストが表示されます。Image Builder には、リストされたリソースに関する以下の詳細が表示されます。

[イメージ名]

管理イメージの名前。レシピを作成すると、ベースイメージのデフォルトは [クイックスタート (Amazon 管理)] です。このタブに一覧表示されるイメージは、レシピの作成時にベースイメージ用に選択したオペレーティングシステムプラットフォームに関連付けられた [イメージ名] リストに入力します。

タイプ

このイメージリソース (AMI またはコンテナイメージ) に対して Image Builder が作成する出力イメージのタイプ。

バージョン

「Windows」や「Linux」など、イメージリソースバージョンのオペレーティングシステムプラットフォーム。

プラットフォーム

「Windows」や「Linux」など、イメージリソースバージョンのオペレーティングシステムプラットフォーム。

作成時刻

Image Builder がユーザーと共有していたバージョンのイメージリソースを作成した日付と時刻。

[所有者]

管理イメージは Amazon が所有しています。

ARN

共有されたイメージリソースバージョンの Amazon リソースネーム (ARN)。

AWS CLI コマンドを使用してイメージを一覧表示する

で list-images コマンドを実行すると AWS CLI、所有しているイメージまたはアクセスできるイメージのリストを取得できます。

次のコマンド例は、フィルターなしで list-images コマンドを使用し、所有しているすべての Image Builder イメージリソースを一覧表示する方法を示しています。

例: すべてのイメージを一覧表示する

```
aws imagebuilder list-images
```

出力:

```
{
 "requestId": "1abcd234-e567-8fa9-0123-4567b890cd12",
 "imageVersionList": [
  {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-name/1.0.0",
    "name": "image-recipe-name",
    "type": "AMI",
    "version": "1.0.0",
    "platform": "Linux",
    "owner": "123456789012",
    "dateCreated": "2022-04-28T01:38:23.286Z"
  },
  {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-win/1.0.1",
    "name": "image-recipe-win",
    "type": "AMI",
    "version": "1.0.1",
    "platform": "Windows",
    "owner": "123456789012",
    "dateCreated": "2022-04-28T01:38:23.286Z"
  }
 ]
}
```

list-images コマンドを実行すると、次の例のようにフィルターを適用して結果を効率化できます。結果をフィルタリングする方法の詳細については、AWS CLI Command Referenceのlist-imagesコマンドを参照してください。

例: Linux イメージのフィルタ処理

```
aws imagebuilder list-images --filters name="platform",values="Linux"
```

出力:

```
{
 "requestId": "1abcd234-e567-8fa9-0123-4567b890cd12",
 "imageVersionList": [
  {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-name/1.0.0",
    "name": "image-recipe-name",
    "type": "AMI",
    "version": "1.0.0",
    "platform": "Linux",
    "owner": "123456789012",
    "dateCreated": "2022-04-28T01:38:23.286Z"
  }
 ]
}
```

アクション待ちのイメージを一覧表示する

イメージワークフローで WaitForAction ステップアクションを使用すると、処理を再開するか、ワークフローを中止するシグナルを送信するまで、ワークフローは一時停止します。このステップアクションは、続行する前に実行する必要のある外部プロセスがある場合に使用できます。その後、SendWorkflowStepAction を使用して、一時停止のステップへのシグナルを RESUME または STOP に送信できます。コンソールからワークフローを停止または再開することもできます。

以下のタブは、再開または停止のシグナルを待つために現在一時停止されているワークフローステップを含む、アカウント内のすべてのイメージリソースのリストを取得する方法を示しています。タブには、コンソールのステップと AWS CLI コマンドが表示されます。

API や SDK を使用して、アクションを待っているワークフローステップのリストを取得することもできます。API アクションについては、EC2 Image Builder API リファレンスListWaitingWorkflowStepsの「」を参照してください。関連する SDK リクエストについては、同じページの「関連項目」リンクを参照してください。

Console

コンソールの [アクション待ち] タブに移動するには、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから [イメージ] を選択します。[イメージ] リストページが開きます。
3. リストページから [アクション待ち] タブを選択します。
4. (オプション) ステップを停止または再開するには、名前の横にあるチェックボックスを選択し、[ステップを停止] または [ステップを再開] を選択します。複数のチェックボックスを選択して、選択したすべてのステップに対して同じアクションを実行できます。

保留中のワークフローステップの詳細

保留中のステップのワークフローの詳細には以下が含まれます。

- [イメージ名] — 保留中のステップがあるイメージリソースの名前。名前のリンクを選択すると、そのイメージの詳細ページを表示できます。
- [保留中のステップ名] — アクションを待っているワークフローステップの名前。
- [ステップの実行 ID] — ワークフローステップのランタイムインスタンスを一意に識別します。リンクされた ID を選択すると、ステップのランタイム詳細を表示できます。
- [ステップの開始] – ワークフローステップのランタイムインスタンスが開始されたときのタイムスタンプ。
- [ワークフロー ARN] — 保留中のステップが適用されているワークフローの Amazon リソースネーム (ARN)。
- [アクション] — 待機状態にあるステップアクション。

AWS CLI

で list-waiting-workflow-steps コマンドを実行すると AWS CLI、イメージ作成プロセスを完了する前にアクションを待っているワークフローステップがあるアカウント内のすべてのイメージのリストが表示されます。

以下のコマンド例は、list-waiting-workflow-steps コマンドを使用して、アクション待ちのワークフローステップを含むアカウント内のすべてのイメージを一覧表示する方法を示しています。

例: 待機中のワークフローステップを含むアカウント内のイメージを一覧表示する

```
aws imagebuilder list-waiting-workflow-steps
```

出力:

この例の出力には、アクション待ちのステップを含むアカウント内の 1 つのイメージが表示されます。

```
{
    "steps": [
        {
            "imageBuildVersionArn": "arn:aws:imagebuilder:us-
west-2:111122223333:image/example-image/1.0.0/8",
            "name": "WaitForAction",
            "workflowExecutionId": "wf-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "stepExecutionId": "step-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "workflowBuildVersionArn": "arn:aws:imagebuilder:us-
west-2:111122223333:workflow/test/wait-for-action/1.0.0/1",
            "startTime": "2023-11-21T23:21:23.609Z",
            "action": "WaitForAction"
        }
    ]
}
```

イメージビルドバージョンを一覧表示する

Image Builder コンソールの [イメージビルドバージョン] ページには、ビルドバージョンのリストと、所有しているイメージリソースの追加情報が表示されます。Image Builder API、 SDKs、またはでコマンドまたはアクションを使用して、イメージビルドバージョン AWS CLI を一覧表示することもできます。

以下の方法のいずれかを使用して、所有しているイメージリソースのイメージビルドバージョンを一覧表示できます。API アクションについては、EC2 Image Builder API リファレンスListImageBuildVersionsの「」を参照してください。関連する SDK リクエストについては、同じページの 「関連項目」リンクを参照してください。

Console

バージョンの詳細

Image BuilderコンソールのImage build versionsページには、以下の詳細が記載されています：

- [バージョン] — イメージリソースのビルドバージョン。Image Builder コンソールでは、バージョンはイメージの詳細ページにリンクしています。

- [タイプ] — このイメージリソース (AMI またはコンテナイメージ) を作成したときに Image Builder が配信した出力のタイプ。
- [作成日] — Image Builder がイメージビルドバージョンを作成した日付と時刻。
- [イメージステータス] — イメージビルドバージョンの現在のステータス。ステータスは、イメージのビルドまたは廃棄に関連する場合があります。たとえば、ビルドプロセス中に、`Building` または `Distributing` のステータスが表示される場合があります。イメージの廃棄については、`Deprecated` または `Deleted` のステータスが表示される場合があります。
- [障害の理由] — イメージステータスの理由。Image Builder コンソールには、ビルドが失敗した理由 (イメージステータス は `Failed` と等しい) のみが表示されます。
- [セキュリティ検出結果] — 参照先のイメージビルドバージョンのイメージスキャン結果の集約です。
- [ARN] — イメージリソースの Amazon リソースネーム (ARN)。
- [ログストリーム] — 参照先のイメージビルドバージョンのログストリームの詳細へのリンク。

バージョンを一覧表示する

Image Builder コンソールにイメージビルドバージョンを一覧表示するには、次の手順を実行します。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから [イメージ] を選択します。デフォルトでは、イメージリストには所有している各イメージの現在のバージョンが表示されます。
3. イメージのすべてのバージョンのリストを表示するには、現在のバージョンリンクを選択します。このリンクをクリックすると、特定のイメージのすべてのビルドバージョンを一覧表示する [イメージビルドバージョンページ] が開きます。

AWS CLI

で list-image-build-versions コマンドを実行すると AWS CLI、指定したイメージリソースのビルドバージョンの完全なリストが表示されます。このコマンドを実行するには、イメージを所有する必要があります。

以下のコマンド例は、list-image-build-versions コマンドを使用して指定したイメージのすべてのビルドバージョンを一覧表示する方法を示しています。

例: 特定のイメージのビルド・バージョンをリストアップする。

```
aws imagebuilder list-image-build-versions --image-version-arn
 arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-name/1.0.0
```

出力:

この例の出力には、指定したイメージレシピの 2 つのビルドバージョンが含まれています。

```
{
  "requestId": "12f3e45d-67cb-8901-af23-45ed678c9b01",
  "imageSummaryList": [
    {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-
name/1.0.0/2",
    "name": "image-recipe-name",
    "type": "AMI",
    "version": "1.0.0/2",
    "platform": "Linux",
    "osVersion": "Amazon Linux 2",
    "state": {
      "status": "AVAILABLE"
    },
    "owner": "123456789012",
    "dateCreated": "2023-03-10T01:04:40.609Z",
    "outputResources": {
      "amis": [
        {
        "region": "us-west-2",
        "image": "ami-012b3456789012c3d",
        "name": "image-recipe-name 2023-03-10T01-05-12.541Z",
        "description": "First verison of image-recipe-name",
        "accountId": "123456789012"
        }
      ]
    },
    "tags": {}
    },
    {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-
name/1.0.0/1",
    "name": "image-recipe-name",
    "type": "AMI",
```

```
    "version": "1.0.0/1",
    "platform": "Linux",
    "osVersion": "Amazon Linux 2",
    "state": {
      "status": "AVAILABLE"
    },
    "owner": "123456789012",
    "dateCreated": "2023-03-10T00:07:16.384Z",
    "outputResources": {
      "amis": [
        {
        "region": "us-west-2",
        "image": "ami-0d1e23456789f0a12",
        "name": "image-recipe-name 2023-03-10T00-07-18.146132Z",
        "description": "First verison of image-recipe-name",
        "accountId": "123456789012"
        }
      ]
    },
    "tags": {}
    }
  ]
}
```


> **Note**
>
> 現時点では、list-image-build-versions コマンドの出力にはセキュリティ検出結果やログストリームは含まれていません。

イメージの詳細を表示す

Image Builder コンソールのイメージ詳細ページでは、所有している特定のイメージリソースの詳細を表示できます。Image Builder API や SDK および AWS CLI でコマンドやアクションを使用したり、イメージの詳細を取得したりすることもできます。

AWS Resource Access Manager （ AWS RAM) リソース共有を通じて別の が AWS アカウント 共有したリソースの詳細については、AWS RAM 「 ユーザーガイド」の「共有された AWS リソースへのアクセス」を参照してください。

内容

- Image Builderコンソールでイメージの詳細を表示する
- イメージポリシーの詳細を取得 (AWS CLI)

Image Builderコンソールでイメージの詳細を表示する

Image Builder コンソールのイメージ詳細ページには概要セクションがあり、追加情報はタブにまとめられています。ページの見出しは、イメージを作成したレシピの名前とビルドバージョンです。

コンソールの詳細セクションとタブ

- Summary (概要) セクション
- 「出力リソース」タブ
- インフラストラクチャ設定タブ
- ディストリビューション設定タブ
- ワークフロー タブ
- 「セキュリティ結果」タブ
- Tags (タグ) タブ

Summary (概要) セクション

概要セクションはページの幅いっぱいに広がり、以下の詳細が含まれます。これらの詳細は常に表示されます。

レシピ

ビルドバージョンを含まないレシピ名とバージョン。たとえば、ビルドバージョンがsample-linux-recipe | 1.0.1/2の場合、レシピはsample-linux-recipe | 1.0.1で、ビルドバージョンは2です。

作成日

Image Builder バージョンがImage Builder により作成された日時。

イメージステータス

イメージビルドバージョンの現在のステータス。ステータスは、イメージのビルドまたは廃棄に関連する場合があります。たとえば、ビルドプロセス中に、Building または Distributing のステータスが表示される場合があります。イメージの廃棄については、Deprecated または Deleted のステータスが表示される場合があります。

失敗の理由。

イメージステータスの理由。Image Builder コンソールには、ビルドが失敗した理由 (イメージステータス は `Failed` と等しい) のみが表示されます。

「出力リソース」タブ

出力リソースタブには、現在表示されているイメージリソースの出力と配信の詳細が一覧表示されます。Image Builder に表示される情報は、パイプラインがイメージの作成に使用したレシピの種類によって次のように異なります。

イメージのレシピ

- リージョン — Image 列で指定されている出力 Amazon マシンイメージ (AMI) のディストリビューションリージョン。
- イメージ — Image Builder が宛先に配布した AMI の ID。この ID は、Amazon EC2 コンソールの Amazon マシンイメージ (AMI) ページにリンクされています。

> **Note**
>
> Image Builder は、出力イメージリソースを作成した後、AMI を宛先に配布する前に AMI を作成します。

- 名前 — Image Builder が宛先に配布した AMI の名前。
- 説明 — パイプラインが出力イメージリソースの作成に使用したイメージレシピの説明 (オプション)。
- アカウント – 現在表示されている Image Builder イメージリソースを所有 AWS アカウント する 。

コンテナレシピ

Image Builder は、コンテナレシピから作成された出力について以下の詳細を表示します。

- リージョン — Image URI 列で指定されているコンテナーイメージの配布リージョン。
- イメージ URI — Image Builder が宛先リージョンの ECR リポジトリに配布した出力コンテナイメージの URI。

> **Note**
>
> Image Builder は、宛先ごとに 1 行を表示します。出力イメージには、イメージを作成したアカウントに配布するためのエントリが常に 1 つ以上含まれます。追加の送信先には、リージョン間のディストリビューション、 AWS アカウント、または を含めることができます AWS Organizations。詳細については、「EC2 Image Builder ディストリビューション設定の管理」を参照してください。

インフラストラクチャ設定タブ

インフラストラクチャ設定タブには、Image Builder が現在表示されているイメージの構築とテストに使用した Amazon EC2 インフラストラクチャ設定が表示されます。Image Builder には常にインフラストラクチャ設定リソースの名前 (設定名) とその Amazon リソースネーム (ARN) が表示されます。インフラストラクチャ設定によって値が設定される場合、インフラストラクチャの詳細には以下が含まれる場合があります。

- インスタンスのタイプ
- インスタンスプロファイル
- ネットワークインフラストラクチャ
- セキュリティグループ設定
- Image Builder がアプリケーションログを保存する Amazon S3 の場所
- トラブルシューティング用の Amazon EC2 key pair
- 通知用の Amazon SNS トピック

詳細については、「EC2 Image Builder インフラストラクチャ設定の管理」を参照してください。

ディストリビューション設定タブ

配布設定タブには、Image Builder が出力イメージの配布に使用した設定が表示されます。Image Builder には常にディストリビューション設定リソースの名前 (設定名) とその Amazon リソースネーム (ARN) が表示されます。その他のディストリビューションの詳細は、Image Builder パイプラインがイメージの作成に使用したレシピの種類によって次のように異なります。

イメージのレシピ

ディストリビューション設定リソースが値を設定する場合、その他のディストリビューションの詳細には以下が含まれる場合があります。

- リージョン — 出力 Amazon マシンイメージ (AMI) のディストリビューションリージョン。
- 出力 AMI 名 — Image Builder が宛先に配布した AMI の名前。
- 暗号化 (KMS キー) — 設定されている場合、Image Builder AWS KMS key がイメージを暗号化してターゲットリージョンに配布するときに使用します。
- ディストリビューションのターゲットアカウント – クロスアカウントディストリビューションを設定した場合、この列には、ターゲットリージョンで出力イメージを共有 AWS アカウント するためのカンマ区切りのリストが表示されます。
- 共有アクセス許可を持つプリンシパル – イメージを起動するアクセス許可を持つ AWS プリンシパルのカンマ区切りリスト。たとえば、 AWS アカウント グループ AWS Organizations 、組織単位 (OUs

> **Note**
>
> 他のプリンシパルにイメージを起動するアクセス許可を付与しても、Image. AWS bills は、Amazon EC2 がイメージから起動するすべてのインスタンスのアカウントで所有されます。

- 起動設定を高速化するためのターゲットアカウント —
- 関連するライセンス構成 — 指定したリージョンの AMI に関連付けるライセンLicense Manager のライセンス構成 ARN。
- テンプレート設定の起動
- 起動テンプレートのデフォルトバージョンを設定 —

コンテナレシピ

コンテナディストリビューションには常に以下の詳細が含まれます。

- リージョン — Image URI 列で指定されたコンテナーイメージのディストリビューションリージョン。
- イメージ URI — Image Builder が宛先リージョンの Amazon ECR リポジトリに配布した出力コンテナイメージの URI。

> **Note**
>
> Image Builder は、宛先ごとに 1 行を表示します。出力イメージには、イメージを作成したアカウントに配布するためのエントリが常に 1 つ以上含まれます。追加の送信先には、リー

ジョン間のディストリビューション、 AWS アカウント、または を含めることができます AWS Organizations。詳細については、「EC2 Image Builder ディストリビューション設定の管理」を参照してください。

ワークフロー タブ

ワークフローは、Image Builder が新しいイメージを作成するときに実行する一連のステップを定義します。すべてのイメージには、ビルドおよびテストワークフローがあります。コンテナには配布用のワークフローが追加されています。ワークフロータブには、Image Builder がイメージに対して実行した該当するワークフローが表示されます。

ワークフローの種類を絞り込む

Image Builder は、デフォルトでビルドワークフローの概要とワークフローステップを最初に表示します。ただし、ワークフローフィルターには、イメージに対して進行中または完了したワークフローがすべて表示されます。別のワークフローを表示するには、以下のようにリストから選択してください。

イメージワークフロー (AMI 出力)

- `build-image`
- `test-image`

コンテナワークフロー (コンテナ出力)

- `build-container`
- `test-container`
- `distribute-container`

> **Note**
>
> ワークフローがまだ開始されていない場合、ワークフローはリストに表示されません。たとえば、イメージビルドが開始されたばかりの場合、`build-image`がリストに表示される唯一のワークフロータイプです。`test-image`はこの場合い、次のワークフローが開始されると、Image Builder はそのワークフローをリストに追加します。

ワークフローフィルターに続いて、選択したワークフローには、ワークフロータイプごとに以下の詳細を含むランタイムサマリーが表示されます。

ワークフローステータス

このワークフローの現在のランタイムステータス。以下の値を含めることができます。

- 保留中
- スキップ
- 実行中
- 完了
- 失敗
- Rollback-in-progress
- ロールバック完了

実行 ID

Image Builder がワークフローを実行するたびにランタイムリソースを追跡するために割り当てる一意の識別子。

開始

このワークフローのランタイムインスタンスが開始されたときのタイムスタンプ。

終了

このワークフローのランタイムインスタンスが終了したときのタイムスタンプ。

合計ステップ数

ワークフロー内のステップの総数。これは、成功した、スキップされた、失敗したステップのステップ数の合計と等しくなるはずです。

ステップは成功しました。

ワークフロー内で正常に実行されたステップ数のランタイム数。

ステップが失敗しました。

失敗したワークフロー内のステップ数のランタイムカウント。

スキップされたステップ

ワークフロー内でスキップされたステップ数のランタイムカウント。

次のリストの詳細は、ワークフローのこのランタイムインスタンスに含まれるすべてのステップの現在のステータスを報告します。Image Builder では、すべてのイメージタイプで同じ詳細が表示されます。

ステップ #

Image Builder がワークフローステップを実行する順序を表す数値。

ステップ ID

ランタイムに割り当てられる、ワークフローの一意の識別子。

ステップステータス

指定したワークフローステップの現在のランタイムステータス。

ロールバックステータス

ワークフローのこのランタイムインスタンスが失敗した場合の現在のロールバックステータス。

ステップ名

指定されたワークフローステップの名前。

開始

ワークフローのこのランタイムインスタンスに対して指定されたステップが開始されたときのタイムスタンプ。

終了

ワークフローのこのランタイムインスタンスの指定されたステップが終了したときのタイムスタンプ。

「セキュリティ結果」タブ

スキャンを有効にすると、「セキュリティ結果」タブに「一般的な脆弱性と暴露 (CVE)」の結果が表示されます。Amazon Inspector は、Image Builder が新しいイメージを作成するために起動したテストインスタンスでこれらの検出結果を特定しました。Image Builder がイメージの結果をキャプチャするようにするには、次のようにスキャンを設定する必要があります。

1. アカウントの Amazon Inspector スキャンを有効にしてください。詳細については、Amazon Inspector ユーザーガイドの Amazon Inspector を使用するを参照してください。

2. このイメージを作成するパイプラインのセキュリティ結果を有効にしてください。パイプラインのセキュリティ結果を有効にすると、Image Builder はテストインスタンスを終了する前に検出結果のスナップショットを保存します。詳細については、「で Image Builder イメージのセキュリティスキャンを設定する AWS Management Console」を参照してください。

セキュリティ結果タブには、Amazon Inspector がイメージについて特定した各脆弱性に関する以下の詳細が含まれます。

重要度

CVE検出結果の重大度レベル。値は次のとおりです。

- トリアージされていない
- 情報
- 低
- 中程度
- 高い
- [非常事態]

検出結果 ID

Amazon Inspector がテストインスタンスをスキャンしたときにイメージについて検出した CVE 検出結果の固有識別子。ID は [セキュリティ検出結果] > [脆弱性別] ページにリンクされています。詳細については、「で Image Builder イメージのセキュリティ検出結果を管理する AWS Management Console」を参照してください。

ソース

CVE検出結果の検出結果のソース。

年齢

イメージで検出結果が最初に観測されてからの日数。

Inspector スコア

Amazon Inspector が CVE の検出結果に割り当てたスコア。

Tags (タグ) タブ

タグ タブには、イメージに定義したタグがすべて表示されます。

イメージポリシーの詳細を取得 (AWS CLI)

次の例は、Amazon リソースネーム (ARN) によりイメージポリシーの詳細を取得する方法を示しています。

```
aws imagebuilder get-image-policy --image-arn arn:aws:imagebuilder:us-
west-2:123456789012:image/example-image/2019.12.02
```

イメージの作成

このセクションでは、Image Builder イメージを作成し、進行中のビルドをキャンセルする方法を示します。

内容

イメージの作成

新しい Image Builder イメージを作成する方法はいくつかあります。例えば、次のいずれかの方法を使用して、 AWS Management Console または でイメージを作成できます AWS CLI。CreateImage API アクションを使用することもできます。関連する SDK リクエストについては、「EC2 Image Builder API」リファレンスのそのコマンドの「関連項目」リンクを参照してください。

AWS Management Console

既存のパイプラインで新しいイメージを作成するには、次のようにパイプラインを手動で実行できます。パイプラインウィザードを使用して新しいイメージを一から作成することもできます。作成するイメージのタイプに応じて、イメージパイプラインを作成します。 または イメージパイプラインを作成します(Docker) を参照してください。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから、[イメージパイプライン] を選択します。
3. 実行したいパイプライン名の横にあるチェックボックスを選択します。
4. イメージを作成するには、アクション メニューから パイプラインを実行 を選択します。パイプラインが開始されます。

パイプラインを実行するスケジュールを指定したり、Amazon EventBridge を使用して設定したルールに基づいてパイプラインを実行したりすることもできます。

AWS CLI

で create-image コマンドを実行する前に AWS CLI、次のリソースがまだ存在しない場合は作成する必要があります。

必要なリソース

- レシピ - 以下のように、イメージに正確に1つのレシピを指定する必要があります：

  イメージのレシピ

  Image Builder の Amazon リソースネーム (ARN) を `--image-recipe-arn` パラメータで指定します。

  コンテナレシピ

  コンテナレシピリソースの ARN を `--container-recipe-arn` パラメータで指定します。

- インフラストラクチャ構成[- `--infrastructure-configuration-arn`パラメータでインフラストラクチャ構成リソースのARNを指定します。

Image Builder が必要とする以下のいずれかのリソースを指定できます。

オプションのリソースと設定

- 配布設定 - デフォルトでは、Image Builder は、create-imageコマンドを実行したリージョンのアカウントに出力イメージリソースを配布します。ディストリビューションに追加の宛先または設定を指定するには、`--distribution-configuration-arn` パラメータを使用してディストリビューション設定リソースの ARN を指定します。
- イメージスキャン –イメージまたはコンテナテストインスタンス上で Amazon Inspector の検出結果用のスナップショットを構成するには、`--image-scanning-configuration` パラメータを使用します。コンテナイメージの場合は、Amazon Inspector がスキャンに使用する ECR リポジトリも指定します。
- イメージテスト - Image Builderのテストステージを抑制するには、`--image-tests-configuration`パラメータを使用します。または、実行時間のタイムアウトを設定することもできます。
- イメージタグ - `--tags`パラメータを使って出力イメージにタグを追加します。

- イメージワークフロー — ビルドワークフローまたはテストワークフローを指定しない場合、Image Builder はデフォルトのイメージワークフローでイメージを作成します。作成したワークフローを指定するには、`--workflows` パラメータを使用します。

> **Note**
>
> イメージワークフローを指定する場合は、Image Builder がワークフローアクションを実行するために使用する IAM ロールの名前または ARN も `--execution-role` パラメータで指定する必要があります。

次の例は、「create-image」 AWS CLI コマンドを使用してImage Builder を使用してImage Builder を作成する方法を示しています。詳細については、『AWS CLI コマンドリファレンス』を参照してください。

例 デフォルトのディストリビューションで基本的なイメージを作成する。

```
aws imagebuilder create-image --image-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-recipe/simple-recipe-linux/1.0.0 --infrastructure-
configuration-arn arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-
configuration/simple-infra-config-linux
```

出力:

```
{
    "requestId": "1abcd234-e567-8fa9-0123-4567b890cd12",
    "imageVersionList": [
        {
            "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/simple-recipe-
linux/1.0.0",
            "name": "simple-recipe-linux",
            ...
        }
    ]
}
```

イメージ作成をキャンセルします(AWS CLI)

進行中のイメージビルドをキャンセルするには、以下のcancel-image-creationコマンドを使用します。

```
aws imagebuilder cancel-image-creation --image-build-version-arn
 arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-recipe/2019.12.03/1
```

VM イメージのインポート

Image Builder は Amazon EC2 VM Import/Export API と統合されているため、インポートプロセスをバックグラウンドで非同期的に実行できます。Image Builder は VM インポートのタスク ID を参照して進行状況を追跡し、出力として Image Builder イメージリソースを作成します。これにより、VM のインポートが完了する前に、レシピ内の Image Builder イメージリソースを参照できます。

VM をインポートする (コンソール)

Image Builder コンソールで VM をインポートするには、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから [イメージ] を選択します。
3. [イメージのインポート] を選択します。
4. [イメージのインポート] ページで、以下のセクションの詳細を入力します。完了したら、イメージのインポート] を選択します。

全般

1. ベースイメージに固有の 名前 を指定します。
2. ベースイメージの バージョン を指定します。形式は以下のようになります: *major.minor.patch*
3. オプションで、ベースイメージの [説明] を入力することもできます。

ベースイメージオペレーティングシステム

1. お使いの VM OS プラットフォームに合った [イメージオペレーティングシステム (OS)] オプションを選択します。

2. お使いの VM バージョンと一致する [OS バージョン] をリストから選択します。

VM インポート設定

VM を仮想化環境からエクスポートすると、そのプロセスによって 1 つ以上のディスクコンテナファイルのセットが作成されます。これらは VM の環境、設定、データのスナップショットとして機能します。これらのファイルを使用して、VM をイメージレシピのベースイメージとしてインポートできます。Image Builder での VM インポートの詳細については、「VM イメージのインポートとエクスポート」を参照してください。

インポートソースの場所を指定するには、次の手順に従います。

インポートソース

ディスクコンテナ 1 セクションで、インポートする最初の VM イメージディスクコンテナまたはスナップショットのソースを指定します。

1. ソース — これは S3 バケットでも EBS スナップショットでもかまいません。
2. ディスクのS3の場所を選択 - ディスクイメージが保存されているAmazon S3の場所を入力します。場所を参照するには、[S3 を参照] を選択します。
3. ディスクコンテナを追加するには、[ディスクコンテナを追加] を選択します。

IAM ロール

IAM ロールを VM インポート設定に関連付けるには、[IAM ロール] ドロップダウンリストからロールを選択するか、[新規ロールを作成] を選択して新しいロールを作成します。新しいロールを作成すると、[IAM ロール] コンソールページが別のタブで開きます。

詳細設定 - オプション

次のオプション設定はオプショナルです。これらの設定により、インポートによって作成されるベースイメージの暗号化、ライセンス、タグなどを設定できます。

ベースイメージアーキテクチャ

VM インポートソースのアーキテクチャを指定するには、アーキテクチャ リストから値を選択します。

暗号化

VM ディスクイメージが暗号化されている場合は、インポートプロセスに使用するキーを指定する必要があります。インポート用の KMS キーを指定するには、[暗号化 (KMS キー)] リストから値を選択します。このリストには、現在のリージョンでアカウントがアクセスできる KMS キーが含まれています。

ライセンス管理

VM をインポートすると、インポートプロセスによって VM OS が自動的に検出され、適切なライセンスがベースイメージに適用されます。お使いの OS プラットフォームに応じて、ライセンスの種類は次のとおりです。

- License included - あなたのプラットフォームに適した AWS ライセンスがベースイメージに適用されます。
- Bring-Your-Own-License (BYOL) - VMのライセンスを保持します（該当する場合）。

で作成されたライセンス設定をベースイメージ AWS License Manager にアタッチするには、ライセンス設定名リストから を選択します。License Manager の詳細については、「 の使用 AWS License Manager」を参照してください。

> **Note**
>
> - ライセンスコンフィギュレーションには、企業契約の条件に基づくライセンスルールが含まれています。
> - Linux は BYOL ライセンスのみをサポートします。

タグ (ベースイメージ)

タグはキーと値のペアを使用して、検索可能なテキストを Image Builder リソースに割り当てます。インポートしたベースイメージのタグを指定するには、[キー] ボックスと [値] ボックスを使用してキーと値のペアを入力します。

タグを追加するには、タグの追加 を選択します。タグを削除するには、[タグの削除] を選択します。

VM (AWS CLI) をインポートする

VM をディスクから AMI にインポートし、すぐに参照できる Image Builder イメージリソースを作成するには、 AWS CLIから以下の手順に従ってください。

1. AWS CLIのAmazon EC2 VM Import/Export import-imageコマンドで、VMのインポートを開始する。コマンド・レスポンスで返されるタスクIDをメモしておく。これは次のステップで必要になります。詳細については、VM Import/Export ユーザーガイドの「VM Import/Export を使用してイメージとして VM をインポート」を参照してください。
2. CLI 入力 JSON ファイルの作成

   で使用される Image Builder import-vm-image コマンドを効率化するために AWS CLI、コマンドに渡すすべてのインポート設定を含む JSON ファイルを作成します。

   > **Note**
   >
   > JSON ファイル内のデータ値の命名規則は、Image Builder API アクションリクエストパラメータに指定されているパターンに従います。API コマンドリクエストパラメータを確認するには、EC2 Image Builder API リファレンス」のImportVmImage「 コマンド」を参照してください。
   > データ値をコマンドラインパラメータとして指定するには、AWS CLI コマンドリファレンスで指定されているパラメータ名を参照してください。 オプションとして、イメージビルダーimport-vm-imageコマンドに指定します。

   以下に、これらの例で指定するパラメータの概要を示します。

   - 名前 (文字列、必須) — インポートからの出力として作成する Image Builder イメージリソースの名前。
   - semanticVersion (文字列、必須) — 出力イメージのセマンティックバージョンは次の形式で表されます: 各位置に特定のバージョン (「メジャー」「マイナー」「パッチ」) を示す数値が付いています。例えば `1.0.0` です。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。
   - 説明 (文字列) — イメージレシピの説明。
   - platform (文字列、必須) — インポートされた VM のオペレーティングシステムプラットフォーム。
   - vmImportTaskID (文字列、必須) — Amazon EC2 VM インポートプロセスの `ImportTaskId` (AWS CLI) 。Image Builder はインポートプロセスを監視して作成した AMI を取り込み、レシピですぐに使用できる Image Builder イメージリソースを構築します。

- clientToken(文字列、必須) - リクエストの冪等性を保証するために提供する、大文字と小文字を区別する一意の識別子。詳細については、Amazon EC2 API Referenceの冪等性の確保を参照してください。
- タグ (文字列マップ) — タグはインポートリソースに添付されるキーと値のペアです。最大 50 個のキーと値のペアを使用できます。

ファイルに `import-vm-image.json` という名前を付けて保存し、Image Builder import-vm-image コマンドで使用します。

```
{
    "name": "example-request",
    "semanticVersion": "1.0.0",
    "description": "vm-import-test",
    "platform": "Linux",
    "vmImportTaskId": "import-ami-01ab234567890cd1e",
    "clientToken": "asz1231231234cs3z",
    "tags": {
     "Usage": "VMIE"
    }
}
```

3. イメージのインポート

作成したファイルを入力として、import-vm-image コマンドを実行します。

```
aws imagebuilder import-vm-image --cli-input-json file://import-vm-image.json
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

Image Builder イメージのセキュリティ検出結果の管理

Amazon Inspector でセキュリティスキャンを有効にすると、アカウント内のマシンイメージと実行中のインスタンスが継続的にスキャンされ、オペレーティングシステムとプログラミング言語の脆弱性が検出されます。有効にすると、セキュリティスキャンが自動的に実行され、Image Builder は、新しいイメージを作成するときに、テストインスタンスの検出結果のスナップショットを保存できます。Amazon Inspector は有料機能です。

Amazon Inspector は、ソフトウェアまたはネットワーク設定の脆弱性を発見すると、次のアクションを実行します。

- 検出結果があったこと通知します。
- 検出結果の重要度評価 重要度評価では、検出結果の優先順位付けに役立つように脆弱性を分類します。評価には次の値が含まれます。
  - トリアージされていない
  - 情報
  - 低
  - 中程度
  - 高い
  - [非常事態]
- 検出結果に関する情報と、詳細情報が含まれる追加リソースへのリンクを提供します。
- 検出結果を生成した問題の解決に役立つ修正ガイダンスを提供します。

で Image Builder イメージのセキュリティスキャンを設定する AWS Management Console

アカウントの Amazon Inspector を有効にした場合、Amazon Inspector は Image Builder が起動する EC2 インスタンスを自動的にスキャンして、新しいイメージをビルドしてテストします。これらのインスタンスはビルドとテストのプロセス中は有効期間が短く、検出結果は通常、インスタンスがシャットダウンするとすぐに期限切れになります。新しいイメージの検出結果の調査と修正に役立つように、Image Builder では、ビルドプロセス中に Amazon Inspector がテストインスタンスについて特定した検出結果をオプションでスナップショットとして保存できます。

ステップ 1: アカウントの Amazon Inspector セキュリティスキャンを有効にする

Image Builder コンソールからアカウントの Amazon Inspector セキュリティスキャンを有効にするには、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインで、[セキュリティスキャン設定] を選択します。[セキュリティスキャン] ダイアログボックスが開きます。

   アカウントのスキャンステータスが、ダイアログボックスに表示されます。Amazon Inspector がアカウントですでに有効化されている場合、ステータスは [有効] と表示されます。
3. 説明のステップ 1 と 2 に従って Amazon Inspector のスキャンを有効にします。

> **Note**
>
> Amazon Inspector では料金が発生します。詳細については、「Amazon Inspector の料金」を参照してください。

パイプラインのスキャンを有効にしている場合、Image Builder は、新しいイメージを作成するときに、ビルドインスタンスに対する検出結果のスナップショットを取得します。これにより、Image Builder がビルドインスタンスを終了した後で、検出結果にアクセスできます。

ステップ 2: 脆弱性検出結果のスナップショットを保存するようにパイプラインを設定する

パイプラインの脆弱性検出結果スナップショットを設定するには、次の手順を実行します。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから、[イメージパイプライン] を選択します。
3. パイプラインの詳細を指定するには、次のいずれかの方法を選択します。

   新規パイプラインを作成する

   1. [パイプライン] ページで、[イメージパイプラインの作成] をクリックします。パイプラインウィザードの [パイプラインの詳細を指定] ページが開きます。

   既存のパイプラインを更新する

   1. [イメージパイプライン] ページから、更新するパイプラインの [パイプライン名] リンクを選択します。パイプラインの詳細ページが開きます。

> **Note**
>
> 更新するパイプラインの名前の横にあるチェックボックスを選択し、[詳細を表示] を選択することもできます。

2. パイプライン詳細ページの [アクション] メニューから [パイプラインの編集] を選択します。「パイプラインの編集」ページが開きます。

4. パイプラインウィザードの [全般] セクションまたは [パイプラインの編集] ページで、[セキュリティスキャンを有効にする] チェックボックスを選択します。

> **Note**
>
> 後でスナップショットをオフにする場合は、パイプラインを編集してチェックボックスをオフにできます。これによってアカウントの Amazon Inspector スキャンが無効になるわけではありません。Amazon Inspectorのスキャンを無効にするには、Amazon InspectorユーザーガイドのAmazon Inspectorの無効化を参照してください。

で Image Builder イメージのセキュリティ検出結果を管理する AWS Management Console

[セキュリティ検出結果]リストページには、リソースに関する検出結果に関する概要情報と、適用可能な数種類のフィルタに基づくビューが表示されます。各ビューの上部には、ビューを変更するための以下のオプションが表示されます。

- [すべてのセキュリティ結果] — Image Builder コンソールのナビゲーションペインから [セキュリティ検出結果] ページを選択した場合のデフォルトビューです。
- [脆弱性別] — このビューには、アカウント内の検出結果のあるすべてのイメージリソースの概要リストが表示されます。[検出結果 ID] は、検出結果に関する詳細情報を確認できます。この情報は、ページの右側にあるパネルに表示されます。パネルには次の情報が含まれます。
  - 検出結果の詳細説明。
  - [検出結果の詳細] タブ。このタブには、検出結果の概要、影響を受けるパッケージ、修復アドバイスの概要、脆弱性の詳細、および関連する脆弱性が含まれます。[脆弱性 ID] は、National Vulnerability Database の詳細な脆弱性情報にリンクしています。

  - [スコアの内訳] タブ。このタブには CVSS スコアと Amazon Inspector スコアside-by-side の比較が含まれており、該当する場合は Amazon Inspector がスコアを変更した場所を確認できます。
- [イメージパイプライン別] — このビューには、アカウント内の各イメージパイプラインの検出結果の数が表示されます。Image Builder では、重要度が中程度以上の結果の数と、すべての検出結果の合計が表示されます。リスト内のすべてのデータは次のようにリンクされています。
  - [イメージパイプライン名列] は、指定されたイメージパイプラインの詳細ページにリンクします。
  - 重要度列のリンクをクリックすると、関連するイメージパイプライン名と重要度レベルでフィルタリングされた [すべてのセキュリティ検出結果] ビューが開きます。

  検索条件を使用して、結果を絞り込むこともできます。
- [イメージ別] — このビューには、アカウント内の各イメージビルドの検出結果数が表示されます。Image Builder では、重要度が中程度以上の結果の数と、すべての検出結果の合計が表示されます。リスト内のすべてのデータは次のようにリンクされています。
  - [イメージ名] 列は、指定したイメージビルドのイメージ詳細ページにリンクします。詳細については、「イメージの詳細を表示す」を参照してください。
  - 重要度列のリンクをクリックすると、関連するイメージパイプライン名と重要度レベルでフィルタリングされた [すべてのセキュリティ検出結果] ビューが開きます。

  検索条件を使用して、結果を絞り込むこともできます。

Image Builder では、デフォルトの [すべてのセキュリティ検出結果] ビューの「検出結果リスト」セクションに次の詳細が表示されます。

重要度

CVE検出結果の重大度レベル。値は次のとおりです。

- トリアージされていない
- 情報
- 低
- 中程度
- 高い
- [非常事態]

検出結果 ID

Amazon Inspector がビルドインスタンスをスキャンしたときにイメージについて検出した CVE 検出結果の固有識別子。ID は [セキュリティ検出結果] > [脆弱性別] ページにリンクされています。

イメージ ARN

[検出結果の ID] 列で指定された、検出結果を含むイメージの Amazon リソースネーム (ARN)。

パイプライン

[イメージ ARN] 列で指定されたイメージを構築したパイプライン。

説明

検出結果の簡単な説明。

Inspector スコア

Amazon Inspector が CVE の検出結果に割り当てたスコア。

修正

検出結果を修正するための推奨アクション方針に関する詳細へのリンク。

[公開日]

この脆弱性がベンダーのデータベースに最初に追加された日付と時刻。

リソースをクリーンアップする

予期しない料金が発生しないように、このガイドの例で作成したリソースとパイプラインは必ずクリーンアップしてください。Image Builder でのリソースの削除については、「EC2 Image Builder resourcesの削除」を参照してください。

EC2 Image Builder インフラストラクチャ設定の管理

インフラストラクチャ設定を使用して、Image Builder が EC2 Image Builder イメージの構築とテストに使用する Amazon EC2 インフラストラクチャを指定できます。インフラストラクチャには次のような設定が含まれています。

- ビルドおよびテストインフラストラクチャーのインスタンスタイプ。複数のインスタンスタイプを指定することをお勧めします。これにより、Image Builder は十分な容量のプールからインスタンスを起動できます。これにより、一時的なビルドエラーを減らすことができます。

- ビルドインスタンスとテストインスタンスにカスタマイズアクティビティの実行に必要な権限を与えるインスタンスプロファイル。たとえば、Amazon S3 からリソースを取得するコンポーネントがある場合、インスタンスプロファイルにはそれらのファイルにアクセスするためのアクセス権限が必要です。インスタンスプロファイルには、EC2 Image Builder がインスタンスと正常に通信するための最小限の権限セットも必要です。詳細については、「前提条件」を参照してください。
- パイプラインのビルドインスタンスとテストインスタンスの VPC、サブネット、およびセキュリティグループ。
- Image Builder がビルドとテストのアプリケーションログを保存する Amazon S3 の場所。ロギングを設定する場合、インフラストラクチャ構成で指定されたインスタンスプロファイルは、ターゲットバケット(`arn:aws:s3:::BucketName/*`)に対して`s3:PutObject`の権限を持っている必要があります。
- ビルドが失敗し、`terminateInstanceOnFailure` を `false` に設定していた場合、Amazon EC2 キーペアを通じてインスタンスにログオンし、トラブルシューティングを行うことができます。
- Image Builder がイベント通知を送信する SNS トピックです。Image Builder と Amazon SNS との統合の詳細については、「Image BuilderにおけるAmazon SNSの統合」を参照してください。

> **Note**
>
> SNS トピックが暗号化されている場合、このトピックを暗号化するキーは、Image Builder サービスが実行されるアカウントにある必要があります。Image Builder は、他のアカウントのキーで暗号化された SNS トピックに通知を送信できません。

Image Builder コンソール、Image Builder API、または AWS CLIの imagebuilder コマンドを使用して、インフラストラクチャ設定を作成および管理できます。

内容

- インフラストラクチャ設定の詳細を一覧表示および表示します。
- インフラストラクチャ構成を作成します。
- インフラストラクチャ設定を更新します。
- EC2 Image Builder とインターフェイス VPC エンドポイント (AWS PrivateLink)

> **Tip**
>
> 同じ型のリソースが複数にある場合に、割り当てたタグに基づいて特定のリソースをすばやく識別できます。の Image Builder コマンドを使用してリソースにタグを付ける方法の詳細については AWS CLI、このガイドのリソースのタグ付け「」セクションを参照してください。

インフラストラクチャ設定の詳細を一覧表示および表示します。

このセクションでは、EC2 Image Builder インフラストラクチャ設定の情報を検索したり詳細を表示したりするさまざまな方法について説明します。

インフラストラクチャ設定の詳細

- インフラストラクチャ設定を一覧表示 (AWS CLI)
- インフラストラクチャ設定の詳細 (AWS CLI)を所得

インフラストラクチャ設定を一覧表示 (AWS CLI)

次の例では、list-infrastructure-configurationsコマンドを AWS CLIコマンドで使用し、すべてのインフラストラクチャーコンフィギュレーションをリストアップする方法を示している。

```
aws imagebuilder list-infrastructure-configurations
```

インフラストラクチャ設定の詳細 (AWS CLI)を所得

次の例は、 で get-infrastructure-configuration コマンドを使用して、Amazon リソースネーム (ARN) を指定してインフラストラクチャ設定の詳細 AWS CLI を取得する方法を示しています。

```
aws imagebuilder get-infrastructure-configuration --infrastructure-configuration-arn
 arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/my-example-
infrastructure-configuration
```

インフラストラクチャ構成を作成します。

このセクションでは、 で Image Builder コンソールまたは imagebuilder コマンドを使用してインフラストラクチャ設定 AWS CLI を作成する方法について説明します。

Console

Image Builder コンソールでインフラストラクチャ設定リソースを作成するには、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインで、インフラストラクチャー設定 を選択します。
3. インフラストラクチャー構成の作成 を選択します。
4. キュー セクションに、以下の情報を入力します。
   - インフラストラクチャー設定リソースの名前 を入力します。
   - ビルドインスタンスとテストインスタンスのコンポーネントアクセス許可についてインスタンスプロファイルに関連付ける [IAM ロール] を選択します。Image Builder は、これらのアクセス許可を使用して、コンポーネントをダウンロードして実行し、 にログをアップロードし CloudWatch、レシピ内のコンポーネントが指定する追加のアクションを実行します。
5. AWS インフラストラクチャパネルでは、使用可能な残りのインフラストラクチャ設定を設定できます。以下の必須情報を入力します。
   - インスタンスタイプ — このビルドに使用するインスタンスタイプを 1 つ以上指定できます。サービスは可用性に基づいてこれらのインスタンスタイプから 1 つを選択します。
   - SNS トピック (オプション) — EC2 Image Builder から通知とアラートを受信する SNS トピックを選択します。

   以下の設定に値を指定しない場合、該当する場合、サービス固有のデフォルトが使用されます。

   - VPC、サブネット、セキュリティグループ — Image Builder はデフォルトの VPC とサブネットを使用します。VPC インターフェイスエンドポイントの設定の詳細については、「EC2 Image Builder とインターフェイス VPC エンドポイント (AWS PrivateLink)」を参照してください。
   - トラブルシューティング設定 セクションでは、以下の値を設定できます。
     - デフォルトでは、障害発生時にインスタンスを終了 チェックボックスが選択されています。ただし、ビルドが失敗した場合、EC2 インスタンスにログオンしてトラブルシュー

ティングを行うことができます。ビルドが失敗した後もインスタンスを実行し続けるには、このチェックボックスをオフにします。

- キーペア — ビルドが失敗した後も EC2 インスタンスが実行され続ける場合は、key pair を作成するか、既存のキーペアを使用してインスタンスにログオンし、トラブルシューティングを行うことができます。
- ログ — Image Builder がビルドとテストのトラブルシューティングに役立つアプリケーションログを書き込める S3 バケットを指定できます。S3 バケットを指定しない場合、Image Builder はアプリケーションログをインスタンスに書き込みます。

- インスタンスメタデータ設定 セクションでは、Image Builder がイメージのビルドとテストに使用する EC2 インスタンスに適用する次の値を設定できます。
  - メタデータバージョン を選択して、EC2 がインスタンスのメタデータの取得リクエストで、署名付きトークンヘッダーを必要とするかどうかを判断します。
    - V1 と V2 (トークンはオプション) — 何も選択しない場合のデフォルト値。
    - V2 (トークンが必要)

    > **Note**
    >
    > Image Builder がパイプラインビルドから起動するすべての EC2 インスタンスを IMDSv2 を使用するように設定して、インスタンスメタデータの取得リクエストに署名付きトークンヘッダーが必要になるようにすることをお勧めします。

  - メタデータトークンの応答ホップ上限数 – メタデータトークンに輸送できるネットワークホップ数。最小ホップ:1、最大ホップ:64、デフォルトは 1 ホップ。

6. インフラストラクチャタグセクション (オプション) では、Image Builder がビルドプロセス中に起動する Amazon EC2 インスタンスにメタデータタグを割り当てることができます。タグはキーと値のペアとして入力されます。
7. タグセクション (オプション) では、Image Builder が出力として作成するインフラストラクチャ設定リソースにメタデータタグを割り当てることができます。タグはキーと値のペアとして入力されます。

AWS CLI

次の例は、 の Image Builder create-infrastructure-configuration コマンドを使用してイメージのインフラストラクチャを設定する方法を示しています AWS CLI。

1. CLI 入力 JSON ファイルの作成

   このインフラストラクチャー設定例では、m5.large と m5.xlarge の二つのインスタンスタイプを指定しています。複数のインスタンスタイプを指定することをお勧めします。これにより、Image Builder は十分な容量のプールからインスタンスを起動できます。これにより、一時的なビルドエラーを減らすことができます。

   instanceProfileName はプロファイルがカスタマイズアクティビティを実行するのに必要な権限をインスタンスに提供するインスタンスプロファイルを指定します。たとえば、Amazon S3 からリソースを取得するコンポーネントがある場合、インスタンスプロファイルにはそれらのファイルにアクセスするためのアクセス権限が必要です。インスタンスプロファイルには、EC2 Image Builder がインスタンスと正常に通信するための最小限の権限セットも必要です。詳細については、「前提条件」を参照してください。

   ファイル編集ツールを使用して、次の例に示すキーと環境に有効な値を含む JSON ファイルを作成します。この例では、create-infrastructure-configuration.jsonという名前のファイルを使用します。

```
{
    "name": "MyExampleInfrastructure",
    "description": "An example that will retain instances of failed builds",
    "instanceTypes": [
        "m5.large", "m5.xlarge"
    ],
    "instanceProfileName": "myIAMInstanceProfileName",
    "securityGroupIds": [
        "sg-12345678"
    ],
    "subnetId": "sub-12345678",
    "logging": {
        "s3Logs": {
            "s3BucketName": "my-logging-bucket",
            "s3KeyPrefix": "my-path"
        }
    },
    "keyPair": "myKeyPairName",
    "terminateInstanceOnFailure": false,
    "snsTopicArn": "arn:aws:sns:us-west-2:123456789012:MyTopic"
}
```

2. 以下のコマンドを実行する際には、作成したファイルを入力として使用します。

```
aws imagebuilder create-infrastructure-configuration --cli-input-json
 file://create-infrastructure-configuration.json
```

インフラストラクチャ設定を更新します。

このセクションでは、 の Image Builder コンソールまたは imagebuilder コマンドを使用してインフラストラクチャ設定リソース AWS CLI を更新する方法について説明します。リソースを追跡するには、次のようにタグを適用します。タグはキーと値のペアとして入力されます。

- リソースタグは、Image Builder がビルドプロセス中に起動する Amazon EC2 インスタンスにメタデータタグを割り当てます。
- タグは、Image Builder が出力として作成するインフラストラクチャ設定リソースにメタデータタグを割り当てます。

Console

Image Builder コンソールから以下のインフラストラクチャー設定の詳細を編集できます。

- インフラストラクチャ設定の 説明。
- [IAM ロール] をインスタンスプロファイルに関連付けます。
- AWS インフラストラクチャ。これには、インスタンスタイプと通知用の SNS トピックが含まれます。
- VPC、サブネット、セキュリティグループ。
- 障害発生時にインスタンスを終了する、接続用のキーペア、インスタンスLog用のオプションの S3 バケットの場所などの トラブルシューティング設定。

Image Builder コンソールからインフラストラクチャ構成リソースを更新するには、以下の手順に従います：

既存の Image Builder インフラストラクチャー構成を選択してください

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。

2. アカウントのインフラストラクチャー設定リソースのリストを表示するには、ナビゲーションペインから インフラストラクチャー設定 を選択します。
3. インフラストラクチャー設定の詳細を表示したり編集したりするには、「設定名」リンクを選択します。これにより、インフラストラクチャ設定の詳細ビューが開きます。

> **Note**
>
> 設定名 の横にあるボックスをオンにして、詳細を表示 を選択することもできます。

4. インフラストラクチャー詳細 パネルの右上隅から 編集 を選択します。
5. インフラストラクチャー設定に加えた更新を保存する準備ができたら、変更を保存 を選択します。

AWS CLI

次の例は、 の Image Builder update-infrastructure-configuration コマンドを使用してイメージのインフラストラクチャ設定を更新する方法を示しています AWS CLI。

1. CLI 入力 JSON ファイルの作成

   このインフラストラクチャー設定例では、create の例と同じ設定を使用していますが、`terminateInstanceOnFailure`設定を `false` に更新している点が異なります。update-infrastructure-configurationコマンドを実行した後、このインフラストラクチャ構成を使用するパイプラインは、ビルドが失敗するとビルドインスタンスとテストインスタンスを終了します。

   ファイル編集ツールを使用して、次の例に示すキーと環境に有効な値を含む JSON ファイルを作成します。この例では、`update-infrastructure-configuration.json`という名前のファイルを使用します。

```
{
"infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/my-example-infrastructure-
configuration",
"description": "An example that will terminate instances of failed builds",
"instanceTypes": [
    "m5.large", "m5.2xlarge"
],
"instanceProfileName": "myIAMInstanceProfileName",
```

```
"securityGroupIds": [
    "sg-12345678"
],
"subnetId": "sub-12345678",
"logging": {
    "s3Logs": {
        "s3BucketName": "my-logging-bucket",
        "s3KeyPrefix": "my-path"
    }
},
"terminateInstanceOnFailure": true,
"snsTopicArn": "arn:aws:sns:us-west-2:123456789012:MyTopic"
}
```

2. 以下のコマンドを実行する際には、作成したファイルを入力として使用します。

```
aws imagebuilder update-infrastructure-configuration --cli-input-json
 file://update-infrastructure-configuration.json
```

EC2 Image Builder とインターフェイス VPC エンドポイント (AWS PrivateLink)

VPCとEC2 Image Builderの間にプライベート接続を確立するには、インターフェイス VPC エンドポイントを作成します。インターフェイスエンドポイントは、インターネットゲートウェイAWS PrivateLink、NAT デバイス、VPN 接続、 AWS Direct Connect 接続のいずれも必要とせずに Image Builder APIs にプライベートにアクセスできるテクノロジーである を利用しています。VPC のインスタンスは、パブリック IP アドレスがなくても API と通信できます。VPCとImage Builder間のトラフィックは、Amazonのネットワークから出ることはありません。

各インターフェースエンドポイントは、サブネット内の 1 つ以上の Elastic Network Interface によって表されます。新しいイメージを作成するときに、インフラストラクチャ設定で VPC subnet-id を指定できます。

> **Note**
>
> VPC 内からアクセスする各サービスには、独自のエンドポイントポリシーを持つ独自のインターフェイスエンドポイントがあります。Image Builder は AWSTOE コンポーネントマネージャーアプリケーションをダウンロードし、S3 バケットからマネージドリソースにアクセスしてカスタムイメージを作成します。これらのバケットへのアクセスを許可するには、S3 エ

ンドポイントポリシーを更新して許可する必要があります。詳細については、「S3 バケットアクセスのカスタムポリシー」を参照してください。

VPC エンドポイントの詳細については、Amazon VPC ユーザーガイドの「インターフェイス VPC エンドポイント (AWS PrivateLink PrivateLink)」を参照してください。

Image Builder VPC エンドポイントに関する考慮事項

Image Builder用のインターフェースVPCエンドポイントを設定する前に、Amazon VPC User Guideのインターフェースエンドポイントのプロパティと制限事項を確認してください。

Image Builderは、VPCからすべてのAPIアクションの呼び出しをサポートしています。

Image Builder用のインターフェースVPCエンドポイントを作成する

Image Builder サービスの VPC エンドポイントを作成するには、Amazon VPC コンソールまたは AWS Command Line Interface () を使用できますAWS CLI。詳細については、 Amazon VPC ユーザーガイド のインターフェイスエンドポイントの作成を参照してください。

以下のサービス名を使用して、Image Builder用のVPCエンドポイントを作成します：

- com.amazonaws.*region*.imagebuilder

エンドポイントのプライベート DNS を有効にすると、リージョンのデフォルト DNS 名 (`imagebuilder.us-east-1.amazonaws.com` など) を使用して、QLDB への API リクエストを実行できます。対象のリージョンに適用されるエンドポイントを調べるには、Amazon Web Services 全般のリファレンスの]EC2 Image Builderのエンドポイントとクォータを参照する。

詳細については、Amazon VPC User Guideのインターフェースエンドポイントを介したサービスへのアクセスを参照してください。

Image Builder用VPCエンドポイントポリシーの作成

VPC エンドポイントには、 へのアクセスを制御するエンドポイントポリシーをアタッチできます。このポリシーでは、以下の情報を指定します。

- アクションを実行できるプリンシパル。
- 実行可能なアクション。

- このアクションを実行できるリソース。

レシピで Amazon が管理するコンポーネントを使用している場合、Image Builder の VPC エンドポイントは、以下のサービス所有のコンポーネントライブラリへのアクセスを許可する必要があります。

`arn:aws:imagebuilder:region:aws:component/*`

> **Important**
>
> EC2 Image Builder のインターフェイス VPC エンドポイントにデフォルト以外のポリシーを適用すると、 からの失敗など、失敗した特定の API リクエストはRequestLimitExceeded、 AWS CloudTrail または Amazon にログ記録されない場合があります CloudWatch。

詳細については、「Amazon VPC ユーザーガイド」の「VPC エンドポイントによるサービスのアクセスコントロール」を参照してください。

S3 バケットアクセスのカスタムポリシー

Image Builder は、公開されている S3 バケットを使用して、コンポーネントなどの管理対象リソースを保存し、アクセスします。また、 AWSTOE コンポーネント管理アプリケーションを別の S3 バケットからダウンロードします。環境でAmazon S3のVPCエンドポイントを使用している場合、S3 VPCエンドポイントポリシーでImage Builderが以下のS3バケットにアクセスできるようにする必要があります。バケット名は、 AWS リージョン (#####) とアプリケーション環境 (##) ごとに一意です。Image Builder と は、、prod、preprodおよび のアプリケーション環境 AWSTOE をサポートしますbeta。

- AWSTOE コンポーネントマネージャーバケット :

```
s3://ec2imagebuilder-toe-region-environment
```

例： s3://ec2imagebuilder-toe-us-west-2-prod/*

- Image Builder 管理リソースバケット:

```
s3://ec2imagebuilder-managed-resources-region-environment/components
```

例： s3://ec2imagebuilder-managed-resources-us-west-2-prod/components/*

VPC エンドポイントのポリシーの例

このセクションには、カスタム VPC エンドポイントポリシーの例が含まれています。

Image Builderアクションの一般的なVPCエンドポイントポリシー

次の Image Builder のエンドポイントポリシー例では、Image Builder のイメージとコンポーネントを削除する権限を拒否しています。このポリシー例では、EC2 Image Builderの他のすべてのアクションを実行する権限も付与している。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Action": "imagebuilder:*",
        "Effect": "Allow",
        "Resource": "*"
    },
    {
        "Action": [
            "imagebuilder: DeleteImage"
        ],
        "Effect": "Deny",
        "Resource": "*",
    },
    {
        "Action": [
            "imagebuilder: DeleteComponent"
        ],
        "Effect": "Deny",
        "Resource": "*",
    }]
}
```

組織ごとにアクセスを制限し、管理対象コンポーネントへのアクセスを許可します。

次のエンドポイントポリシーの例は、組織に属する ID とリソースへのアクセスを制限し、Amazon が管理する AWSTOE コンポーネントへのアクセスを提供する方法を示しています。region 、*principal-org-id*、および を組織の値*resource-org-id*に置き換えます。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowRequestsByOrgsIdentitiesToOrgsResources",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalOrgID": "principal-org-id",
          "aws:ResourceOrgID": "resource-org-id"
        }
      }
    },
    {
      "Sid": "AllowAccessToEC2ImageBuilderComponents",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "imagebuilder:GetComponent"
      ],
      "Resource": [
        "arn:aws:imagebuilder:region:aws:component/*"
      ]
    }
  ]
}
```

Amazon S3 バケットアクセスの VPC エンドポイントポリシー

以下の S3 エンドポイントポリシーの例では、Image Builder がカスタムイメージの構築に使用する S3 バケットへのアクセスを提供する方法を示しています。 ##### と ## を組織の値に置き換えてください。アプリケーションの要件に基づいて、その他の必要な権限をポリシーに追加します。

> **Note**
>
> Linux イメージの場合、イメージレシピでユーザーデータを指定しない場合、Image Builder は、イメージのビルドインスタンスとテストインスタンスに Systems Manager エージェントをダウンロードしてインストールするスクリプトを追加します。エージェントをダウンロードするには、Image Builder がビルドリージョンの S3 バケットにアクセスします。Image Builder がビルドインスタンスとテストインスタンスをブートストラップできるようにするには、S3 エンドポイントポリシーに次のリソースを追加します。
> `"arn:aws:s3:::amazon-ssm-region/*"`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowImageBuilderAccessToAppAndComponentBuckets",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::ec2imagebuilder-toe-region-environment/*",
        "arn:aws:s3:::ec2imagebuilder-managed-resources-region-environment/components/
*"
      ]
    }
  ]
}
```

EC2 Image Builder ディストリビューション設定の管理

Image Builder でディストリビューション設定を作成した後は、Image Builder コンソール、Image Builder API、または AWS CLIの imagebuilder コマンドを使用してディストリビューション設定を管理できます。ディストリビューション設定では、以下のアクションを実行できます。

OS ディストリビューション

- 出力 AMI の名前と説明を指定します。
- 所有者のアカウントから AMI を起動することを他の AWS アカウント、組織、OU に許可します。OUs 所有者アカウントには、AMI に関連する料金が請求されます。

> **Note**
>
> AMI をパブリックにするには、起動許可アカウントを `all` に設定します。EC2でAMIを公開する例を参照されたいModifyImageAttribute。

- 宛先リージョン内の指定されたターゲットアカウント、組織、OUs のそれぞれについて、出力 AMI のコピーを作成します。対象となるアカウント、組織、OUs はそれぞれの AMI コピーを所有しており、関連する料金はすべて請求されます。および OUs「組織または AWS Organizations OU と AMI を共有する」を参照してください。 OUs
- AMI を他の の所有者のアカウントにコピーします AWS リージョン。
- VM Image Diskを Amazon Simple Storage Service (Amazon S3) にエクスポートします。詳細については、「出力VMディスクのディストリビューション設定を作成する(AWS CLI)」を参照してください。

コンテナイメージの配布

- Image Builder が配布リージョン内の出力イメージを保存する ECR リポジトリを指定します。

ディストリビューション設定を次の方法で使用して、ターゲットリージョン、アカウント、 AWS Organizations 組織単位 (OUs回、またはパイプラインビルドごとに配信できます。

- 更新されたイメージを指定された地域、アカウント、Organizations、OUs に自動的に配信するには、スケジュールに従って実行される Image Builder パイプラインの配信設定を使用します。
- 新しいイメージを作成して、指定したリージョン、アカウント、Organizations、OUs に配信するには、Image Builder コンソールから [アクション] メニューの [パイプラインの実行] を使用して、Image Builder コンソールから 1 回実行する Image Builder パイプラインの配布設定を使用します。
- 新しいイメージを作成して、指定したリージョン、アカウント、Organizations、OUs に配信するには、次の API アクションまたは AWS CLIの Image Builder コマンドで配布設定を使用します。
  - Image Builder API での「CreateImage」アクション。

  - AWS CLIのcreate-imageコマンド。
- 通常のイメージビルドプロセスの一環として、仮想マシン (VM) イメージディスクをターゲットリージョンの S3 バケットにエクスポートすること。

> **Tip**
>
> 同じ型のリソースが複数にある場合に、割り当てたタグに基づいて特定のリソースをすばやく識別できます。の Image Builder コマンドを使用してリソースにタグを付ける方法の詳細については AWS CLI、このガイドのリソースのタグ付け「」セクションを参照してください。

このトピックでは、ディストリビューション設定を一覧表示、表示、作成する方法について説明します。

内容

- ディストリビューション設定の一覧と表示
- AMI ディストリビューション設定の作成と更新
- コンテナイメージのディストリビューション設定を作成および更新します。
- Image Builder でクロスアカウント AMI ディストリビューションのセットアップ
- Amazon EC2 起動テンプレートを使用するように AMI ディストリビューション設定を設定する

ディストリビューション設定の一覧と表示

このセクションでは、EC2 Image Builderのディストリビューション設定の詳細を確認する方法について説明します。

配信設定の詳細

- ディストリビューション設定を一覧表示する (コンソール)
- ディストリビューション構成の詳細の表示 (コンソール)
- AWS CLIディストリビューションのリスト表示
- ディストリビューション設定の詳細を取得 (AWS CLI)

ディストリビューション設定を一覧表示する (コンソール)

自分のアカウントで作成されたディストリビューション設定の一覧を Image Builder コンソールで確認するには、以下の手順に従います：

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから配信設定を選択します。アカウントで作成されたディストリビューション設定のリストが表示されます
3. 詳細を表示したり、新しいディストリビューション設定を作成したりするには、[設定名]リンクを選択します。ディストリビューション設定の詳細ビューが開きます。

> **Note**
>
> また、設定名の横にあるチェックボックスを選択し、詳細を表示を選択することもできます。

ディストリビューション構成の詳細の表示 (コンソール)

Image Builder コンソールを使用して特定のディストリビューション設定の詳細を表示するには、「ディストリビューション設定を一覧表示する (コンソール)」で説明されている手順を使用して、確認する設定を選択します。

ディストリビューションの詳細ページでは、次のことができます。

- ディストリビューションの設定を削除する Image Builder でのリソースの削除については、「EC2 Image Builder resourcesの削除」を参照してください。
- ディストリビューションの詳細を [編集] します。

AWS CLIディストリビューションのリスト表示

次の例は、 で list-distribution-configurations コマンドを使用してすべてのディストリビューションを AWS CLI 一覧表示する方法を示しています。

```
aws imagebuilder list-distribution-configurations
```

ディストリビューション設定の詳細を取得 (AWS CLI)

次の例は、 で get-distribution-configuration コマンドを使用して、Amazon リソースネーム (ARN) を指定してディストリビューション設定の詳細 AWS CLI を取得する方法を示しています。

```
aws imagebuilder get-distribution-configuration --distribution-configuration-arn
 arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-example-
distribution-configuration
```

AMI ディストリビューション設定の作成と更新

このセクションでは、Image Builder AMI のディストリビューション設定の作成と更新について説明します。

内容

- AMIディストリビューション構成 (コンソール) の作成
- 出力 AMI の配信設定を作成します (AWS CLI)
- AMI ディストリビューション設定の更新 (コンソール)
- EC2の高速起動を有効にしたWindows AMIのディストリビューション設定を作成する (AWS CLI)
- 出力VMディスクのディストリビューション設定を作成する(AWS CLI)
- AMI ディストリビューション設定 (AWS CLI) の更新

AMIディストリビューション構成 (コンソール) の作成

ディストリビューション設定には、出力 AMI 名、暗号化用の特定のリージョン設定、起動許可、出力 AMI を起動できる AWS アカウント、組織、組織単位 (OUs)、ライセンス設定が含まれます。

新しいAMIディストリビューション構成を作成するには

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから配信設定を選択します。アカウントで作成されたディストリビューション設定のリストが表示されます
3. [ディストリビューション設定]パネルの上部にある[ディストリビューション設定を作成]を選択します。
4. イメージタイプセクションで、Amazon マシンイメージ (AMI) 出力タイプを選択します。

5. 一般セクションで、ディストリビューション設定の[名前]とオプションの説明を入力します。
6. [リージョン設定]セクションで、AMI を配布する各リージョンについて次の詳細を入力します。

   a. AMI は、デフォルトで現在のリージョン ([リージョン 1]) に配布されます。[リージョン 1]は配信のソースです。[リージョン 1]の一部の設定は編集できません。追加するどのリージョンでも、[リージョン]ドロップダウンリストからリージョンを選択できます。

      Kms キーは AWS KMS key 、ターゲットリージョン内のイメージの EBS ボリュームを暗号化するために使用される を識別します。これは、ビルドがソースリージョン ([リージョン 1]) のアカウントで作成した元の AMI には適用されないことに注意してください。ビルドの配布フェーズで実行される暗号化は、他のアカウントまたはリージョンに配布されるイメージのみに適用されます。

      アカウントのソースリージョンで作成された AMI の EBS ボリュームを暗号化するには、イメージレシピブロックデバイスマッピング (コンソールの[ストレージ (ボリューム)]) で KMS キーを設定する必要があります。

      Image Builder は、リージョンに指定した[ターゲットアカウント]に AMI をコピーします。

      > **前提条件**
      >
      > アカウントをまたいでイメージをコピーするには、ターゲットリージョンのすべてのターゲットアカウントに `EC2ImageBuilderDistributionCrossAccountRole` ロールを作成し、Ec2ImageBuilderCrossAccountDistributionAccess ポリシー管理ポリシーをロールにアタッチする必要があります。

      [出力 AMI 名] はオプションです。名前を指定すると、最終的に出力される AMI 名には、AMI が構築されたときのタイムスタンプが付加されます。名前を指定しないと、Image Builder はビルドタイムスタンプをレシピ名に追加します。これにより、ビルドごとに一意の AMI 名が保証されます。

      i. AMI 共有を使用すると、指定した AWS プリンシパルに AMI からインスタンスを起動するためのアクセスを許可できます。[AMI 共有] セクションを展開すると、次の詳細を入力できます。

         - 起動許可 – AMI をプライベートに保ち、特定の AWS プリンシパルがプライベート AMI からインスタンスを起動するためのアクセスを許可する場合は、プライベート

を選択します。AMI をパブリックにする場合は、[パブリック]を選択します。プリンAWS シパルは、パブリック AMI からインスタンスを起動できます。

- プリンシパル – インスタンスを起動するために、次のタイプの AWS プリンシパルへのアクセスを許可できます。
  - AWS アカウント – 特定の AWS アカウントへのアクセス権を付与します。
  - [組織単位 (OU)] — OU とそのすべての子エンティティへのアクセスを許可します。子エンティティには OUsと AWS アカウントが含まれます。
  - Organization – AWS Organizationsとそのすべての子エンティティへのアクセスを許可します。子エンティティには OUsと AWS アカウントが含まれます。

  まず、プリンシパルタイプを選択します。次に、アクセスを許可したい AWS プリンシパルの ID をドロップダウンリストの右側のボックスに入力します。異なるタイプの複数の ID を入力できます。

ii. ライセンス設定セクションを展開して、 で作成したライセンス設定を Image Builder イメージ AWS License Manager にアタッチできます。ライセンスコンフィギュレーションには、企業契約の条件に基づくライセンスルールが含まれています。Image Builder には、ベース AMI に関連付けられたライセンス設定が自動的に含まれます。

iii. [起動テンプレート設定]セクションを展開して、作成した AMI からインスタンスを起動するために使用する EC2 起動テンプレートを指定できます。

EC2 起動テンプレートを使用している場合は、ビルドの完了後に最新の AMI ID を含む起動テンプレートの新しいバージョンを作成するように Image Builder に指示できます。起動テンプレートを更新するには、次のように設定を行います。

- [起動テンプレート名] — Image Builder で更新する起動テンプレートの名前を選択します。
- [デフォルトバージョンを設定] — 起動テンプレートのデフォルトバージョンを新しいバージョンに更新するには、このチェックボックスを選択します。

別のローンチテンプレート設定を追加するには、Add launch template configurationを選択します。リージョンあたり最大 5 つの起動テンプレート設定を持つことができます。

b. 別のリージョンのディストリビューション設定を追加するには、[リージョンを追加]を選択します。

7. 完了したらCreate settingsを選択します。

出力 AMI の配信設定を作成します (AWS CLI)

ディストリビューション設定では、出力 AMI の名前と説明を指定し、他の AWS アカウント に AMI の起動を許可し、AMI を他のアカウントにコピーし、AMI を他の AWS リージョンにレプリケートできます。また、AMI を Amazon Simple Storage Service (Amazon S3) にエクスポートしたり、出力 Windows AMI 用に EC2 高速起動を設定したりすることもできます。AMI をパブリックにするには、起動許可アカウントを `all` に設定します。EC2でAMIを公開する例を参照されたいModifyImageAttribute。

次の例では、create-distribution-configuration コマンドを使用して AMI の新しいディストリビューション設定を作成し、 AWS CLIを使用して AMI の新しいディストリビューション設定を作成する方法を示します。

1. CLI 入力 JSON ファイルの作成

   ファイル編集ツールを使って、以下の例のいずれかのキーと、あなたの環境で有効な値を持つJSONファイルを作成 これらの例では AWS アカウント、指定したリージョンに配布する AMI を起動するアクセス許可を持つ 、 AWS Organizations または組織単位 (OU) を定義します。 OUs 次のステップで使用するのファイルに`create-ami-distribution-configuration.json`で名前を付けます。

   Accounts

   この例では、AMI を 2 つのリージョンに配布し、各リージョンに起動権限を持つ AWS アカウント を指定します。

```
{
    "name": "MyExampleAccountDistribution",
    "description": "Copies AMI to eu-west-1, and specifies accounts that can
 launch instances in each Region.",
    "distributions": [
        {
            "region": "us-west-2",
            "amiDistributionConfiguration": {
                "name": "Name {{imagebuilder:buildDate}}",
                "description": "An example image name with parameter
 references",
                "amiTags": {
                    "KeyName": "Some Value"
                },
                "launchPermission": {
```

```
                    "userIds": [
                        "987654321012"
                    ]
                }
            }
        },
        {
            "region": "eu-west-1",
            "amiDistributionConfiguration": {
                "name": "My {{imagebuilder:buildVersion}} image
 {{imagebuilder:buildDate}}",
                "amiTags": {
                    "KeyName": "Some value"
                },
                "launchPermission": {
                    "userIds": [
                        "100000000001"
                    ]
                }
            }
        }
    ]
}
```

Organizations and OUs

この例では、AMI をソースリージョンに配布し、組織と OU の起動権限を指定します。

```
{
  "name": "MyExampleAWSOrganizationDistribution",
  "description": "Shares AMI with the Organization and OU",
  "distributions": [
    {
      "region": "us-west-2",
      "amiDistributionConfiguration": {
        "name": "Name {{ imagebuilder:buildDate }}",
        "launchPermission": {
          "organizationArns": [
            "arn:aws:organizations::123456789012:organization/o-
myorganization123"
          ],
          "organizationalUnitArns": [
```

```
            "arn:aws:organizations::123456789012:ou/o-123example/ou-1234-
myorganizationalunit"
          ]
        }
      }
    }
  ]
}
```

2. 作成したファイルを入力として使用し、次のコマンドを実行します。

```
aws imagebuilder create-distribution-configuration --cli-input-json file://create-ami-distribution-configuration.json
```

> **Note**
>
> - JSON ファイルパスの先頭にfile://ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

詳細については、AWS CLI コマンドリファレンスのcreate-distribution-configurationを参照してください。

AMI ディストリビューション設定の更新 (コンソール)

Image Builder コンソールを使用して AMI ディストリビューション設定を変更できます。更新されたディストリビューション設定は、今後すべての自動および手動パイプラインデプロイに使用されます。ただし、行った変更は、Image Builder がすでに配布しているリソースには適用されません。たとえば、後でディストリビューションから削除するリージョンに AMI を配布した場合、すでに配布されていた AMI は、手動で削除するまでそのリージョンに残ります。

AMIディストリビューション設定の更新

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。

2. ナビゲーションペインから配信設定を選択します。アカウントで作成されたディストリビューション設定のリストが表示されます
3. ディストリビューション設定の詳細を表示したり、更新したりするには、設定名リンクを選択します。ディストリビューション設定の詳細ビューが開きます。

> **Note**
>
> また、設定名の横にあるチェックボックスを選択し、詳細を表示を選択することもできます。

4. ディストリビューション設定を編集するには、[ディストリビューションの詳細]セクションの右上隅にある[編集]を選択します。ディストリビューション設定の[名前]や、[リージョン 1] と表示されるデフォルトの[リージョン] など、一部のフィールドはロックされています。ディストリビューション設定の詳細については、「AMIディストリビューション構成（コンソール）の作成」を参照してください。
5. 完了したら、変更を保存 を選択します。

EC2の高速起動を有効にしたWindows AMIのディストリビューション設定を作成する（AWS CLI）

次の例は、 create-distribution-configuration コマンドを使用して、 から AMI 用に EC2 Fast Launch が設定されたディストリビューション設定を作成する方法を示しています AWS CLI。

> **Note**
>
> Image Builder は、EC2 Fast Launch が事前に有効になっている AMIsクロスアカウントディストリビューションをサポートしていません。EC2 高速起動は、送信先アカウントから有効にする必要があります。

1. CLI 入力 JSON ファイルの作成

   ファイル編集ツールを使って、以下の例のようなキーと、あなたの環境で有効な値を持つJSONファイルを作成

   この例では、parallel 起動の最大数がターゲットリソース数よりも多いため、すべてのターゲットリソースのインスタンスを同時に起動します。次のステップで示すコマンドの例では、このファイルは「`ami-dist-config-win-fast-launch.json`」と名付けられています。

```
{
"name": "WinFastLaunchDistribution",
"description": "An example of Windows AMI EC2 Fast Launch settings in the
 distribution configuration.",
"distributions": [
    {
        "region": "us-west-2",
        "amiDistributionConfiguration": {
            "name": "Name {{imagebuilder:buildDate}}",
            "description": "Includes Windows AMI EC2 Fast Launch settings.",
            "amiTags": {
                "KeyName": "Some Value"
            }
        },
        "fastLaunchConfigurations": [{
            "enabled": true,
            "snapshotConfiguration": {
                "targetResourceCount": 5
            },
            "maxParallelLaunches": 6,
            "launchTemplate": {
                "launchTemplateId": "lt-0ab1234c56d789012",
                "launchTemplateVersion": "1"
             }
        }],
        "launchTemplateConfigurations": [{
                    "launchTemplateId": "lt-0ab1234c56d789012",
                    "setDefaultVersion": true
          }]
    }]
}
```

> **Note**
>
> launchTemplate のlaunchTemplateIdの代わりにlaunchTemplateNameを指定することはできるが、名前とIDの両方を指定することはできない。

2. 作成したファイルを入力として使用し、次のコマンドを実行します。

```
aws imagebuilder create-distribution-configuration --cli-input-json file://ami-dist-config-win-fast-launch.json
```

> **Note**
>
> - JSON ファイルパスの先頭にfile://ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

詳細については、AWS CLI コマンドリファレンスのcreate-distribution-configurationを参照してください。

出力VMディスクのディストリビューション設定を作成する(AWS CLI)

次の例は、create-distribution-configurationコマンドを使用して、イメージをビルドするたびに VM イメージディスクを Amazon S3 にエクスポートするディストリビューション設定を作成する方法を示しています。

1. CLI 入力 JSON ファイルの作成

   AWS CLIで使うcreate-distribution-configurationコマンドを効率化できる。そのためには、コマンドに渡すすべてのエクスポート設定を含む JSON ファイルを作成します。

> **Note**
>
> JSON ファイル内のデータ値の命名規則は、Image Builder API アクションリクエストパラメータに指定されているパターンに従います。APIコマンドのリクエストパラメータを確認するには、EC2 Image Builder API ReferenceのCreateDistributionConfigurationコマンドを参照してください。

データ値をコマンドラインパラメータとして指定するには、AWS CLI コマンドリファレンスで指定されているパラメータ名を参照してください。 オプションとして、イメージビルダーcreate-distribution-configurationコマンドに指定します。

以下は、この例のs3ExportConfigurationJSONオブジェクトに指定するパラメーターの概要であります。

- RoleName (文字列、必須) — S3 バケットにイメージをエクスポートするための VM Import/Export 権限を付与するロールの名前。
- diskImageFormat (文字列、必須) — 更新されたディスクイメージを次のいずれかのサポートされている形式にエクスポートします。
  - 仮想ハードディスク（VHD） - Citrix XenおよびMicrosoft Hyper-V仮想化製品と互換性があります。
  - ストリーム最適化ESX仮想マシンディスク（VMDK） - VMware ESXおよびVMware vSphereバージョン4、5、6と互換性があります。
  - Raw - Rawフォーマット。
- S3Bucket (文字列、必須) — VM の出力ディスクイメージを保存する S3 バケット。

export-vm-disks.json という名前でファイルを保存します。create-distribution-configuration コマンドではファイル名を使用します。

```
{
    "name": "example-distribution-configuration-with-vm-export",
    "description": "example",
    "distributions": [
        {
            "region": "us-west-2",
            "amiDistributionConfiguration": {
             "description": "example-with-vm-export"

            },
            "s3ExportConfiguration": {
                "roleName": "vmimport",
                "diskImageFormat": "RAW",
                "s3Bucket": "vm-bucket-export"
            }
        }],
```

```
    "clientToken": "abc123def4567ab"
}
```

2. 作成したファイルを入力として使用し、次のコマンドを実行します。

```
aws imagebuilder create-distribution-configuration --cli-input-json file://export-vm-disks.json
```

> **Note**
>
> - JSON ファイルパスの先頭にfile://ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

詳細については、AWS CLI コマンドリファレンスのcreate-distribution-configurationを参照してください。

AMI ディストリビューション設定 (AWS CLI) の更新

次の例では、update-distribution-configurationコマンドを使用して、 AWS CLIコマンドを使用してAMIのディストリビューション設定を更新しています。

1. CLI 入力 JSON ファイルの作成

   お好みのファイル編集ツールを使って、以下の例に示すキーと、あなたの環境で有効な値を加えたJSONファイルを作成します。この例では、update-ami-distribution-configuration.jsonという名前のファイルを使用します。

```
{
    "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/update-ami-distribution-
configuration.json",
    "description": "Copies AMI to eu-west-2, and specifies accounts that can launch
 instances in each Region.",
    "distributions": [
      {
```

```
            "region": "us-west-2",
            "amiDistributionConfiguration": {
                "name": "Name {{imagebuilder:buildDate}}",
                "description": "An example image name with parameter references",
                "launchPermissions": {
                    "userIds": [
                        "987654321012"
                    ]
                }
            }
        },
        {
            "region": "eu-west-2",
            "amiDistributionConfiguration": {
                "name": "My {{imagebuilder:buildVersion}} image
 {{imagebuilder:buildDate}}",
                "tags": {
                    "KeyName": "Some value"
                },
                "launchPermissions": {
                    "userIds": [
                        "100000000001"
                    ]
                }
            }
        }
    ]
}
```

2. 作成したファイルを入力として使用し、次のコマンドを実行します。

```
aws imagebuilder update-distribution-configuration --cli-input-json file://update-ami-distribution-configuration.json
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照する

ためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

詳細については、AWS CLI コマンドリファレンスのupdate-distribution-configurationを参照してください。ディストリビューション設定リソースのタグを更新するには、「リソースのタグ付け」セクションを参照してください。

コンテナイメージのディストリビューション設定を作成および更新します。

このセクションでは、Image Builder AMI のディストリビューション設定の作成と更新について説明します。

内容

Image Builder コンテナイメージ (AWS CLI) の配布設定を作成する

ディストリビューション設定を使用すると、出力コンテナイメージの名前と説明を指定し、コンテナイメージを他の AWS リージョンにレプリケートできます。ディストリビューション設定リソースと各リージョン内のコンテナイメージに別々のタグを適用することもできます。

1. CLI 入力 JSON ファイルの作成

   お好みのファイル編集ツールを使って、以下の例に示すキーと、あなたの環境で有効な値を加えたJSONファイルを作成します。この例では、`create-container-distribution-configuration.json`という名前のファイルを使用します。

```
{
 "name": "distribution-configuration-name",
 "description": "Distributes container image to Amazon ECR repository in two
 regions.",
 "distributions": [
     {
         "region": "us-west-2",
         "containerDistributionConfiguration": {
```

```
            "description": "My test image.",
            "targetRepository": {
                "service": "ECR",
                "repositoryName": "testrepo"
            },
            "containerTags": ["west2", "image1"]
        }
    },
    {
        "region": "us-east-1",
        "containerDistributionConfiguration": {
            "description": "My test image.",
            "targetRepository": {
                "service": "ECR",
                "repositoryName": "testrepo"
            },
           "containerTags": ["east1", "imagedist"]
        }
    }
 ],
 "tags": {
    "DistributionConfigurationTestTagKey1":
 "DistributionConfigurationTestTagValue1",
    "DistributionConfigurationTestTagKey2":
 "DistributionConfigurationTestTagValue2"
 }
}
```

2. 作成したファイルを入力として使用し、次のコマンドを実行します。

```
aws imagebuilder create-distribution-configuration --cli-input-json file://create-container-distribution-configuration.json
```

> **Note**
>
> - JSON ファイルパスの先頭にfile://ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

詳細については、AWS CLI コマンドリファレンスのcreate-distribution-configurationを参照してください。

コンテナイメージ (AWS CLI) のディストリビューション設定を更新します。

以下の例では、update-distribution-configurationコマンドを使用して、 AWS CLIコマンドを使用して、コンテナイメージのディストリビューション設定を更新する方法を示しています。各リージョン内のコンテナイメージのタグを更新することもできます。

1. CLI 入力 JSON ファイルの作成

   お好みのファイル編集ツールを使って、以下の例に示すキーと、環境で有効な値を含むJSONファイルを作成します。この例では、update-container-distribution-configuration.jsonという名前のファイルを使用します。

```
{
    "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/update-container-distribution-
configuration.json",
 "description": "Distributes container image to Amazon ECR repository in two
 regions.",
 "distributions": [
     {
         "region": "us-west-2",
         "containerDistributionConfiguration": {
             "description": "My test image.",
             "targetRepository": {
                 "service": "ECR",
                 "repositoryName": "testrepo"
             },
             "containerTags": ["west2", "image1"]
         }
     },
     {
         "region": "us-east-2",
         "containerDistributionConfiguration": {
             "description": "My test image.",
             "targetRepository": {
                 "service": "ECR",
                 "repositoryName": "testrepo"
```

```
            },
            "containerTags": ["east2", "imagedist"]
        }
    }
 ]
}
```

2. 作成したファイルを入力として使用し、次のコマンドを実行します。

```
aws imagebuilder update-distribution-configuration --cli-input-json file://update-container-distribution-configuration.json
```

> **Note**
>
> - JSON ファイルパスの先頭にfile://ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

詳細については、AWS CLI コマンドリファレンスのupdate-distribution-configurationを参照してください。ディストリビューション設定リソースのタグを更新するには、「リソースのタグ付け」セクションを参照してください。

Image Builder でクロスアカウント AMI ディストリビューションのセットアップ

このセクションでは、指定した他のアカウントに Image Builder AMI を配信するためのディストリビューション設定を行う方法について説明します。

その後、宛先アカウントは、必要に応じて AMI を起動または変更できます。

> **Note**
>
> AWS CLI このセクションの コマンドの例は、イメージレシピとインフラストラクチャ設定 JSON ファイルを以前に作成したことを前提としています。イメージレシピの JSON ファイルを作成するには、「でイメージレシピを作成する AWS CLI」を参照してください。インフ

ラストラクチャー設定用の JSON ファイルを作成するには、「インフラストラクチャ構成を作成します。」を参照してください。

前提条件

ターゲットアカウントが Image Builder イメージからインスタンスを正常に起動できるようにするには、すべてのリージョンのすべての宛先アカウントに適切な権限を設定する必要があります。

AWS Key Management Service（ AWS KMS) を使用して AMI を暗号化する場合は、新しいイメージの暗号化に使用される AWS KMS key アカウントの を設定する必要があります。

Image Builder が暗号化された AMI のクロスアカウント配信を実行すると、ソースアカウントのイメージが復号化されてターゲットリージョンにプッシュされ、そのリージョンに指定されたキーを使用して再暗号化されます。Image Builder はターゲットアカウントに代わって動作し、宛先リージョンで作成した IAM ロールを使用するため、そのアカウントはソースリージョンとターゲットリージョンの両方のキーにアクセスできる必要があります。

暗号化キー

AWS KMSを使用してイメージを暗号化する場合、以下の前提条件が必要です。IAM の前提条件として以下で説明します。

ソースアカウント要件

- AMI を構築して配布するすべてのリージョンのアカウントに KMS キーを作成します。既存のグループを使用することもできます。
- 宛先アカウントがキーを使用できるように、これらすべてのキーのキーポリシーを更新してください。

送信先アカウント要件

- 暗号化された AMI を配布するために、必要なアクションをロールが実行できるようにするインラインポリシーを `EC2ImageBuilderDistributionCrossAccountRole` に追加します。IAM の設定手順については、「IAM ポリシー前提条件」セクションを参照してください。

を使用したクロスアカウントアクセスの詳細については AWS KMS、「 AWS Key Management Service デベロッパーガイド」の「他のアカウントのユーザーに KMS キーの使用を許可する」を参照してください。

以下のように、イメージレシピに暗号化キーを指定します。

- Image Builder コンソールを使用している場合は、レシピの ストレージ (ボリューム) セクションにある 暗号化 (KMS エイリアス) ドロップダウンリストから暗号化キーを選択します。
- CreateImageRecipe API アクション、または の create-image-recipe コマンドを使用している場合は AWS CLI、JSON 入力の の `ebs`セクション`blockDeviceMappings`でキーを設定します。

  次の JSON スニペットは、イメージレシピの暗号化設定を示しています。暗号化キーを提供するだけでなく、`encrypted` フラグを `true` に設定する必要もあります。

```
{
 ...
 "blockDeviceMappings": [
 {
  "deviceName": "Example root volume",
  "ebs": {
   "deleteOnTermination": true,
   "encrypted": true,
   "iops": 100,
   "kmsKeyId": "image-owner-key-id",
   ...
  },
  ...
 }],
 ...
}
```

IAM ポリシー

AWS Identity and Access Management (IAM) でクロスアカウントディストリビューションのアクセス許可を設定するには、次の手順に従います。

1. 複数のアカウントに分散されている Image Builder AMI を使用するには、宛先アカウントの所有者が自分の `EC2ImageBuilderDistributionCrossAccountRole` というアカウントで新しい IAM ロールを作成する必要があります。
2. アカウント間の配信を有効にするには、「Ec2ImageBuilderCrossAccountDistributionAccess ポリシー」をロールにアタッチする必要があります。マネージドポリシーの詳細については、AWS Identity and Access Management IAM ユーザーガイドの「マネージドポリシーとインラインポリシー」を参照してください。

3. ソースアカウント ID が、宛先アカウントの IAM ロールにアタッチされた信頼ポリシーに追加されていることを確認します。信頼ポリシーの詳細については、AWS Identity and Access Management User GuideのResource-Based Policiesを参照してください。

4. 配布する AMI が暗号化されている場合、宛先アカウントの所有者は、KMS キーを使用できるように、アカウントで EC2ImageBuilderDistributionCrossAccountRole に次のインラインポリシーを追加する必要があります。Principal セクションにはアカウント番号が含まれています。これにより、Image Builder は、 AWS KMS を使用して各リージョンの適切なキーで AMI を暗号化および復号するときに、Image Builder に代わって動作できるようになります。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowRoleToPerformKMSOperationsOnBehalfOfTheDestinationAccount",
   "Effect": "Allow",
   "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:ReEncrypt*",
    "kms:GenerateDataKey*",
    "kms:DescribeKey",
    "kms:CreateGrant",
    "kms:ListGrants",
    "kms:RevokeGrant"
   ],
   "Resource": "*"
  }
 ]
}
```

インラインポリシーの詳細については、AWS Identity and Access Management ユーザーガイドの「インラインポリシー」を参照してください。

5. launchTemplateConfigurationsを使用して Amazon EC2 起動テンプレートを指定する場合は、各宛先アカウントの EC2ImageBuilderDistributionCrossAccountRole に次のポリシーを追加する必要もあります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:CreateLaunchTemplateVersion",
                "ec2:ModifyLaunchTemplate"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeLaunchTemplates"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "arn:aws:ec2:*:*:launch-template/*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "EC2 Image Builder"
                }
            }
        }
    ]
}
```

クロスアカウント配信の制限

Image Builder イメージを複数のアカウントに配布する場合、いくつかの制限があります。

- 宛先アカウントは、宛先リージョンごとに同時 AMI コピーが 50 個に制限されています。
- 準仮想化 (PV) 仮想化 AMI を別のリージョンにコピーする場合、コピー先のリージョンが PV 仮想化 AMI をサポートしている必要があります。詳細については、「Linux AMI 仮想化タイプ」を参照してください。

- 暗号化されていないスナップショットの暗号化されたコピーを作成することはできません。KmsKeyId パラメータに AWS Key Management Service（AWS KMS）カスタマーマネージドキーを指定しない場合、Image BuilderはAmazon Elastic Block Store（Amazon EBS）のデフォルトキーを使用します。詳細については、Amazon Elastic Compute Cloudユーザーガイドの「Amazon EBS Encryption」を参照してください。

詳細については、EC2 Image Builder API リファレンスCreateDistributionConfigurationの「」を参照してください。

Image Builder AMI のクロスアカウント配信の設定 (コンソール)

このセクションでは、 AWS Management Consoleを使用して Image Builder AMI のクロスアカウント配信用のディストリビューション設定を作成および設定する方法について説明します。クロスアカウント配信を設定するには、特定の IAM 権限が必要です。続行する前に、このセクションの「前提条件」を完了する必要があります。

Image Builder コンソールでディストリビューション設定を作成するには、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから配信設定を選択します。これにより、アカウントで作成されたディストリビューション設定のリストが表示されます。
3. ディストリビューション設定ページの上部で、ディストリビューション設定を作成を選択します。これにより、ディストリビューション設定の作成 ページが表示されます。
4. イメージタイプ セクションで、出力タイプ として Amazon マシンイメージ (AMI) を選択します。これはデフォルトの設定です。
5. 全般セクションに、作成する配布設定リソースの名前を入力します（必須）。
6. リージョン設定で、選択したリージョンのターゲットアカウントにAMIを配布したい12桁のアカウントIDを入力し、Enterを押す。これにより正しい形式が確認され、入力したアカウント ID がボックスの下に表示されます。このプロセスを繰り返して、さらにアカウントを追加します。

   入力したアカウントを削除するには、アカウント ID の右に表示される X を選択します。

   各リージョンの出力 AMI 名を入力します。
7. 必要な追加設定を続けて指定し、設定の作成 を選択して新しい配布設定リソースを作成します。

Image Builder AMI (AWS CLI) のクロスアカウント配信の設定

このセクションでは、ディストリビューション設定ファイルを設定し、 の create-image コマンドを使用して Image Builder AMI AWS CLI を構築してアカウント間で配布する方法について説明します。

クロスアカウント配信を設定するには、特定の IAM 権限が必要です。create-imageコマンドを実行する前に、このセクションの前提条件を完了する必要があります。

1. ディストリビューション設定ファイルを設定します。

   の create-image コマンドを使用して、別のアカウントに配布される Image Builder AMI AWS CLI を作成する前に、AmiDistributionConfiguration設定でターゲットアカウント IDs を指定する DistributionConfiguration JSON 構造を作成する必要があります。ソースリージョンで、少なくとも 1 つのAmiDistributionConfigurationエントリを指定する必要があります。

   次の create-distribution-configuration.json というサンプルファイルは、ソースリージョンでのクロスアカウントイメージ配信の設定を示しています。

```
{
 "name": "cross-account-distribution-example",
 "description": "Cross Account Distribution Configuration Example",
 "distributions": [
  {
   "amiDistributionConfiguration": {
    "targetAccountIds": ["123456789012", "987654321098"],
    "name": "Name {{ imagebuilder:buildDate }}",
    "description": "ImageCopy Ami Copy Configuration"
   },
   "region": "us-west-2"
  }
 ]
}
```

2. ディストリビューション設定を作成します。

   の create-distribution-configuration コマンドを使用して Image Builder ディストリビューション設定リソースを作成するには AWS CLI、 コマンドで次のパラメータを指定します。

   - ポリシーの名前を --name パラメータに入力します。

- `--cli-input-json` パラメータで作成したディストリビューション設定 JSON ファイルを添付します。

```
aws imagebuilder create-distribution-configuration --name my distribution name --
cli-input-json file://create-distribution-configuration.json
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

また、`--distributions`パラメーターを使用して、コマンドで直接JSONを指定することもできます。

Amazon EC2 起動テンプレートを使用するように AMI ディストリビューション設定を設定する

ターゲットアカウントとリージョンでの Image Builder AMI の一貫した起動環境を確保するために、`launchTemplateConfigurations` を使用してディストリビューション設定で Amazon EC2 起動テンプレートを指定できます。`launchTemplateConfigurations` は配布プロセス中に存在する場合、Image Builder は、テンプレートの元の設定とビルドの新しい AMI ID をすべて含む起動テンプレートの新しいバージョンを作成します。起動テンプレートを使用して EC2 インスタンスを起動の詳細については、ターゲットのオペレーティングシステムに応じて、以下のいずれかのリンクを参照してください。

- 起動テンプレートからLinuxインスタンスを起動する
- 起動テンプレートからWindowsインスタンスを起動する

> **Note**
>
> イメージに Windows Fast Launch を有効にする起動テンプレートを含める場合、Image Builder がユーザーに代わって Windows Fast Launch を有効にするには、起動テンプレートに次のタグを含める必要があります。
>
> `CreatedBy: EC2 Image Builder`

Amazon EC2 起動テンプレートを使用するように AMI ディストリビューション設定を設定する

出力 AMI に起動テンプレートを提供するには、コンソールで次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから配信設定を選択します。これにより、アカウントで作成されたディストリビューション設定のリストが表示されます。
3. ディストリビューション設定ページの上部で、ディストリビューション設定を作成を選択します。ディストリビューション設定の作成 ページが開きます。
4. イメージタイプセクションで、Amazon マシンイメージ (AMI) 出力タイプを選択します。これはデフォルトの設定です。
5. 全般セクションに、作成する配布設定リソースの名前を入力します (必須) 。
6. リージョン設定 で、リストから EC2 起動テンプレートの名前を選択します。アカウントに起動テンプレートがない場合は、Create new Launch template を選択すると、EC2 ダッシュボードに 起動テンプレート が開きます。

   デフォルトバージョンを設定チェックボックスを選択して、起動テンプレートのデフォルトバージョンを、Image Builder が出力 AMI で作成する新しいバージョンに更新します。

   選択したリージョンに別の起動テンプレートを追加するには、起動テンプレート設定を追加を選択します。

   起動テンプレートを削除するには、削除 を選択します。
7. 必要な追加設定を続けて指定し、設定の作成 を選択して新しい配布設定リソースを作成します。

AMI ディストリビューション設定 (AWS CLI) に Amazon EC2 起動テンプレートを追加します

このセクションでは、起動テンプレートで配布設定ファイルを構成し、 AWS CLI のcreate-imageコマンドを使用してImage Builder AMIとそれを使用する起動テンプレートの新バージョンをビルドして配布する方法について説明します。

1. ディストリビューション設定ファイルを設定します。

   起動テンプレートを使用して Image Builder AMI を作成する前に、 を使用して AWS CLI、launchTemplateConfigurations設定を指定するディストリビューション設定 JSON 構造を作成する必要があります。ソースリージョンで、少なくとも 1 つの launchTemplateConfigurations エントリを指定する必要があります。

   次のcreate-distribution-config-launch-template.jsonというサンプルファイルは、ソースリージョンでの起動テンプレート設定で考えられるいくつかのシナリオを示しています。

```
{
    "name": "NewDistributionConfiguration",
    "description": "This is just a test",
    "distributions": [
        {
            "region": "us-west-2",
            "amiDistributionConfiguration": {
                "name": "test-{{imagebuilder:buildDate}}-
{{imagebuilder:buildVersion}}",
                "description": "description"
            },
            "launchTemplateConfigurations": [
                {
                    "launchTemplateId": "lt-0a1bcde2fgh34567",
                    "accountId": "935302948087",
                    "setDefaultVersion": true
                },
                {
                    "launchTemplateId": "lt-0aaa1bcde2ff3456"
                },
                {
                    "launchTemplateId": "lt-12345678901234567",
                    "accountId": "123456789012"
                }
```

```
            ]
        }
    ],
    "clientToken": "clientToken1"
}
```

2. ディストリビューション設定を作成します。

   の create-distribution-configuration コマンドを使用して Image Builder ディストリビューション設定リソースを作成するには AWS CLI、 コマンドで次のパラメータを指定します。

   - ポリシーの名前を `--name` パラメータに入力します。
   - `--cli-input-json` パラメータで作成したディストリビューション設定 JSON ファイルを添付します。

```
aws imagebuilder create-distribution-configuration --name my distribution name--
cli-input-json file://create-distribution-config-launch-template.json
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

また、`--distributions`パラメーターを使用して、コマンドで直接JSONを指定することもできます。

EC2 Image Builder イメージのライフサイクルポリシーの管理

カスタムイメージを作成する場合、それらのイメージが古くなる前に廃止する計画を立てることが重要です。Image Builder パイプラインでは、アップデートとセキュリティパッチを自動的に適用できます。ただし、ビルドするたびに、イメージの新しいバージョンと、それによって配布されるすべての関連リソースが作成されます。以前のバージョンは、手動で削除するか、タスクを実行するスクリプトを作成するまでアカウントに残ります。

Image Builder のライフサイクル管理ポリシーを使用すると、古くなったイメージやそれに関連するリソースの廃止、無効化、削除のプロセスを自動化できます。関連リソースには、 全体で他の AWS アカウント、組織、組織単位 (OUsに配布した出力イメージを含めることができます AWS リージョン。ライフサイクルプロセスの各ステップをいつどのように実行するか、またどのステップをポリシーに含めるかについてのルールを定義します。

自動ライフサイクル管理の利点

自動ライフサイクル管理の全体的な利点には、次のようなものがあります。

- イメージと関連リソースを自動的に廃止することで、カスタムイメージのライフサイクル管理を簡素化します。
- 古いイメージを使用して新しいインスタンスを起動することによるコンプライアンスリスクの防止に役立ちます。
- 古いイメージを削除することで、イメージインベントリを最新の状態に保ちます。
- 削除したイメージの関連リソースをオプションで削除することで、ストレージとデータ転送のコストを削減できます。

コスト削減の実現

カスタム EC2 Image Builder を使用してカスタム AMI またはコンテナイメージを作成してもコストはかかりません。ただし、このプロセスで使用される他のサービスには標準価格が適用されます。未使用のイメージや古いイメージ、および関連するリソースを から削除すると AWS アカウント、次の方法で時間とコスト削減を実現できます。

- 未使用または古いイメージにもパッチを適用しない場合、既存のイメージにパッチを適用するのにかかる時間を短縮できます。
- 削除した AMI イメージリソースについては、配布された AMI とそれに関連するスナップショットも削除するように選択できます。この方法により、スナップショットの保存コストを節約できます。
- 削除するコンテナイメージリソースについては、基になるリソースを削除するように選択できます。この方法により、ECR リポジトリに保存されている Docker イメージの Amazon ECR ストレージコストとデータ転送速度を節約できます。

> **Note**
>
> Image Builder では、Auto Scaling グループや起動テンプレートなど、考えられるすべてのダウンストリームの依存関係に対する潜在的な影響を評価することはできません。ポリシーアクションを設定するときは、イメージのダウンストリームの依存関係を考慮する必要があります。

内容

EC2 Image Builder イメージのライフサイクル管理の前提条件

イメージリソースの EC2 Image Builder ライフサイクル管理ポリシーとルールを定義する前に、次の前提条件を満たす必要があります。

- Image Builder にライフサイクルポリシーを実行する権限を付与する IAM ロールを作成します。ロールを作成するには、「Image Builder ライフサイクル管理用の IAM ロールを作成する」を参照してください。
- 複数のアカウントに分散されていた関連リソースの IAM ロールを移行先アカウントで作成します。このロールは、関連するリソースの送信先アカウントでライフサイクルアクションを実行する権限を Image Builder に付与します。ロールを作成するには、「Image Builder クロスアカウントライフサイクル管理用の IAM ロールを作成する」を参照してください。

  > **Note**
  >
  > 出力 AMI の起動権限を付与した場合、この前提条件は適用されません。起動権限があれば、共有したアカウントは共有 AMI から起動されたインスタンスを所有しますが、すべての AMI リソースはアカウントに残ります。

- コンテナイメージの場合、以下のタグを ECR リポジトリに追加して、Image Builder がリポジトリに保存されているコンテナイメージに対してライフサイクルアクションである `LifecycleExecutionAccess: EC2 Image Builder` を実行するためのアクセス権を付与する必要があります。

Image Builder ライフサイクル管理用の IAM ロールを作成する

Image Builder にライフサイクルポリシーを実行する権限を付与するには、まず Image Builder がライフサイクルアクションを実行するために使用する IAM ロールを作成する必要があります。次の手順に従って、権限を付与するサービスロールを作成します。

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで、[ロール] を選択します。
3. [ロールを作成] を選択します。これにより、[信頼されたエンティティを選択] のプロセスの最初のステップが開いて、ロールを作成します。
4. [信頼されたエンティティタイプ] に、[カスタム信頼ポリシー] オプションを選択します。
5. 以下の JSON 信頼ポリシーをコピーして、[カスタム信頼ポリシー] のテキスト領域に貼り付け、サンプルテキストと置き換えます。この信頼ポリシーにより、Image Builder はライフサイクルアクションを実行するために作成したロールを引き継ぐことができます。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "imagebuilder.amazonaws.com"
                ]
            }
        }
    ]
}
```

6. リストから管理ポリシー [EC2ImageBuilderLifecycleExecutionPolicy] を選択し、[次へ] を選択します。これにより、[名前、確認、および作成] ページが開きます。

> **Tip**
>
> `image` をフィルターして結果を効率化します。

7. [Role name] (ロール名) に入力します。

8. 設定を確認したら、[ロールを作成] を選択します。

Image Builder クロスアカウントライフサイクル管理用の IAM ロールを作成する

Image Builder に関連リソースの宛先アカウントでライフサイクルアクションを実行する権限を付与するには、まず、Image Builder がそれらのアカウントでライフサイクルアクションを実行するために使用する IAM ロールを作成する必要があります。送信先アカウントでロールを作成する必要があります。

次の手順に従って、送信先アカウントの権限を付与するサービスロールを作成します。

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. ナビゲーションペインで、[ロール] を選択します。
3. [ロールを作成] を選択します。これにより、[信頼されたエンティティを選択] のプロセスの最初のステップが開いて、ロールを作成します。
4. [信頼されたエンティティタイプ] に、[カスタム信頼ポリシー] オプションを選択します。
5. 以下の JSON 信頼ポリシーをコピーして、[カスタム信頼ポリシー] のテキスト領域に貼り付け、サンプルテキストと置き換えます。この信頼ポリシーにより、Image Builder はライフサイクルアクションを実行するために作成したロールを引き継ぐことができます。

> **Note**
>
> Image Builder が送信先アカウントでこのロールを使用して、複数のアカウントに分散されていた関連リソースを処理する場合、Image Builder は送信先アカウントの所有者に代わって動作します。信頼ポリシー`aws:SourceAccount`で として AWS アカウント 設定する は、Image Builder がそれらのリソースを配布したアカウントです。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "imagebuilder.amazonaws.com"
                ]
            },
```

```
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "444455556666"
                },
                "StringLike": {
                    "aws:SourceArn": "arn:*:imagebuilder:*:*:image/*/*/*"
                }
            }
        }
    ]
}
```

6. リストから管理ポリシー [EC2ImageBuilderLifecycleExecutionPolicy] を選択し、[次へ] を選択します。これにより、[名前、確認、および作成] ページが開きます。

> **Tip**
>
> image をフィルターして結果を効率化します。

7. Ec2ImageBuilderCrossAccountLifecycleAccess を [ロール名] として入力します。

> **Important**
>
> Ec2ImageBuilderCrossAccountLifecycleAccess は、このロールの名前でなければなりません。

8. 設定を確認したら、[ロールを作成] を選択します。

EC2 Image Builder イメージリソースのライフサイクル管理ポリシー

イメージライフサイクルポリシーを使用すると、古くなったイメージや関連リソースを廃止、無効化、削除するプロセスを通じて、古くなったイメージや関連リソースを廃棄するリソース管理戦略を定義できます。このセクションでは、ポリシーを一覧表示する方法、ポリシーの詳細を表示する方法、および AMI とコンテナイメージの新しいポリシーを作成する方法を示します。

内容

- ライフサイクルポリシーの作成

Image Builder イメージリソースのライフサイクル管理ポリシーを一覧表示する

のライフサイクルポリシーリストページのキー詳細列を含むイメージライフサイクル管理ポリシーのリスト、または Image Builder API AWS Management Console、 SDKs、または のコマンドまたはアクションを取得できます AWS CLI。

以下の方法のいずれかを使用して、 AWS アカウントの Image Builder イメージライフサイクルのポリシーリソースを一覧表示できます。API アクションについては、EC2 Image Builder API リファレンスListLifecyclePoliciesの「」を参照してください。関連する SDK リクエストについては、同じページの「関連項目」リンクを参照してください。

AWS Management Console

既存のポリシーに関する次の詳細がコンソールに表示されます。任意の列を選択して、結果のソート順序を変更できます。ポリシーリストは、最初は [ポリシー名] でソートされます。現在のソート順序の列名は太字です。

結果が複数ページある場合は、パネルの右上隅にあるページング矢印がアクティブになります。検索バーを使用して、ポリシー名、ポリシーステータス、出力イメージタイプ、およびイメージリソース ARN で結果をフィルタリングできます。

- [ポリシー名] – ポリシーの名前。
- [ポリシーステータス] — ポリシーがアクティブか非アクティブか。
- [タイプ] — 新しいイメージバージョン (AMI またはコンテナイメージ) を作成したときに Image Builder が配信する出力のタイプ。
- [最終実行日] — ライフサイクルポリシーが最後に実行された時間。
- [作成日] — ライフサイクルポリシーの作成時のタイムスタンプ。
- [ARN] – ライフサイクルポリシーの Amazon リソースネーム (ARN)。

でライフサイクルポリシーを一覧表示するには AWS Management Console、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。

2. ナビゲーションペインから [ライフサイクルポリシー] を選択します。これにより、アカウント内のイメージライフサイクルポリシーのリストが表示されます。

使用可能なアクション

[ライフサイクルポリシー] のリストページから、ライフサイクルポリシーに対して次のアクションを実行することもできます。

新しいイメージライフサイクルポリシーを作成するには、[ライフサイクルポリシーを作成] を選択します。ポリシーを作成する方法については、「ライフサイクルポリシーの作成」を参照してください。

以下のすべてのアクションでは、最初にポリシーを選択する必要があります。ポリシーを選択するには、[ポリシー名] の横にあるチェックボックスをオンにします。

- ポリシーを無効または有効にするには、[アクション] メニューから [ポリシーを無効にする] または [ポリシーを有効にする] を選択します。
- ポリシーを変更するには、[アクション] メニューから [ポリシーを編集] を選択します。
- ポリシーを削除するには、[アクション] メニューから [ポリシーを削除] を選択します。
- 選択したポリシーをベースライン設定に使用する新しいポリシーを作成するには、[アクション] メニューから [ポリシーのクローンを作成] を選択します。

AWS CLI

次のコマンド例は、 を使用して特定の のイメージライフサイクルポリシーを AWS CLI 一覧表示する方法を示しています AWS リージョン。このコマンドで使用できるパラメータとオプションの詳細については、 AWS CLI コマンドリファレンスの list-lifecycle-policies コマンドを参照してください。

例:

```
aws imagebuilder list-lifecycle-policies \
--region us-west-1
```

出力:

```
{
    "lifecyclePolicySummaryList": [
        {
```

```
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:lifecycle-policy/
sample-lifecycle-policy1",
            "name": "sample-lifecycle-policy1",
            "status": "DISABLED",
            "executionRole": "arn:aws:iam::111122223333:role/sample-lifecycle-role",
            "resourceType": "AMI_IMAGE",
            "dateCreated": "2023-11-07T14:57:01.603000-08:00",
            "tags": {}
        },
        {
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:lifecycle-policy/
sample-lifecycle-policy2",
            "name": "sample-lifecycle-policy2",
            "status": "ENABLED",
            "executionRole": "arn:aws:iam::111122223333:role/sample-lifecycle-role",
            "resourceType": "AMI_IMAGE",
            "dateCreated": "2023-09-06T10:43:21.436000-07:00",
            "dateLastRun": "2023-11-13T04:43:46.106000-08:00",
            "tags": {}
        },
        {
            "arn": "arn:aws:imagebuilder:us-west-2:111122223333:lifecycle-policy/
sample-lifecycle-policy3",
            "name": "sample-lifecycle-policy3",
            "status": "ENABLED",
            "executionRole": "arn:aws:iam::111122223333:role/sample-lifecycle-role",
            "resourceType": "AMI_IMAGE",
            "dateCreated": "2023-10-19T15:16:40.046000-07:00",
            "dateUpdated": "2023-10-21T20:07:15.958000-07:00",
            "dateLastRun": "2023-11-12T09:27:45.830000-08:00"
}]}
```

> **Note**
>
> デフォルトの を使用するには AWS リージョン、 `--region`パラメータを指定せずにこのコマンドを実行します。

ライフサイクルポリシーの詳細の表示

Image Builder コンソールのサイフサイクルポリシー詳細ページには概要セクションがあり、追加情報はタブにまとめられています。ページの見出しは、ポリシー名です。

Image Builder コンソールのライフサイクルポリシーの詳細ページでは、特定のライフサイクルポリシーの詳細を表示できます。Image Builder API、SDK または AWS CLI でコマンドやアクションを使用して、ポリシー詳細を取得することもできます。

内容

Image Builder コンソールでライフサイクルポリシーの詳細を表示します

Image Builder コンソールのイメージ詳細ページには概要セクションがあり、追加情報はタブにまとめられています。ページの見出しは、イメージを作成したレシピの名前とビルドバージョンです。

コンソールの詳細セクションとタブ

Summary (概要) セクション

概要セクションはページの幅いっぱいに広がり、以下の詳細が含まれます。これらの詳細は常に表示されます。

[ポリシーステータス]

ポリシーがアクティブか非アクティブか。

タイプ

新しいイメージバージョン (AMI またはコンテナイメージ) を作成したときに Image Builder が配信した出力のタイプ。

作成日

ライフサイクルポリシーの作成時のタイムスタンプ。

[変更日]

ライフサイクルポリシーが最後に更新された時間。

[最終実行日]

ライフサイクルポリシーが最後に実行された時間。

IAM ロール

Image Builder がライフサイクルアクションを実行するために使用する IAM ロール。

ARN

ライフサイクルポリシーリソースの Amazon リソースネーム (ARN)。

説明

ライフサイクルポリシーの説明 (入力した場合)。

[ルール] タブ

[ルール] タブには、表示しているポリシーに設定したライフサイクルルールが表示されます。このタブには、次の詳細が含まれます。

- [名前] — ルールの名前。これらの名前は固定されており、設定可能なポリシーアクションに基づいています。
  - `Deprecation rule`
  - `Disable rule`
  - `Deletion rule`
- [ルール] — ルールに設定されているアクションの簡単な説明。
- [ルール条件] — 関連するリソース処理の構成、ルールの例外、保持設定 (該当する場合) を一覧表示します。

ルール設定の詳細については、「ライフサイクルルールの仕組み」を参照してください。

スコープタブ

[スコープ] タブには、表示しているポリシーに設定されているリソース選択条件が表示されます。このタブには、次の詳細が含まれます。

- [フィルター: **#########**] – スコープの定義に使用したフィルタータイプ。フィルタータイプは、次のいずれかになります。
  - `recipes` — ライフサイクルポリシーが適用されるイメージの作成に使用されたレシピ。

- tags — Image Builder がライフサイクルポリシーを適用するイメージリソースを選択するために使用するタグのセット。
- 検索バー — リストを [名前] でフィルタリングして、タブに表示される結果を効率化できます。
- [名前] — 各行には、フィルター条件に設定した名前またはタグが含まれます。
- [バージョン] — レシピフィルターを設定している場合、Image Builder はレシピのバージョンを表示します。

RunLog タブ

設定したリソースのポリシーを実行するたびに、Image Builder はランタイムの詳細を保存します。テーブルの各行は 1 つのランタイムインスタンスを表します。このタブには、次の詳細が含まれます。

- [実行 ID] — ライフサイクルポリシーのランタイムインスタンスを識別します。
- [実行ステータス] — ポリシーアクションが現在実行中か、正常に実行されたか、失敗したか、またはキャンセルされたかを報告するランタイムステータス。
- [影響を受けたリソース] — ランタイムインスタンスがライフサイクルアクションの対象となるイメージリソースを識別したかどうかを示します。
- [開始日] — ランタイムインスタンスが開始されたときのタイムスタンプ。
- [終了日] — ランタイムインスタンスが終了されたときのタイムスタンプ。

ライフサイクルポリシーの作成

新しい EC2 Image Builder ライフサイクルポリシーを作成する場合、設定はポリシーの対象となるイメージの種類によって異なります。AMI イメージリソースとコンテナイメージリソースのライフサイクルポリシーを作成する API アクションは同じです (CreateLifecyclePolicy)。ただし、イメージリソースと関連するリソースの設定は異なります。このセクションでは、両方のライフサイクル管理ポリシーを作成する方法を説明します。

> **Note**
>
> ライフサイクルポリシーを作成する前に、前提条件 を満たしていることを確認してください。

Image Builder AMI イメージリソースのライフサイクル管理ポリシーを作成します

以下の方法のいずれかを使用して、 AWS Management Console または で AMI イメージライフサイクルポリシーを作成できます AWS CLI。CreateLifecyclePolicy API アクションを使用することもできます。関連する SDK リクエストについては、「EC2 Image Builder API」リファレンスのそのコマンドの「関連項目」リンクを参照してください。

AWS Management Console

で AMI イメージリソースのライフサイクルポリシーを作成するには AWS Management Console、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから [ライフサイクルポリシー] を選択します。
3. [ライフサイクルポリシーを作成] を選択します。
4. 以下の手順で説明するポリシーを設定します。
5. 設定を行った後にライフサイクルポリシーを作成するには、[ポリシーを作成] を選択します。

ポリシーの [全般] 設定を設定します。

1. [ポリシータイプ]から [AMI] オプションを選択します。
2. [ポリシー名]を入力します。
3. オプションで、ライフサイクルポリシーの [説明] を入力します。
4. デフォルトでは、[アクティブ] はオンになっています。デフォルト設定ではライフサイクルポリシーが有効になり、すぐにスケジュールに追加されます。最初に非アクティブ化されたポリシーを作成するには、[アクティブ] をオフにします。
5. ライフサイクルポリシー権限用に作成した [IAM ロール]を選択します。このロールをまだ作成していない場合は、「前提条件」で詳細を確認してください。

ポリシーの [ルールスコープ] を設定します。

このセクションでは、使用するフィルタの種類に基づいて、ライフサイクルポリシーのリソース選択を設定します。

1. [フィルタータイプ: レシピ] – イメージリソースを作成したレシピに基づいてライフサイクルルールをイメージリソースに適用するには、ポリシー用に最大 50 のレシピバージョンを選択します。
2. [フィルタータイプ: タグ] – リソースタグに基づいてライフサイクルルールをイメージリソースに適用するには、ポリシーが照合する最大 50 のキーと値のペアのリストを入力します。

次の [ライフサイクルルール] を 1 つ以上有効にして、ライフサイクルポリシーが選択したリソースに適用します。ポリシーの実行時にリソースが複数のライフサイクルルールと一致する場合、Image Builder は 1) 非推奨、2) 無効化、3) 削除の順序でルールアクションを実行します。

ルールの非推奨

Image Builder のイメージリソースのステータスを `Deprecated` に設定します。Image Builder パイプラインは、廃止されたイメージでも引き続き実行されます。新しいインスタンスを起動することに影響を与えることなく、関連する AMI の非推奨期間をオプションで設定できます。

- [ユニット数] — イメージリソースが作成されてから `Deprecated` としてマークされるまでに経過する必要がある時間の整数値を指定します。
- [単位] — 使用する時間範囲を選択します。範囲は、`Days`、`Weeks`、`Months`、または `Years` とすることができます。
- [AMI の廃止] — チェックボックスを選択して、関連する Amazon EC2 AMI に廃止日をマークします。AMI は引き続き使用でき、AMI から新しいインスタンスを起動できます。

ルールの無効化

Image Builder のイメージリソースのステータスを `Disabled` に設定します。これにより、このイメージでは Image Builder パイプラインが実行されなくなります。オプションで、関連する AMI を無効にして、新しいインスタンスが起動しないようにすることができます。

- [ユニット数] — イメージリソースが作成されてから `Disabled` としてマークされるまでに経過する必要がある時間の整数値を指定します。
- [単位] — 使用する時間範囲を選択します。範囲は、`Days`、`Weeks`、`Months`、または `Years` とすることができます。
- [AMI を無効にする] — チェックボックスを選択して、関連する Amazon EC2 AMI を無効にします。AMI を使用したり、AMI から新しいインスタンスを起動したりすることはできなくなります。

ルールの削除

イメージリソースを経過時間または数別に削除します。ニーズを満たすしきい値を定義します。Image Builder のイメージリソースはしきい値を超えると削除されます。関連する AMI の登録を解除したり、それらの AMI のスナップショットを削除したりすることもできます。しきい値を超えて保持するリソースのタグを指定することもできます。

[削除ルール] を経過時間で設定すると、Image Builder は設定した一定期間後にイメージリソースを削除します。例えば、6 か月後にイメージリソースを削除します。数で設定する場合、Image Builder は指定した最新の数、または可能な限りその数に近い数のイメージを保持し、以前のバージョンを削除します。

- 保持期間別
  - [ユニット数] — イメージリソースが作成されてから削除されるまでに経過する必要がある時間の整数値を指定します。
  - [単位] — 使用する時間範囲を選択します。範囲は、`Days`、`Weeks`、`Months`、または `Years` とすることができます。
  - [レシピごとに少なくとも 1 つのイメージを保持] — このルールが適用されるレシピバージョンごとに使用可能な最新のイメージリソースを保持するには、このチェックボックスを選択します。

  カウント別

  - [イメージ数] — レシピバージョンごとに保持する最新のイメージリソースの数を整数値で指定します。
- [AMI の登録を解除] — 関連する Amazon EC2 AMI の登録を解除するには、このチェックボックスを選択します。AMI を使用したり、AMI から新しいインスタンスを起動したりすることはできなくなります。
- [関連するタグが付いたイメージ、AMI、スナップショットを保持] — チェックボックスを選択して、保持したいイメージリソースのタグのリストを入力します。タグはイメージリソースと Amazon EC2 AMI に適用されます。最大 50 個のキーと値のペアを入力できます。

[タグ (省略可能)]

ライフサイクルポリシーにタグを追加します。

AWS CLI

新しい Image Builder ライフサイクルポリシーを作成するには、 AWS CLIの create-lifecycle-policy コマンドを使用できます。

Image Builder コンテナイメージリソースのライフサイクル管理ポリシーを作成する

次の方法のいずれかを使用して、 AWS Management Console または でコンテナイメージのライフサイクルポリシーを作成できます AWS CLI。CreateLifecyclePolicy API アクションを使用することもできます。関連する SDK リクエストについては、「EC2 Image Builder API」リファレンスのそのコマンドの「関連項目」リンクを参照してください。

AWS Management Console

でコンテナイメージリソースのライフサイクルポリシーを作成するには AWS Management Console、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから [ライフサイクルポリシー] を選択します。
3. [ライフサイクルポリシーを作成] を選択します。
4. 以下の手順で説明するポリシーを設定します。
5. 設定を行った後にライフサイクルポリシーを作成するには、[ポリシーを作成] を選択します。

ポリシー設定: 全般設定

ポリシーの [全般] 設定を設定します。

1. [ポリシータイプ]から [AMI] オプションを選択します。
2. [ポリシー名]を入力します。
3. オプションで、ライフサイクルポリシーの [説明] を入力します。
4. デフォルトでは、[アクティブ] はオンになっています。デフォルト設定ではライフサイクルポリシーが有効になり、すぐにスケジュールに追加されます。最初に非アクティブ化されたポリシーを作成するには、[アクティブ] をオフにします。
5. ライフサイクルポリシー権限用に作成した [IAM ロール]を選択します。このロールをまだ作成していない場合は、「前提条件」で詳細を確認してください。

ポリシーの [ルールスコープ] を設定します。

このセクションでは、使用するフィルタの種類に基づいて、ライフサイクルポリシーのリソース選択を設定します。

1. [フィルタータイプ: レシピ] – イメージリソースを作成したレシピに基づいてライフサイクルルールをイメージリソースに適用するには、ポリシー用に最大 50 のレシピバージョンを選択します。
2. [フィルタータイプ: タグ] – リソースタグに基づいてライフサイクルルールをイメージリソースに適用するには、ポリシーが照合する最大 50 のキーと値のペアのリストを入力します。

ルールの削除

コンテナイメージの場合、このルールは Image Builder コンテナイメージリソースを削除します。ECR リポジトリに配布された Docker イメージをオプションで削除して、新しいコンテナーの実行に使用されないようにすることができます。

[削除ルール] を経過時間で設定すると、Image Builder は設定した一定期間後にイメージリソースを削除します。例えば、6 か月後にイメージリソースを削除します。数で設定する場合、Image Builder は指定した最新の数、または可能な限りその数に近い数のイメージを保持し、以前のバージョンを削除します。

- 保持期間別
  - [ユニット数] — イメージリソースが作成されてから削除されるまでに経過する必要がある時間の整数値を指定します。
  - [単位] — 使用する時間範囲を選択します。範囲は、`Days`、`Weeks`、`Months`、または `Years` とすることができます。
  - [少なくとも 1 つのイメージを保持] — このルールが適用される各レシピバージョンで使用可能な最新のイメージリソースのみを保持するには、このチェックボックスを選択します。

  カウント別
  - [イメージ数] — レシピバージョンごとに保持する最新のイメージリソースの数を整数値で指定します。
- [ECR コンテナイメージを削除] — ECR リポジトリに保存されている関連するコンテナーイメージを削除するには、このチェックボックスを選択します。コンテナイメージをベースとして使用し、新しいイメージを作成したり、新しいコンテナを実行したりすることはできなくなります。

- [関連するタグが付いた画像を保持] — 保持したいイメージリソースのタグのリストを入力するには、このチェックボックスを選択します。

[タグ (省略可能)]

ライフサイクルポリシーにタグを追加します。

AWS CLI

新しい Image Builder ライフサイクルポリシーを作成するには、 AWS CLIの create-lifecycle-policy コマンドを使用できます。

EC2 Image Builder イメージリソースのライフサイクル管理ルールの仕組み

イメージライフサイクルポリシーは、定義したライフサイクルルールを使用して全体的なリソース管理戦略を実装します。定義するルールは、利用可能なイメージを最新の状態に保ち、出力 AMI 用のスナップショットストレージ、コンテナイメージの ECR リポジトリストレージとデータ転送速度などの基盤となるインフラストラクチャのコストを最小限に抑えるのに役立ちます。

ポリシーについては、次のタイプのルールを指定できます。

ルールの非推奨

Image Builder のイメージリソースのステータスを `Deprecated` に設定します。Image Builder パイプラインは、廃止されたイメージでも引き続き実行されます。新しいインスタンスを起動することに影響を与えることなく、関連する AMI の非推奨期間をオプションで設定できます。

AMI が廃止されると、一般的な検索では無視されます。例えば、 で Amazon EC2 describe-images コマンドを実行しても AWS CLI、結果セットで非推奨の AMIs返されません。ただし、廃止された AMI は AMI ID で引き続き確認できます。

このルールはコンテナイメージには適用されません。

ルールの無効化

Image Builder のイメージリソースのステータスを `Disabled` に設定します。これにより、このイメージでは Image Builder パイプラインが実行されなくなります。オプションで、関連する AMI を無効にして、新しいインスタンスが起動しないようにすることができます。

AMI を無効にすると、その AMI はプライベートになり、その AMI を使用して新しいインスタンスを起動できなくなります。AMI をアカウント、組織、または組織単位と共有している場合、AMI がプライベートになると、その AMI にアクセスできなくなります。

このルールはコンテナイメージには適用されません。

ルールの削除

イメージリソースを経過時間または数別に削除します。ニーズを満たすしきい値を定義します。Image Builder のイメージリソースはしきい値を超えると削除されます。関連する AMI の登録を解除したり、それらの AMI のスナップショットを削除したりすることもできます。しきい値を超えて保持するリソースのタグを指定することもできます。

コンテナイメージの場合、このルールは Image Builder コンテナイメージリソースを削除します。ECR リポジトリに配布されたコンテナイメージをオプションで削除して、新しいコンテナの実行に使用されないようにすることができます。

内容

- 除外ルール (API/SDK/CLI)
- ポリシーのライフサイクル管理ルールの詳細を表示します。

除外ルール (API/SDK/CLI)

以下の除外ルールは AMI のライフサイクルルールの例外を定義します。除外ルールで指定された条件を満たす AMI はライフサイクルアクションから除外されます。除外ルールは AWS Management Consoleでは利用できません。

次の用語では、`LifecyclePolicyDetailExclusionRules` データ型の API 表記法を使用しています。

除外ルール

amis

以下のリストに示す `LifecyclePolicyDetailExclusionRulesAmis` での設定が含まれます。

tagMap

リソースのタイプにかかわらず、ライフサイクルアクションをスキップする最大 50 のタグのリストを提供できます。

次の用語では、`LifecyclePolicyDetailExclusionRulesAmis` データ型の API 表記法を使用しています。

AMI 除外ルール

isPublic

パブリック AMI をライフサイクルアクションから除外するかどうかを設定します。

lastLaunched

ライフサイクルアクションから最新のリソースを除外するための Image Builder の設定の詳細を指定します。

regions

ライフサイクルアクションから除外 AWS リージョン される を設定します。

sharedAccounts

ライフサイクルアクションから除外されるリソース AWS アカウント を指定します。

tagMap

タグを含む AMI のライフサイクルアクションから除外すべきタグを一覧表示します。

ポリシーのライフサイクル管理ルールの詳細を表示します。

ルールは、Image Builder イメージリソース用に作成したライフサイクル管理ポリシー内で定義されます。コンソールのライフサイクルポリシー詳細ページには、ポリシーに設定したルールの詳細を表示する [ルール] タブ があります。

でポリシーの詳細を取得するには AWS CLI、 get-lifecycle-policy コマンドを実行します。レスポンス内のポリシー詳細には、ポリシーに対して定義したアクション (ルール) のリストが含まれます。これには、設定したすべての設定が含まれます。

EC2 Image Builder イメージのビルドワークフローとテストワークフローを管理する

イメージワークフローは、イメージ作成プロセスのビルドステージとテストステージで EC2 Image Builder が実行する一連のステップを定義します。これは、Image Builder ワークフローフレームワーク全体の一部です。

イメージワークフローの利点

- イメージワークフローを使用すると、イメージ作成プロセスの柔軟性、可視性、制御性が向上します。
- ワークフロードキュメントを定義するときにカスタマイズされたワークフローステップを追加することも、Image Builder のデフォルトワークフローを使用することもできます。
- デフォルトのイメージワークフローに含まれるワークフローステップは除外できます。
- ビルドプロセスを完全にスキップするテスト専用のワークフローを作成できます。ビルド専用のワークフローを作成する場合も同様です。

> **Note**
>
> 既存のワークフローは変更できませんが、クローンを作成したり、新しいバージョンを作成することはできます。

ワークフローフレームワーク: ステージ

イメージワークフローをカスタマイズするには、イメージ作成ワークフローフレームワークを構成するワークフローステージを理解することが重要です。

イメージ作成ワークフローフレームワークには、次の 2 つの異なるステージがあります。

1. ビルドステージ (スナップショット前) — ビルドステージでは、ベースイメージを実行している Amazon EC2 ビルドインスタンスに変更を加え、新しいイメージのベースラインを作成します。例えば、レシピには、アプリケーションをインストールしたり、オペレーティングシステムのファイアウォール設定を変更したりするコンポーネントを含めることができます。

   この段階が正常に完了すると、Image Builder はテスト段階以降に使用するスナップショットまたはコンテナイメージを作成します。
2. テストステージ (スナップショット後) — テストステージでは、AMI を作成するイメージとコンテナイメージにいくつかの違いがあります。AMI ワークフローでは、Image Builder はビルドステージの最終ステップとして作成したスナップショットから EC2 インスタンスを起動します。新しいインスタンスでテストを実行して設定を検証し、インスタンスが期待どおりに機能していることを確認します。コンテナワークフローでは、テストはビルドに使用したのと同じインスタンスで実行されます。

ワークフローフレームワークには配布ステージも含まれています。ただし、その段階のワークフローは Image Builder が処理します。

サービスアクセス

イメージワークフローを実行するには、Image Builder にワークフローアクションを実行する権限が必要です。AWSServiceRoleForImageBuilder サービスリンクロールを指定することも、次のように、サービスアクセス用の独自のカスタムロールを指定することもできます。

- コンソール — パイプラインウィザードの [ステップ 3: イメージ作成プロセスの定義] で、[サービスアクセス] パネルの [IAM ロール] リストから、サービスリンクロールまたは独自のカスタムロールを選択します。
- Image Builder API – CreateImageアクションリクエストで、 `executionRole`パラメータの値としてサービスにリンクされたロールまたは独自のカスタムロールを指定します。

サービスロールの作成方法の詳細については、「 ユーザーガイド」の「 AWS サービスにアクセス許可を委任するロールの作成AWS Identity and Access Management 」を参照してください。

内容

イメージワークフローを一覧表示する

Image Builder コンソールの [イメージワークフロー] リストページでは、所有している、またはアクセスできるイメージワークフローリソースの一覧と、これらのリソースに関する重要な詳細情報を確認できます。Image Builder API、 SDKs、または でコマンドまたはアクションを使用して、アカウントのイメージワークフローを AWS CLI 一覧表示することもできます。

以下の方法のいずれかを使用して、所有またはアクセスできるイメージワークフローリソースを一覧表示できます。API アクションについては、EC2 Image Builder API リファレンスListWorkflowsの「」を参照してください。関連する SDK リクエストについては、同じページの「関連項目」リンクを参照してください。

Console

ワークフローの詳細

Image Builder コンソールの[イメージワークフロー]リストページには、以下の詳細が含まれます。

- [ワークフロー] — イメージワークフローリソースの最新バージョンの名前。Image Builder コンソールの [ワークフロー] 列は、ワークフローの詳細ページにリンクしています。
- [バージョン] — イメージワークフローリソースの最新バージョン。
- [タイプ] – ワークフロータイプ: `BUILD` または `TEST`。
- [所有者] – ワークフローリソースの所有者。
- [作成日] — Image Builder が最新バージョンのイメージワークフローリソースを作成した日付と時刻。
- [ARN] — 最新バージョンのイメージワークフローリソースの Amazon リソースネーム (ARN)。

イメージワークフローを一覧表示する

Image Builder コンソールにイメージワークフローリソースを一覧表示するには、次の手順を実行します。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから [イメージワークフロー] を選択します。

結果のフィルタ処理

[イメージワークフロー] リストページでは、特定のイメージワークフローを検索して結果をフィルタリングできます。イメージワークフローでは、次のフィルターを使用できます。

`Workflow`

結果を効率化するために、ワークフロー名の全体または一部を入力できます。デフォルトでは、リスト内のすべてのワークフローが表示されます。

`Version`

結果を効率化するために、バージョン番号の全体または一部を入力できます。デフォルトでは、リスト内のすべてのバージョンが表示されます。

`Type`

ワークフローのタイプでフィルタリングすることも、すべてのタイプを表示することもできます。デフォルトでは、リスト内のすべてのワークフローのタイプが表示されます。

- BUILD
- TEST

`Owner`

検索バーから所有者フィルターを選択すると、Image Builder はアカウントのイメージワークフローの所有者のリストを表示します。リストから所有者を選択すると、検索結果を効率化できます。デフォルトでは、リスト内のすべての所有者が表示されます。

- AWS アカウント– ワークフローリソースを所有するアカウント。
- Amazon – Amazon が所有および管理するワークフローリソース。

AWS CLI

で list-workflows コマンドを実行すると AWS CLI、所有している、またはアクセスできるイメージワークフローのリストを取得できます。

次のコマンド例は、フィルターなしで list-workflows コマンドを使用し、所有している、もしくはアクセス権のあるすべての Image Builder イメージワークフローリソースを一覧表示する方法を示しています。

例:すべてのイメージワークフローを一覧表示する

```
aws imagebuilder list-workflows
```

出力:

```
{
 "workflowVersionList": [
  {
    "name": "example-test-workflow",
    "dateCreated": "2023-11-21T22:53:14.347Z",
    "version": "1.0.0",
    "owner": "111122223333",
    "type": "TEST",
    "arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/test/example-test-
workflow/1.0.0"
```

```
  },
  {
    "name": "example-build-workflow",
    "dateCreated": "2023-11-20T12:26:10.425Z",
    "version": "1.0.0",
    "owner": "111122223333",
    "type": "BUILD",
    "arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/example-
build-workflow/1.0.0"
  }
 ]
}
```

list-workflows コマンドを実行すると、次の例のようにフィルターを適用して結果を効率化できます。結果をフィルタリングする方法の詳細については、「AWS CLI コマンドリファレンス」の「list-workflows」コマンドを参照してください。

例: ビルドワークフローのフィルター

```
aws imagebuilder list-workflows --filters name="type",values="BUILD"
```

出力:

```
{
 "workflowVersionList": [
  {
    "name": "example-build-workflow",
    "dateCreated": "2023-11-20T12:26:10.425Z",
    "version": "1.0.0",
    "owner": "111122223333",
    "type": "BUILD",
    "arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/example-
build-workflow/1.0.0"
  }
 ]
}
```

イメージワークフローの作成

イメージワークフローを作成すると、イメージの作成プロセスをより細かく制御できます。Image Builder がイメージをビルドするときに実行するワークフローと、イメージをテストするときに実行

するワークフローを指定できます。また、カスタマーマネージドキーを指定してワークフローリソースを暗号化することもできます。ワークフローリソースの暗号化の詳細については、「EC2 Image Builder での暗号化とキー管理」を参照してください。

イメージの作成には、ビルドステージのワークフローを 1 つと、テストステージのワークフローを 1 つ以上指定できます。必要に応じて、ビルドステージやテストステージを完全にスキップすることもできます。ワークフローが使用する YAML 定義ドキュメントで、ワークフローが実行するアクションを設定します。YAML ドキュメントの構文については「YAML ワークフロードキュメントを作成する」を参照してください。

ビルドワークフローまたはテストワークフローを新しく作成する手順については、使用する環境に合ったタブを選択してください。

AWS Management Console

以下のプロセスを使用して、Image Builder コンソールで新しいワークフローを作成できます。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. ナビゲーションペインから [イメージワークフロー] を選択します。これにより、アカウントが所有している、またはアクセスできるイメージワークフローのリストが表示されます。

   > **Note**
   >
   > Image Builder がデフォルトのワークフローに使用する Amazon マネージドワークフローリソースは、常にリストに表示されます。これらのワークフローの詳細を表示するには、[ワークフロー] リンクを選択します。

3. 新しいワークフローを作成するには、[イメージワークフローを作成] を選します。これにより、[イメージワークフローを作成] ページが表示されます。
4. 新しいワークフローの詳細を設定します。ビルドワークフローを作成するには、フォームの上部にある [ビルド] オプションを選択します。テストワークフローを作成するには、フォームの上部にある [テスト] オプションを選択します。Image Builder は、このオプションに基づいて [テンプレート] リストにデータを入力します。ビルドワークフローおよびテストワークフローのその他のステップは、すべて同じです。

   全般

一般セクションには、名前や説明など、ワークフローリソースに適用される設定が含まれます。一般設定には以下が含まれます。

- [イメージワークフロー名] (必須) — イメージワークフローの名前。新しい名前は アカウント内で一意である必要があります。名前の最大長は 128 文字です。使用可能な文字は、文字、数字、スペース、-、および _ です。
- [バージョン] (必須) — 作成するワークフローリソースのセマンティックバージョン (major.minor.patch)。
- [説明] (オプション) — オプションでワークフローの説明を追加します。
- [KMS キー] (オプション) — カスタマーマネージドキーを使用してワークフローリソースを暗号化できます。詳細については、「カスタマーマネージドキーを使用してイメージワークフローを暗号化する」を参照してください。

定義ドキュメント

YAML ワークフロードキュメントには、ワークフローのすべての設定が含まれています。

使用を開始する

- Image Builder のデフォルトテンプレートをワークフローのベースラインとして開始するには、[テンプレートから開始] オプションを選択します。このオプションはデフォルト選択。[テンプレート] リストから使用するテンプレートを選択すると、選択したテンプレートのデフォルト設定が新しいワークフロードキュメントの [コンテンツ] にコピーされ、そこで変更を加えることができます。
- ワークフロードキュメントを最初から定義するには、[ゼロから開始] オプションを選択します。これにより、ドキュメント形式の重要な部分の簡単な概要が [コンテンツ] に入力され、作業を開始しやすくなります。

[コンテンツ] パネルの下部には、YAML ドキュメントに関する警告やエラーを示すステータスバーがあります。YAML ワークフロードキュメントの作成方法の詳細については、「YAML ワークフロードキュメントを作成する」を参照してください。

5. ワークフローが完了したら、または進行状況を保存して後で戻りたい場合は、[ワークフローを作成] を選択します。

AWS CLI

で create-workflow コマンドを実行する前に AWS CLI、ワークフローのすべての設定を含む YAML ドキュメントを作成する必要があります。詳細については、「YAML ワークフロードキュメントを作成する」を参照してください。

次の例は、create-workflow AWS CLI コマンドを使用してビルドワークフローを作成する方法を示しています。--data パラメータは、作成したワークフローのビルド設定を含む YAML ドキュメントを指します。

例: ワークフローの作成

```
aws imagebuilder create-workflow --name example-build-workflow --semantic-
version 1.0.0 --type BUILD --data file://example-build-workflow.yml
```

出力:

```
{
"workflowBuildVersionArn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/
build/example-build-workflow/1.0.0/1",
"clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

次の例は、create-workflow AWS CLI コマンドを使用してテストワークフローを作成する方法を示しています。--data パラメータは、作成したワークフローのビルド設定を含む YAML ドキュメントを指します。

例: テストワークフローの作成

```
aws imagebuilder create-workflow --name example-test-workflow --semantic-
version 1.0.0 --type TEST --data file://example-test-workflow.yml
```

出力:

```
{
"workflowBuildVersionArn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/
test/example-test-workflow/1.0.0/1",
"clientToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222"
}
```

YAML ワークフロードキュメントを作成する

YAML 形式の定義ドキュメントでは、イメージビルドプロセスのビルド段階とテスト段階の入力、出力、ワークフローのステップを設定します。標準化されたステップを含むテンプレートから始めることも、ゼロから始めて独自のワークフローを定義することもできます。テンプレートを使用するかゼロから始めるかにかかわらず、ワークフローをニーズに合わせてカスタマイズできます。

YAML ワークフロードキュメントの構造

Image Builder がイメージビルドとテストアクションを実行するために使用する YAML ワークフロードキュメントは、次のように構成されています。

- 識別
- 入力パラメータ
- ステップ
- 出力

識別

ワークフローを一意に識別します。このセクションは次の属性を含むことができます。

フィールド	説明	タイプ	必須
name	ワークフロードキュメントの名前。	文字列	いいえ
説明	ドキュメントの説明。	文字列	いいえ
schemaVersion	ドキュメントのスキーマバージョン。現在は 1.0。	文字列	はい

例

```
---
name: sample-test-image
description: Workflow for a sample image, with extra configuration options exposed
 through workflow parameters.
schemaVersion: 1.0
```

入力パラメータ

ワークフロードキュメントのこの部分では、呼び出し側が指定できる入力パラメータを定義します。パラメータがない場合、このセクションは省略できます。パラメータを指定する場合、各パラメータには以下の属性を含めることができます。

フィールド	説明	タイプ	必須	制約
name	パラメータの名前。	文字列	はい	
説明	パラメータの説明。	文字列	いいえ	
デフォルト	値が指定されていない場合のパラメータのデフォルト値。パラメータ定義にデフォルト値を含めない場合、ランタイム時にパラメータ値が必要になります。	パラメータのデータ型に一致します。	いいえ	
type	パラメータのデータ型。パラメータ定義にデ	文字列	はい	パラメータのデータ型は、次のいずれかであ

フィールド	説明	タイプ	必須	制約
	フォルト値を含めない場合、パラーメータ型のデフォルトはランタイム時に必要な文字列値になります。			る必要があります。 • string • integer • boolean • stringList

例

ワークフロードキュメントでパラメータを指定します。

```
parameters:
  - name: waitForActionAtEnd
    type: boolean
    default: true
    description: "Wait for an external action at the end of the workflow"
```

ワークフロードキュメントでパラメータ値を使用します。

```
$.parameters.waitForActionAtEnd
```

ステップ

ワークフローに最大 15 のステップアクションを指定します。ステップは、ワークフロードキュメント内で定義されている順序で実行されます。失敗した場合、ロールバックは逆の順序で実行されます。失敗したステップから始まり、前のステップまでさかのぼって実行されます。

各ステップは、前のステップアクションの出力を参照できます。これをチェーニング、または参照と呼びます。前のステップアクションの出力を参照するには、JSONPath セレクターを使用できます。例:

```
$.stepOutputs.step-name.output-name
```

詳細については、「ワークフロードキュメントでは動的変数を使用します」を参照してください。

> **Note**
>
> ステップ自体には出力属性はありませんが、ステップアクションからの出力はすべてステップの `stepOutput` に含まれます。

各ステップには、次の属性を含めることができます。

フィールド	説明	タイプ	必須	デフォルト値	制約
アクション	このステップが実行するワークフローアクション。	文字列	はい		Image Builder ワークフロードキュメントでサポートされているステップアクションである必要があります。
`if`、その後に `if` オペレータを変更する一連の条件ステートメントが続きます。	条件ステートメントは、ワークフローステップの本文に制御フローの決定ポイントを追加します。	dict	いいえ		Image Builder は、`if` 演算子の修飾子として以下の条件ステートメントをサポートしています。 • 分岐条件と修飾子: `if`、`and`、`or`、`not` 分岐条件は

フィールド	説明	タイプ	必須	デフォルト値	制約
					1 行に単独で指定されます。 • 比較演算子: booleanEquals 、numberEquals 、numberGreaterThan 、numberGreaterThanEquals 、numberLessThan 、numberLessThanEquals 、stringEquals 。
説明	手順の説明。	文字列	いいえ		空の文字列は使用できません。含める場合、長さは 1 ～ 1024 文字である必要があります。

フィールド	説明	タイプ	必須	デフォルト値	制約
入力	ステップアクションの実行に必要なパラメータが含まれます。キー値は静的な値として指定することも、正しいデータ型に解決される JSONPath 変数で指定することもできます。	dict	はい		
name	ステップの名前。この名前はワークフロードキュメント内で一意である必要があります。	文字列	はい		長さは 3 ~ 128 文字にする必要があります。 英数字と _ を含めることができます。スペースは使用できません。

フィールド	説明	タイプ	必須	デフォルト値	制約
onFailure	ステップが失敗した場合に実行するアクションを次のように構成します。 Behavior • Abort – ステップを失敗させ、ワークフローを失敗させ、失敗したステップの後に残っているステップを実行しません。ロールバックが有効な場合、ロールバックは失敗したステップから始まり、ロールバックを許可するすべてのステップがロールバックされ	文字列	いいえ	Abort	Abort | Continue

フィールド	説明	タイプ	必須	デフォルト値	制約
	るまで続きます。 • Continue – ステップは失敗しますが、失敗したステップの後に残りのステップは引き続き実行されます。この場合、ロールバックは行われません。				
rollbackEnabled	障害が発生した場合にステップをロールバックするかどうかを設定します。静的なブール値を使用するか、ブール値に解決される動的な JSONPath 変数を使用できます。	ブール値	いいえ	true	true | false | または true または false に解決される JSONPath 変数。

フィールド	説明	タイプ	必須	デフォルト値	制約
timeoutSeconds	再試行が適用される場合に、ステップが失敗して再試行されるまでの最大実行時間 (秒単位)。	整数	いいえ	該当する場合、ステップアクションに定義されているデフォルトによって異なります。	1 ~ 86400 秒 (最大 24 時間)

例

```
steps:
  - name: LaunchTestInstance
    action: LaunchInstance
    onFailure: Abort
    inputs:
      waitFor: "ssmAgent"

  - name: ApplyTestComponents
    action: ExecuteComponents
    onFailure: Abort
    inputs:
      instanceId.$: "$.stepOutputs.LaunchTestInstance.instanceId"

  - name: TerminateTestInstance
    action: TerminateInstance
    onFailure: Continue
    inputs:
      instanceId.$: "$.stepOutputs.LaunchTestInstance.instanceId"

  - name: WaitForActionAtEnd
    action: WaitForAction
    if:
      booleanEquals: true
      value: "$.parameters.waitForActionAtEnd"
```

出力

ワークフローの出力を定義します。各出力は、出力の名前と値を指定するキーと値のペアです。出力を使用してランタイム時に後続のワークフローで使用できるデータをエクスポートできます。このセクションはオプションです。

定義する各出力には以下の属性が含まれます。

フィールド	説明	タイプ	必須
name	出力の名前。名前は、パイプラインに含めるワークフロー間で一意である必要があります。	文字列	はい
value	出力の値。文字列の値は、ステップアクションからの出力ファイルなどの動的変数でもかまいません。詳細については、「ワークフロードキュメントでは動的変数を使用します」を参照してください。	文字列	はい

例

createProdImage ステップからのステップ出力を含むワークフロードキュメントの出力イメージ ID を作成します。

```
outputs:
  - name: 'outputImageId'
    value: '$.stepOutputs.createProdImage.imageId'
```

次のワークフローのワークフロー出力を参照してください。

```
$.workflowOutputs.outputImageId
```

ワークフロードキュメントでサポートされているステップアクション

このセクションには、Image Builder がサポートするステップアクションの詳細が含まれています。

このセクションで使用する用語

AMI

Amazon マシンイメージ

ARN

Amazon リソースネーム

サポートされているアクション

- BootstrapInstanceForContainer
- CollectImageMetadata
- CollectImageScanFindings
- CreateImage
- ExecuteComponents
- LaunchInstance
- RunCommand
- RunSysPrep
- SanitizeInstance
- TerminateInstance
- WaitForAction

BootstrapInstanceForContainer

このステップアクションは、コンテナワークフローを実行するための最小要件でインスタンスをブートストラップするサービススクリプトを実行します。Image Builder は、Systems Manager API

の sendCommand を使用してこのスクリプトを実行します。詳細については、「AWS Systems Manager Run Command」を参照してください。

> **Note**
>
> ブートストラップスクリプトは、Image Builder が Docker コンテナを正常に構築するための前提条件である AWS CLI および Docker パッケージをインストールします。このステップアクションを含めないと、イメージビルドは失敗する可能性があります。

デフォルトタイムアウト: 60 分

ロールバック: このステップアクションにはロールバックはありません。

インプット: 以下の表には、このステップアクションでサポートされる入力が含まれています。

入力名	説明	タイプ	必須	デフォルト	制約
instanceId	ブートストラップするインスタンスの ID。	文字列	はい		これは、このワークフローのインスタンスを起動したワークフローステップの出力インスタンス ID でなければなりません。

出力: 次の表には、このステップアクションの出力が含まれています。

出力名	説明	[Type] (タイプ)
runCommandId	インスタンスでブートストラップスクリプトを実行した Systems Manager sendCommand の ID。	文字列

出力名	説明	[Type] (タイプ)
status	Systems Manager sendCommand から返されたステータス。	文字列
output	Systems Manager sendCommand から返された出力。	文字列

例

ワークフロードキュメントでステップアクションを指定します。

```
- name: ContainerBootstrapStep
  action: BootstrapInstanceForContainer
  onFailure: Abort
  inputs:
      instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

ワークフロードキュメント内のステップアクション値の出力を使用します。

```
$.stepOutputs.ContainerBootstrapStep.status
```

CollectImageMetadata

このステップアクションはビルドワークフローでのみ有効です。

EC2 Image Builder は、イメージを構築してテストするために、起動した EC2 インスタンスで AWS Systems Manager (Systems Manager) エージェントを実行します。Image Builder は、「Systems Manager インベントリ」を使用してビルドフェーズ中に使用されたインスタンスに関する追加情報を収集します。この情報には、オペレーティングシステム (OS) の名前とバージョン、オペレーティングシステムによって報告されたパッケージとそれぞれのバージョンのリストが含まれます。

> **Note**
>
> このステップアクションは AMI を作成するイメージにのみ有効です。

デフォルトタイムアウト: 30 分

ロールバック:Image Builder は、このステップで作成された Systems Manager リソースをすべてロールバックします。

インプット: 以下の表には、このステップアクションでサポートされる入力が含まれています。

入力名	説明	タイプ	必須	デフォルト	制約
instanceId	メタデータ設定を適用するビルドインスタンス。	文字列	はい		これは、このワークフローのビルドインスタンスを起動したワークフローステップの出力インスタンス ID でなければなりません。

出力: 次の表には、このステップアクションの出力が含まれています。

出力名	説明	[Type] (タイプ)
osVersion	ビルドインスタンスから収集されたオペレーティングシステム名とバージョン。	文字列
associationId	インベントリ収集に使用される Systems Manager アソシエーション ID。	文字列

例

ワークフロードキュメントでステップアクションを指定します。

```
- name: CollectMetadataStep
```

```
  action: CollectImageMetadata
  onFailure: Abort
  inputs:
      instanceId: $.stepOutputs.LaunchStep.instanceId
```

ワークフロードキュメント内のステップアクションからの出力を使用します。

```
$.stepOutputs.CollectMetadataStep.osVersion
```

CollectImageScanFindings

アカウントで Amazon Inspector が有効になっていて、パイプラインでイメージスキャンが有効になっている場合、このステップアクションは Amazon Inspector によって報告されたテストインスタンス用のイメージスキャンの検出結果を収集します。このステップアクションはビルドワークフローでは利用できません。

デフォルトタイムアウト: 120 分

ロールバック: このステップアクションにはロールバックはありません。

インプット: 以下の表には、このステップアクションでサポートされる入力が含まれています。

入力名	説明	タイプ	必須	デフォルト	制約
instanceId	スキャンが実行されたインスタンスの ID。	文字列	はい		これは、このワークフローのインスタンスを起動したワークフローステップの出力インスタンス ID でなければなりません。

出力: 次の表には、このステップアクションの出力が含まれています。

出力名	説明	[Type] (タイプ)
runCommandId	結果を収集するためにスクリプトを実行した Systems Manager sendCommand の ID。	文字列
status	Systems Manager sendCommand から返されたステータス。	文字列
output	Systems Manager sendCommand から返された出力。	文字列

例

ワークフロードキュメントでステップアクションを指定します。

```
- name: CollectFindingsStep
  action: CollectImageScanFindings
  onFailure: Abort
  inputs:
      instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

ワークフロードキュメント内のステップアクション値の出力を使用します。

```
$.stepOutputs.CollectFindingsStep.status
```

CreateImage

このステップアクションは、Amazon EC2 CreateImage API を使用して実行中のインスタンスからイメージを作成します。作成プロセス中、ステップアクションは必要に応じてリソースが正しい状態になったことを確認してから続行します。

デフォルトタイムアウト: 720 分

ロールバック: このステップアクションにはロールバックはありません。

インプット: 以下の表には、このステップアクションでサポートされる入力が含まれています。

入力名	説明	タイプ	必須	デフォルト	制約
instanceId	新しいイメージを作成するインスタンス。	文字列	はい		指定したインスタンス ID のインスタンスは、このステップの開始時点の `running` 状態にある必要があります。

出力: 次の表には、このステップアクションの出力が含まれています。

出力名	説明	[Type] (タイプ)
imageId	作成されたイメージの AMI ID。	文字列

例

ワークフロードキュメントでステップアクションを指定します。

```
- name: CreateImageFromInstance
  action: CreateImage
  onFailure: Abort
  inputs:
      instanceId.$: "i-1234567890abcdef0"
```

ワークフロードキュメント内のステップアクション値の出力を使用します。

```
$.stepOutputs.CreateImageFromInstance.imageId
```

ExecuteComponents

このステップアクションは、ビルド中の現在のイメージのレシピで指定されているコンポーネントを実行します。ビルドワークフローは、ビルドインスタンスでビルドコンポーネントを実行します。テストワークフローは、テストインスタンスでのみテストコンポーネントを実行します。

Image Builder は、Systems Manager API の sendCommand を使用してコンポーネントを実行します。詳細については、「AWS Systems Manager Run Command」を参照してください。

デフォルトタイムアウト: 720 分

ロールバック: このステップアクションにはロールバックはありません。

インプット: 以下の表には、このステップアクションでサポートされる入力が含まれています。

入力名	説明	タイプ	必須	デフォルト	制約
instanceId	コンポーネントを実行する必要のあるインスタンスの ID。	文字列	はい		これは、このワークフローのインスタンスを起動したワークフローステップの出力インスタンス ID でなければなりません。

出力: 次の表には、このステップアクションの出力が含まれています。

出力名	説明	[Type] (タイプ)
runCommandId	インスタンスでコンポーネントを実行した Systems Manager sendCommand の ID。	文字列

出力名	説明	[Type] (タイプ)
status	Systems Manager sendCommand から返されたステータス。	文字列
output	Systems Manager sendCommand から返された出力。	文字列

例

ワークフロードキュメントでステップアクションを指定します。

```
- name: ExecComponentsStep
  action: ExecuteComponents
  onFailure: Abort
  inputs:
      instanceId: $.stepOutputs.LaunchStep.instanceId
```

ワークフロードキュメント内のステップアクションからの出力を使用します。

```
$.stepOutputs.ExecComponentsStep.status
```

LaunchInstance

このステップアクションは、 でインスタンスを起動 AWS アカウント し、Systems Manager エージェントがインスタンスで実行されるまで待ってから、次のステップに進みます。起動アクションでは、レシピの設定と、イメージに関連するインフラストラクチャ設定リソースを使用します。例えば、起動するインスタンスタイプはインフラストラクチャ設定に基づいています。出力は、起動したインスタンスのインスタンス ID です。

`waitFor` 入力は、ステップ完了要件を満たす条件を設定します。

デフォルトタイムアウト: 60 分

ロールバック: ビルドインスタンスの場合、ロールバックはインフラストラクチャ設定リソースで設定したアクションを実行します。デフォルトでは、イメージの作成に失敗するとビルドインスタンスは終了します。ただし、インフラストラクチャ設定には、ビルドインスタンスをトラブルシューティング用に保持する設定があります。

インプット: 以下の表には、このステップアクションでサポートされる入力が含まれています。

入力名	説明	タイプ	必須	デフォルト	制約
waitFor	ワークフローステップを完了して次のステップに進む前に待つ条件。	文字列	はい		Image Builder は現在 ssmAgent をサポートしています。

出力: 次の表には、このステップアクションの出力が含まれています。

出力名	説明	[Type] (タイプ)
instanceId	起動したインスタンスのインスタンス ID。	文字列

例

ワークフロードキュメントでステップアクションを指定します。

```
- name: LaunchStep
  action: LaunchInstance
  onFailure: Abort
  inputs:
    waitFor: ssmAgent
```

ワークフロードキュメント内のステップアクションからの出力を使用します。

```
$.stepOutputs.LaunchStep.instanceId
```

RunCommand

このステップアクションは、ワークフローのコマンドドキュメントを実行します。Image Builder は、Systems Manager API の sendCommand を使用して実行します。詳細については、「AWS Systems Manager Run Command」を参照してください。

デフォルトタイムアウト: 12 時間

ロールバック: このステップアクションにはロールバックはありません。

インプット: 以下の表には、このステップアクションでサポートされる入力が含まれています。

入力名	説明	タイプ	必須	デフォルト	制約
instanceId	コマンドドキュメントを実行するインスタンスのID。	文字列	はい		これは、このワークフローのインスタンスを起動したワークフローステップの出力インスタンス ID でなければなりません。
documentName	実行する Systems Manager コマンドドキュメントの名前。	文字列	はい		
parameters	コマンドドキュメントに必要なすべてのパラメータのキーと値のペアのリスト。	dictionary<string, list<string>>	条件付き		
documentVersion	実行するコマンドドキュメントのバージョン。	文字列	いいえ	$DEFAULT	

出力: 次の表には、このステップアクションの出力が含まれています。

出力名	説明	[Type] (タイプ)
runCommandId	インスタンスでコマンドドキュメントを実行した Systems Manager sendCommand の ID。	文字列
status	Systems Manager sendCommand から返されたステータス。	文字列
output	Systems Manager sendCommand から返された出力。	文字列のリスト

例

ワークフロードキュメントでステップアクションを指定します。

```
- name: RunCommandDoc
  action: RunCommand
  onFailure: Abort
  inputs:
    documentName: SampleDocument
    parameters:
        osPlatform:
          - "linux"
    instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

ワークフロードキュメント内のステップアクション値の出力を使用します。

```
$.stepOutputs.RunCommandDoc.status
```

RunSysPrep

このステップアクションでは、ビルドインスタンスがスナップショット用にシャットダウンされる前に、Systems Manager API の sendCommand を使用して Windows インスタンス用の AWSEC2-

RunSysprep ドキュメントを実行します。これらのアクションはAWS 、イメージの強化とクリーニングのベストプラクティスに従います。

デフォルトタイムアウト: 60 分

ロールバック: このステップアクションにはロールバックはありません。

インプット: 以下の表には、このステップアクションでサポートされる入力が含まれています。

入力名	説明	タイプ	必須	デフォルト	制約
instanceId	AWSEC2-RunSysprep ドキュメントを実行するインスタンスの ID。	文字列	はい		これは、このワークフローのインスタンスを起動したワークフローステップの出力インスタンス ID でなければなりません。

出力: 次の表には、このステップアクションの出力が含まれています。

出力名	説明	[Type] (タイプ)
runCommandId	インスタンスで AWSEC2-RunSysprep ドキュメントを実行した Systems Manager sendCommand の ID。	文字列
status	Systems Manager sendCommand から返されたステータス。	文字列
output	Systems Manager sendCommand から返された出力。	文字列

例

ワークフロードキュメントでステップアクションを指定します。

```
- name: RunSysprep
  action: RunSysPrep
  onFailure: Abort
  inputs:
      instanceId.$: $.stepOutputs.LaunchStep.instanceId
```

ワークフロードキュメント内のステップアクション値の出力を使用します。

```
$.stepOutputs.RunSysprep.status
```

SanitizeInstance

このステップアクションは、スナップショットのためにビルドインスタンスをシャットダウンする前に Linux インスタンス用の推奨サニタイズスクリプトを実行します。サニタイズスクリプトにより、最終的なイメージがセキュリティのベストプラクティスに従っていることを確認し、スナップショットに引き継がれてはならないビルドアーティファクトや設定をすべて削除することができます。スクリプトの詳細については、「ビルド後のクリーンアップが必要です。」を参照してください。このステップアクションはコンテナーイメージには適用されません。

Image Builder は、Systems Manager API の sendCommand を使用してこのスクリプトを実行します。詳細については、「AWS Systems Manager Run Command」を参照してください。

デフォルトタイムアウト: 60 分

ロールバック: このステップアクションにはロールバックはありません。

インプット: 以下の表には、このステップアクションでサポートされる入力が含まれています。

入力名	説明	タイプ	必須	デフォルト	制約
instanceId	サニタイズするインスタンスの ID。	文字列	はい		これは、このワークフローのインスタンスを起動したワークフローステップの出

入力名	説明	タイプ	必須	デフォルト	制約
					カインスタンス ID でなければなりません。

出力: 次の表には、このステップアクションの出力が含まれています。

出力名	説明	[Type] (タイプ)
runCommandId	インスタンスでサニタイズスクリプトを実行した Systems Manager sendCommand の ID。	文字列
status	Systems Manager sendCommand から返されたステータス。	文字列
output	Systems Manager sendCommand から返された出力。	文字列

例

ワークフロードキュメントでステップアクションを指定します。

```
- name: SanitizeStep
  action: SanitizeInstance
  onFailure: Abort
  inputs:
      instanceId: $.stepOutputs.LaunchStep.instanceId
```

ワークフロードキュメント内のステップアクション値の出力を使用します。

```
$.stepOutputs.SanitizeStep.status
```

TerminateInstance

このステップアクションは、入力として渡されたインスタンス ID でインスタンスを終了します。

デフォルトタイムアウト: 30 分

ロールバック: このステップアクションにはロールバックはありません。

インプット: 以下の表には、このステップアクションでサポートされる入力が含まれています。

入力名	説明	タイプ	必須	デフォルト	制約
instanceId	終了するインスタンスの ID。	文字列	はい		

出力: このステップアクションには出力がありません。

例

ワークフロードキュメントでステップアクションを指定します。

```
- name: TerminateInstance
  action: TerminateInstance
  onFailure: Continue
  inputs:
      instanceId.$: i-1234567890abcdef0
```

WaitForAction

このステップアクションは、実行中のワークフローを一時停止し、Image Builder SendWorkflowStepAction API アクションからの外部アクションの受信を待ちます。このステップでは、詳細タイプ のデフォルトの EventBridge イベントバスに EventBridge イベントを発行しますEC2 Image Builder Workflow Step Waiting。SNS トピック ARN を提供した場合、ステップは SNS 通知を送信することもできます。

デフォルトタイムアウト: 3 日

ロールバック: このステップアクションにはロールバックはありません。

インプット: 以下の表には、このステップアクションでサポートされる入力が含まれています。

入力名	説明	タイプ	必須	デフォルト	制約
snsTopicArn	ワークフローステップが保留になっているときに通知を送信するオプションの SNS トピック ARN。	文字列	いいえ		

出力: 次の表には、このステップアクションの出力が含まれています。

出力名	説明	[Type] (タイプ)
アクション	SendWorkflowStepAction API アクションが返すアクション。	文字列 (RESUME または STOP)
理由	反されたアクションの理由。	文字列

例

ワークフロードキュメントでステップアクションを指定します。

```
- name: SendEventAndWait
  action: WaitForAction
  onFailure: Abort
  inputs:
    snsTopicArn: arn:aws:sns:us-west-2:111122223333:ExampleTopic
```

ワークフロードキュメント内のステップアクション値の出力を使用します。

```
$.stepOutputs.SendEventAndWait.reason
```

ワークフロードキュメントでは動的変数を使用します

ワークフロードキュメントで動的変数を使用して、イメージ作成プロセスのランタイム時に変化する値を表すことができます。動的変数値は、ターゲット変数を一意に識別する構造ノードを備えた JSONPath セレクターとして表されます。

JSONPath の動的ワークフロー変数構造

```
$.<document structure>.[<step name>].<variable name>
```

ルート ($) の後の最初のノードは、`stepOutputs` または Image Builder システム変数の場合は、`imageBuilder` などのワークフロードキュメント構造を表します。以下のリストには、サポートされている JSONPath ワークフロードキュメント構造ノードが含まれています。

ドキュメント構造ノード

- parameters – ワークフローパラメータ
- stepOutputs – 同じワークフロードキュメント内のステップからの出力
- workflowOutputs – すでに実行されているワークフロードキュメントからの出力
- imagebuilder – Image Builder システム変数

`parameters` および `stepOutputs` ドキュメント構造ノードには、ステップ名のオプションノードが含まれます。これにより、すべてのステップで変数名が一意になるようになります。

JSONPath の最後のノードは、`instanceId` などのターゲット変数の名前です。

各ステップは、これらの JSONPath 動的変数を備えた前のステップアクションの出力を参照できます。これをチェーニング、または参照とも呼びます。前のステップアクションの出力を参照するには、次の動的変数を使用できます。

```
$.stepOutputs.step-name.output-name
```

例

```
  - name: ApplyTestComponents
    action: ExecuteComponents
    onFailure: Abort
    inputs:
```

```
      instanceId.$: "$.stepOutputs.LaunchTestInstance.instanceId"
```

Image Builder システム変数を使用する

Image Builder には、ワークフロードキュメントで使用できる以下のシステム変数があります。

変数名	説明	[Type] (タイプ)	値の例
cloudWatchLogグループ	出力 CloudWatch ログのロググループの名前。 形式: `/aws/imagebuilder/`*<recipe-name>*	文字列	`/aws/imagebuilder/`*sampleImageRecipe*
cloudWatchLogストリーム	出力 CloudWatch ログのログストリームの名前。	文字列	*1.0.0/1*
collectImageMetadata	インスタンスメタデータを収集するかどうかを Image Builder に指示する設定。	ブール値	`true` | `false`
collectImageScan検出結果	Image Builder が画像スキャン検出結果を収集できるようにする設定の現在の値。	ブール値	`true` | `false`
imageBuildNumber	イメージのビルドバージョン番号。	整数	*1*
imageId		文字列	

変数名	説明	[Type] (タイプ)	値の例
	ベースイメージの AMI ID。		*ami-12345 67890abcdef1*
imageName	イメージの名前。	文字列	*sampleImage*
imageType	画像出力タイプ。	文字列	`AMI | Docker`
imageVersionNumber	イメージのバージョン番号。	文字列	*1.0.0*
instanceProfileName	Image Builder がビルドインスタンスとテストインスタンスを起動するために使用するインスタンスプロファイルロールの名前。	文字列	*SampleImageBuilder InstanceProfileRole*
platform	ビルドされたイメージのオペレーティングシステムプラットフォーム。	文字列	`Linux | Windows | MacOS`
s3Logs	Image Builder が書き込む S3 ログの設定を含む JSON オブジェクト。	JSON オブジェクト	{'s3Logs ': {'s3BucketName': '*sample-bucket* ', 's3KeyPrefix': '*ib-logs* '}}

変数名	説明	[Type] (タイプ)	値の例
securityGroups	ビルドインスタンスとテストインスタンスに適用されるセキュリティグループ ID。	List [String]	*[sg-12345 67890abcd ef1, sg-111122 22333344445]*
sourceImageARN	ワークフローがビルドステージとテストステージに使用する Image Builder イメージリソースの Amazon リソースネーム (ARN)。	文字列	arn:aws:imagebuild er:*us-east-1* :*111122223 333* :image/*sampleIma ge* /*1.0.0/1*
subnetId	ビルドインスタンスとテストインスタンスを起動するサブネットの ID。	文字列	*subnet-12 34567890a bcdef1*
terminateInstanceOn失敗	障害発生時にインスタンスを終了するか、トラブルシューティングのために保持するよう Image Builder に指示する設定の現在の値。	ブール値	`true` | `false`
workflowPhase	ワークフロー実行のために実行中の現在のステージ。	文字列	`Build` | `Test`

変数名	説明	[Type] (タイプ)	値の例
workingDirectory	作業ディレクトリへのパス。	文字列	/tmp

ワークフローステップで条件ステートメントを使用する

条件ステートメントは if ステートメントのドキュメント属性で始まります。if ステートメントの最終的な目的は、ステップアクションを実行するか、スキップするかを決定することです。if ステートメントが true に解決されると、ステップアクションが実行されます。false に解決された場合、Image Builder はステップアクションをスキップし、SKIPPED のステップステータスをログに記録します。

if ステートメントは分岐ステートメント (and、or) と条件付き修飾子 (not) をサポートします。また、比較するデータ型 (文字列または数値) に基づいて値比較 (等しい、より小さい、より大きい) を実行する次の比較演算子もサポートしています。

サポートされている比較演算子

- booleanEquals
- numberEquals
- numberGreaterThan
- numberGreaterThanEquals
- numberLessThan
- numberLessThanEquals
- stringEquals

分岐ステートメントと条件付き修飾子のルール

分岐ステートメント(and、or) と条件付き修飾子 (not) に以下のルールが適用されます。

- 分岐ステートメントと条件付き修飾子は、単独で 1 行に表示する必要があります。
- 分岐ステートメントと条件付き修飾子は、レベルのルールに従う必要があります。
  - 親レベルにはステートメントが 1 つしか存在できません。

- 子ブランチまたは修飾子はそれぞれ新しいレベルを開始します。

  レベルの詳細については、「ネストレベル」を参照してください。
- 各分岐ステートメントには少なくとも 1 つの子条件ステートメントが必要ですが、10 個以下でなければなりません。
- 条件付き修飾子は 1 つの子条件文に対してのみ動作します。

ネストレベル

条件ステートメントは、独自のセクション内の複数のレベルで動作します。例えば、if ステートメント属性はワークフロードキュメント内のステップ名とアクションと同じレベルに表示されます。これが条件ステートメントの基本です。

条件ステートメントのレベルは最大 4 つまで指定できますが、親レベルに表示できるのは 1 つだけです。他のすべての分岐ステートメント、条件修飾子、または条件演算子は、そこからレベルごとに 1 つずつインデントされます。

次の概要は、条件ステートメントのネストレベルの最大数を示しています。

```
base:
  parent:
    - child (level 2)
        - child (level 3)
            child (level 4)
```

if 属性

if 属性は、条件ステートメントをドキュメント属性として指定します。これはレベル 0 です。

親レベル

これは条件ステートメントのネストの最初のレベルです。このレベルにはステートメントが 1 つしか存在できません。分岐や修飾子が不要な場合は、子ステートメントを含まない条件演算子でもかまいません。このレベルでは、条件演算子を除いてダッシュ表記は使用しません。

子レベル

レベル 2~4 は子レベルとみなされます。子ステートメントには、分岐ステートメント、条件修飾子、または条件演算子を含めることができます。

例: ネストレベル

次の例は、条件ステートメントのレベルの最大数を示しています。

```
if:
  and:                              #first level
    - stringEquals: 'my_string'     #second level
      value: 'my_string'
    - and:                          #also second level
        - numberEquals: '1'         #third level
          value: 1
        - not:                      #also third level
            stringEquals: 'second_string'      #fourth level
            value: "diff_string"
```

ネストルール

- 子レベルのブランチまたは修飾子はそれぞれ新しいレベルを開始します。
- 各レベルはインデントされます。
- 親レベルには 1 つのステートメント、修飾子、または演算子を含め、最大 4 つのレベルがあり、さらに最大 3 つのレベルを追加できます。

例

この一連の例は、条件ステートメントのさまざまな側面を示しています。

分岐: and

and 分岐ステートメントは、ブランチの子である式のリストに基づいて動作します。これらの式はすべて、`true` と評価される必要があります。Image Builder は、リストに表示される順序で式を評価します。いずれかの式が `false` と評価されると、処理は停止し、分岐は `false` と見なされます。

次の例では、両方の式が `true` と評価されるため、`true` と評価されます。

```
if:
  and:
    - stringEquals: 'test_string'
      value: 'test_string'
    - numberEquals: 1
      value: 1
```

分岐: or

or 分岐ステートメントは、ブランチの子である式のリストに基づいて動作します。これらの式の少なくとも 1 つは、true と評価される必要があります。Image Builder は、リストに表示される順序で式を評価します。いずれかの式が true と評価されると、処理は停止し、分岐は true と見なされます。

次の例では、最初の式が false であるにもかかわらず、true と評価されます。

```
if:
  or:
    - stringEquals: 'test_string'
      value: 'test_string_not_equal'
    - numberEquals: 1
      value: 1
```

条件付き修飾子: not

not 条件付き修飾子は、ブランチの子である条件ステートメントを無効にします。

次の例では、not 修飾子が stringEquals 条件ステートメントを無効にした場合 true と評価します。

```
if:
  not:
    - stringEquals: 'test_string'
      value: 'test_string_not_equal'
```

条件ステートメント: booleanEquals

booleanEquals 比較演算子はブール値を比較し、ブール値が完全に一致する場合は true を返します。

次の例では、collectImageScanFindings が有効かどうかを調べます。

```
if:
  - booleanEquals: true
    value: '$.imagebuilder.collectImageScanFindings'
```

条件ステートメント: stringEquals

stringEquals 比較演算子は 2 つの文字列を比較し、文字列が完全に一致する場合は true を返します。いずれかの値が文字列でない場合、Image Builder は比較する前にそれを文字列に変換します。

次の例では、プラットフォームシステム変数を比較して、ワークフローが Linux プラットフォームで実行されているかどうかを判断します。

```
if:
  - stringEquals: 'Linux'
    value: '$.imagebuilder.Platform'
```

条件ステートメント: numberEquals

numberEquals 比較演算子は 2 つの数値を比較し、数値が等しい場合は true を返します。比較する数値は、以下の形式のいずれかである必要があります。

- 整数
- 浮動小数点数
- 次の正規表現パターンに一致する文字列: ^-?[0-9]+(\.)?[0-9]+$。

次の比較例では、すべて と評価されますtrue。

```
if:
  # Value provider as a number
  numberEquals: 1
  value: '1'

  # Comparison value provided as a string
  numberEquals: '1'
  value: 1

  # Value provided as a string
  numberEquals: 1
  value: '1'

  # Floats are supported
  numberEquals: 5.0
  value: 5.0

  # Negative values are supported
  numberEquals: -1
  value: -1
```

EC2 イメージ Builder による仮想マシン (VM) イメージのインポートとエクスポート

VM を仮想化環境からエクスポートすると、そのプロセスによって VM 環境、設定、データのスナップショットとして機能する 1 つ以上のディスクコンテナファイルのセットが作成されます。これらのファイルを使用してVMをインポートし、イメージレシピのベースイメージとして使用することができます。

Image Builder は VM ディスクコンテナの以下のファイル形式をサポートしています。

- オープン仮想化アーカイブ (OVA)
- 仮想マシンディスク (VMDK)
- バーチャルハードディスク (VHD/VHDX)
- Raw

インポートでは、ディスクを使用して Amazon マシンイメージ (AMI) と Image Builder イメージリソースを作成します。いずれもカスタムイメージレシピのベースイメージとして使用できます。インポートするには、VM ディスクを S3 バケットに保存する必要があります。別の方法として、既存の EBS スナップショットからインポートすることもできます。

Image Builder コンソールでは、イメージを直接インポートし、出力イメージまたは AMI をレシピで使用したり、レシピまたはレシピバージョンを作成するときにインポートパラメータを指定したりできます。直接インポートの詳細については、VM をインポートする (コンソール)を参照のこと。イメージレシピの一部としてインポートする方法の詳細については、「VM インポート設定」を参照してください。

VM をImage Builder (AWS CLI) にインポートする

VM をディスクから AMI にインポートし、すぐに参照できる Image Builder イメージリソースを作成するには、 AWS CLIから以下の手順に従ってください。

1. AWS CLIのAmazon EC2 VM Import/Export import-imageコマンドで、VMのインポートを開始する。コマンド・レスポンスで返されるタスクIDをメモしておく。これは次のステップで必要になります。詳細については、VM Import/Export ユーザーガイドの「VM Import/Export を使用してイメージとして VM をインポート」を参照してください。

2. CLI 入力 JSON ファイルの作成

で使用される Image Builder import-vm-image コマンドを効率化するために AWS CLI、コマンドに渡すすべてのインポート設定を含む JSON ファイルを作成します。

> **Note**
>
> JSON ファイル内のデータ値の命名規則は、Image Builder API アクションリクエストパラメータに指定されているパターンに従います。API コマンドリクエストパラメータを確認するには、EC2 Image Builder API リファレンス」のImportVmImage「 コマンド」を参照してください。
> データ値をコマンドラインパラメータとして指定するには、AWS CLI コマンドリファレンスで指定されているパラメータ名を参照してください。 オプションとして、イメージビルダーimport-vm-imageコマンドに指定します。

以下に、これらの例で指定するパラメータの概要を示します。

- 名前 (文字列、必須) — インポートからの出力として作成する Image Builder イメージリソースの名前。
- semanticVersion (文字列、必須) — 出力イメージのセマンティックバージョンは次の形式で表されます: 各位置に特定のバージョン (「メジャー」「マイナー」「パッチ」) を示す数値が付いています。例えば `1.0.0` です。Image Builder リソースのセマンティックバージョニングの詳細については、セマンティックバージョニングを参照してください。
- 説明 (文字列) — イメージレシピの説明。
- platform (文字列、必須) — インポートされた VM のオペレーティングシステムプラットフォーム。
- vmImportTaskID (文字列、必須) — Amazon EC2 VM のインポートプロセスからの `ImportTaskId` (AWS CLI) 。Image Builder はインポートプロセスを監視して作成した AMI を取り込み、レシピですぐに使用できる Image Builder イメージリソースを構築します。
- clientToken(文字列、必須) - リクエストの冪等性を保証するために提供する、大文字と小文字を区別する一意の識別子。詳細については、Amazon EC2 API Referenceの冪等性の確保を参照してください。
- タグ (文字列マップ) — タグはインポートリソースに添付されるキーと値のペアです。最大 50 個のキーと値のペアを使用できます。

ファイルに import-vm-image.json という名前を付けて保存し、Image Builder import-vm-image コマンドで使用します。

```
{
    "name": "example-request",
    "semanticVersion": "1.0.0",
    "description": "vm-import-test",
    "platform": "Linux",
    "vmImportTaskId": "import-ami-01ab234567890cd1e",
    "clientToken": "asz1231231234cs3z",
    "tags": {
     "Usage": "VMIE"
    }
}
```

3. イメージのインポート

作成したファイルを入力として、import-vm-image コマンドを実行します。

```
aws imagebuilder import-vm-image --cli-input-json file://import-vm-image.json
```

> **Note**
>
> - JSON ファイルパスの先頭にfile://ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

イメージビルド (AWS CLI) から VM ディスクを配布します。

AWS CLIの Image Builder ディストリビューション設定を使用して、サポートされている VM ディスクフォーマットファイルの通常のイメージビルドプロセスの一部として、ターゲットリージョンの S3 バケットへの配布を設定できます。詳細については、「出力VMディスクのディストリビューション設定を作成する(AWS CLI)」を参照してください。

EC2 Image Builder リソースを共有

EC2 Image Builder は AWS Resource Access Manager （ AWS RAM) と統合されており、特定のリソースを AWS アカウント または を通じて共有できます AWS Organizations。共有できる EC2 Image Builder リソースは下記のとおりです。

- コンポーネント
- イメージ
- recipe

このセクションでは、 EC2 Image Builder リソースの共有に役立つ情報を提供します。

セクションの内容

EC2 Image Builder での共有コンポーネント、イメージ、レシピの使用

コンポーネント、イメージ、レシピの共有により、リソース所有者はソフトウェア設定を他の AWS アカウント または AWS 組織内で共有できます。リソース共有を一元的に管理し、設定を共有できるアカウントのセットを定義できます。

このモデルでは、コンポーネント、イメージ、またはレシピを所有 AWS アカウント する (所有者) は、それを他の AWS アカウント （ コンシューマー) と共有します。コンシューマーは、共有コンポーネントをイメージパイプラインに関連付けることにより、共有コンポーネント、イメージ、またはレシピの更新を自動的に利用することができます。

コンポーネント、イメージ、またはレシピの所有者は、これらのリソースを下記のユーザーと共有できます。

- の組織 AWS アカウント 内外に固有 AWS Organizations。
- AWS Organizationsの組織内の組織単位 (OU)
- AWS Organizationsの組織全体。
- AWS Organizations または組織外の OUs。

コンポーネント、イメージ、レシピを共有するための前提条件

Image Builder コンポーネント、イメージ、またはレシピを共有するには :

- AWS アカウントでコンポーネント、イメージ、またはレシピがあるのは必要です。自身が共有を受けているリソースを他者に共有することはできません。
- 暗号化されたリソースに関連付けられた AWS Key Management Service （ AWS KMS) キーは、ターゲットアカウント、組織、または OUs と明示的に共有する必要があります。
- を使用して Image Builder リソースを AWS Organizations および OUs と共有するには AWS RAM、共有を有効にする必要があります。詳細については、「AWS RAM ユーザーガイド」の「AWS Organizationsで共有を有効化する」を参照してください。
- で暗号化されたイメージを異なるリージョンのアカウント AWS KMS 間で配布する場合は、各ターゲットリージョンに KMS キーとエイリアスを作成する必要があります。さらに、それらのリージョンでインスタンスを起動するユーザーは、キーポリシーで指定された KMS キーにアクセスする必要があります。

Image Builder がパイプラインビルドから作成する以下のリソースは、Image Builder リソースとは見なされません。むしろ、Image Builder がアカウント、およびディストリビューション設定で指定した AWS リージョン、アカウント、組織、または組織単位 (OUsに配布する外部リソースです。

- Amazon マシンイメージ (AMI)
- Amazon ECR にあるコンテナイメージ。

AMI ディストリビューション設定の詳細については、「AMI ディストリビューション設定の作成と更新」を参照してください。Amazon ECR にあるコンテナイメージのディストリビューション設定の詳細については、「コンテナイメージのディストリビューション設定を作成および更新します。」を参照してください。

AWS Organizations および OUs「組織または OUs」を参照してください。

関連サービス

AWS Resource Access Manager

コンポーネント、イメージ、レシピの共有は AWS Resource Access Manager () と統合されます AWS RAM。 AWS RAM は、任意の AWS アカウントまたは を通じて AWS リソースを共有できるようにするサービスです AWS Organizations。では AWS RAM、リソース共有を作成して、所有しているリソースを共有します。リソース共有は共有するリソースと、それらを共有するコンシューマーを指定します。コンシューマーは、個々の AWS アカウント、組織単位、または 内の組織全体にすることができます AWS Organizations。

の詳細については AWS RAM、「 AWS RAM ユーザーガイド」を参照してください。

リージョン間での共有

共有コンポーネント、イメージ、レシピは、指定された AWS リージョンでのみ共有できます。これらのリソースを共有する際、リージョン間でレプリケートされません。

コンポーネント、イメージ、またはレシピを共有する

Image Builder コンポーネント、イメージ、またはレシピを共有するには、リソース共有に追加する必要があります。リソース共有は、アカウント間で AWS RAM AWS リソースを共有できる リソースです。リソース共有では、共有対象のリソースと、共有先のコンシューマーを指定します。コンポーネント、イメージ、またはレシピを新しいリソース共有に追加するには、まず AWS RAM コンソールを使用してリソース共有を作成する必要があります。

ユーザーが の組織に属 AWS Organizations していて、組織内での共有が有効になっている場合、組織内のコンシューマーには共有コンポーネント、イメージ、またはレシピへのアクセス許可が自動的に付与されます。これに該当しない場合、コンシューマーはリソースへの参加の招待を受け取り、その招待を受け入れた後で、リソースの共有に対するアクセス許可が付与されます。

リソースの共有するには、次のオプションが利用できます:

オプション 1: RAM リソース共有を作成する

RAM リソース共有を作成する時、所有しているコンポーネント、イメージ、またはレシピを 1 つのステップで共有できます。次のいずれかの方法を使用して、リソース共有を作成してください:

- コンソール

  AWS RAM コンソールを使用してリソース共有を作成するには、「 ユーザーガイド」の「所有する AWS リソースの共有AWS RAM 」を参照してください。

- AWS CLI

  AWS RAM コマンドラインインターフェイスを使用してリソース共有を作成するには、 で create-resource-share コマンドを実行します AWS CLI。

オプション 2: リソースポリシーを適用して RAM リソース共有に昇格する

リソースを共有するための 2 番目のオプションには、両方の AWS CLI でコマンドを実行する 2 つのステップが含まれます。最初のステップでは、 の Image Builder コマンド AWS CLI を使用して、リソースベースのポリシーを共有リソースに適用します。2 番目のステップでは、 の promote-resource-share-created-from-policy AWS RAM コマンドを使用してリソースを RAM リソース共有に昇格 AWS CLI し、リソースを共有したすべてのプリンシパルにリソースが表示されるようにします。

1. リソースポリシーを適用します。

   リソースポリシーを正常に適用するには、共有を受けているアカウントが基礎的なリソースへのアクセス権限があることを確認する必要があります。

   該当するコマンドのリソースタイプに合ったタブを選択します。

   Image

   リソースポリシーをイメージに適用して、他のユーザーがレシピのベースイメージとして使用できるようにすることができます。

   で put-image-policy Image Builder コマンドを実行して AWS CLI、イメージを共有する AWS プリンシパルを特定します。

   ```
   aws imagebuilder put-image-policy --image-arn arn:aws:imagebuilder:us-
   west-2:123456789012:image/my-example-image/2019.12.03/1 --policy
    '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal":
    { "AWS": [ "123456789012" ] }, "Action": ["imagebuilder:GetImage",
    "imagebuilder:ListImages"], "Resource": [ "arn:aws:imagebuilder:us-
   west-2:123456789012:image/my-example-image/2019.12.03/1" ] } ] }'
   ```

Component

クロスアカウント共有を有効にするには、リソースポリシーを適用して、コンポーネントをビルトまたはテストします。このコマンドは、他のアカウントにレシピ内のあなたのコンポーネントを使用する権限を付与します。リソースポリシーを正常に適用するには、共有を受けているアカウントが基礎的なリソースへのアクセス権限があることを確認する必要があります。

で put-component-policy Image Builder コマンドを実行して AWS CLI、コンポーネントを共有する AWS プリンシパルを特定します。

```
aws imagebuilder put-component-policy --component-arn arn:aws:imagebuilder:us-
west-2:123456789012:component/my-example-component/2019.12.03/1 --policy
 '{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Principal":
 { "AWS": [ "123456789012" ] }, "Action": [ "imagebuilder:GetComponent",
 "imagebuilder:ListComponents" ], "Resource": [ "arn:aws:imagebuilder:us-
west-2:123456789012:component/my-example-component/2019.12.03/1" ] } ] }'
```

Image recipe

リソースポリシーをイメージ レシピに適用して、クロスアカウント共有を有効にできます。このコマンドは、レシピを使用して自分のアカウントにイメージを作成する権限を他のアカウントに与えます。リソースポリシーを正常に適用するには、共有するアカウントに、ベースイメージや選択されたコンポーネントなど、レシピが参照するすべてのリソースにアクセスする権限があることを確認する必要があります。

で put-image-recipe-policy Image Builder コマンドを実行して AWS CLI、イメージを共有する AWS プリンシパルを識別します。

```
aws imagebuilder put-image-recipe-policy --image-recipe-arn
 arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-
image-recipe/2019.12.03 --policy '{ "Version": "2012-10-17", "Statement":
 [ { "Effect": "Allow", "Principal": { "AWS": [ "123456789012" ] }, "Action":
 [ "imagebuilder:GetImageRecipe", "imagebuilder:ListImageRecipes" ], "Resource":
 [ "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-image-
recipe/2019.12.03" ] } ] }'
```

Container recipe

リソースポリシーをイメージ レシピに適用して、クロスアカウント共有を有効にできます。このコマンドは、レシピを使用して自分のアカウントにイメージを作成する権限を他のアカウントに与えます。リソースポリシーを正常に適用するには、共有するアカウントに、ベースイメージや選択されたコンポーネントなど、レシピが参照するすべてのリソースにアクセスする権限があることを確認する必要があります。

で put-container-recipe-policy Image Builder コマンドを実行して AWS CLI、イメージを共有する AWS プリンシパルを特定します。

```
aws imagebuilder put-container-recipe-policy --container-recipe-arn
 arn:aws:imagebuilder:us-west-2:123456789012:container-recipe/my-example-
container-recipe/2021.12.03 --policy '{ "Version": "2012-10-17", "Statement":
 [ { "Effect": "Allow", "Principal": { "AWS": [ "123456789012" ] }, "Action":
 [ "imagebuilder:GetContainerRecipe", "imagebuilder:ListContainerRecipes" ],
 "Resource": [ "arn:aws:imagebuilder:us-west-2:123456789012:container-recipe/my-
example-container-recipe/2021.12.03" ] } ] }'
```

> **Note**
>
> リソースの共有と共有解除に関する正しいポリシーを設定するには、`imagebuilder:put*` リソース所有者に権限が必要です。

2. RAM リソース共有として昇格する

リソースを共有したすべてのプリンシパルにリソースが表示されるようにするには、 で promote-resource-share-created-from-policy AWS RAM コマンドを実行します AWS CLI。

共有コンポーネント、イメージ、またはレシピの共有を解除する

所有する共有コンポーネント、イメージ、またはレシピを共有または共有を解除するには、リソース共有から削除する必要があります。これは、 AWS Resource Access Manager コンソールまたは を使用して実行できます AWS CLI。

> **Note**
>
> コンポーネント、イメージ、レシピの共有を解除する場合、コンシューマーはそれらに依存することはできません。コンシューマーは、所有者が共有を解除する前に、共有リソースの依存関係をすべて削除する必要があります。

AWS Resource Access Manager コンソールを使用して、所有する共有コンポーネント、イメージ、またはレシピを共有解除できます。

AWS RAM ユーザーガイドのリソース共有の更新を参照してください。

AWS CLIを使用して所有するコンポーネント、イメージ、またはレシピを共有解除できます。

「disassociate-resource-share」コマンドを使用してリソースの共有を停止します。

共有コンポーネント、イメージ、またはレシピを識別しています。

所有者と利用者は、Image Builder コマンドを AWS CLIで 使用して、共有コンポーネント、イメージ、およびイメージレシピを識別できます。

共有コンポーネントの識別

list-components コマンドを実行して、所有しているコンポーネントと共有しているコンポーネントのリストを取得します。get-component コマンドは、コンポーネント所有者の AWS アカウント ID を表示します。

共有イメージの識別

list-images コマンドを実行して、所有しているイメージと共有されているイメージのリストを取得します。get-image コマンドは、イメージ所有者の AWS アカウント ID を表示します。

共有コンテナイメージを識別

list-images コマンドを実行して、所有しているイメージと共有されているイメージのリストを取得します。「get-image」コマンドはイメージ所有者の AWS アカウント ID を表示します。

共有イメージレシピを識別

list-image-recipes コマンドを実行して、所有しているイメージレシピと共有されているイメージレシピのリストを取得します。get-image-recipe コマンドは、イメージレシピ所有者の AWS アカウント ID を表示します。

共有コンテナレシピを識別

list-container-recipes コマンドを実行して、所有しているコンテナレシピと共有されているコンテナレシピのリストを取得します。get-container-recipe コマンドは、コンテナレシピ所有者の AWS アカウント ID を表示します。

共有コンポーネント、イメージ、及びレシピのアクセス許可

所有者のアクセス許可

所有者は、共有コンポーネント、イメージ、またはイメージレシピが共有されなくなるまで削除できません。所有者は、利用者が誰もリソースに依存しなくなるまで、これらのリソースの共有を解除することはできません。

コンシューマーのアクセス許可

コンシューマーはコンポーネント、イメージ、またはイメージレシピを読み取ることはできますが、変更することはできません。リソースの所有者や他のコンシューマーが所有している場合、リソースを表示したり変更したりすることはできません。コンシューマーは共有コンポーネントとイメージをイメージレシピの中で使用して独自のカスタムイメージを作成できます。コンシューマーは共有イメージレシピを使用して独自のカスタムイメージを作成できます。

請求と使用量測定

EC2 Image Builder の使用には料金はかかりません。

リソースの制限

共有コンポーネント、イメージ、およびイメージレシピは、所有者のみが対応するリソース制限にカウントされます。共有されたリソースは、コンシューマーのリソース制限に影響はありません。

のタグ付けEC2 Image Builder リソースにタグを付けます。

リソースにタグを付けると、リソースコストやその他のカテゴリのフィルタリングや追跡に役立ちます。タグに基づいてアクセスを制御することもできます。タグベースの認可についての詳細は、Image Builder タグに基づく認可を参照してください。

Image Builder は以下の動的タグをサポートしています。

- `- {{imagebuilder:buildDate}}`

  ビルド時にビルドの日付/時刻に解決します。

- `- {{imagebuilder:buildVersion}}`

  Image Builder Amazon リソースネーム (ARN) の末尾にある番号であるビルドバージョンを解決します。たとえば、`"arn:aws:imagebuilder:us-west-2:123456789012:component/myexample-component/2019.12.02/1"` はビルドバージョンを 1 としてと表示します。

配布した Amazon マシンイメージ (AMIsを追跡しやすくするために、Image Builder は出力 AMIs に次のタグを自動的に追加します。

- `"CreatedBy":"EC2 Image Builder"`
- `"Ec2ImageBuilderArn":"arn:aws:imagebuilder:us-west-2:123456789012:image/simple-recipe-linux/1.0.0/10"`。 このタグには、AMI の作成に使用された Image Builder イメージリソースの ARN が含まれています。

内容

(AWS CLI)リソースをタグ付けします。

次の例は、imagebuilder CLI コマンドを使用して EC2 Image Builder でリソースを追加し、タグを付ける方法を示しています。`resourceArn` とタグを指定して適用する必要があります。

例`tag-resource.json`の内容は以下の通り：

```
{
    "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-example-pipeline",
    "tags": {
        "KeyName": "KeyValue"
    }
}
```

上記の tag-resource.json ファイルを参照する次のコマンドを実行します。

```
aws imagebuilder tag-resource --cli-input-json file://tag-resource.json
```

リソースのタグを外します (AWS CLI)

次の例は、imagebuilder CLI コマンドを使用してリソースからタグを削除する方法を示しています。タグを削除するには、resourceArn とキーを指定する必要があります。

例untag-resource.jsonの内容は以下の通り：

```
{
    "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-example-pipeline",
    "tagKeys": [
        "KeyName"
    ]
}
```

上記の untag-resource.json ファイルを参照する次のコマンドを実行します。

```
aws imagebuilder untag-resource --cli-input-json file://untag-resource.json
```

特定のリソース (AWS CLI) のすべてのタグを一覧表示します。

次の例は、imagebuilder CLI コマンドを使用して特定のリソースのすべてのタグを一覧表示する方法を示しています。

```
aws imagebuilder list-tags-for-resource --resource-arn arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-example-pipeline
```

EC2 Image Builder resourcesの削除

Image Builder 環境は、ご自宅と同様、必要なものを見つけて散らかることなくタスクを完了できるように、定期的なメンテナンスが必要です。テスト用に作成した一時リソースは定期的にクリーンアップしてください。そうしないと、それらのリソースのことを忘れてしまい、後でそのリソースが何に使用されたかを思い出せなくなる可能性があります。その時までには、それらを安全に取り除くことができるかどうかがはっきりしないかもしれません。

リソースを削除しても、イメージビルドプロセス中に作成された Amazon EC2 AMI や Amazon ECR コンテナイメージは削除されません。これらは、適切な Amazon EC2 または Amazon ECR コンソールアクション、または API または AWS CLI コマンドを使用して個別にクリーンアップする必要があります。

> **Tip**
>
> リソースを削除するときの依存関係エラーを防ぐため、必ず次の順序でリソースを削除してください。
>
> 1. イメージパイプライン
> 2. イメージのレシピ
> 3. 残っているすべてのリソース

AWS マネジメントコンソールを使用してリソースを削除する

イメージパイプラインとそのリソースを削除するには、以下の手順に従います。

パイプラインを削除します。

1. アカウントで作成されたビルドパイプラインのリストを表示するには、ナビゲーションペインから [Image pipelines] を選択します。
2. パイプライン名 の横にあるチェックボックスをオンにして、削除するパイプラインを選択します。
3. イメージパイプラインパネルの上部にある「アクション」メニューで、「削除」を選択します。
4. `Delete` と入力して削除を確認し、削除 を選択します。

レシピを削除します。

1. アカウントで作成されたレシピのリストを見るには、ナビゲーションペインからイメージレシピを選択します。
2. レシピ名の横にあるチェックボックスを選択して、削除するレシピを選択します。
3. イメージレシピパネルの上部にある「アクション」メニューで、「レシピを削除」を選択します。
4. `Delete` と入力して削除を確認し、削除 を選択します。

インフラストラクチャ設定を削除します。

1. アカウントのインフラストラクチャー設定リソースのリストを表示するには、ナビゲーションペインから [インフラストラクチャー設定] を選択します。
2. 構成名の横にあるチェックボックスを選択して、削除するインフラストラクチャー構成を選択します。
3. 「インフラストラクチャー設定」パネルの上部にある「削除」を選択します。
4. Delete と入力して削除を確認し、削除 を選択します。

ディストリビューション設定

1. アカウントで作成された配布設定のリストを表示するには、ナビゲーションペインから [配布設定] を選択します。
2. 設定名の横にあるチェックボックスをオンにして、このチュートリアル用に作成したディストリビューション設定を選択します。
3. ディストリビューション設定パネルの上部で、「削除」を選択します。
4. Delete と入力して削除を確認し、削除 を選択します。

イメージを削除します。

1. 自分のアカウントで作成されたイメージの一覧を表示するには、ナビゲーションペインからイメージを選択します。
2. 削除するイメージのバージョンを選択します。これにより、「イメージビルドバージョン」ページが開きます。
3. 削除したいイメージのバージョンの横にあるチェックボックスを選択します。一度に複数のイメージバージョンを選択することもできます。
4. 「イメージビルドバージョン」パネルの上部にある「バージョンを削除」を選択します。
5. Delete と入力して削除を確認し、削除 を選択します。

を使用してイメージパイプラインを削除する AWS CLI

以下の例では、 AWS CLIを使用して Image Builder リソースを削除する方法を示しています。前述のように、依存関係のエラーを避けるため、リソースは次の順序で削除する必要があります。

1. イメージパイプライン

2. イメージのレシピ

3. 残っているすべてのリソース

イメージパイプラインを削除します (AWS CLI)

次の例では、ARN を指定してイメージパイプラインを削除する方法を示しています。

```
aws imagebuilder delete-image-pipeline --image-pipeline-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-pipeline/my-example-pipeline
```

イメージレシピ (AWS CLI) の削除

次の例では、ARN を指定してイメージレシピを削除する方法を示しています。

```
aws imagebuilder delete-image-recipe --image-recipe-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-recipe/my-example-recipe/2019.12.03
```

インフラストラクチャ設定を削除します。

次の例では、ARN を指定してインフラストラクチャー設定リソースを削除する方法を示しています。

```
aws imagebuilder delete-infrastructure-configuration --infrastructure-configuration-arn
 arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/my-example-
infrastructure-configuration
```

ディストリビューション設定

次の例では、ARN を指定してディストリビューション設定リソースを削除する方法を示しています。

```
aws imagebuilder delete-distribution-configuration --distribution-configuration-arn
 arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-example-
distribution-configuration
```

イメージを削除します。

次の例では、ARN を指定してイメージビルドバージョンを削除する方法を示しています。

```
aws imagebuilder delete-image --image-build-version-arn arn:aws:imagebuilder:us-
west-2:123456789012:image/my-example-image/2019.12.02/1
```

コンポーネントを削除します。

次の例は、imagebuilder CLI コマンドを使用して ARN を指定してコンポーネントのビルドバージョンを削除する方法を示しています。

```
aws imagebuilder delete-component --component-build-version-arn
 arn:aws:imagebuilder:us-west-2:123456789012:component/my-example-
component/2019.12.02/1
```

> **Important**
>
> 削除する前に、コンポーネントのビルドバージョンを参照するレシピがないことを確認してください。そうしないと、パイプラインに障害が発生する可能性があります。

コンソールを使用して EC2 Image Builder パイプラインを管理する

Image Builder のイメージパイプラインは、カスタム AMI とコンテナイメージを作成および管理するための自動化フレームワークを提供します。パイプラインには以下の機能があります。

- ンがベースイメージ、ビルドとテスト用のコンポーネント、インフラストラクチャ設定、およびディストリビューション設定を組み立てる方法について説明します。
- コンソールウィザードの `Schedule builder` を使用するか、イメージを定期的に更新するための cron 式を入力することで、自動メンテナンスプロセスのスケジュールを簡単に設定できます。
- ベースイメージとコンポーネントの変更検出を有効にして、変更がない場合はスケジュールされたビルドを自動的にスキップします。
- Amazon を通じてルールベースのオートメーションを有効にします EventBridge。

> **Note**
>
> EventBridge API を使用してルールを表示または変更する方法の詳細については、「Amazon EventBridge API リファレンス」を参照してください。でeventsコマンドを使用して EventBridgeルールを表示または変更 AWS CLI する方法の詳細については、「 コマンドリファレンス」の「 イベント」を参照してください。 AWS CLI

内容

パイプラインの詳細を一覧表示して表示します。

このセクションでは、EC2 Image Builder インフラストラクチャ設定の情報を検索したり詳細を表示したりするさまざまな方法について説明します。

パイプラインの詳細

- イメージパイプライン (AWS CLI) を一覧表示する
- イメージパイプラインの詳細を取得する (AWS CLI)

イメージパイプライン (AWS CLI) を一覧表示する

次の例は、 で list-image-pipelines コマンドを使用してすべてのイメージパイプラインを AWS CLI 一覧表示する方法を示しています。

```
aws imagebuilder list-image-pipelines
```

イメージパイプラインの詳細を取得する (AWS CLI)

次の例は、 の get-image-pipeline コマンドを使用して、ARN を介してイメージパイプラインの詳細 AWS CLI を取得する方法を示しています。

```
aws imagebuilder get-image-pipeline --image-pipeline-arn arn:aws:imagebuilder:us-
west-2:123456789012:image-pipeline/my-example-pipeline
```

AMI イメージパイプラインの作成と更新

AMI イメージパイプラインは、Image Builder コンソールから、または Image Builder API、または AWS CLIの imagebuilder コマンドを使用して設定、構成、管理できます。[イメージパイプラインの作成] コンソールウィザードを使用して、次の手順を実行できます。

- 名前、説明、およびリソースタグなどのパイプラインの詳細を指定します。
- クイックスタートマネージドイメージからの基本イメージ、または自分で作成· 共有したイメージを含む AMI イメージレシピを選択します。レシピには、Image Builder がイメージの構築に使用する EC2 インスタンスで以下のタスクを実行するコンポーネントも含まれています。
  - ソフトウェアの追加と削除

  - 設定とスクリプトをカスタマイズ
  - 選択したテストを実行する
- ワークフローを指定して、パイプラインが実行するイメージビルドとテストのステップを設定します。
- デフォルト設定または自分で行った設定を使用して、パイプラインのインフラストラクチャー設定を定義します。設定には、イメージに使用するインスタンスタイプとキーペア、セキュリティとネットワークの設定、ログストレージとトラブルシューティングの設定、SNS 通知が含まれます。

  これは任意の手順です。Image Builder は、ユーザーが設定を自分で定義しない場合、デフォルトのインフラストラクチャー設定を使用します。
- 配信先の AWS リージョンとアカウントにイメージを配信するための配信設定を定義します。暗号化用の KMS キーを指定したり、AMI の共有やライセンスを設定したり、配布する AMI の起動テンプレートを設定したりできます。

  これは任意の手順です。設定を自分で定義しない場合、Image Builder は出力 AMI にデフォルトの命名を使用し、AMI をソースリージョンに配布します。ソースリージョンは、パイプラインを実行するリージョンです。

提供されているデフォルト値でイメージパイプラインの作成コンソールウィザードを使用する方法の詳細と step-by-step チュートリアルについては、「」を参照してくださいEC2 Image Builder コンソールウィザードを使用してコンテナイメージパイプラインを作成します。。

内容

AMIイメージパイプラインの作成 (AWS CLI)

AWS CLIの create-image-pipeline コマンドの入力として、設定の詳細を含むJSONファイルを使用してAMIイメージパイプラインを作成できます。

ベースイメージとコンポーネントからの保留中の更新を組み込むために、パイプラインが新しいイメージをビルドする頻度は、設定した `schedule` 内容によって異なります。各 `schedule` には次の属性があります。

- scheduleExpression— パイプラインの実行スケジュールを設定して、そのパイプラインを評価し、 pipelineExecutionStartCondition とビルドを開始すべきかどうかを判断します。スケジュールは cron 式で設定されます。Image Builder で cron 式を書式設定する方法については、「EC2 Image Builder で cron 式を使用する」を参照してください。
- pipelineExecutionStartCondition— パイプラインでビルドを開始するかどうかを決定します。有効な値を次に示します。
  - EXPRESSION_MATCH_ONLY— cron 式が現在の時刻と一致するたびに、パイプラインが新しいイメージが作成されます。
  - EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE— ベースイメージまたはコンポーネントに保留中の変更がない限り、パイプラインは新しいイメージビルドを開始しません。

で create-image-pipeline コマンドを実行する場合 AWS CLI、設定リソースの多くはオプションです。ただし、パイプラインが作成するイメージのタイプによっては、条件付きの要件があるリソースもあります。AMI イメージパイプラインには次のリソースが必要です。

- イメージレシピ ARN
- インフラストラクチャ設定ARN

1. CLI 入力 JSON ファイルの作成

   お気に入りのファイル編集ツールを使って、以下のキーと、あなたの環境で有効な値を持つJSONファイルを作成します。この例では、create-image-pipeline.jsonという名前のファイルを使用します。

```
{
 "name": "MyWindows2019Pipeline",
 "description": "Builds Windows 2019 Images",
 "enhancedImageMetadataEnabled": true,
 "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-recipe/2020.12.03",
 "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/my-example-infrastructure-configuration",
 "distributionConfigurationArn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-example-distribution-configuration",
 "imageTestsConfiguration": {
  "imageTestsEnabled": true,
```

```
  "timeoutMinutes": 60
 },
 "schedule": {
  "scheduleExpression": "cron(0 0 * * SUN *)",
  "pipelineExecutionStartCondition":
 "EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
 },
 "status": "ENABLED"
}
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

2. 作成したファイルを入力として使用し、次のコマンドを実行します。

```
aws imagebuilder create-image-pipeline --cli-input-json file://create-image-
pipeline.json
```

AMI イメージパイプラインの更新 (コンソール)

AMI イメージ用の Image Builder イメージパイプラインを作成したら、Image Builder コンソールからインフラストラクチャ設定とディストリビューション設定を変更できます。

イメージパイプラインを新しいイメージレシピで更新するには、 AWS CLIを使用する必要があります。詳細については、このガイドの「AMI イメージパイプラインの更新 (AWS CLI)」を参照してください。

既存の Image Builder パイプラインを選択

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. アカウントで作成されたビルドパイプラインのリストを表示するには、ナビゲーションペインから [Image pipelines] を選択します。

> **Note**
>
> イメージパイプラインのリストには、パイプラインによって作成される出力イメージのタイプ (AMI または Docker) のインジケータが含まれています。

3. 詳細を表示したりパイプラインを編集したりするには、[パイプライン名] のリンクを選択します。パイプラインの詳細ビューが開きます。

> **Note**
>
> また、パイプライン名の横にあるチェックボックスを選択し、詳細を表示を選択することもできます。

パイプラインの詳細

Demographics ページには、次のセクションが表示されます。

[概要]

ページ上部のセクションには、詳細タブを開くと表示されるパイプラインの主要な詳細がまとめられています。このセクションに表示される詳細は、それぞれの詳細タブでのみ編集できます。

詳細 タブ

- 出力イメージ — パイプラインが生成した出力イメージを表示します。
- イメージレシピ — レシピの詳細を表示します。レシピを作成すると、その編集はできなくなります。レシピの新しいバージョンは、Image Builder コンソールのイメージレシピページから、または AWS CLIで Image Builder コマンドを使用して作成する必要があります。詳細については、「レシピの管理」を参照してください。
- インフラストラクチャー設定 — ビルドパイプラインインフラストラクチャーを構成するための編集可能な情報を表示します。
- ディストリビューション設定 — AMI ディストリビューションの編集可能な情報を表示します。
- EventBridge rules – 選択したイベントバス について、 は現在のパイプラインをターゲットとする EventBridge ルールを表示します。 EventBridge イベントバスの作成とコンソールにリンクするルールアクションの作成が含まれます。詳細については、「 EventBridge ルールを使用する」を参照してください。

パイプラインのインフラストラクチャー設定を編集します。

インフラストラクチャ設定には以下の詳細が含まれており、パイプラインの作成後に編集できます。

- インフラストラクチャ設定の 説明。
- [IAM ロール] をインスタンスプロファイルに関連付けます。
- AWS インフラストラクチャ。これには、インスタンスタイプと通知用の SNS トピックが含まれます。
- VPC、サブネット、セキュリティグループ。
- 障害発生時にインスタンスを終了する、接続用のキーペア、インスタンスLog用のオプションの S3 バケットの場所などの トラブルシューティング設定。

パイプラインの詳細ページからインフラストラクチャー設定を編集するには、以下の手順に従います。

1. [インフラストラクチャー構成の作成] を選択します。
2. [インフラストラクチャー詳細] パネルの右上隅から [編集] を選択します。
3. インフラストラクチャー設定に加えた更新を保存する準備ができたら、[変更を保存] を選択します。

パイプラインのディストリビューション設定を編集します。

ディストリビューション設定には以下の詳細が含まれており、パイプラインの作成後に編集できます。

- ディストリビューション設定の説明です。
- イメージを配布するリージョンの リージョン設定。リージョン 1 のデフォルトは、パイプラインを作成したリージョンです。リージョンを追加 ボタンで配信するリージョンを追加でき、リージョン 1 を除くすべてのリージョンを削除できます。

  地域設定には以下が含まれます。

  - ターゲット リージョン
  - 出力 AMI の名前
  - 起動権限、およびそれらを共有するアカウント
  - 関連ライセンス (関連ライセンス設定)

> **Note**
>
> License Manager の設定は、 (香港) AWS リージョンと ap-east-1 (me-south-1バーレーン) リージョンなど、アカウントで有効にする必要があるリージョン間でレプリケートされません。

パイプラインの詳細ページからディストリビューション設定を編集するには、次の手順に従います。

1. ディストリビューション設定タブを選択します。
2. ディストリビューション詳細パネルの右上隅から編集を選択します。
3. 更新を保存する準備ができたら、変更を保存の順に選択します。

パイプラインのビルドスケジュールを編集します。

パイプラインの編集ページには、パイプラインの作成後に編集できる以下の詳細が含まれています。

- パイプラインの [説明]。
- メタデータの収集が強化されました。デフォルトで有効になっています。オフにするには、「拡張メタデータ収集を有効にする」チェックボックスをオフにします。
- パイプラインの ビルドスケジュール。スケジュールオプション とすべての設定をここで変更できます。

パイプラインの詳細ページからパイプラインを編集するには、以下の手順に従います：

1. ページの右上にある アクション メニューで 削除 を選択します。
2. 更新を保存する準備ができたら、変更を保存の順に選択します。

> **Note**
>
> cron式を使ったビルドのスケジューリングについては、EC2 Image Builder で cron 式を使用するを参照してください。

AMI イメージパイプラインの更新 (AWS CLI)

AWS CLIのupdate-image-pipelineコマンドの入力としてJSONファイルを使用して、AMIイメージパイプラインを更新することができます。JSON ファイルを設定するには、以下の既存のリソースを参照する Amazon リソースネーム (ARN) が必要です。

- 更新するイメージパイプライン
- イメージのレシピ
- インフラストラクチャ設定
- ディストリビューションの設定

AMI イメージパイプラインは、 AWS CLI 次のように の update-image-pipeline コマンドで更新できます。

> **Note**
>
> UpdateImagePipeline は、パイプラインの選択的な更新をサポートしていません。更新リクエストでは、変更されたプロパティだけでなく、必要なプロパティをすべて指定する必要があります。

1. CLI 入力 JSON ファイルの作成

   お気に入りのファイル編集ツールを使って、以下のキーと、あなたの環境で有効な値を持つJSONファイルを作成します。この例では、`create-component.json`という名前のファイルを使用します。

```
 {
 "imagePipelineArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-
pipeline/my-example-pipeline",
 "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-
example-recipe/2019.12.08",
 "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/my-example-infrastructure-
configuration",
 "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/my-example-distribution-
configuration",
 "imageTestsConfiguration": {
```

```
  "imageTestsEnabled": true,
  "timeoutMinutes": 120
 },
 "schedule": {
  "scheduleExpression": "cron(0 0 * * MON *)",
  "pipelineExecutionStartCondition":
 "EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
 },
 "status": "DISABLED"
}
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

2. 作成したファイルを入力として使用し、次のコマンドを実行します。

```
aws imagebuilder update-image-pipeline --cli-input-json file://update-image-
pipeline.json
```

コンテナイメージパイプラインの作成と更新

Image Builder コンソール、Image Builder API、または AWS CLIの imagebuilder コマンドを使用して、コンテナイメージパイプラインを設定、構成、管理できます。[イメージパイプライン作成]コンソールウィザードには開始アーティファクトが表示され、以下のステップを案内します。

- クイックスタートマネージドイメージ、Amazon ECR、または Docker Hub リポジトリからベースイメージを選択します。
- ソフトウェアの追加と削除
- 設定とスクリプトをカスタマイズ
- 選択したテストを実行する
- 事前に設定されたビルド時変数を使用して Dockerfile を作成します。

- イメージを AWS リージョンに配信する

イメージパイプラインの作成コンソールウィザードの使用に関する詳細と step-by-step チュートリアルについては、「」を参照してくださいEC2 Image Builder コンソールウィザードを使用してコンテナイメージパイプラインを作成します。。

内容

コンテナイメージパイプライン (AWS CLI) を作成する

AWS CLIのcreate-image-pipelineコマンドの入力としてJSONファイルを使用してコンテナイメージパイプラインを作成することができます。

ベースイメージとコンポーネントからの保留中の更新を組み込むために、パイプラインが新しいイメージをビルドする頻度は、設定した `schedule` 内容によって異なります。各 `schedule` には次の属性があります。

- `scheduleExpression`— パイプラインの実行スケジュールを設定して、そのパイプラインを評価し、 `pipelineExecutionStartCondition` とビルドを開始すべきかどうかを判断します。スケジュールは cron 式で設定されます。Image Builder で cron 式を書式設定する方法については、「EC2 Image Builder で cron 式を使用する」を参照してください。
- `pipelineExecutionStartCondition`— パイプラインでビルドを開始するかどうかを決定します。有効な値を次に示します。
  - `EXPRESSION_MATCH_ONLY`— cron 式が現在の時刻と一致するたびに、パイプラインが新しいイメージが作成されます。
  - `EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE`— ベースイメージまたはコンポーネントに保留中の変更がない限り、パイプラインは新しいイメージビルドを開始しません。

で create-image-pipeline コマンドを実行する場合 AWS CLI、設定リソースの多くはオプションです。ただし、パイプラインが作成するイメージのタイプによっては、条件付きの要件があるリソースもあります。コンテナイメージパイプラインには以下のリソースが必要です：

- コンテナレシピ ARN

- インフラストラクチャ設定ARN

コマンドを実行するときにディストリビューション設定リソースを含めない場合、出力イメージは、create-image-pipelineコマンドを実行するリージョンのコンテナレシピでターゲットリポジトリとして指定した ECR リポジトリに保存されます。パイプラインにディストリビューション設定リソースを含めると、ディストリビューションの最初のリージョンに指定したターゲットリポジトリが使用されます。

1. CLI 入力 JSON ファイルの作成

   お気に入りのファイル編集ツールを使って、以下のキーと、あなたの環境で有効な値を持つJSONファイルを作成します。この例では、`create-image-pipeline.json`という名前のファイルを使用します。

```
{
 "name": "MyWindows2019Pipeline",
 "description": "Builds Windows 2019 Images",
 "enhancedImageMetadataEnabled": true,
 "containerRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:container-
recipe/my-example-recipe/2020.12.03",
 "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/my-example-infrastructure-
configuration",
 "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/my-example-distribution-
configuration",
 "imageTestsConfiguration": {
  "imageTestsEnabled": true,
  "timeoutMinutes": 60
 },
 "schedule": {
  "scheduleExpression": "cron(0 0 * * SUN *)",
  "pipelineExecutionStartCondition":
 "EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
 },
 "status": "ENABLED"
}
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

2. 作成したファイルを入力として使用し、次のコマンドを実行します。

```
aws imagebuilder create-image-pipeline --cli-input-json file://create-image-pipeline.json
```

コンテナイメージパイプラインの更新 (コンソール)

Docker イメージ用に Image Builder コンテナイメージパイプラインを作成したら、Image Builder コンソールからインフラストラクチャ構成と配布設定を変更できます。

コンテナイメージパイプラインを新しいコンテナレシピで更新するには、 AWS CLIを使用する必要があります。詳細については、このガイドの「コンテナイメージパイプライン (AWS CLI) の更新」を参照してください。

既存のImage Builder Dockerイメージパイプラインを選択します。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. アカウントで作成されたビルドパイプラインのリストを表示するには、ナビゲーションペインから [Image pipelines] を選択します。

> **Note**
>
> イメージパイプラインのリストには、パイプラインによって作成される出力イメージのタイプ (AMI または Docker) のインジケータが含まれています。

3. 詳細を表示したりパイプラインを編集したりするには、[パイプライン名] のリンクを選択します。パイプラインの詳細ビューが開きます。

> **Note**
>
> また、パイプライン名の横にあるチェックボックスを選択し、詳細を表示を選択することもできます。

パイプラインの詳細

EC2 Image Builderのパイプライン詳細ページには、以下のセクションがあります：

[概要]

ページ上部のセクションには、詳細タブを開くと表示されるパイプラインの主要な詳細がまとめられています。このセクションに表示される詳細は、それぞれの詳細タブでのみ編集できます。

詳細 タブ

- 出力イメージ — パイプラインが生成した出力イメージを表示します。
- [コンテナレシピ] — レシピの詳細を表示します。レシピを作成すると、その編集はできなくなります。[コンテナレシピ]ページからレシピの新しいバージョンを作成する必要があります。詳細については、「新しいイメージレシピのバージョンを作成」を参照してください。
- インフラストラクチャー設定 — ビルドパイプラインインフラストラクチャーを構成するための編集可能な情報を表示します。
- [ディストリビューション設定] — Docker イメージディストリビューションに関する編集可能な情報を表示します。
- EventBridge rules – 選択したイベントバス について、 は現在のパイプラインをターゲットとする EventBridge ルールを表示します。 EventBridge イベントバスの作成とコンソールにリンクするルールアクションの作成が含まれます。詳細については、「 EventBridge ルールを使用する」を参照してください。

パイプラインのインフラストラクチャー設定を編集します。

インフラストラクチャ設定には以下の詳細が含まれており、パイプラインの作成後に編集できます。

- インフラストラクチャ構成のDescription。
- [IAM ロール] をインスタンスプロファイルに関連付けます。

- AWS インフラストラクチャ 。これには、インスタンスタイプと通知用の SNS トピックが含まれます。
- VPC、サブネット、セキュリティグループ。
- 障害発生時にインスタンスを終了する、接続用のキーペア、インスタンスLog用のオプションの S3 バケットの場所などの トラブルシューティング設定。

パイプラインの詳細ページからインフラストラクチャー設定を編集するには、以下の手順に従います。

1. [インフラストラクチャー構成の作成] を選択します。
2. [インフラストラクチャー詳細] パネルの右上隅から [編集] を選択します。
3. インフラストラクチャー設定に加えた更新を保存する準備ができたら、[変更を保存] を選択します。

パイプラインのディストリビューション設定を編集します。

ディストリビューション設定には以下の詳細が含まれており、パイプラインの作成後に編集できます。

- ディストリビューション 設定の説明。
- イメージを配布するリージョンの リージョン設定。リージョン 1 のデフォルトは、パイプラインを作成したリージョンです。リージョンを追加 ボタンで配信するリージョンを追加でき、リージョン 1 を除くすべてのリージョンを削除できます。

  地域設定には以下が含まれます。

  - ターゲット リージョン
  - サービスのデフォルトは「ECR」で、編集はできません。
  - リポジトリ名 - ターゲットリポジトリの名前（Amazon ECRの場所を含まない）。リポジトリ名と場所は以下の例のようになります。

    ```
    <account-id>.dkr.ecr.<region>.amazonaws.com/<repository-name>
    ```

> **Note**
>
> リポジトリ名を変更すると、名前が変更された後に作成されたイメージだけが新しい名前で追加されます。パイプラインで以前に作成したイメージは、すべて元のリポジトリに残ります。

パイプラインの詳細ページからディストリビューション設定を編集するには、次の手順に従います。

1. ディストリビューション設定タブを選択します。
2. ディストリビューション詳細パネルの右上隅から編集を選択します。
3. ディストリビューション設定に加えた更新を保存する準備ができたら、「変更を保存」を選択します。

パイプラインのビルドスケジュールを編集します。

パイプラインの編集ページには、パイプラインの作成後に編集できる以下の詳細が含まれています。

- パイプラインの [説明]。
- メタデータの収集が強化されました。デフォルトで有効になっています。オフにするには、「拡張メタデータ収集を有効にする」チェックボックスをオフにします。
- パイプラインの ビルドスケジュール。[スケジュールオプション]とこのセクションのすべての設定を変更できます。

パイプラインの詳細ページからパイプラインを編集するには、以下の手順に従います：

1. ページの右上にある アクション メニューで 削除 を選択します。
2. 更新を保存する準備ができたら、変更を保存の順に選択します。

> **Note**
>
> cron式を使ったビルドのスケジューリングについては、EC2 Image Builder で cron 式を使用するを参照してください。

コンテナイメージパイプライン (AWS CLI) の更新

AWS CLIのupdate-image-pipelineコマンドの入力として、JSONファイルを使用してコンテナイメージパイプラインを更新することができます。JSON ファイルを設定するには、以下の既存のリソースを参照する Amazon リソースネーム (ARN) が必要です。

- 更新するイメージパイプライン
- コンテナレシピ
- インフラストラクチャ設定
- ディストリビューション設定 (現在のパイプラインに含まれている場合)

> **Note**
>
> 配布設定リソースが含まれている場合、コマンドが実行されるリージョン (リージョン 1) のディストリビューション設定でターゲットリポジトリとして指定されている ECR リポジトリが、コンテナレシピで指定されているターゲットリポジトリよりも優先されます。

以下の手順に従い、 AWS CLIのupdate-image-pipelineコマンドを使用してコンテナイメージパイプラインを更新します。

> **Note**
>
> UpdateImagePipeline は、パイプラインの選択的な更新をサポートしていません。更新リクエストでは、変更されたプロパティだけでなく、必要なプロパティをすべて指定する必要があります。

1. CLI 入力 JSON ファイルの作成

   お気に入りのファイル編集ツールを使って、以下のキーと、あなたの環境で有効な値を持つJSONファイルを作成します。この例では、`create-component.json`という名前のファイルを使用します。

   ```
   {
    "imagePipelineArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-
   pipeline/my-example-pipeline",
   ```

```
 "containerRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:container-
recipe/my-example-recipe/2020.12.08",
 "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:infrastructure-configuration/my-example-infrastructure-
configuration",
 "distributionConfigurationArn": "arn:aws:imagebuilder:us-
west-2:123456789012:distribution-configuration/my-example-distribution-
configuration",
 "imageTestsConfiguration": {
  "imageTestsEnabled": true,
  "timeoutMinutes": 120
 },
 "schedule": {
  "scheduleExpression": "cron(0 0 * * MON *)",
  "pipelineExecutionStartCondition":
 "EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
 },
 "status": "DISABLED"
}
```

> **Note**
>
> - JSON ファイルパスの先頭に`file://`ノテーションを含める必要があります。
> - JSON ファイルのパスは、コマンドを実行するベースオペレーティングシステムに適した規則に従う必要があります。例えば、Windows ではディレクトリパスを参照するためにバックスプラッシュ (|) が使用され、Linux ではフォーワードスラッシュ (/) が使用されます。

2. 作成したファイルを入力として使用し、次のコマンドを実行します。

```
aws imagebuilder update-image-pipeline --cli-input-json file://update-image-
pipeline.json
```

EC2 Image Builder パイプラインのイメージワークフローを設定する

イメージワークフローでは、パイプラインが実行するワークフローを必要に応じてカスタマイズしてイメージを構築およびテストできます。定義するワークフローは、Image Builder ワークフローフ

レームワークのコンテキスト内で実行されます。ワークフローフレームワークを構成するステージの詳細については、「EC2 Image Builder イメージのビルドワークフローとテストワークフローを管理する」を参照してください。

ビルドワークフロー

ビルドワークフローは、ワークフローフレームワークの Build 段階で実行されます。パイプラインに指定できるビルドワークフローは 1 つのみです。または、ビルドを完全にスキップして、テスト専用パイプラインを設定することもできます。

テストワークフロー

テストワークフローはワークフローフレームワークの Test 段階で実行されます。最大 10 個のテストワークフローをパイプラインに指定できます。パイプラインのみをビルドしたい場合は、テストを完全にスキップすることもできます。

テストワークフローのテストグループを定義します。

テストワークフローはテストグループ内で定義されます。最大 10 個のテストワークフローをパイプラインに実行できます。テストワークフローを特定の順序で実行するか、できるだけ多くのワークフローを同時に実行するかを決定します。実行方法は、テストグループの定義方法によって異なります。以下のシナリオは、テストワークフローを定義するいくつかの方法を示したものです。

> **Note**
>
> コンソールを使用してワークフローを作成する場合は、テストグループを定義する前に、時間をかけてテストワークフローをどのように実行するかを計画することをお勧めします。コンソールでは、テストワークフローとグループを追加または削除できますが、順序を変更することはできません。

シナリオ 1: 一度に 1 つのテストワークフローを実行する

すべてのテストワークフローを 1 つずつ実行するには、それぞれに 1 つのテストワークフローを含むテストグループを 10 個まで設定できます。テストグループは、パイプラインに追加した順に 1 つずつ実行します。これは、テストワークフローが特定の順序で 1 つずつ実行されるようにするための 1 つの方法です。

シナリオ 2: 複数のテストワークフローを同時に実行する

順序は重要ではなく、できるだけ多くのテストワークフローを同時に実行したい場合は、1 つのテストグループを設定し、そのグループに最大数のテストワークフローを設定できます。Image Builder は同時に最大 5 つのテストワークフローを起動し、各テストワークフローが完了すると新しいテストワークフローを開始します。テストワークフローをできるだけ速く実行することが目標であれば、これがその方法の 1 つです。

シナリオ 3: 混合と組み合わせ

同時に実行できるテストワークフローと 1 つずつ実行する必要があるテストワークフローが混在するシナリオでは、この目標を達成するようにテストグループを設定できます。テストグループの設定方法に関する唯一の制限は、パイプラインで実行できるテストワークフローの最大数です

Image Builder パイプライン (コンソール) でのワークフローパラメータの設定

ワークフローパラメータは、ビルドワークフローとテストワークフローで同じように機能します。パイプラインを作成または更新するときは、含めたいビルドワークフローとテストワークフローを選択します。選択したワークフローのワークフロードキュメントでパラメータを定義した場合、Image Builder は [パラメータ] パネルにそのパラメータを表示します。パラメータが定義されていないワークフローでは、このパネルは非表示になります。

各パラメータには、ワークフロードキュメントで定義されている以下の属性が表示されます。

- [名前] (編集不可) – パラメータの名前。
- 型 (編集不可) - パラメータ値のデータ型。
- 値 – パラメータの値。パラメータ値を編集してパイプラインに設定できます。

Image Builder がワークフローアクションを実行するために使用する IAM サービスロールを指定します。

サービスアクセス

イメージワークフローを実行するには、Image Builder にワークフローアクションを実行する権限が必要です。AWSServiceRoleForImageBuilder サービスリンクロールを指定することも、次のように、サービスアクセス用の独自のカスタムロールを指定することもできます。

- コンソール — パイプラインウィザードの [ステップ 3: イメージ作成プロセスの定義] で、[サービスアクセス] パネルの [IAM ロール] リストから、サービスリンクロールまたは独自のカスタムロールを選択します。
- Image Builder API – CreateImageアクションリクエストで、 `executionRole`パラメータの値としてサービスにリンクされたロールまたは独自のカスタムロールを指定します。

サービスロールの作成方法の詳細については、「 ユーザーガイド」の「 AWS サービスにアクセス許可を委任するロールの作成AWS Identity and Access Management 」を参照してください。

イメージパイプラインを実行する

パイプラインに手動スケジュールオプションを選択した場合、パイプラインは手動でビルドを開始したときにのみ実行されます。自動スケジューリングオプションのいずれかを選択した場合は、定期的にスケジュールされた実行の合間に手動で実行することもできます。たとえば、通常は月に 1 回実行されるパイプラインがあるが、前回の実行の 2 週間後にコンポーネントの 1 つに更新を組み込む必要がある場合は、パイプラインを手動で実行することを選択できます。

Console

Image Builder コンソールのパイプライン詳細ページからパイプラインを実行するには、ページ上部のアクションメニューからパイプラインを実行を選択します。パイプラインが開始されたことや、エラーが発生したことを知らせるステータスメッセージが表示されます。

1. ページの右上にある アクション メニューで 削除 を選択します。
2. パイプラインの現在のステータスは [出力イメージ] タブの [ステータス] 列で確認できます。

AWS CLI

次の例では、 AWS CLI のstart-image-pipeline-executionコマンドを使って、イメージパイプラインを手動で開始する方法を示しています。このコマンドを実行すると、パイプラインは新しいイメージをビルドして配布します。

```
aws imagebuilder start-image-pipeline-execution --image-pipeline-arn
 arn:aws:imagebuilder:us-west-2:111122223333:image-pipeline/my-example-pipeline
```

ビルドパイプラインの実行時にどのようなリソースが作成されるかを確認するには、「作成されたリソース」を参照してください。

EC2 Image Builder で cron 式を使用する

EC2 Image Builder の cron 式を使用して、パイプラインのベースイメージとコンポーネントに適用される更新でイメージを更新するタイムウィンドウを設定します。パイプライン更新のタイムウィンドウは、cron 式で設定した時間から始まります。cron エクスプレッションの時間は分単位で設定できます。パイプラインビルドは開始時間以降にも実行できます。

ビルドの実行が開始されるまでに数秒、最長で 1 分かかることがあります。

> **Note**
>
> Cron 式は、デフォルトで協定世界時 (UTC) タイムゾーンを使用するか、タイムゾーンを指定できます。UTC 時間の詳細とタイムゾーンのオフセットについては、「タイムゾーンの略語 — ワールドワイドリスト」を参照してください。

Image Builderでサポートされるcron式の値

EC2 Image Builder は 6 つの必須フィールドで構成される cron 形式を使用します。各フィールドは、先頭や末尾にスペースを入れずに、間にスペースを入れて他のフィールドと区切られます。

<Minute> <Hour> <Day> <Month> <Day of the week> <Year>

次の表は、サポートされている必須 cron エントリの値の一覧です。

cron 式でサポートされている値

フィールド	値	ワイルドカード
分	0-59	, - * /
時	0-23	, - * /
日	1-31	, - * ? / L W
月	1-12 または jan-dec	, - * /
曜日。	1-7 または sun-sat	, - * ? L #
年	1970-2199	, - * /

ワイルドカード

次の表で、Image Builder で cron 式にワイルドカードを使用する方法について説明します。指定した時間が経過してからビルドが開始されるまでに最大 1 分かかる場合があることに注意してください。

cron 式でサポートされているワイルドカード

ワイルドカード	説明
,	, (カンマ) のワイルドカードには、追加の値が含まれます。 フィールドの、 `jan,feb,mar` は、1 月、2 月、3 月を含みます。
-	- (ダッシュ) のワイルドカードは、範囲を指定します。月日フィールドの1-15には、指定された月の1日から15日までが含まれ
*	* (アスタリスク) のワイルドカードには、フィールドのすべての値が含まれます。
?	? (疑問符) ワイルドカードは、フィールド値が別の設定に依存することを指定します。Day フィールドと D ay-of-week フィールドの場合、一方が指定されるか、可能なすべての値 (*) が含まれている場合、もう一方のフィールドは である必要があります?。両方を指定することはできません。例えば、7Day フィールドに を入力した場合 (その月の 7 日にビルドを実行) 、D ay-of-week ポジションには が含まれている必要があります?。
/	/ (スラッシュ) のワイルドカードは、増分を指定します。たとえば、ビルドを 1 日おきに実行したい場合は、「日」*/2 フィールドに入力します。
L	日付フィールドのどちらかにLのワイルドカードを指定すると、Lastの曜日を指定します：

ワイルドカード	説明
	28-31は月によって、日曜日は曜日によって指定されます。
W	Day-of-month フィールドの W ワイルドカードは平日を指定します。D ay-of-month フィールドに の前に数値を入力するとW、その日に最も近い曜日をターゲットにしたいことを意味します。たとえば、3W を指定した場合、ビルドは月の 3 日目に最も近い平日に実行する必要があります。
#	#(ハッシュ) は曜日フィールドにのみ使用でき、その後に1～5の数字を入力する必要があります。この数字は、特定の月のどの週をビルドの実行に適用するかを指定します。たとえば、ビルドを毎月の第 2 金曜日に実行したい場合は、「曜日」フィールドに fri#2 を使用してください。

制限事項

- 同じ cron 式で D フィールドay-of-month と D ay-of-week フィールドを指定することはできません。一方のフィールドに値 または*を指定する場合、もう一方のフィールドで?を使用する必要があります。
- 1 分より短い間隔を導き出す cron 式はサポートされていません。

EC2 Image Builderでのcron式の例

Cron 式は、Image Builder コンソールの場合と API や CLI の場合の入力方法が異なります。例を見るには、該当するタブを選択してください。

Image Builder console

以下の例は、ビルドスケジュールに合わせてコンソールに入力できる cron 式を示しています。UTC 時間形式。24 時間形式。

毎日午前 10:00 (UTC) に実行

```
0 10 * * ? *
```

毎日午後12時15分 (UTC) に実行

```
15 12 * * ? *
```

毎日午前0時 (UTC) に実行

```
0 0 * * ? *
```

平日毎朝10:00 (UTC) に実行

```
0 10 ? * 2-6 *
```

平日夕方6時 (UTC) に実行

```
0 18 ? * mon-fri *
```

毎月1日午前8時 (UTC) に実行

```
0 8 1 * ? *
```

毎月第2火曜日の午後 10 時 30 分 (UTC) に実行

```
30 22 ? * tue#2 *
```

> **Tip**
>
> 実行中にパイプラインジョブを翌日に延長したくない場合は、開始時間を指定する際にビルドの時間を考慮に入れてください。

API/CLI

以下の例は、CLIコマンドまたはAPIリクエストを使ってビルドスケジュールに入力できるcron式を示して cron 式のみが表示されます。

毎日午前 10:00 (UTC) に実行

```
cron(0 10 * * ? *)
```

毎日午後12時15分 (UTC) に実行

`cron(15 12 * * ? *)`

毎日午前0時 (UTC) に実行

`cron(0 0 * * ? *)`

平日毎朝10:00 (UTC) に実行

`cron(0 10 ? * 2-6 *)`

平日夕方6:00 (UTC) に実行

`cron(0 18 ? * mon-fri *)`

毎月1日午前8時 (UTC) に実行

`cron(0 8 1 * ? *)`

毎月第2火曜日の午後 10 時 30 分 (UTC) に実行

`cron(30 22 ? * tue#2 *)`

> **Tip**
>
> 実行中にパイプラインジョブを翌日に延長したくない場合は、開始時間を指定する際にビルドの時間を考慮に入れてください。

EC2 Image Builderのrate 式

rate 式は、予定されたイベントルールを作成すると開始され、その定義済されたスケジュールに基づいて実行されます。

rate 式は 2 つの必須フィールドがあります。フィールドは空白で区切ります。

構文

```
rate(value unit)
```

value

正数。

単位

時刻の単位。値 1 には、`minute` などさまざまな単位が必要です。また、1 を超える値には `minutes` などの単位が必要です。

有効な値: minute | minutes | hour | hours | day | days

制限事項

値が 1 に等しい場合、単位は単数形であることが必要です。同様に、1 より大きい値の場合、単位は複数であることが必要です。たとえば、`rate(1 hours)` と `rate(5 hour)` は有効ではありませんが、`rate(1 hour)` と `rate(5 hours)` は有効です。

Image Builder パイプラインで EventBridge ルールを使用する

さまざまな AWS およびパートナーサービスからのイベントは、ほぼリアルタイムで Amazon EventBridge イベントバスにストリーミングされます。カスタムイベントを生成し、独自のアプリケーションから にイベントを送信することもできます EventBridge。イベントバスはルールを使用してイベントデータのルーティング先を決定します。

Image Builder パイプラインは EventBridge ルールターゲットとして使用できます。つまり、バス上のイベントに応答するために作成したルール、またはスケジュールに基づいて Image Builder パイプラインを実行できます。

Image Builder が に送信するシステム生成イベントの概要については EventBridge、「」を参照してくださいImage Builder が送信するイベントメッセージ。

> **Note**
>
> イベントバスはリージョン固有です。ルールとターゲットは同じリージョンに存在する必要があります。

内容

EventBridge 用語

このセクションでは、 が Image Builder パイプラインとどのように EventBridge 統合されるかを理解するのに役立つ用語の概要を示します。

イベント

1つまたは複数のアプリケーションリソースに影響を与える可能性のある環境の変更について記述 環境は、 AWS 環境、SaaS パートナーサービスまたはアプリケーション、またはアプリケーションまたはサービスのいずれかです。タイムラインに予定されたイベントをセットアップすることもできます。

イベントバス

アプリケーションやサービスからイベントデータを受け取るパイプライン。

ソース

イベントをイベントバスに送信したサービスまたはアプリケーション。

Target

ルールに一致すると が呼び出すリソースまたはエンドポイント EventBridge で、イベントからターゲットにデータを配信します。

ルール

ルールは、受信したイベントをマッチングし、処理のためにターゲットに送信します。1つのルールで複数のターゲットにイベントを送信し、それを並行して実行することができます。ルールは、イベントパターンまたはスケジュールに基づいて作成します。

パターン

イベントパターンは、ターゲットアクションを開始するために、ルールがマッチするイ ベント構造とフィールドを定義します。

スケジュール

スケジュールルールは、Image Builder パイプラインを実行して四半期ごとにイメージを更新するなど、スケジュールに基づいてアクションを実行します。スケジュール表現には次の 2 種類があります。

- Cron 式 — たとえば、特定の日に毎週実行するなど、単純な基準を概説できる cron 構文を使用して、特定のスケジューリング条件を照合します。また、月の 5 日目の午前 2 時から午前 4 時の間に四半期ごとに実行するなど、より複雑な条件を設定することもできます。

- レート表現 — 12 時間ごとなど、ターゲットを定期的に呼び出す間隔を指定します。

Image Builder パイプラインの EventBridge ルールを表示する

Image Builder Image Pipelines の詳細ページのEventBridge ルールタブには、アカウントがアクセスできる EventBridge イベントバスと、現在のパイプラインに適用される選択したイベントバスのルールが表示されます。このタブは、新しいリソースを作成するためにEventBridge コンソールに直接リンクします。

EventBridge コンソールにリンクするアクション

- イベントバスを作成
- ルールの作成

の詳細については EventBridge、「Amazon EventBridge ユーザーガイド」の以下のトピックを参照してください。

- Amazon とは EventBridge
- Amazon EventBridge イベントバス
- Amazon EventBridge イベント
- Amazon EventBridge ルール

EventBridge ルールを使用してパイプライン構築をスケジュールする

この例では、rate 式を使用してデフォルトイベントバス用の新しいスケジュールルールを作成します。この例のルールは 90 日ごとにイベントバスでイベントを生成します。このイベントは、イメージを更新するパイプラインの構築を開始します。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. アカウントで作成されたビルドパイプラインのリストを表示するには、ナビゲーションペインから [Image pipelines] を選択します。

> **Note**
>
> イメージパイプラインのリストには、パイプラインによって作成される出力イメージのタイプ (AMI または Docker) のインジケータが含まれています。

3. 詳細を表示したりパイプラインを編集したりするには、[パイプライン名] のリンクを選択します。パイプラインの詳細ビューが開きます。

> **Note**
>
> また、パイプライン名の横にあるチェックボックスを選択し、詳細を表示を選択することもできます。

4. EventBridge ルールタブを開きます。
5. Event Busパネルであらかじめ選択されているデフォルトのイベントバスはそのままにしておきます。
6. ルールの作成を選択します。これにより、Amazon EventBridge コンソールのルールの作成ページに移動します。
7. ルールの名前と説明を入力します。ルール名は、選択したリージョンのイベントバス内で一意である必要があります。
8. パターンを定義パネルで、スケジュールオプションを選択します。これによりパネルが拡張され、すべての固定レートオプションが選択されます。
9. 90 最初のボックスに入力し、ドロップダウンリストからDaysを選択します。
10. ターゲットの選択 パネルで以下のアクションを実行します。
    a. Targetのドロップダウンリストから`EC2 Image Builder`を選択します。
    b. Image Builder パイプラインにルールを適用するには、Image Pipelineドロップダウンリストからターゲットパイプラインを選択します。
    c. EventBridge には、選択したパイプラインのビルドを開始するためのアクセス許可が必要です。この例では、この特定のリソースに対して新しいロールを作成するをデフォルトのままにします。
    d. Add target を選択します。
11. 作成を選択します。

> **Note**
>
> この例で説明されていないレート式ルールの設定の詳細については、「Amazon EventBridge ユーザーガイド」の「レート式」を参照してください。

EC2 Image Builder に製品およびサービスの統合

EC2 Image Builder は AWS Marketplace 、堅牢で安全なカスタムマシンイメージの作成に役立つように、 およびその他の AWS のサービス およびアプリケーションと統合されています。

製品

Image Builder レシピには、次のように、 AWS Marketplace および Image Builder マネージドコンポーネントからのイメージ製品を組み込むことで、特殊なビルドおよびテスト機能を提供できます。

- AWS Marketplace イメージ製品 – CIS Hardening などの組織基準を満たすために、レシピのベースイメージ AWS Marketplace として のイメージ製品を使用します。Image Builder コンソールからレシピを作成する場合、既存のサブスクリプションから選択するか、 AWS Marketplaceの特定の製品を検索できます。Image Builder API、CLI、または SDK からレシピを作成する場合、ベースイメージとして使用するイメージ製品の Amazon リソースネーム (ARN) を指定できます。
- AWSTOE コンポーネント – recipe で指定するコンポーネントは、ソフトウェアのインストールやコンプライアンス検証など、ビルドおよびテストアクションを実行できます。サブスクライブしている一部の AWS Marketplace のイメージ製品には、レシピで使用できるコンパニオンコンポーネントが含まれている場合があります。CIS 強化イメージには、レシピで CIS Benchmarks Level 1 ガイドラインを設定に適用するために使用できる一致する AWSTOE コンポーネントが含まれています。

> **Note**
>
> コンプライアンス関連製品の詳細については、「Image Builder イメージ用のコンプライアンス製品」を参照してください。

サービス

Image Builder は以下と統合 AWS のサービス され、詳細なイベントメトリクス、ログ記録、モニタリングを提供します。この情報は、アクティビティの追跡、イメージビルドの問題のトラブルシューティング、イベント通知に基づく自動化の作成に役立ちます。

- AWS CloudTrail – に送信される Image Builder イベントをモニタリングします CloudTrail。の詳細については CloudTrail、「 ユーザーガイド」の「 とは AWS CloudTrailAWS CloudTrail 」を参照してください。

- Amazon CloudWatch Logs – Image Builder ログファイルのモニタリング、保存、アクセスを行います。オプションで、ログを S3 バケットに保存できます。 CloudWatch ログの詳細については、「Amazon CloudWatch Logs ユーザーガイド」の「Amazon Logs とは CloudWatch 」を参照してください。
- Amazon EventBridge – アカウント内の Image Builder アクティビティからのリアルタイムイベントデータのストリームに接続します。の詳細については EventBridge、「Amazon ユーザーガイド」の「Amazon EventBridgeとは」を参照してください。 EventBridge
- Amazon Inspector – Image Builder が新しいイメージの作成を起動する EC2 テストインスタンスの自動スキャンにより、ソフトウェアとネットワーク設定の脆弱性を検出します。Image Builder は出力イメージリソースの検出結果を保存するので、テストインスタンスの終了後に調査と修正を行うことができます。スキャンと価格の詳細については、Amazon Inspectorユーザーガイドの Amazon Inspectorとは を参照してください。

  拡張スキャンを設定した場合、Amazon Inspector は ECR リポジトリをスキャンすることもできます。詳細については、Amazon Inspector User Guideの Amazon ECRコンテナイメージのスキャン を参照してください。

  > **Note**
  >
  > Amazon Inspector は有料機能です。

- AWS Marketplace - 現在の AWS Marketplace 製品サブスクリプションのリストを表示し、Image Builderから直接イメージ製品を検索できます。購読しているイメージプロダクトを Image Builder レシピのベースイメージとして使用することもできます。 AWS Marketplace サブスクリプションの管理の詳細については、「 AWS Marketplace 購入者ガイド」を参照してください。
- Amazon Simple Notification Service (Amazon SNS) - 設定されている場合、あなたが購読しているSNSトピックにイメージのステータスに関する詳細なメッセージを公開します。Amazon SNS の詳細については、Amazon Simple Notification Service デベロッパーガイドの、「Amazon SNS とは」を参照してください。

製品およびサービスの統合に関するトピック

- AWS CloudTrail Image Builder での統合
- Image Builder での Amazon CloudWatch Logs の統合
- Image Builder での Amazon EventBridge の統合
- Image Builder へAmazon Inspector の統合

- AWS Marketplace Image Builder での統合
- Image BuilderにおけるAmazon SNSの統合
- Image Builder イメージ用のコンプライアンス製品

AWS CloudTrail Image Builder での統合

このサービスは をサポートします AWS CloudTrail。 CloudTrail は、 の AWS 呼び出しを記録 AWS アカウント し、ログファイルを Amazon S3 バケットに配信するサービスです。によって収集された情報を使用して CloudTrail、 に対して正常に行われたリクエスト AWS のサービス、リクエスト者、リクエスト日時などを判断できます。Image Builder との統合の詳細については CloudTrail、「」を参照してください を使用した EC2 Image Builder API コールのログ記録 AWS CloudTrail。

をオンにしてログファイルを検索する方法など CloudTrail、 の詳細については、AWS CloudTrail 「 ユーザーガイド」を参照してください。

Image Builder での Amazon CloudWatch Logs の統合

CloudWatch ログのサポートはデフォルトで有効になっています。ログはビルドプロセス中にインスタンスに保持され、 CloudWatch ログにストリーミングされます。インスタンスログは、イメージが作成される前にインスタンスから削除されます。

ビルドログは、Image Builder CloudWatch Logs グループとストリームに従って にストリーミングされます。

LogGroup:

```
/aws/imagebuilder/ImageName
```

LogStream (x.x.x/x):

```
ImageVersion/ImageBuildVersion
```

インスタンスプロファイルに関連付けられている次のアクセス許可を削除することで、 CloudWatch ログストリーミングをオプトアウトできます。

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
```

```
            "logs:CreateLogStream",
            "logs:CreateLogGroup",
            "logs:PutLogEvents"
        ],
        "Resource": "arn:aws:logs:*:*:log-group:/aws/imagebuilder/*"
    }
]
```

高度なトラブルシューティングを行う場合は、「AWS Systems Manager コマンドを実行」を使用して定義済みのコマンドとスクリプトを実行できます。詳細については、「EC2Image Builderのトラブルシューティング」を参照してください。

Image Builder での Amazon EventBridge の統合

Amazon EventBridge は、Image Builder アプリケーションを他の からの関連データに接続するために使用できるサーバーレスイベントバスサービスです AWS のサービス。では EventBridge、ルールは受信イベントを照合し、処理のためにターゲットに送信します。1つのルールで複数のターゲットにイベントを送信でき、これらのイベントは並行して実行されます。

を使用すると EventBridge、 を自動化 AWS のサービス し、アプリケーションの可用性の問題やリソースの変更などのシステムイベントに自動的に対応できます。からの AWS のサービス イベントは、ほぼリアルタイムで EventBridge に配信されます。受信イベントに反応するルールを設定して、アクションを開始できます。例えば、EC2 インスタンスのステータスが保留中から実行中に変わったときに Lambda 関数にイベントを送信する場合などです。これらはパターンと呼ばれる。イベントパターンに基づいてルールを作成するには、「Amazon ユーザーガイド」の「イベントに反応するAmazon EventBridge ルールの作成」を参照してください。 EventBridge

自動的に開始されるアクションには以下のようなものがありまれます。

- AWS Lambda 関数を呼び出す
- Amazon EC2 コマンドを実行 を呼び出す
- Amazon Kinesis Data Streams へのイベントを中継する
- AWS Step Functions ステートマシンをアクティブ化する
- Amazon SNSトピックまたはAmazon SQSキューへの通知

また、Image Builder パイプラインを実行して四半期ごとにイメージを更新するなど、デフォルトのイベントバスに定期的にアクションを実行するスケジューリングルールを設定することもできます。スケジュール表現には次の 2 種類があります。

- cron式 - 以下のcron式の例は、毎日正午（UTC+0）にタスクを実行するようにスケジュールします：

  ```
  cron(0 12 * * ? *)
  ```

  で cron 式を使用する方法の詳細については EventBridge、「Amazon EventBridge ユーザーガイド」の「Cron 式」を参照してください。
- rate 式 - 以下の rate 式の例は、12時間ごとにタスクを実行するようにスケジュールします：

  ```
  rate(12 hour)
  ```

  での rate 式の使用の詳細については EventBridge、「Amazon EventBridge ユーザーガイド」の「Rate 式」を参照してください。

EventBridge ルールと Image Builder イメージパイプラインの統合方法の詳細については、「」を参照してくださいImage Builder パイプラインで EventBridge ルールを使用する。

Image Builder が送信するイベントメッセージ

Image Builder は、Image Builder リソースのステータスに重大な変更 EventBridge があると、イベントメッセージを に送信します。例えば、イメージの状態が変更された場合などです。次の例は、Image Builder が送信する可能性のある一般的な JSON イベントメッセージを示しています。

EC2 Image Builder イメージの状態変更

Image Builder は、イメージの作成中にイメージリソースの状態が変更されたときにこのイベントを送信します。例えば、次のようにイメージのステータスが 1 つの状態から別の状態に変わった場合です。

- `building` から `testing` へ
- `testing` から `distribution` へ
- `testing` から `failed` へ
- `integrating` から `available` へ

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "EC2 Image Builder Image State Change",
    "source": "aws.imagebuilder",
```

```
    "account": "111122223333",
    "time": "2024-01-18T17:50:56Z",
    "region": "us-west-2",
    "resources": ["arn:aws:imagebuilder:us-west-2:111122223333:image/
cmkencryptedworkflowtest-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222/1.0.0/1"],
    "detail": {
        "previous-state": {
            "status": "TESTING"
        },
        "state": {
            "status": "AVAILABLE"
        }
    }
}
```

EC2 Image Builder CVE が検出されました

イメージに対して CVE 検出が有効になっている場合、Image Builder はイメージスキャンが完了するたびに結果を含むメッセージを送信します。

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "EC2 Image Builder CVE Detected",
    "source": "aws.imagebuilder",
    "account": "111122223333",
    "time": "2023-03-01T16:59:09Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:imagebuilder:us-east-1:111122223333:image/test-image/1.0.0/1",
        "arn:aws:imagebuilder:us-east-1:111122223333:image-pipeline/test-pipeline"
    ],
    "detail": {
        "resource-id": "i-1234567890abcdef0",
        "finding-severity-counts": {
            "all": 0,
            "critical": 0,
            "high": 0,
            "medium": 0
        }
    }
}
```

EC2 Image Builder ワークフローステップ待機

Image Builder は、WaitForActionワークフローステップが一時停止して非同期アクションが完了するのを待つときにメッセージを送信します。

```
{
    "version": "0",
    "id": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "detail-type": "EC2 Image Builder Workflow Step Waiting",
    "source": "aws.imagebuilder",
    "account": "111122223333",
    "time": "2024-01-18T16:54:44Z",
    "region": "us-west-2",
    "resources": ["arn:aws:imagebuilder:us-west-2:111122223333:image/
workflowstepwaitforactionwithvalidsnstopictest-a1b2c3d4-5678-90ab-cdef-
EXAMPLE22222/1.0.0/1", "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/
build-workflow-a1b2c3d4-5678-90ab-cdef-EXAMPLE33333/1.0.0/1"],
    "detail": {
        "workflow-execution-id": "wf-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
        "workflow-step-execution-id": "step-a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "workflow-step-name": "TestAutoSNSStop"
    }
}
```

Image Builder へAmazon Inspector の統合

Amazon Inspector でセキュリティスキャンを有効にすると、アカウント内のマシンイメージと実行中のインスタンスが継続的にスキャンされ、オペレーティングシステムとプログラミング言語の脆弱性が検出されます。有効にすると、セキュリティスキャンが自動的に実行され、Image Builder は、新しいイメージを作成するときに、テストインスタンスの検出結果のスナップショットを保存できます。Amazon Inspector は有料機能です。

Amazon Inspector は、ソフトウェアまたはネットワーク設定の脆弱性を発見すると、次のアクションを実行します。

- 検出結果があったこと通知します。
- 検出結果の重要度評価 重要度評価では、検出結果の優先順位付けに役立つように脆弱性を分類します。評価には次の値が含まれます。
  - トリアージされていない

- 情報
- 低
- 中程度
- 高い
- [非常事態]

- 検出結果に関する情報と、詳細情報が含まれる追加リソースへのリンクを提供します。
- 検出結果を生成した問題の解決に役立つ修正ガイダンスを提供します。

セキュリティスキャンの設定

アカウントの Amazon Inspector を有効にした場合、Amazon Inspector は Image Builder が起動する EC2 インスタンスを自動的にスキャンして、新しいイメージをビルドしてテストします。これらのインスタンスはビルドとテストのプロセス中は有効期間が短く、検出結果は通常、インスタンスがシャットダウンするとすぐに期限切れになります。新しいイメージの検出結果の調査と修正に役立つように、Image Builder では、ビルドプロセス中に Amazon Inspector がテストインスタンスについて特定した検出結果をオプションでスナップショットとして保存できます。

パイプラインのセキュリティスキャンを設定する方法については、「で Image Builder イメージのセキュリティスキャンを設定する AWS Management Console」を参照してください。

セキュリティ調査結果を確認する

Image Builder コンソールでは、すべての Image Builder リソースのセキュリティ結果を 1 か所に表示できます。すべての結果は セキュリティ概要 セクションの セキュリティ結果 ページで確認することも、脆弱性別、イメージパイプライン別、またはイメージ別にグループ化することもできます。コンソールにはデフォルトですべてのセキュリティ結果が表示されます。すべてのセキュリティ結果オプションの概要パネルには、各重要度レベルの結果の数が表示されます。詳細については、「で Image Builder イメージのセキュリティ検出結果を管理する AWS Management Console」を参照してください。

Amazon Inspectorの脆弱性調査結果の詳細については、Amazon InspectorユーザーガイドのAmazon Inspectorの調査結果を理解するを参照してください。

AWS Marketplace Image Builder での統合

AWS Marketplace は、ビジネスニーズに合ったソリューションの構築に役立つサードパーティーのソフトウェア、データ、サービスを検索してサブスクライブできる厳選されたデジタルカタログで

す。 は、認証済みの購入者と登録済みの販売者を、セキュリティ、ネットワーク、ストレージ、機械学習などの一般的なカテゴリのソフトウェアリストとともに AWS Marketplace 提供します。

AWS Marketplace 販売者は、独立系ソフトウェアベンダー (ISV)、リセラー、または AWS 製品やサービスと連携する何かを提供する個人です。販売者が で製品を送信すると AWS Marketplace、製品の価格と利用規約が定義されます。買い手は、オファーに設定された価格、条件、条項に同意します。の詳細については AWS Marketplace、「 とは」を参照してください AWS Marketplace。

> **Note**
>
> データ製品プロバイダーは、 AWS Data Exchange の資格要件を満たしている必要があります。詳細については、AWS Data Exchange User Guide の AWS データ交換 でのデータ製品の提供を参照してください。

AWS Marketplace 統合機能

Image Builder は と統合 AWS Marketplace され、Image Builder コンソールから直接以下の機能を提供します。

- で利用可能なイメージ製品を検索します AWS Marketplace。
- 現在の AWS Marketplace 製品サブスクリプションのリストを参照してください。
- AWS Marketplace Image Builder レシピのベースイメージとして Image 製品を使用します。

関連付けられた AWS Task Orchestrator and Executor （ AWSTOE) コンポーネントを含む製品の場合、コンソールおよび API、SDK、CLI で製品所有者をフィルタリングできます。詳細については、「AWSTOE コンポーネントを一覧表示する」を参照してください。

Image Builder コンソールから AWS Marketplace イメージ製品を検索する

Image Builder は と統合され AWS Marketplace 、Image Builder コンソールの AWS Marketplaceセクションから直接イメージ製品のサブスクリプションを表示します。Image Builder コンソールを離れずに、イメージ製品ページから AWS Marketplace イメージ製品を検索することもできます。

Image Builder コンソールから AWS Marketplace イメージ製品を検索するには、次の手順に従います。

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。

2. ナビゲーションペインから、AWS Marketplace セクション内のイメージ製品を選択します。
3. イメージ製品ページには、サブスクリプションタブでサブスクリプションしているイメージ製品の概要が表示されます。また、AWS Marketplace タブでイメージ製品を検索することもできます。

   Image Builder は、Image Builder レシピで使用できるマシンイメージに焦点を当て AWS Marketplace るために、 から製品を事前にフィルタリングします。Image Builder と AWS Marketplace の統合の詳細については、表示したいものに一致するタブを選択してください。

   AWS Marketplace

   このタブには2つのパネルがあります。左側の結果の絞り込みパネルでは、結果を絞り込んで購読したい製品を見つけることができます。右側の「製品を検索」パネルには、フィルター条件を満たす製品が表示されます。また、製品名で検索することもできます。

   結果を絞り込む

   以下のリストは、商品検索に適用できるフィルターのほんの一部を示しています。

   - インフラストラクチャーソフトウェアや機械学習など、1 つ以上の製品カテゴリを選択します。
   - イメージ製品のオペレーティングシステムを選択するか、特定のオペレーティングシステムプラットフォーム用のすべての製品All Linux/Unixなどを選択します。
   - 1 つまたは複数のパブリッシャーを選択して、販売可能な製品を表示します。すべて表示リンクを選択すると、適用したフィルターに適合する製品を販売しているすべてのパブリッシャーが表示されます。

     > **Note**
     >
     > パブリッシャー名はアルファベット順に並んでいません。たとえば、特定のパブリッシャーをお探しの場合は `Center for Internet Security`、すべての出版者 ダイアログの上部にある検索ボックスに名前の一部を入力できます。名前は略語として入力してください。たとえば `CIS` 探している結果が得られない場合があります。
     > パブリッシャー名をページごとに閲覧することもできます。

フィルターの選択肢は動的です。選択するたびに、他のすべてのカテゴリのオプションに影響します。には何千もの製品があるため AWS Marketplace、フィルタリングできるほど、必要な製品を見つける可能性が高くなります。

製品を検索します

特定の製品を名前で検索するには、パネル上部の検索バーに名前の一部を入力します。各製品結果には以下の詳細が含まれます。

- 製品名とロゴ。これらは両方とも、 AWS Marketplaceの製品詳細ページにリンクされています。詳細ページはブラウザの新しいタブで開きます。Image Builder レシピで使用したい場合は、そこからイメージプロダクトをサブスクライブできます。詳しくはAWS Marketplace バイヤーガイドの商品購入をご覧ください。

  でイメージ製品をサブスクライブする場合は AWS Marketplace、ブラウザの Image Builder タブに戻り、サブスクライブしたイメージ製品のリストを更新して表示します。

  > **Note**
  >
  > 新しいサブスクリプションが利用可能になるまでに数分かかることがあります。

- パブリッシャー名。これは、 のパブリッシャーの詳細ページにリンクされています AWS Marketplace。パブリッシャーの詳細ページがブラウザの新しいタブで開きます。
- 製品バージョン。
- 商品の星評価と、 AWS Marketplaceの商品詳細ページのレビューセクションへの直接リンク。詳細ページはブラウザの新しいタブで開きます。
- 製品説明の最初の数行。

検索バーのすぐ下には、検索によって生成された結果の数と、現在表示されている結果のサブセットが表示されます。パネルの右側にあるその他のコントロールを使用して、一度に表示する商品の数や、結果に適用するソート順序の設定を調整できます。また、ページネーションコントロールを使用して、結果を確認することもできます。

Subscriptions

このタブには、 でサブスクライブしているイメージ製品のリストが表示されます AWS Marketplace。登録した各製品には、以下の詳細が表示されます。

- 製品名。これは、 の製品詳細ページにリンクされています AWS Marketplace。購読製品の製品詳細ページが、ブラウザの新しいタブで開きます。
- パブリッシャー名。これは、 のパブリッシャーの詳細ページにリンクされています AWS Marketplace。パブリッシャーの詳細ページがブラウザの新しいタブで開きます。
- 購読している製品バージョン。
- サブスクライブした製品に関連コンポーネントが含まれている場合、Image Builder は AWSTOE コンポーネントの詳細へのリンクを表示します。

ページの上部では、特定の製品を名前で検索したり、ページ区切りコントロールを使用して結果をページ表示したりできます。サブスクライブされた商品を新しいレシピのベースイメージとして使用するには、サブスクライブされた商品を選択して Create new recipe を選択します。Image Builder は、デフォルトでリストの最初の商品を事前に選択します。

> **Note**
>
> サブスクライブしたばかりの商品を探していて、その商品がリストに表示されない場合は、タブ上部の更新ボタンを使用して結果を更新してください。新しいサブスクリプションがリストに表示されるまでに数分かかることがあります。

Image Builder レシピで AWS Marketplace イメージ製品を使用する

Image Builder コンソールでは、サブスクライブしているイメージプロダクトの 1 つに基づいて新しいイメージレシピを作成する方法が 2 つあります。

1. 以下のようにイメージプロダクトページから開始できます。

    1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
    2. ナビゲーションペインから、AWS Marketplace セクション内のイメージ製品を選択します。
    3. サブスクリプションタブを開きます。
    4. レシピのベースとして使用する登録済みのイメージ製品を選択します。
    5. レシピを作成する を選択します。これにより、レシピの作成ページが開き、AWS Marketplace イメージオプションが表示され、サブスクライブしたイメージ製品があらかじめ選択されています。

6. 通常どおり、レシピの残りの設定を行います。イメージレシピについては[イメージレシピの新しいバージョンを作成します。]を参照。

2. レシピの作成ページを開き、ベース AWS Marketplace イメージとして使用するイメージ製品を選択することもできます。

   1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
   2. ナビゲーションペインから、AWS Marketplace セクションのイメージレシピを選択します。作成したイメージレシピのリストが表示されます。
   3. [イメージレシピの作成] を選択します。これにより、レシピの作成ページが開きます。
   4. レシピの詳細 セクションで、名前 と バージョン を入力します。
   5. ベースイメージセクションで、AWS Marketplace イメージオプションを選択します。サブスクリプションタブにサブスクライブしている AWS Marketplace イメージ製品のリストが表示されます。リストからベースイメージを選択できます。

      AWS Marketplace タブ AWS Marketplace から直接 で利用可能な他のイメージ製品を検索することもできます。製品を追加 を選択するか、AWS Marketplace タブを直接開きます。でフィルターと検索を設定する方法の詳細については、 AWS Marketplace「」を参照してくださいImage Builder コンソールから AWS Marketplace イメージ製品を検索する。
   6. 残りの詳細を通常どおり入力し、Create recipeを選択します。

> **Note**
>
> イメージ製品のサブスクリプションに AWSTOE ビルドコンポーネントが含まれている場合は、ビルドコンポーネントリストから選択できます。コンポーネント所有者のタイプリストから `Third party managed` を選択すると表示されます。製品サブスクリプションに AWSTOE テストコンポーネントが含まれている場合は、テストコンポーネントリストと同じ手順に従ってください。

Image BuilderにおけるAmazon SNSの統合

Amazon Simple Notification Service (Amazon SNS)は、パブリッシャーからサブスクライバー (プロデューサーとコンシューマーとも呼ばれる) への非同期メッセージ配信を提供するマネージドサービス

SNS トピックはインフラストラクチャ設定で指定できます。イメージを作成したりパイプラインを実行したりすると、Image Builder はイメージステータスに関する詳細なメッセージをこのトピックに公開できます。イメージステータスが以下のいずれかの状態になると、Image Builder はメッセージを公開します。

- `AVAILABLE`
- `FAILED`

Image Builder からの SNS メッセージの例については、「メッセージ形式」を参照してください。新しい SNS トピックを作成したい場合は、Amazon Simple Notification Service Developer Guide の Getting started with Amazon SNS を参照してください。

暗号化 SNS トピック

SNS トピックが暗号化されている場合は、Image Builder サービスロールが以下のアクションを実行するためのアクセス許可を AWS KMS key ポリシーで付与する必要があります。

- `kms:Decrypt`
- `kms:GenerateDataKey`

> **Note**
>
> SNS トピックが暗号化されている場合、このトピックを暗号化するキーは、Image Builder サービスが実行されるアカウントにある必要があります。Image Builder は、他のアカウントのキーで暗号化された SNS トピックに通知を送信できません。

KMS キーポリシーの追加例

次の例は、KMS キーポリシーに追加するセクションを示しています。Image Builder イメージを最初に作成したときに Image Builder がアカウントで作成した IAM サービスにリンクされたロールには Amazon リソースネーム (ARN) を使用してください。Image Builderのサービス連動ロールについては、EC2 Image Builder サービスにリンクされたロールを使用するを参照してください。

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
```

```
      "AWS": "arn:aws:iam::123456789012:role/aws-service-role/
imagebuilder.amazonaws.com/AWSServiceRoleForImageBuilder"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

ARNを取得するには、以下のいずれかの方法を使用できます。

AWS Management Console

Image Builder がアカウントで作成したサービスにリンクされたロールの ARN を から取得するには AWS Management Console、次の手順に従います。

1. IAM コンソール (https://console.aws.amazon.com/iam/) を開きます。
2. 左のナビゲーションペインで、[ロール] を選択します。
3. `ImageBuilder` を検索して、`AWSServiceRoleForImageBuilder` の結果から次のロール名を選択します。ロールの詳細ページが表示されます。
4. Query ID (クエリ ID) の横にあるアイコンをクリックして、ID をクリップボードにコピーします。

AWS CLI

Image Builder がアカウントで作成したサービスにリンクされたロールの ARN を から取得するには AWS CLI、次のように IAM get-role コマンドを使用します。

```
aws iam get-role --role-name AWSServiceRoleForImageBuilder
```

部分的なサンプル出力：

```
{
    "Role": {
        "Path": "/aws-service-role/imagebuilder.amazonaws.com/",
        "RoleName": "AWSServiceRoleForImageBuilder",
        ...
```

```
        "Arn": "arn:aws:iam::123456789012:role/aws-service-role/
imagebuilder.amazonaws.com/AWSServiceRoleForImageBuilder",
        ...
}
```

メッセージ形式

Image Builder が Amazon SNS トピックにメッセージを公開すると、そのトピックを購読する他のサービスがメッセージ形式をフィルタリングして、今後のアクションの基準を満たしているかどうかを判断できます。たとえば、成功メッセージによって、 AWS Systems Manager パラメータストアを更新するタスクや、出力 AMI の外部コンプライアンステストワークフローを起動するタスクが開始される場合があります。

次の例は、パイプラインビルドが完了するまで実行され、Linux イメージを作成したときに Image Builder が公開する一般的なメッセージの JSON ペイロードを示しています。

```
{
  "versionlessArn": "arn:aws:imagebuilder:us-west-1:123456789012:image/example-linux-
image",
  "semver": 123794003928538027489912 4227,
  "arn": "arn:aws:imagebuilder:us-west-1:123456789012:image/example-linux-
image/1.0.0/3",
  "name": "example-linux-image",
  "version": "1.0.0",
  "type": "AMI",
  "buildVersion": 3,
  "state": {
    "status": "AVAILABLE"
  },
  "platform": "Linux",
  "imageRecipe": {
    "arn": "arn:aws:imagebuilder:us-west-1:123456789012:image-recipe/example-linux-
image/1.0.0",
    "name": "amjule-barebones-linux",
    "version": "1.0.0",
    "components": [
      {
        "componentArn": "arn:aws:imagebuilder:us-west-1:123456789012:component/update-
linux/1.0.2/1"
      }
    ],
```

```
    "platform": "Linux",
    "parentImage": "arn:aws:imagebuilder:us-west-1:987654321098:image/amazon-linux-2-
x86/2022.6.14/1",
    "blockDeviceMappings": [
      {
        "deviceName": "/dev/xvda",
        "ebs": {
          "encrypted": false,
          "deleteOnTermination": true,
          "volumeSize": 8,
          "volumeType": "gp2"
        }
      }
    ],
    "dateCreated": "Feb 24, 2021 12:31:54 AM",
    "tags": {
      "internalId": "1a234567-8901-2345-bcd6-ef7890123456",
      "resourceArn": "arn:aws:imagebuilder:us-west-1:123456789012:image-recipe/example-
linux-image/1.0.0"
    },
    "workingDirectory": "/tmp",
    "accountId": "462045008730"
  },
  "sourcePipelineArn": "arn:aws:imagebuilder:us-west-1:123456789012:image-pipeline/
example-linux-pipeline",
  "infrastructureConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-1:123456789012:infrastructure-configuration/
example-linux-infra-config-uswest1",
    "name": "example-linux-infra-config-uswest1",
    "instanceProfileName": "example-linux-ib-baseline-admin",
    "tags": {
      "internalId": "234abc56-d789-0123-a4e5-6b789d012c34",
      "resourceArn": "arn:aws:imagebuilder:us-west-1:123456789012:infrastructure-
configuration/example-linux-infra-config-uswest1"
    },
    "logging": {
      "s3Logs": {
        "s3BucketName": "12345-example-linux-testbucket-uswest1"
      }
    },
    "keyPair": "example-linux-key-pair-uswest1",
    "terminateInstanceOnFailure": true,
    "snsTopicArn": "arn:aws:sns:us-west-1:123456789012:example-linux-ibnotices-
uswest1",
```

```
    "dateCreated": "Feb 24, 2021 12:31:55 AM",
    "accountId": "123456789012"
  },
  "imageTestsConfigurationDocument": {
    "imageTestsEnabled": true,
    "timeoutMinutes": 720
  },
  "distributionConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-1:123456789012:distribution-configuration/
example-linux-distribution",
    "name": "example-linux-distribution",
    "dateCreated": "Feb 24, 2021 12:31:56 AM",
    "distributions": [
      {
        "region": "us-west-1",
        "amiDistributionConfiguration": {}
      }
    ],
    "tags": {
      "internalId": "345abc67-8910-12d3-4ef5-67a8b90c12de",
      "resourceArn": "arn:aws:imagebuilder:us-west-1:123456789012:distribution-
configuration/example-linux-distribution"
    },
    "accountId": "123456789012"
  },
  "dateCreated": "Jul 28, 2022 1:13:45 AM",
  "outputResources": {
    "amis": [
      {
        "region": "us-west-1",
        "image": "ami-01a23bc4def5a6789",
        "name": "example-linux-image 2022-07-28T01-14-17.416Z",
        "accountId": "123456789012"
      }
    ]
  },
  "buildExecutionId": "ab0cd12e-34fa-5678-b901-2c3456d789e0",
  "testExecutionId": "6a7b8901-cdef-234a-56b7-8cd89ef01234",
  "distributionJobId": "1f234567-8abc-9d0e-1234-fa56b7c890de",
  "integrationJobId": "432109b8-afe7-6dc5-4321-0ba98f7654e3",
  "accountId": "123456789012",
  "osVersion": "Amazon Linux 2",
  "enhancedImageMetadataEnabled": true,
  "buildType": "USER_INITIATED",
```

```
  "tags": {
    "internalId": "901e234f-a567-89bc-0123-d4e567f89a01",
    "resourceArn": "arn:aws:imagebuilder:us-west-1:123456789012:image/example-linux-
image/1.0.0/3"
  }
}
```

次の例は、Linux イメージのパイプラインビルドに失敗した場合に Image Builder が発行する典型的なメッセージの JSON ペイロードを示しています。

```
{
  "versionlessArn": "arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-
image",
  "semver": 12379400392853802748991242 31,
  "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-image/1.0.0/7",
  "name": "My Example Image",
  "version": "1.0.0",
  "type": "AMI",
  "buildVersion": 7,
  "state": {
    "status": "FAILED",
    "reason": "Image Failure reason."
  },
  "platform": "Linux",
  "imageRecipe": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-
image/1.0.0",
    "name": "My Example Image",
    "version": "1.0.0",
    "description": "Testing Image recipe",
    "components": [
      {
        "componentArn": "arn:aws:imagebuilder:us-west-2:123456789012:component/my-
example-image-component/1.0.0/1"
      }
    ],
    "platform": "Linux",
    "parentImage": "ami-0cd12345db678d90f",
    "dateCreated": "Jun 21, 2022 11:36:14 PM",
    "tags": {
      "internalId": "1a234567-8901-2345-bcd6-ef7890123456",
      "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-
example-image/1.0.0"
```

```
    },
    "accountId": "123456789012"
  },
  "sourcePipelineArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-pipeline/my-
example-image-pipeline",
  "infrastructureConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/
my-example-infra-config",
    "name": "SNS topic Infra config",
    "description": "An example that will retain instances of failed builds",
    "instanceTypes": [
      "t2.micro"
    ],
    "instanceProfileName": "EC2InstanceProfileForImageBuilder",
    "tags": {
      "internalId": "234abc56-d789-0123-a4e5-6b789d012c34",
      "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-
configuration/my-example-infra-config"
    },
    "terminateInstanceOnFailure": true,
    "snsTopicArn": "arn:aws:sns:us-west-2:123456789012:example-pipeline-notification-
topic",
    "dateCreated": "Jul 5, 2022 7:31:53 PM",
    "accountId": "123456789012"
  },
  "imageTestsConfigurationDocument": {
    "imageTestsEnabled": true,
    "timeoutMinutes": 720
  },
  "distributionConfiguration": {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-
example-distribution-config",
    "name": "New distribution config",
    "dateCreated": "Dec 3, 2021 9:24:22 PM",
    "distributions": [
      {
        "region": "us-west-2",
        "amiDistributionConfiguration": {},
        "fastLaunchConfigurations": [
          {
            "enabled": true,
            "snapshotConfiguration": {
              "targetResourceCount": 2
            },
```

```
            "maxParallelLaunches": 2,
            "launchTemplate": {
              "launchTemplateId": "lt-01234567890"
            },
            "accountId": "123456789012"
          }
        ]
      }
    ],
    "tags": {
      "internalId": "1fecd23a-4f56-7f89-01e2-345678abbe90",
      "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-
configuration/my-example-distribution-config"
    },
    "accountId": "123456789012"
  },
  "dateCreated": "Jul 5, 2022 7:40:15 PM",
  "outputResources": {
    "amis": []
  },
  "accountId": "123456789012",
  "enhancedImageMetadataEnabled": true,
  "buildType": "SCHEDULED",
  "tags": {
    "internalId": "456c78b9-0e12-3f45-afb6-7e89b0f1a23b",
    "resourceArn": "arn:aws:imagebuilder:us-west-2:123456789012:image/my-example-
image/1.0.0/7"
  }
}
```

Image Builder イメージ用のコンプライアンス製品

セキュリティ標準は絶えず進化しているため、コンプライアンスを維持し、組織をサイバー脅威から守ることは容易ではありません。カスタムイメージが確実に準拠し、パブリッシャーが新しいバージョンをリリースするときに自動更新を続けるために、Image Builder は AWS Marketplace コンプライアンス製品および AWSTOE コンポーネントと統合されます。

Image Builder は以下のコンプライアンス製品と統合されています。

- センターフォーインターネットセキュリティ（CIS）ベンチマークハードニング

CIS 強化イメージと関連する CIS 強化コンポーネントを使用して、最新の CIS Benchmarks Level 1 ガイドラインに準拠するカスタムイメージを作成できます。CIS 強化イメージは で利用できます AWS Marketplace。CIS 強化イメージと強化コンポーネントの設定方法と使用方法の詳細については、CIS Web サイトサポートポータルの「クイックスタートガイド」を参照してください。

> **Note**
>
> CIS 強化イメージを購読すると、CIS Benchmark Level 1 ガイドラインを設定に適用するスクリプトを実行する関連するビルドコンポーネントにもアクセスできます。詳細については、「CIS Fardening のコンポーネント」を参照してください。

- セキュリティ技術実装ガイド (STIG)

  STIGコンプライアンスのために、 は Image Builder レシピで Amazon マネージド AWS Task Orchestrator and Executor （ AWSTOE) STIG コンポーネントを使用できます。STIG コンポーネントはビルドインスタンスの設定ミスをスキャンし、見つかった問題を修正するための修正スクリプトを実行します。Image Builder でビルドしたイメージの STIG コンプライアンスは保証できません。組織のコンプライアンスチームと協力して、最終的なイメージが準拠していることを確認する必要があります。Image Builder レシピで使用できる AWSTOE STIG コンポーネントの完全なリストについては、「」を参照してくださいEC2 Image Builder用の Amazon managed STIG 強化コンポーネント。

EC2 Image Builder でのイベントとログのモニタリング

EC2 Image Builder パイプラインの信頼性、可用性、およびパフォーマンスを維持するには、イベントとログをモニタリングすることが重要です。イベントとログは、API 呼び出しが失敗した場合に全体像を把握し、詳細を掘り下げるのに役立ちます。Image Builder は、設定した条件にイベントが一致したときにアラートを送信したり、自動応答を開始したりできるサービスと統合されています。

以下のトピックでは、Image Builder と統合するサービスを通じて使用できる監視手法について説明します。

イベントとログを監視する。

- を使用した EC2 Image Builder API コールのログ記録 AWS CloudTrail

を使用した EC2 Image Builder API コールのログ記録 AWS CloudTrail

EC2 Image Builder は、Image Builder API を通じてユーザー AWS CloudTrail、ロール、または サービスによって実行されたすべての API コールのアクションを記録する AWS サービスである と統合されています。CloudTrail は Image Builder をイベントとしてキャプチャします。キャプチャされる呼び出しには、Image Builder コンソールからの呼び出しと、Image Builder API 操作へのコード呼び出しが含まれます。

証跡を作成する場合は、Image Builder の CloudTrail イベントなど、S3 バケットへのイベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、 CloudTrail コンソールの イベント履歴 で最新のイベントを表示できます。で収集された情報を使用して CloudTrail、Image Builder に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

の詳細については CloudTrail、「 AWS CloudTrail ユーザーガイド」を参照してください。

の Image Builder 情報 CloudTrail

CloudTrail アカウントを作成する AWS アカウント と、 で が有効になります。Image Builder でアクティビティが発生すると、そのアクティビティは CloudTrail イベント履歴 の他の AWS サービスイベントとともにイベントに記録されます。で最近のイベントを表示、検索、ダウンロードできます

AWS アカウント。詳細については、「イベント履歴で CloudTrail イベントを表示する」を参照してください。

Image Builder のイベントなど AWS アカウント、 のイベントの継続的な記録については、証跡を作成します。証跡により、 はログファイル CloudTrail を S3 バケットに配信できます。デフォルトでは、コンソールで証跡を作成するときに、証跡がすべての AWS リージョンに適用されます。証跡は、 AWS パーティション内のすべてのリージョンからのイベントをログに記録し、指定した S3 バケットにログファイルを配信します。さらに、他の を設定 AWS のサービス して、 CloudTrail ログで収集されたイベントデータをさらに分析し、それに基づく対応を行うことができます。詳細については、次を参照してください。

- 証跡作成の概要
- CloudTrail がサポートする サービスと統合。
- の Amazon SNS 通知の設定 CloudTrail。
- 複数のリージョン からの CloudTrail ログファイルの受信。
- 複数のアカウント からのCloudTrail ログファイルの受信。

CloudTrail は、EC2 Image Builder API リファレンス に記載されているすべての Image Builder アクションを記録します。例えば、`CreateImagePipeline`、、および `StartImagePipelineExecution`アクションを呼び出すと`UpdateInfrastructureConfiguration`、 CloudTrail ログファイルにエントリが生成されます。

各イベントまたはログエントリには、誰がリクエストを生成したかという情報が含まれます。同一性情報は次の判断に役立ちます。

- リクエストが、ルートと IAM ユーザー認証情報のどちらを使用して送信されたか。
- リクエストがロールまたはフェデレーションユーザーの一時的なセキュリティ認証情報を使用して行われたかどうか。
- リクエストが別の AWS サービスによって行われたかどうか。

イベントをリクエストしたユーザーを決定する方法の詳細については、CloudTrail userIdentity 要素」を参照してください。

EC2 Image Builder でのセキュリティ

のクラウドセキュリティが最優先事項 AWS です。 AWS のお客様は、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャからメリットを得られます。

セキュリティは、 AWS とユーザーの間で共有される責任です。責任共有モデルではこれを、クラウドのセキュリティ、およびクラウド内でのセキュリティと説明しています:

- クラウドのセキュリティ — クラウド AWS のサービス で実行されるインフラストラクチャを保護する責任 AWS は にあります AWS 。 AWS また、 では、安全に使用できるサービスも提供しています。コンプライアンスAWS プログラムコンプライアンスプログラム の一環として、サードパーティーの監査者は定期的にセキュリティの有効性をテストおよび検証。EC2 Image Builderに適用されるコンプライアンスプLogラムについては、AWS のサービス コンプライアンスプLogラム別適用範囲を参照してください。
- クラウドのセキュリティ — お客様の責任は、使用する AWS サービスによって決まります。また、お客様は、データの機密性、会社の要件、適用される法律や規制など、その他の要因についても責任を負います。

このドキュメントは、Image Builder を使用する際の責任共有モデルの適用方法を理解するのに役立ちます。以下のトピックでは、セキュリティおよびコンプライアンス目標を満たすために Image Builder を設定する方法について説明します。また、Image Builder リソースのモニタリングや保護 AWS のサービス に役立つ他の の使用方法についても説明します。

トピック

- EC2 Image Builder でのデータ保護
- EC2Image Builderのアイデンティティとアクセス管理
- EC2 Image Builder のコンプライアンス検証
- EC2 Image Builder でのレジリエンス
- Image Builder でのインフラストラクチャセキュリティ
- EC2 Image Builder でのパッチ管理
- EC2 Image Builder でのセキュリティのベストプラクティス

EC2 Image Builder でのデータ保護

責任 AWS 共有モデル、EC2 Image Builder でのデータ保護に適用されます。このモデルで説明されているように、 AWS はすべての を実行するグローバルインフラストラクチャを保護する責任があります AWS クラウド。お客様は、このインフラストラクチャでホストされているコンテンツに対する管理を維持する責任があります。また、使用する AWS のサービス のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、「データプライバシーのよくある質問」を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された記事「AWS 責任共有モデルおよび GDPR」を参照してください。

データ保護の目的で、 認証情報を保護し AWS アカウント 、 AWS IAM Identity Center または AWS Identity and Access Management (IAM) を使用して個々のユーザーを設定することをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して AWS リソースと通信します。TLS 1.2 は必須であり TLS 1.3 がお勧めです。
- で API とユーザーアクティビティのログ記録を設定します AWS CloudTrail。
- AWS 暗号化ソリューションと、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度なマネージドセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API AWS を介して にアクセスするときに FIPS 140-2 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-2」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これは、コンソール、API、または AWS CLI SDK を使用して Image Builder または他の AWS のサービス を操作する場合も同様です。 AWS SDKs 名前に使用する自由記述のテキストフィールドやタグに入力したデータは、課金や診断ログに使用される場合があります。外部サーバーへの URL を提供する場合は、そのサーバーへのリクエストを検証するための認証情報を URL に含めないように強くお勧めします。

EC2 Image Builder での暗号化とキー管理

Image Builder は、以下を除き、デフォルトでサービス所有の KMS キーを使用して転送中および保存中のデータを暗号化します。

- カスタムコンポーネント — Image Builder は、デフォルトの KMS キーまたはサービス所有の KMS キーを使用してカスタムコンポーネントを暗号化します。
- イメージワークフロー – Image Builder は、ワークフローの作成時にカスタマーマネージドキーを指定すると、イメージワークフローをそのキーで暗号化できます。Image Builder は、キーを使用して暗号化と復号化を処理し、画像に設定したワークフローを実行します。

を使用して独自のキーを管理できます AWS KMS。ただし、Image Builder が所有するImage Builder KMS キーを管理する権限はありません。で KMS キーを管理する方法の詳細については AWS Key Management Service、「 AWS Key Management Service デベロッパーガイド」の「開始方法」を参照してください。

暗号化コンテキスト

暗号化されたデータの整合性と信頼性をさらに確認するために、データを暗号化するときに暗号化コンテキストを含めることもできます。リソースが暗号化コンテキストで暗号化されると、 はコンテキストを暗号文に AWS KMS 暗号的にバインドします。リソースを復号化できるのは、リクエスタがコンテキストと完全に一致させ、大文字と小文字を区別して一致させた場合のみです。

このセクションでのポリシー例では、Image Builder ワークフローリソースの Amazon リソースネーム (ARN) に似た暗号化コンテキストを使用しています。

カスタマーマネージドキーを使用してイメージワークフローを暗号化する

保護を強化するために、Image Builder ワークフローリソースを独自のカスタマーマネージドキーで暗号化できます。カスタマーマネージドキーを使用して作成した Image Builder ワークフローを暗号化する場合、Image Builder がワークフローリソースを暗号化および復号化するときにそのキーを使用するように、キーポリシーでアクセス権を付与する必要があります。アクセスはいつでも取り消すことができます。ただし、キーへのアクセス権を取り消すと、Image Builder はすでに暗号化されているワークフローにアクセスできなくなります。

Image Builder にカスタマーマネージドキーを使用するアクセス権を付与するプロセスには、次の 2 つのステップがあります。

ステップ 1: Image Builder ワークフローにキーポリシー権限を追加する

Image Builder がワークフローを作成または使用するときにワークフローリソースを暗号化および復号化できるようにするには、KMS キーポリシーで権限を指定する必要があります。

このサンプルキーポリシーは、Image Builder パイプラインにアクセス権を付与して、作成プロセス中にワークフローリソースを暗号化し、ワークフローリソースを復号化して使用できるようにしま

す。このポリシーでは、キー管理者にもアクセス権限が付与されます。暗号化コンテキストとリソース仕様では、ワークフローリソースがあるすべてのリージョンを対象にワイルドカードを使用しています。

イメージワークフローを使用するための前提条件として、Image Builder にワークフローアクションを実行する権限を付与する IAM ワークフロー実行ロールを作成しました。このキーポリシーの例で示した最初のステートメントのプリンシパルは、IAM ワークフロー実行ロールを指定する必要があります。

カスタマーマネージドキーの詳細については、「AWS Key Management Service デベロッパーガイド」の「カスタマーマネージドキーへのアクセスを管理する」を参照してください。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "Allow access to build images with encrypted workflow",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/YourImageBuilderExecutionRole"
   },
   "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
   ],
   "Resource": "*",
   "Condition": {
    "StringLike": {
     "kms:EncryptionContext:aws:imagebuilder:arn":
 "arn:aws:imagebuilder:*:111122223333:workflow/*"
    }
   }
  },
  {
   "Sid": "Allow access for key administrators",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:root"
   },
   "Action": [
    "kms:*"
   ],
   "Resource": "arn:aws:kms:*:111122223333:key/"
```

```
  }
 ]
}
```

ステップ 2: ワークフロー実行ロールへのキーアクセス権を付与する

Image Builder がワークフローを実行するために引き受ける IAM ロールには、カスタマーマネージドキーを使用する権限が必要です。キーにアクセスできないと、Image Builder はキーを使用してワークフローリソースを暗号化または復号化できません。

ワークフロー実行ロールのポリシーを編集して、次のポリシーステートメントを追加します。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "Allow access to the workflow key",
   "Effect": "Allow",
   "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
   ],
   "Resource": "arn:aws:kms:us-west-2:111122223333:key/key_ID",
   "Condition": {
    "StringLike": {
     "kms:EncryptionContext:aws:imagebuilder:arn":
 "arn:aws:imagebuilder:*:111122223333:workflow/*"
    }
   }
  }
 ]
}
```

AWS CloudTrail イメージワークフローの イベント

次の例は、カスタマーマネージドキーで保存されているイメージワークフローを暗号化および復号するための一般的な AWS CloudTrail エントリを示しています。

例： GenerateDataKey

この例では、Image Builder が Image Builder API アクションから API CreateWorkflowアクションを AWS KMS GenerateDataKey呼び出すと、 CloudTrail イベントがどのようになるかを示しま

す。Image Builder は、ワークフローリソースを作成する前に新しいワークフローを暗号化する必要があります。

```
{
 "eventVersion": "1.08",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "PRINCIPALID1234567890:workflow-role-name",
  "arn": "arn:aws:sts::111122223333:assumed-role/Admin/workflow-role-name",
  "accountId": "111122223333",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "PRINCIPALID1234567890",
    "arn": "arn:aws:iam::111122223333:role/Admin",
    "accountId": "111122223333",
    "userName": "Admin"
   },
   "webIdFederationData": {},
   "attributes": {
    "creationDate": "2023-11-21T20:29:31Z",
    "mfaAuthenticated": "false"
   }
  },
  "invokedBy": "imagebuilder.amazonaws.com"
 },
 "eventTime": "2023-11-21T20:31:03Z",
 "eventSource": "kms.amazonaws.com",
 "eventName": "GenerateDataKey",
 "awsRegion": "us-west-2",
 "sourceIPAddress": "imagebuilder.amazonaws.com",
 "userAgent": "imagebuilder.amazonaws.com",
 "requestParameters": {
  "encryptionContext": {
   "aws:imagebuilder:arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/
sample-encrypted-workflow/1.0.0/*",
   "aws-crypto-public-key": "key value"
  },
  "keyId": "arn:aws:kms:us-west-2:111122223333:alias/ExampleKMSKey",
  "numberOfBytes": 32
 },
 "responseElements": null,
```

```
 "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa",
 "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
 "readOnly": true,
 "resources": [
  {
   "accountId": "111122223333",
   "type": "AWS::KMS::Key",
   "ARN": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLEzzzzz"
  }
 ],
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "111122223333",
 "eventCategory": "Management"
}
```


例: Decrypt

この例では、Image Builder が Image Builder API アクションから API GetWorkflowアクションを AWS KMS Decrypt呼び出すと、 CloudTrail イベントがどのようになるかを示します。Image Builder パイプラインは、ワークフローリソースを使用する前に復号化する必要があります。


```
{
 "eventVersion": "1.08",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "PRINCIPALID1234567890:workflow-role-name",
  "arn": "arn:aws:sts::111122223333:assumed-role/Admin/workflow-role-name",
  "accountId": "111122223333",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "PRINCIPALID1234567890",
    "arn": "arn:aws:iam::111122223333:role/Admin",
    "accountId": "111122223333",
    "userName": "Admin"
   },
   "webIdFederationData": {},
   "attributes": {
    "creationDate": "2023-11-21T20:29:31Z",
    "mfaAuthenticated": "false"
   }
```

```
  },
  "invokedBy": "imagebuilder.amazonaws.com"
 },
 "eventTime": "2023-11-21T20:34:25Z",
 "eventSource": "kms.amazonaws.com",
 "eventName": "Decrypt",
 "awsRegion": "us-west-2",
 "sourceIPAddress": "imagebuilder.amazonaws.com",
 "userAgent": "imagebuilder.amazonaws.com",
 "requestParameters": {
  "keyId": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-
EXAMPLEzzzzz",
  "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
  "encryptionContext": {
   "aws:imagebuilder:arn": "arn:aws:imagebuilder:us-west-2:111122223333:workflow/build/
sample-encrypted-workflow/1.0.0/*",
   "aws-crypto-public-key": "ABC123def4567890abc12345678/90dE/F123abcDEF+4567890abc123D
+ef1=="
  }
 },
 "responseElements": null,
 "requestID": "a1b2c3d4-5678-90ab-cdef-EXAMPLEbbbbb",
 "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
 "readOnly": true,
 "resources": [
  {
   "accountId": "111122223333",
   "type": "AWS::KMS::Key",
   "ARN": "arn:aws:kms:us-west-2:111122223333:key/a1b2c3d4-5678-90ab-cdef-EXAMPLEzzzzz"
  }
 ],
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "recipientAccountId": "111122223333",
 "eventCategory": "Management"
}
```


EC2 Image Builder のデータストレージ

Image Builder はログをサービスに保存しません。すべてのログは、イメージの構築に使用される Amazon EC2 インスタンス、または Systems Manager 自動化ログに保存されます。

EC2 Image Builder でのネットワーク内のトラフィックプライバシー

接続は、Image Builder とオンプレミスの場所間、 AWS リージョン内の AZs 間、および HTTPS 経由で AWS リージョン間で保護されます。アカウント間には直接接続はありません。

EC2Image Builderのアイデンティティとアクセス管理

トピック

対象者

AWS Identity and Access Management (IAM) の使用方法は、Image Builder で行う作業によって異なります。

サービスユーザ - 業務に Image Builder サービスを使用する場合、管理者が必要な認証情報と権限を提供します。より多くの Image Builder 機能を使用するようになると、追加のパーミッションが必要になる場合があります。アクセスの管理方法を理解しておくと、管理者に適切な許可をリクエストするうえで役立ちます。Image Builderの機能にアクセスできない場合は、EC2 Image Builder アイデンティティとアクセスのトラブルシューティングを参照してください。

サービス管理者 - 会社で Image Builder リソースを管理している場合、おそらく Image Builder にフルアクセスできます。サービス利用者がアクセスすべき Image Builder の機能やリソースを決定するのはあなたの仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。Image BuilderでIAMを使用する方法については、EC2 Image BuilderとIAMの仕組みを参照してください。

IAM管理者 - IAM管理者であれば、Image Builderへのアクセスを管理するためのポリシーの書き方について詳しく知りたいかもしれません。IAM で使用できる Image Builder ID ベースのポリシーの例は、Image Builder のアイデンティティベースのポリシーを参照してください。

アイデンティティを使用した認証

でユーザーとプロセスに認証を提供する方法の詳細については AWS アカウント、IAM ユーザーガイドのhttps://docs.aws.amazon.com/IAM/latest/UserGuide/id.html「アイデンティティ」を参照してください。

EC2 Image BuilderとIAMの仕組み

IAMを使用してImage Builderへのアクセスを管理する前に、Image Builderで使用できるIAM機能について確認してください。

Image Builder およびその他の AWS のサービスがほとんどの IAM 機能と連携する方法の概要を把握するには、「IAM ユーザーガイド」のAWS 「IAM と連携する のサービス」を参照してください。

Image Builder のアイデンティティベースのポリシー

アイデンティティベースポリシーをサポートする	Yes

アイデンティティベースポリシーは、IAM ユーザー、ユーザーグループ、ロールなど、アイデンティティにアタッチできる JSON 権限ポリシードキュメントです。これらのポリシーは、ユーザーとロールが実行できるアクション、リソース、および条件をコントロールします。アイデンティティベースのポリシーを作成する方法については、『IAM ユーザーガイド』の「IAM ポリシーの作成」を参照してください。

IAM アイデンティティベースのポリシーでは、許可または拒否するアクションとリソース、およびアクションを許可または拒否する条件を指定できます。プリンシパルは、それが添付されているユーザーまたはロールに適用されるため、アイデンティティベースのポリシーでは指定できません。JSON ポリシーで使用できるすべての要素については、「IAM ユーザーガイド」の「IAM JSON ポリシーの要素のリファレンス」を参照してください。

Image Builder のアイデンティティベースのポリシー例

Image Builder ID ベースのポリシーの例を見るには、Image Builder のアイデンティティベースのポリシーを参照してください。

Image Builder 内のリソースベースのポリシー

リソースベースのポリシーのサポート	はい

リソースベースのポリシーは、リソースに添付する JSON ポリシードキュメントです。リソースベースのポリシーには例として、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーがあげられます。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスを制御できます。ポリシーがアタッチされているリソースの場合、指定されたプリンシパルがそのリソースに対して実行できるアクションと条件は、ポリシーによって定義されます。リソースベースのポリシーでは、プリンシパルを指定する必要があります。プリンシパルには、アカウント、ユーザー、ロール、フェデレーティッドユーザー、または を含めることができます AWS のサービス。

クロスアカウントアクセスを有効にするには、アカウント全体、または別のアカウントの IAM エンティティをリソースベースのポリシーのプリンシパルとして指定します。リソースベースのポリシーにクロスアカウントのプリンシパルを追加しても、信頼関係は半分しか確立されない点に注意してください。プリンシパルとリソースが異なる にある場合 AWS アカウント、信頼されたアカウントの IAM 管理者は、プリンシパルエンティティ (ユーザーまたはロール) にリソースへのアクセス許可も付与する必要があります。IAM 管理者は、アイデンティティベースのポリシーをエンティティにアタッチすることで権限を付与します。ただし、リソースベースのポリシーで、同じアカウントのプリンシパルへのアクセス権が付与されている場合は、アイデンティティベースのポリシーを追加する必要はありません。詳細については、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

Image Builder ポリシーアクション

ポリシーアクションに対するサポート	はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

JSON ポリシーの`Action`要素には、ポリシー内のアクセスを許可または拒否するために使用できるアクションが記述されます。ポリシーアクションの名前は通常、関連付けられた AWS API オペレー

ションと同じです。一致する API オペレーションのない権限のみのアクションなど、いくつかの例外があります。また、ポリシーに複数アクションが必要なオペレーションもあります。これらの追加アクションは、 依存アクション と呼ばれます。

このアクションは、関連付けられたオペレーションを実行するための権限を付与するポリシーで使用されます。

Image Builderのアクションの一覧は、Service Authorization ReferenceのEC2 Image Builderで定義されたアクションを参照してください。

Image Builder のポリシーアクションは、アクションの前に次の接頭辞を使用します：

```
imagebuilder
```

単一のステートメントで複数のアクションを指定するには、アクションをカンマで区切ります。

```
"Action": [
      "imagebuilder:action1",
      "imagebuilder:action2"
         ]
```

Image Builder ID ベースのポリシーの例を見るには、Image Builder のアイデンティティベースのポリシーを参照してください。

Image Builder ポリシーリソース

ポリシーリソースに対するサポート	はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースにどのような条件でアクションを実行できるかということです。

Resource JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの権限と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Image Builderのリソースタイプとそのarnの一覧は、Service Authorization ReferenceのResources defined by EC2 Image Builderを参照してください。各リソースのarnをどのアクションで指定できるかについては、EC2 Image Builderで定義されているアクションを参照してください。

Image Builder ID ベースのポリシーの例を見るには、Image Builder のアイデンティティベースのポリシーを参照してください。

Image Builderのポリシー条件キー

サービス固有のポリシー条件キーのサポート	はい

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどんなリソースにどんな条件でアクションを実行できるかということです。

`Condition` 要素 (または `Condition` ブロック) を使用すると、ステートメントが有効な条件を指定できます。`Condition` 要素はオプションです。イコールや未満などの 条件演算子 を使用して条件式を作成することで、ポリシーの条件とリクエスト内の値を一致させることができます。

1 つのステートメントに複数の `Condition` 要素を指定するか、1 つの `Condition` 要素に複数のキーを指定すると、 AWS は `AND` 論理演算子を使用してそれらを評価します。1 つの条件キーに複数の値を指定すると、 は論理ORオペレーションを使用して条件 AWS を評価します。ステートメントの権限が付与される前にすべての条件が満たされる必要があります。

条件を指定する際にプレースホルダー変数も使用できます。例えば IAM ユーザーに、IAM ユーザー名がタグ付けされている場合のみリソースにアクセスできる権限を付与することができます。詳細については、『IAM ユーザーガイド』の「IAM ポリシーの要素: 変数およびタグ」を参照してください。

AWS は、グローバル条件キーとサービス固有の条件キーをサポートします。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイド」のAWS 「 グローバル条件コンテキストキー」を参照してください。

Image Builderの条件キーの一覧は、Service Authorization ReferenceのCondition keys for EC2 Image Builderを参照してください。どのアクションとリソースで条件キーを使用できるかについては、EC2 Image Builderで定義されたアクションを参照してください。

Image Builder ID ベースのポリシーの例を見るには、Image Builder のアイデンティティベースのポリシーを参照してください。

Image Builder の ACL

ACL のサポート	No

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための権限を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Image Builder 付き

ABAC (ポリシー内のタグ) のサポート	部分的

属性ベースのアクセスコントロール (ABAC) は、属性に基づいて権限を定義する認可戦略です。では AWS、これらの属性はタグ と呼ばれます。タグは、IAM エンティティ (ユーザーまたはロール) および多くの AWS リソースにアタッチできます。エンティティとリソースのタグ付けは、ABAC の最初の手順です。その後、プリンシパルのタグがアクセスしようとしているリソースのタグと一致した場合に操作を許可するように ABAC ポリシーを設計します。

ABAC は、急成長する環境やポリシー管理が煩雑になる状況で役立ちます。

タグに基づいてアクセスを管理するには、`aws:ResourceTag/key-name`、`aws:RequestTag/key-name`、または `aws:TagKeys` の条件キーを使用して、ポリシーの条件要素でタグ情報を提供します。

サービスがすべてのリソースタイプに対して 3 つの条件キーすべてをサポートする場合、そのサービスの値ははいです。サービスが一部のリソースタイプに対してのみ 3 つの条件キーのすべてをサポートする場合、値は「部分的」になります。

ABAC の詳細については、『IAM ユーザーガイド』の「ABAC とは?」を参照してください。ABAC をセットアップするステップを説明するチュートリアルについては、「IAM ユーザーガイド」の「属性に基づくアクセスコントロール (ABAC) を使用する」を参照してください。

Image Builder での一時的な認証情報の使用

一時的な認証情報のサポート	はい

一部の は、一時的な認証情報を使用してサインインすると機能 AWS のサービス しません。一時的な認証情報 AWS のサービス を使用する などの詳細については、IAM ユーザーガイドのAWS のサービス「IAM と連携する 」を参照してください。

ユーザー名とパスワード以外の AWS Management Console 方法で にサインインする場合、一時的な認証情報を使用します。例えば、会社の Single Sign-On (SSO) リンク AWS を使用して にアクセスすると、そのプロセスによって一時的な認証情報が自動的に作成されます。また、ユーザーとしてコンソールにサインインしてからロールを切り替える場合も、一時的な認証情報が自動的に作成されます。ロールの切り替えに関する詳細については、「IAM ユーザーガイド」の「ロールへの切り替え (コンソール)」を参照してください。

一時的な認証情報は、 AWS CLI または AWS API を使用して手動で作成できます。その後、これらの一時的な認証情報を使用して にアクセスします AWS。 AWS 長期的なアクセスキーを使用する代わりに、一時的な認証情報を動的に生成することをお勧めします。詳細については、「IAM の一時的セキュリティ認証情報」を参照してください。

Image Builderのクロスサービスプリンシパル権限

フォワードアクセスセッション (FAS) をサポート	はい

IAM ユーザーまたはロールを使用して でアクションを実行すると AWS、プリンシパルと見なされます。一部のサービスを使用する際に、アクションを実行することで、別のサービスの別のアクションがトリガーされることがあります。FAS は、 を呼び出すプリンシパルのアクセス許可を AWS のサービス、ダウンストリームサービス AWS のサービス へのリクエストリクエストリクエストと組み合わせて使用します。FAS リクエストは、サービスが他の AWS のサービス またはリソースとのやり取りを完了する必要があるリクエストを受け取った場合にのみ行われます。この場合、両方のアクションを実行するためのアクセス許可が必要です。FAS リクエストを行う際のポリシーの詳細については、「転送アクセスセッション」を参照してください。

Image Builder サービスロール

サービスロールに対するサポート	あり

サービスロールとは、サービスがユーザーに代わってアクションを実行するために引き受ける IAM ロールです。IAM 管理者は、IAM 内からサービスロールを作成、変更、削除できます。詳細については、「IAM ユーザーガイド」の「AWS のサービスに権限を委任するロールの作成」を参照してください。

> **Warning**
>
> サービスロールの権限を変更すると、Image Builderの機能が壊れる可能性があります。サービスロールの編集は、Image Builderのガイダンスに従って行ってください。

Image Builderのサービス連動ロール

サービスにリンクされたロールのサポート	いいえ

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの権限を表示できますが、編集することはできません。

Image Builderサービス連携ロールの詳細については、EC2 Image Builder サービスにリンクされたロールを使用するを参照。してください。

Image Builder のアイデンティティベースのポリシー

IAMアイデンティティベースポリシーでは、許可または拒否されるアクションとリソースを指定し、アクションが許可または拒否される条件も指定できます。Image Builderは、特定のアクション、リソース、条件キーをサポートしています。JSON ポリシーで使用するすべての要素については、IAM ユーザーガイドの「Amazon EC2 Image Builder のアクション、リソース、および条件キー」を参照してください。

アクション

Image Builder のポリシーアクションは、アクションの前に以下の接頭辞を使用します : `imagebuilder:`. ポリシーステートメントには、`Action` または `NotAction` 要素を含める必要があります。Image Builderは、このサービスで実行できるタスクを記述した独自のアクションセットを定義しています。

単一ステートメントに複数アクションを指定するには、次のようにカンマで区切ります:

```
"Action": [
      "imagebuilder:action1",
      "imagebuilder:action2"
```

ワイルドカード (*) を使用して複数アクションを指定できます。例えば、`List` という単語で始まるすべてのアクションを指定するには、次のアクションを含めます。

```
"Action": "imagebuilder:List*"
```

Image Builderのアクションのリストは、IAMユーザーガイドのアクション、リソース、および AWS のサービスの条件キーを参照してください。

ポリシーを使用したアクセスの管理

ポリシーを作成し、IAM ID または AWS リソースにアタッチ AWS して でアクセスを管理する方法の詳細については、IAM ユーザーガイドのポリシーとアクセス許可を参照してください。

インスタンスプロファイルに関連付ける IAM ロールには、イメージに含まれるビルドコンポーネントとテストコンポーネントを実行する権限が必要です。インスタンスプロファイルに関連付けられている IAM ロールに対し、次の IAM ロールポリシーをアタッチする必要があります。

- EC2InstanceProfileForImageBuilder
- EC2InstanceProfileForImageBuilderECRContainerBuilds
- AmazonSSMManagedInstanceCore

リソース

管理者は AWS JSON ポリシーを使用して、誰が何にアクセスできるかを指定できます。つまり、どのプリンシパルがどのリソースにどのような条件でアクションを実行できるかということです。

Resource JSON ポリシー要素は、アクションが適用されるオブジェクトを指定します。ステートメントには、Resource または NotResource要素を含める必要があります。ベストプラクティスとして、Amazon リソースネーム (ARN) を使用してリソースを指定します。これは、リソースレベルの権限と呼ばれる特定のリソースタイプをサポートするアクションに対して実行できます。

オペレーションのリスト化など、リソースレベルの権限をサポートしないアクションの場合は、ステートメントがすべてのリソースに適用されることを示すために、ワイルドカード (*) を使用します。

```
"Resource": "*"
```

Image Builderインスタンスリソースは、以下のAmazon リソースネーム (ARN)) を持ちます。

```
arn:aws:imagebuilder:region:account-id:resource:resource-id
```

ARN の形式の詳細については、「Amazon リソースネーム (ARNs AWS 「サービス名前空間」を参照してください。

例えば、ステートメントでi-1234567890abcdef0インスタンスを指定するには、以下のARNを使用します。

```
"Resource": "arn:aws:imagebuilder:us-east-1:123456789012:instance/i-1234567890abcdef0"
```

特定のアカウントに属するすべてのインスタンスを指定するには、ワイルドカード *を使用します。

```
"Resource": "arn:aws:imagebuilder:us-east-1:123456789012:instance/*"
```

リソースの作成など、一部の Image Builder アクションは、特定のリソースに対して実行できません。このような場合は、ワイルドカード *を使用する必要があります。

```
"Resource": "*"
```

EC2 Image Builder APIアクションの多くは、複数のリソースを使用します。複数リソースを単一ステートメントで指定するには、ARN をカンマで区切ります。

```
"Resource": [
      "resource1",
      "resource2"
```

条件キー

Image Builderはサービス固有の条件キーを提供し、いくつかのグローバル条件キーの使用をサポートします。すべての AWS グローバル条件キーを確認するには、「IAM ユーザーガイドAWS 」の「グローバル条件コンテキストキー」を参照してください。以下のサービス別条件キーがある。

ImageBuilder:CreatedResourceTagKeys

文字列演算子で動作します。

リクエストにタグキーがあるかどうかでアクセスをフィルターするには、 このキーを使用します。これにより、Image Builder が作成するリソースを管理できます。

利用可能性 - このキーはCreateInfrastrucutreConfigurationとUpdateInfrastructureConfigurationのAPIでのみ利用可能です。

imagebuilder:CreatedResourceTag/<key>

文字列演算子で動作します。

このキーを使用して、Image Builder が作成したリソースに添付されているタグのキーと値のペアでアクセスをフィルタリングします。これにより、定義済みのタグを使用して Image Builder リソースを管理できます。

利用可能性 - このキーはCreateInfrastrucutreConfigurationとUpdateInfrastructureConfigurationのAPIでのみ利用可能です。

imagebuilder:Ec2MetadataHttpTokens

文字列演算子で動作します。

このキーを使用して、リクエストで指定された EC2 インスタンスメタデータの HTTP トークン要件によってアクセスをフィルタリングします。

このキーの値はoptionalかrequiredのどちらかなります。

利用可能性 - このキーはCreateInfrastrucutreConfigurationとUpdateInfrastructureConfigurationのAPIでのみ利用可能です。

ImageBuilder:StatusTopicArn

文字列演算子で動作します。

このキーを使用して、端末の状態通知を公開するリクエストで、SNSトピックARNによるアクセスをフィルタリングします。

利用可能性 - このキーは`CreateInfrastrucutreConfiguration`と`UpdateInfrastructureConfiguration`のAPIでのみ利用可能です。

例

Image Builder ID ベースのポリシーの例を見るには、EC2 Image BuilderのIDベースのポリシーを参照してください。

Image Builder 内のリソースベースのポリシー

リソースベースのポリシーは、指定されたプリンシパルが Image Builder リソースに対して実行できるアクションと条件を指定します。Image Builder は、コンポーネント、イメージ、およびイメージレシピのリソースベースのアクセス権限ポリシーをサポートします。リソースベースのポリシーでは、リソースごとに他の アカウントに使用許可を付与することができます。リソースベースのポリシーを使用して、 AWS サービスがコンポーネント、イメージ、イメージレシピにアクセスすることを許可することもできます。

リソースベースのポリシーをコンポーネント、イメージ、またはイメージレシピにアタッチする方法については、「EC2 Image Builder リソースを共有」を参照してください。

> **Note**
>
> Image Builder を使用してリソースポリシーを更新すると、更新は RAM コンソールに表示されます。

Image Builder タグに基づく認可

タグを Image Builder リソースに添付したり、リクエストでタグを Image Builder に渡すことができます。タグに基づいてアクセスを管理するには、`imagebuilder:ResourceTag/key-`

name、aws:RequestTag/*key-name*、または aws:TagKeys の条件キーを使用して、ポリシーの 条件要素でタグ情報を提供します。Image Builder リソースのタグ付けの詳細については、(AWS CLI)リソースをタグ付けします。を参照してください。

Image Builder IAM ロール

IAM ロールは、特定のアクセス許可 AWS アカウント を持つ 内のエンティティです。

Image Builder での一時的な認証情報の使用

一時的な認証情報を使用して、フェデレーションでサインインする、IAM ロールを引き受ける、またはクロスアカウントロールを引き受けることができます。一時的なセキュリティ認証情報を取得するには、 AssumeRoleや などの AWS STS API オペレーションを呼び出しますGetFederationToken。

サービスリンクロール

サービスにリンクされたロールを使用すると AWS のサービス 、 は他の サービスのリソースにアクセスして、ユーザーに代わってアクションを実行できます。サービスリンクロールは IAM アカウント内に表示され、サービスによって所有されます。管理者権限を持つユーザは、サービスリンクされたロールの権限を表示することはできますが、編集することはできません。

Image Builderはサービスリンクされたロールをサポートします。Image Builderサービス連動ロールの作成または管理については、EC2 Image Builder サービスにリンクされたロールを使用するを参照してください。

サービスロール

この機能により、ユーザーに代わってサービスがサービスロールを引き受けることが許可されます。このロールにより、サービスがお客様に代わって他のサービスのリソースにアクセスし、アクションを完了することが許可されます。サービスロールは、IAM アカウントに表示され、アカウントによって所有されます。これは、管理者権限を持つユーザがこのロールの権限を変更できることを意味します。ただし、それにより、サービスの機能が損なわれる場合があります。

EC2 Image BuilderのIDベースのポリシー

トピック

- アイデンティティベースポリシーのベストプラクティス
- Image Builder コンソールの使用

アイデンティティベースポリシーのベストプラクティス

アイデンティティベースのポリシーは、アカウント内の Image Builder リソースを作成、アクセス、または削除できるかどうかを決定します。これらのアクションを実行すると、 AWS アカウントに料金が発生する可能性があります。アイデンティティベースポリシーを作成したり編集したりする際には、以下のガイドラインと推奨事項に従ってください:

- AWS 管理ポリシーを開始し、最小特権のアクセス許可に移行する – ユーザーとワークロードにアクセス許可を付与するには、多くの一般的なユースケースにアクセス許可を付与する AWS 管理ポリシーを使用します。これらは で使用できます AWS アカウント。ユースケースに固有の AWS カスタマー管理ポリシーを定義して、アクセス許可をさらに減らすことをお勧めします。詳細については、『IAM ユーザーガイド』の「AWS マネージドポリシー」または「AWS ジョブ機能の管理ポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで権限を設定するときは、タスクの実行に必要な権限のみを付与します。これを行うには、特定の条件下で特定のリソースに対して実行できるアクションを定義します。これは、最小特権権限とも呼ばれています。IAM を使用して権限を適用する方法の詳細については、『IAM ユーザーガイド』の「IAM でのポリシーと権限」を参照してください。
- IAM ポリシーで条件を使用してアクセスをさらに制限する - ポリシーに条件を追加して、アクションやリソースへのアクセスを制限できます。例えば、ポリシー条件を記述して、すべてのリクエストを SSL を使用して送信するように指定できます。条件を使用して、 などの特定の を介してサービスアクションが使用される場合に AWS のサービス、サービスアクションへのアクセスを許可することもできます AWS CloudFormation。詳細については、『IAM ユーザーガイド』の [IAM JSON policy elements: Condition] (IAM JSON ポリシー要素：条件) を参照してください。
- IAM Access Analyzer を使用して IAM ポリシーを検証し、安全で機能的な権限を確保する - IAM Access Analyzer は、新規および既存のポリシーを検証して、ポリシーが IAM ポリシー言語 (JSON) および IAM のベストプラクティスに準拠するようにします。IAM アクセスアナライザーは 100 を超えるポリシーチェックと実用的な推奨事項を提供し、安全で機能的なポリシーの作成をサポートします。詳細については、『IAM ユーザーガイド』の「IAM Access Analyzer ポリシーの検証」を参照してください。
- 多要素認証 (MFA) を要求する – で IAM ユーザーまたはルートユーザーを必要とするシナリオがある場合は AWS アカウント、セキュリティを強化するために MFA を有効にします。API オペレーションが呼び出されるときに MFA を必須にするには、ポリシーに MFA 条件を追加します。詳細については、『IAM ユーザーガイド』の「MFA 保護 API アクセスの設定」を参照してください。

IAM でのベストプラクティスの詳細については、『IAM ユーザーガイド』の「IAM で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。

Image Builder コンソールの使用

EC2 Image Builderのコンソールにアクセスするには、最低限の権限が必要です。これらの権限により、 AWS アカウントにある Image Builder リソースの一覧を表示し、詳細を確認することができます。最小限必要な許可よりも厳しく制限されたアイデンティティベースポリシーを作成すると、そのポリシーを添付したエンティティ (IAM ユーザーまたはロール) に対してコンソールが意図したとおりに機能しません。

IAM エンティティが Image Builder コンソールを使用できるようにするには、次のいずれかの AWS マネージドポリシーをアタッチする必要があります。

- AWSImageBuilderReadOnlyAccess ポリシー
- AWSImageBuilderFullAccess ポリシー

Image Builderのマネージドポリシーの詳細については、EC2 Image Builder 用のマネージドポリシーの使用を参照してください。

> **Important**
>
> Image Builderサービス連携ロールを作成するには、AWSImageBuilderFullAccessポリシーが必要です。このポリシーを IAM エンティティにアタッチするときは、以下のカスタムポリシーもアタッチする必要があります。また、使用したくて、リソース名には `imagebuilder` が含まれていないリソースを含める必要があります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "sns topic arn"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetInstanceProfile"
            ],
```

```
            "Resource": "instance profile role arn"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "instance profile role arn",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ec2.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "bucket arn"
        }
    ]
}
```

AWS CLI または AWS API のみを呼び出すユーザーには、最小限のコンソールアクセス許可を付与する必要はありません。代わりに、実行しようとしている API オペレーションに一致するアクションのみへのアクセスが許可されます。

EC2 Image Builderリソースベースのポリシー

コンポーネントの作成方法については、Image Builder によるコンポーネントの管理を参照してください。

Image Builder コンポーネントへのアクセスを特定の IP アドレスに制限する

以下の例では、コンポーネントに対して Image Builder 操作を実行する権限を任意のユーザに付与しています。ただし、リクエストは条件で指定された IP アドレス範囲からのリクエストである必要があります。

このステートメントの条件では、54.240.143.* の範囲のインターネットプロトコルバージョン 4 (IPv4) IP アドレスが許可されています。ただし、54.240.143.188 を除きます。

Condition ブロックは、 IpAddress および NotIpAddress条件と、 AWS全体のaws:SourceIp条件キーである 条件キーを使用します。これらの条件キーの詳細については、

「ポリシーでの条件の指定」を参照してください。aws:sourceIpIPv4 値は標準の CIDR 表記を使用します。詳細については、IAM ユーザーガイドの IP アドレス条件演算子 を参照してください。

```
{
  "Version": "2012-10-17",
  "Id": "IBPolicyId1",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "imagebuilder.GetComponent:*",
      "Resource": "arn:aws:imagebuilder:::examplecomponent/*",
      "Condition": {
         "IpAddress": {"aws:SourceIp": "54.240.143.0/24"},
         "NotIpAddress": {"aws:SourceIp": "54.240.143.188/32"}
      }
    }
  ]
}
```

EC2 Image Builder 用のマネージドポリシーの使用

AWS 管理ポリシーは、 によって作成および管理されるスタンドアロンポリシーです AWS。 AWS 管理ポリシーは、多くの一般的なユースケースに対するアクセス許可を付与するように設計されているため、ユーザー、グループ、ロールへのアクセス許可の割り当てを開始できます。

AWS 管理ポリシーは、すべての AWS お客様が使用できるため、特定のユースケースに対して最小特権のアクセス許可を付与しない場合があることに注意してください。ユースケース別にカスタマーマネージドポリシーを定義して、マネージドポリシーを絞り込むことをお勧めします。

AWS 管理ポリシーで定義されているアクセス許可は変更できません。が AWS 管理ポリシーで定義されたアクセス許可 AWS を更新すると、ポリシーがアタッチされているすべてのプリンシパル ID (ユーザー、グループ、ロール) が更新されます。 は、新しい AWS のサービス が起動されたとき、または既存のサービスで新しい API AWS オペレーションが使用可能になったときに、 AWS 管理ポリシーを更新する可能性が最も高くなります。

詳細については、「IAM ユーザーガイド」の「AWS マネージドポリシー」を参照してください。

AWSImageBuilderFullAccess ポリシー

この AWSImageBuilderFullAccess ポリシーは、アタッチされているロールの Image Builder リソースへのフルアクセスを許可し、ロールが Image Builder リソースを一覧表示、説明、作成、更新、削除できるようにします。このポリシー AWS のサービス は、リソースの検証や、アカウントの現在のリソースを に表示するなど、必要な関連 にターゲットを絞ったアクセス許可も付与します AWS Management Console。

許可の詳細

このポリシーには、以下の権限が含まれています。

- Image Builder — ロールが Image Builder リソースを一覧表示、説明、作成、更新、削除できるように、管理者権限が付与されます。
- Amazon EC2 — リソースの存在を確認したり、アカウントに属するリソースのリストを取得したりするために必要な Amazon EC2 Describe アクションへのアクセスが許可されます。
- IAM — 名前に「imagebuilder」が含まれるインスタンスプロファイルの取得と使用、`iam:GetRole` API アクションによる Image Builder サービスにリンクされたロールの存在の確認、および Image Builder サービスにリンクされたロールの作成を行うためのアクセス権が付与されます。
- ライセンス　マネージャ — リソースのライセンス構成またはライセンスを一覧表示するためのアクセス権が付与されます。
- Amazon S3 — アカウントに属するバケットと、名前に「imagebuilder」が含まれる Image Builder バケットを一覧表示するためのアクセスが許可されます。
- Amazon SNS —「imagebuilder」を含むトピックの所有権を確認するための書き込み権限が Amazon SNS に付与されます。

ポリシーの例

以下はAWSImageBuilderFullAccessポリシーの例です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "imagebuilder:*"
            ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:ListTopics"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:*:*:*imagebuilder*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "license-manager:ListLicenseConfigurations",
                "license-manager:ListLicenseSpecificationsForResource"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
imagebuilder.amazonaws.com/AWSServiceRoleForImageBuilder"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetInstanceProfile"
            ],
            "Resource": "arn:aws:iam::*:instance-profile/*imagebuilder*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListInstanceProfiles",
                "iam:ListRoles"
```

```
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": [
                "arn:aws:iam::*:instance-profile/*imagebuilder*",
                "arn:aws:iam::*:role/*imagebuilder*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ec2.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:GetBucketLocation"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3::*:*imagebuilder*"
        },
        {
            "Action": "iam:CreateServiceLinkedRole",
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
imagebuilder.amazonaws.com/AWSServiceRoleForImageBuilder",
            "Condition": {
                "StringLike": {
                    "iam:AWSServiceName": "imagebuilder.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:DescribeImages",
                "ec2:DescribeSnapshots",
                "ec2:DescribeVpcs",
                "ec2:DescribeRegions",
                "ec2:DescribeVolumes",
                "ec2:DescribeSubnets",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeInstanceTypeOfferings",
                "ec2:DescribeLaunchTemplates"
            ],
            "Resource": "*"
        }
    ]
}
```

AWSImageBuilderReadOnlyAccess ポリシー

この AWSImageBuilderReadOnlyAccess ポリシーは、Image Builder のすべてのリソースへの読み取り専用アクセスを提供します。`iam:GetRole` API アクションにより、Image Builder サービスにリンクされたロールが存在することを確認する権限が付与されます。

許可の詳細

このポリシーには、以下の権限が含まれています。

- Image Builder — Image Builder リソースへの読み取り専用アクセスに対してアクセスが許可されます。
- IAM — `iam:GetRole` API アクションにより、Image Builder サービスにリンクされたロールの存在を確認するためのアクセス権限が付与されます。

ポリシーの例

以下はAWSImageBuilderReadOnlyAccessポリシーの例です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "imagebuilder:Get*",
                "imagebuilder:List*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/
imagebuilder.amazonaws.com/AWSServiceRoleForImageBuilder"
        }
    ]
}
```

AWSServiceRoleForImageBuilder ポリシー

このAWSServiceRoleForImageBuilderポリシーは、Image Builder がユーザー AWS のサービス に代わって を呼び出すことを許可します。

許可の詳細

Image Builder サービスにリンクされたロールは、ロールが Systems Manager で作成されたときに、そのロールにアタッチされます。付与される特定の権限を確認するには、このセクションの「ポリシーの例」を参照してください。Image Builderサービス連携ロールの詳細については、EC2 Image Builder サービスにリンクされたロールを使用するを参照してください。

ポリシーには以下のアクセス権限が含まれています。

- CloudWatch ログ – 名前が で始まるロググループに対して CloudWatch ログを作成してアップロードするためのアクセス許可が付与されます`/aws/imagebuilder/`。
- Amazon EC2 — 作成または使用中のイメージ、インスタンス、および `CreatedBy: EC2 Image Builder` あるいは `CreatedBy: EC2 Fast Launch` のタグが付いているボリュームに限り、関連するスナップショット、ボリューム、ネットワークインターフェイス、サブネット、セキュリティグループ、ライセンス設定、およびキーペアを必要に応じて使用して、Image Builder がお客様のアカウントでイメージを作成し、EC2 インスタンスを起動するためのアクセス権が付与されます。

Image Builder は、Amazon EC2 イメージ、インスタンス属性、インスタンスステータス、アカウントで使用できるインスタンスタイプ、起動テンプレート、サブネット、ホスト、Amazon EC2 リソースのタグに関する情報を取得できます。

Image Builder はイメージ設定を更新して、イメージに `CreatedBy: EC2 Image Builder` のタグが付けられているアカウント内の Windows インスタンスの高速起動を有効または無効にすることができます。

さらに、Image Builder では、アカウントで実行されているインスタンスの起動、停止、終了、Amazon EBS スナップショットの共有、イメージの作成と更新、テンプレートの起動、既存のイメージの登録解除、タグの追加、Ec2ImageBuilderCrossAccountDistributionAccess ポリシーによってアクセス権限を付与したアカウント間でのイメージの複製を行うことができます。前述のように、これらすべてのアクションには Image Builder のタグ付けが必要です。

- Amazon ECR — Image Builder には、コンテナイメージの脆弱性スキャンに必要なリポジトリを作成し、作成したリソースにタグを付けて操作の範囲を制限するためのアクセス権限が付与されます。また、Image Builder が脆弱性のスナップショットを取得した後、スキャンのために作成したコンテナイメージを削除するためのアクセス権も付与されます。
- EventBridge – Image Builder には、 EventBridge ルールを作成および管理するためのアクセス許可が付与されます。
- IAM — Image Builder には、アカウント内の任意のロールを Amazon EC2 と VM Import/Exportに渡すためのアクセス権限が付与されます。
- Amazon Inspector — Image Builder には、Amazon Inspector がビルドinstanceのスキャンをいつ完了するかを決定し、それを許可するように設定されたイメージの結果を収集するためのアクセス権限が付与されます。
- AWS KMS — Amazon EBS には Amazon EBS ボリュームを暗号化、復号化、または再暗号化するためのアクセス権限が付与されます。これは、Image Builder がイメージをビルドするときに暗号化されたボリュームが確実に機能するようにするために重要です。
- ライセンス　マネージャ — Image Builder には、`license-manager:UpdateLicenseSpecificationsForResource` を介してライセンス　マネージャの仕様を更新するためのアクセス権が付与されます。
- Amazon SNS — アカウント内のすべての Amazon SNS トピックに書き込み権限が付与されます。
- Systems Manager — Image Builder には、Systems Manager コマンドとその呼び出しおよびインベントリエントリの一覧表示、インスタンス情報とオートメーションの実行ステータスの説明、お

よびコマンド呼び出しを取得するアクセス権限が付与されます。Image Builder は自動化シグナルを送信し、アカウント内の任意のリソースの自動化実行を停止することもできます。

Image Builder は、以下のスクリプトファイルに対して "CreatedBy": "EC2 Image Builder"のタグが付けられたインスタンスに対して実行コマンドを発行することができます：AWS-RunPowerShellScript、AWS-RunShellScript、またはAWSEC2-RunSysprep。Image Builder では、名前が ImageBuilder で始まる自動化ドキュメントの Systems Manager 自動化実行をアカウント内で開始できます。

Image Builder では、関連付けドキュメントが AWS-GatherSoftwareInventory である限り、アカウント内の任意のインスタンスの State Manager 関連付けを作成または削除したり、アカウントに Systems Manager サービスにリンクされたロールを作成したりすることもできます。

- AWS STS — Image Builder には、ロールの信頼ポリシーで許可されている任意のアカウントに、アカウントから指定された EC2ImageBuilderDistributionCrossAccountRole ロールを引き継ぐためのアクセス権限が付与されます。これは、クロスアカウントImage BuilのImage Builder

ポリシーの例

以下は AWSServiceRoleForImageBuilder ポリシーの例です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:RunInstances"
            ],
            "Resource": [
                "arn:aws:ec2:*::image/*",
                "arn:aws:ec2:*::snapshot/*",
                "arn:aws:ec2:*:*:subnet/*",
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:security-group/*",
                "arn:aws:ec2:*:*:key-pair/*",
                "arn:aws:ec2:*:*:launch-template/*",
                "arn:aws:license-manager:*:*:license-configuration:*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "ec2:RunInstances"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*:*:instance/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": [
                        "EC2 Image Builder",
                        "EC2 Fast Launch"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "ec2.amazonaws.com",
                        "ec2.amazonaws.com.cn",
                        "vmie.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StopInstances",
                "ec2:StartInstances",
                "ec2:TerminateInstances"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CopyImage",
                "ec2:CreateImage",
                "ec2:CreateLaunchTemplate",
                "ec2:DeregisterImage",
                "ec2:DescribeImages",
                "ec2:DescribeInstanceAttribute",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceTypeOfferings",
                "ec2:DescribeInstanceTypes",
                "ec2:DescribeSubnets",
                "ec2:DescribeTags",
                "ec2:ModifyImageAttribute",
                "ec2:DescribeImportImageTasks",
                "ec2:DescribeExportImageTasks",
                "ec2:DescribeSnapshots",
                "ec2:DescribeHosts"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ModifySnapshotAttribute"
            ],
            "Resource": "arn:aws:ec2:*::snapshot/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateTags"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "ec2:CreateAction": [
```

```
                    "RunInstances",
                    "CreateImage"
                ],
                "aws:RequestTag/CreatedBy": [
                    "EC2 Image Builder",
                    "EC2 Fast Launch"
                ]
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "ec2:CreateTags"
        ],
        "Resource": [
            "arn:aws:ec2:*::image/*",
            "arn:aws:ec2:*:*:export-image-task/*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "ec2:CreateTags"
        ],
        "Resource": [
            "arn:aws:ec2:*::snapshot/*",
            "arn:aws:ec2:*:*:launch-template/*"
        ],
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/CreatedBy": [
                    "EC2 Image Builder",
                    "EC2 Fast Launch"
                ]
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "license-manager:UpdateLicenseSpecificationsForResource"
        ],
        "Resource": "*"
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:ListCommands",
                "ssm:ListCommandInvocations",
                "ssm:AddTagsToResource",
                "ssm:DescribeInstanceInformation",
                "ssm:GetAutomationExecution",
                "ssm:StopAutomationExecution",
                "ssm:ListInventoryEntries",
                "ssm:SendAutomationSignal",
                "ssm:DescribeInstanceAssociationsStatus",
                "ssm:DescribeAssociationExecutions",
                "ssm:GetCommandInvocation"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ssm:SendCommand",
            "Resource": [
                "arn:aws:ssm:*:*:document/AWS-RunPowerShellScript",
                "arn:aws:ssm:*:*:document/AWS-RunShellScript",
                "arn:aws:ssm:*:*:document/AWSEC2-RunSysprep",
                "arn:aws:s3:::*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:SendCommand"
            ],
            "Resource": [
                "arn:aws:ec2:*:*:instance/*"
            ],
            "Condition": {
```

```
                "StringEquals": {
                    "ssm:resourceTag/CreatedBy": [
                        "EC2 Image Builder"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ssm:StartAutomationExecution",
            "Resource": "arn:aws:ssm:*:*:automation-definition/ImageBuilder*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ssm:CreateAssociation",
                "ssm:DeleteAssociation"
            ],
            "Resource": [
                "arn:aws:ssm:*:*:document/AWS-GatherSoftwareInventory",
                "arn:aws:ssm:*:*:association/*",
                "arn:aws:ec2:*:*:instance/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncryptFrom",
                "kms:ReEncryptTo",
                "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "kms:EncryptionContextKeys": [
                        "aws:ebs:id"
                    ]
                },
                "StringLike": {
                    "kms:ViaService": [
                        "ec2.*.amazonaws.com"
                    ]
```

```
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:DescribeKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:ViaService": [
                        "ec2.*.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "kms:CreateGrant",
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "kms:GrantIsForAWSResource": true
                },
                "StringLike": {
                    "kms:ViaService": [
                        "ec2.*.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::*:role/
EC2ImageBuilderDistributionCrossAccountRole"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
```

```
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/imagebuilder/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateLaunchTemplateVersion",
                "ec2:DescribeLaunchTemplates",
                "ec2:ModifyLaunchTemplate",
                "ec2:DescribeLaunchTemplateVersions"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ExportImage"
            ],
            "Resource": "arn:aws:ec2:*::image/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:ExportImage"
            ],
            "Resource": "arn:aws:ec2:*:*:export-image-task/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CancelExportTask"
            ],
            "Resource": "arn:aws:ec2:*:*:export-image-task/*",
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
```

```
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": [
                        "ssm.amazonaws.com",
                        "ec2fastlaunch.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:EnableFastLaunch"
            ],
            "Resource": [
                "arn:aws:ec2:*::image/*",
                "arn:aws:ec2:*:*:launch-template/*"
            ],
            "Condition": {
                "StringEquals": {
                    "ec2:ResourceTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "inspector2:ListCoverage",
                "inspector2:ListFindings"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "aws:RequestTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:TagResource"
            ],
            "Resource": "arn:aws:ecr:*:*:repository/image-builder-*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchDeleteImage"
            ],
            "Resource": "arn:aws:ecr:*:*:repository/image-builder-*",
            "Condition": {
                "StringEquals": {
                    "ecr:ResourceTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "events:DeleteRule",
                "events:DescribeRule",
                "events:PutRule",
                "events:PutTargets",
                "events:RemoveTargets"
            ],
            "Resource": [
                "arn:aws:events:*:*:rule/ImageBuilder-*"
            ]
        }
    ]
}
```

Ec2ImageBuilderCrossAccountDistributionAccess ポリシー

Ec2ImageBuilderCrossAccountDistributionAccess このポリシーは、Image Builder にターゲットリージョンのアカウントにイメージを配布する権限を付与します。さらに、Image Builder はアカウント内の任意の Amazon EC2 イメージにタグを記述、コピー、および適用できます。このポリシーでは、`ec2:ModifyImageAttribute` API アクションを使用して AMI の権限を変更する機能も付与されます。

許可の詳細

このポリシーには、以下の権限が含まれています。

- Amazon EC2 — Amazon EC2 には、イメージの属性を記述、コピー、変更したり、アカウント内の任意の Amazon EC2 イメージのタグを作成したりするためのアクセス権限が付与されます。

ポリシーの例

以下はEc2ImageBuilderCrossAccountDistributionAccessポリシーの例です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:CreateTags",
            "Resource": "arn:aws:ec2:*::image/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeImages",
                "ec2:CopyImage",
                "ec2:ModifyImageAttribute"
            ],
            "Resource": "*"
        }
    ]
}
```

EC2ImageBuilderLifecycleExecutionPolicy ポリシー

EC2ImageBuilderLifecycleExecutionPolicy ポリシーは、イメージライフサイクル管理タスクの自動ルールをサポートするために、Image Builder イメージリソースとその基盤となるリソース (AMI、スナップショット) の非推奨、無効化、削除などのアクションを実行する権限を Image Builder に付与します。

許可の詳細

このポリシーには、以下の権限が含まれています。

- Amazon EC2 – Amazon EC2 には、`CreatedBy: EC2 Image Builder` のタグが付けられたアカウントの Amazon マシンイメージ (AMI) に対して以下のアクションを実行するためのアクセス権限が付与されます。
  - AMI を有効化および無効化する。
  - イメージ廃止を有効化および無効化する。
  - AMI の記述と登録解除をする。
  - AMI イメージ属性を記述および変更する。
  - AMI に関連付けられているボリュームスナップショットを削除する。
  - リソースのタグを取得する。
  - AMI のタグを追加または削除して廃止する。
- Amazon ECR – Amazon ECR には、`LifecycleExecutionAccess: EC2 Image Builder` タグ付きの ECR リポジトリに対して以下のバッチアクションを実行するためのアクセス権限が付与されます。バッチアクションは、自動コンテナイメージライフサイクルルールをサポートします。
  - `ecr:BatchGetImage`
  - `ecr:BatchDeleteImage`

  `LifecycleExecutionAccess: EC2 Image Builder` のタグが付けられた ECR リポジトリには、リポジトリレベルでアクセス権限が付与されます。
- AWS リソースグループ – Image Builder には、タグに基づいてリソースを取得するためのアクセス許可が付与されます。
- EC2 Image Builder – Image Builder には、Image Builder イメージリソースを削除するためのアクセス権限が付与されます。

ポリシーの例

以下は EC2ImageBuilderLifecycleExecutionPolicy ポリシーの例です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Ec2ImagePermission",
            "Effect": "Allow",
            "Action": [
                "ec2:EnableImage",
                "ec2:DeregisterImage",
                "ec2:EnableImageDeprecation",
                "ec2:DescribeImageAttribute",
                "ec2:DisableImage",
                "ec2:DisableImageDeprecation"
            ],
            "Resource": "arn:aws:ec2:*::image/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
        {
            "Sid": "EC2DeleteSnapshotPermission",
            "Effect": "Allow",
            "Action": "ec2:DeleteSnapshot",
            "Resource": "arn:aws:ec2:*::snapshot/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/CreatedBy": "EC2 Image Builder"
                }
            }
        },
        {
            "Sid": "EC2TagsPermission",
            "Effect": "Allow",
            "Action": [
                "ec2:DeleteTags",
                "ec2:CreateTags"
            ],
            "Resource": [
```

```
                "arn:aws:ec2:*::snapshot/*",
                "arn:aws:ec2:*::image/*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/DeprecatedBy": "EC2 Image Builder",
                    "aws:ResourceTag/CreatedBy": "EC2 Image Builder"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "DeprecatedBy"
                }
            }
        },
        {
            "Sid": "ECRImagePermission",
            "Effect": "Allow",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:BatchDeleteImage"
            ],
            "Resource": "arn:aws:ecr:*:*:repository/*",
            "Condition": {
                "StringEquals": {
                    "ecr:ResourceTag/LifecycleExecutionAccess": "EC2 Image Builder"
                }
            }
        },
        {
            "Sid": "ImageBuilderEC2TagServicePermission",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeImages",
                "tag:GetResources",
                "imagebuilder:DeleteImage"
            ],
            "Resource": "*"
        }
    ]
}
```

EC2InstanceProfileForImageBuilder ポリシー

この EC2InstanceProfileForImageBuilder ポリシーは、EC2 インスタンスが Image Builder と連携するために必要な最小限のアクセス権限を付与します。これには、Systems Manager エージェントを使用するために必要な権限は含まれていません。

許可の詳細

このポリシーには、以下の権限が含まれています。

- CloudWatch ログ – 名前が で始まるロググループに対して CloudWatch ログを作成してアップロードするためのアクセス許可が付与されます`/aws/imagebuilder/`。
- Image Builder — すべてのImage Builder コンポーネントを取得するためのアクセス権が付与されます。
- AWS KMS – Image Builder コンポーネントが を介して暗号化されている場合、そのコンポーネントを復号するためのアクセス許可が付与されます AWS KMS。
- Amazon S3 — 名前が`ec2imagebuilder-`で始まる Amazon S3 バケットに保存されているオブジェクトを取得するためのアクセスが許可されます。

ポリシーの例

以下はEC2InstanceProfileForImageBuilderポリシーの例です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "imagebuilder:GetComponent"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "*",
            "Condition": {
```

```
                "ForAnyValue:StringEquals": {
                    "kms:EncryptionContextKeys": "aws:imagebuilder:arn",
                    "aws:CalledVia": [
                        "imagebuilder.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::ec2imagebuilder*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/imagebuilder/*"
        }
    ]
}
```

EC2InstanceProfileForImageBuilderECRContainerBuilds ポリシー

この EC2InstanceProfileForImageBuilderECRContainerBuilds ポリシーは、Image Builder を使用して Docker イメージを構築し、そのイメージを Amazon ECR コンテナリポジトリに登録して保存する場合に、EC2 インスタンスに必要な最小限のアクセス権限を付与します。これには、Systems Manager エージェントを使用するために必要な権限は含まれていません。

許可の詳細

このポリシーには、以下の権限が含まれています。

- CloudWatch ログ – 名前が で始まるロググループに対して CloudWatch ログを作成してアップロードするためのアクセス許可が付与されます`/aws/imagebuilder/`。
- Amazon ECR — Amazon ECR には、コンテナイメージを取得、登録、保存、および認証トークンの取得を行うためのアクセス権限が付与されます。

- Image Builder — Image Builder コンポーネントまたはコンテナレシピを取得するためのアクセス権が付与されます。
- AWS KMS – Image Builder コンポーネントまたはコンテナレシピが で暗号化されている場合、復号するためのアクセス許可が付与されます AWS KMS。
- Amazon S3 — 名前が`ec2imagebuilder-`で始まる Amazon S3 バケットに保存されているオブジェクトを取得するためのアクセスが許可されます。

ポリシーの例

以下はEC2InstanceProfileForImageBuilderECRContainerBuildsポリシーの例です。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "imagebuilder:GetComponent",
                "imagebuilder:GetContainerRecipe",
                "ecr:GetAuthorizationToken",
                "ecr:BatchGetImage",
                "ecr:InitiateLayerUpload",
                "ecr:UploadLayerPart",
                "ecr:CompleteLayerUpload",
                "ecr:BatchCheckLayerAvailability",
                "ecr:GetDownloadUrlForLayer",
                "ecr:PutImage"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "kms:EncryptionContextKeys": "aws:imagebuilder:arn",
                    "aws:CalledVia": [
                        "imagebuilder.amazonaws.com"
```

```
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:::ec2imagebuilder*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:CreateLogGroup",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:log-group:/aws/imagebuilder/*"
        }
    ]
}
```

Image Builder の AWS マネージドポリシーの更新

このセクションでは、このサービスがこれらの変更の追跡を開始してからの Image Builder の AWS マネージドポリシーの更新に関する情報を提供します。このページの変更に関する自動通知については、ドキュメントの履歴ページの RSS フィードをサブスクライブしてください。

変更	説明	日付
EC2ImageBuilderLifecycleExecutionPolicy - 新しいポリシー	Image Builder は、イメージライフサイクル管理の権限を含む新しい `EC2ImageBuilderLifecycleExecutionPolicy` ポリシーを追加しました。	2023 年 11 月 17 日

変更	説明	日付
AWSServiceRoleForImageBuilder – 既存ポリシーへの更新	Image Builder は、macOS サポートを提供するため、サービスロールに次の変更を加えました。 • ec2:DescribeHosts Enable Image Builder が hostId をポーリングして、インスタンスを起動するための有効な状態がいつあるかを判断するように追加されました。 • Image Builder がコマンド呼び出しの詳細を取得するために使用するメソッドを改善する ssm:GetCommandInvocation, API アクションを追加しました。	2023 年 8 月 28 日

変更	説明	日付
AWSServiceRoleForImageBuilder – 既存ポリシーへの更新	Image Builder はサービスロールに以下の変更を加え、Image Builder ワークフローが AMI と ECR の両方のコンテナイメージビルドの脆弱性結果を収集できるようにしました。新しい権限は CVE 検出およびレポート機能をサポートします。 • Inspector2: ListCoverage および Inspector2: を追加 ListFindings し、Amazon Inspector がテストインスタンススキャンを完了するタイミングを判断し、それを許可するように設定されたイメージの検出結果を収集できるようにしました。 • ecr: を追加CreateRepository しました。Image Builder がリポジトリに `CreatedBy : EC2 Image Builder` () のタグを付けることを要件としましたtag-on-create。また、同じ CreatedBy タグ制約を持つ ecr:TagResource (に必要 tag-on-create) と、リポジトリ名を で始める必要がある追加の制約を追加しましたimage-builder-* 。名前の制約は、権限の昇格を防ぎ、I	2023 年 3 月 30 日

変更	説明	日付
	mage Builder が作成しなかったリポジトリへの変更を防ぎます。 • でタグ付けされた ECR リポジトリBatchDeleteImage の ecr:for を追加しましたCreatedBy: EC2 Image Builder。この権限では、リポジトリ名が image-builder-* で始まる必要があります。 • Image Builder が名前に を含む Amazon EventBridge マネージドルールを作成および管理するためのイベントアクセス許可を追加ImageBuilder-* しました。	
AWSServiceRoleForImageBuilder – 既存ポリシーへの更新	Image Builder は、サービスロールに次の変更を加えました。 • ec2:RunInstance call のリソースとして License Manager ライセンスを追加し、ライセンス設定に関連付けられたベースイメージ AMIs の使用をお客様に許可しました。	2022 年 3 月 22 日

変更	説明	日付
AWSServiceRoleForImageBuilder – 既存ポリシーへの更新	Image Builder は、サービスロールに次の変更を加えました。 • Windows インスタンスの高速起動を有効または無効にする EC2 EnableFastLaunch API アクションのアクセス許可を追加しました。 • ec2:CreateTags action とリソースタグの条件の範囲を厳しくしました。	2022 年 2 月 21 日
AWSServiceRoleForImageBuilder – 既存ポリシーへの更新	Image Builder は、サービスロールに次の変更を加えました。 • VMIE サービスを呼び出して VM をインポートし、そこからベース AMI を作成する権限を追加しました。 • ec2:CreateTags action およびリソースタグ条件の範囲を厳しくしました。	2021 年 11 月 20 日
AWSServiceRoleForImageBuilder – 既存ポリシーへの更新	Image Builder には、複数のインベントリ関連付けによってイメージビルドが停止する問題を修正する新しい権限が追加されました。	2021 年 8 月 11 日

変更	説明	日付
AWSImageBuilderFullAccess – 既存ポリシーへの更新	Image Builder は、サービスロールに次の変更を加えました。 • `ec2:DescribeInstanceTypeOffereings` を許可するパーミッションを追加。 • Image Builder コンソールがアカウントで使用可能なインスタンスタイプを正確に反映できるようにするため `ec2:DescribeInstanceTypeOffereings` の呼び出し権限を追加しました。	2021 年 4 月 13 日
Image Builder が変更の追跡を開始しました	Image Builder が AWS マネージドポリシーの変更の追跡を開始しました。	2021 年 4 月 02 日

EC2 Image Builder サービスにリンクされたロールを使用する

EC2 Image Builder は AWS Identity and Access Management 、 (IAM) サービスにリンクされたロール を使用します。サービス連携ロールは、Image Builderに直接リンクされた独自のタイプのIAMロールです。サービスにリンクされたロールは Image Builder によって事前定義されており、サービスが AWS のサービス ユーザーに代わって他の を呼び出すために必要なすべてのアクセス許可が含まれています。

サービスリンクされたロールは、必要なパーミッションを手動で追加する必要がないため、Image Builderのセットアップをより効率的にします。Image Builderは、サービスリンクされたロールの権限を定義し、他に定義されていない限り、Image Builderだけがその ロールを引き受けることができ

ます。定義されたアクセス許可には、信頼ポリシーとアクセス許可ポリシーが含まれます。アクセス許可ポリシーを他の IAM エンティティにアタッチすることはできません。

サービスにリンクされたロールをサポートする他のサービスについては、「IAM と連携するAWS のサービス」の「サービスにリンクされたロール」列が「はい」になっているサービスを検索してください。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

Image Builderのサービス連動ロールパーミッション

Image Builder は、AWSServiceRoleForImageBuilderサービスにリンクされたロールを使用して、EC2 Image Builder がユーザーに代わって AWS リソースにアクセスできるようにします。 サービスにリンクされたロールは、ロールを引き受ける上で imagebuilder.amazonaws.com サービスを信頼します。

サービスにリンクされたこのロールを手動で作成する必要はありません。 AWS マネジメントコンソール、、 AWS CLIまたは AWS API で最初の Image Builder イメージを作成すると、Image Builder によってサービスにリンクされたロールが作成されます。

次のアクションは新しいイメージを作成します。

- Image Builder コンソールのパイプラインウィザードを実行して、カスタムイメージを作成します。
- 次のいずれかの API アクション、または対応する AWS CLI コマンドを使用します。
  - CreateImage API アクション (create-image の AWS CLI)。
  - ImportVmImage API アクション (import-vm-image の AWS CLI)。
  - StartImagePipelineExecution API アクション (start-image-pipeline-execution の AWS CLI)。

> **Important**
>
> サービスリンクされたロールがアカウントから削除された場合、同じ手順で再度作成することができます。最初の EC2 Image Builder リソースを作成すると、Image Builder はサービスにリンクされたロールを再度作成します。

AWSServiceRoleForImageBuilder ポリシーの権限を確認するには、AWSServiceRoleForImageBuilderページを参照してください。サービスにリンクされたロール

の権限の設定の詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの権限」を参照してください。

Image Builder サービスリンクローラをアカウントから削除します

IAM コンソール、、または AWS API を使用して AWS CLI、Image Builder のサービスにリンクされたロールをアカウントから手動で削除できます。ただし、その前に、それを参照する Image Builder リソースが有効になっていないことを確認する必要があります。

> **Note**
>
> リソースを削除しようとしたときに Image Builder サービスがロールを使用している場合、削除に失敗することがあります。失敗した場合は、数分待ってから操作を再試行してください。

AWSServiceRoleForImageBuilder ロールが使用する Image Builder リソースをクリーンアップします。

1. 開始する前に、パイプラインビルドが実行されていないことを確認してください。実行中のビルドをキャンセルするには、 AWS CLIからcancel-image-creationコマンドを使用します。

   ```
   aws imagebuilder cancel-image-creation --image-build-version-
   arn arn:aws:imagebuilder:us-east-1:123456789012:image-pipeline/sample-pipeline
   ```

2. すべてのパイプラインスケジュールを手動ビルドプロセスを使用するように変更するか、今後使用しない場合は削除してください。リソースの削除については、EC2 Image Builder resources の削除を参照してください。

IAM を使用して、サービスにリンクされたロールを削除します。

IAM コンソール、、または AWS API を使用して AWS CLI、アカウントからAWSServiceRoleForImageBuilderロールを削除できます。詳細については、「IAM ユーザーガイド」の「サービスにリンクされたロールの削除」を参照してください。

EC2 Image Builderサービス連携ロールでサポートされるリージョン

Image Builder は、サービスが利用可能なすべての AWS リージョンでサービスにリンクされたロールの使用をサポートしています。サポートされている AWS リージョンのリストについては、「」を参照してくださいAWS リージョンとエンドポイント。

EC2 Image Builder アイデンティティとアクセスのトラブルシューティング

トピック

Image Builder でアクションを実行する権限がありません。

「I am not authorized to perform an action in Amazon Bedrock」というエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要なimagebuilder:*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 imagebuilder:GetWidget on resource: my-example-widget
```

この場合、imagebuilder:*GetWidget* アクションを使用して *my-example-widget*リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

iam を実行する権限がありません。PassRole

iam:PassRoleのアクションを実行する権限がないというエラーが表示された場合、Image Builder にロールを渡せるようにポリシーを更新する必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

次の例のエラーは、marymajorという IAM ユーザーがコンソールを使用して Image Builder でアクションを実行しようとしたときに発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。Mary には、ロールをサービスに渡す権限がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、 AWS 管理者にお問い合わせください。サインイン資格情報を提供した担当者が管理者です。

自分の 以外のユーザーに Image Builder リソース AWS アカウント へのアクセスを許可したい

他のアカウントのユーザーや組織外の人が、リソースにアクセスするために使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください:

- Image Builderがこれらの機能をサポートしているかどうかについては、EC2 Image BuilderとIAMの仕組みを参照してください。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、IAM ユーザーガイドの「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティー に提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、『IAM ユーザーガイド』の「外部で認証されたユーザー (ID フェデレーション) へのアクセス権限」を参照してください。
- クロスアカウントアクセスでのロールとリソースベースのポリシーの使用の違いの詳細については、『IAM ユーザーガイド』の「IAM ロールとリソースベースのポリシーとの相違点」を参照してください。

EC2 Image Builder のコンプライアンス検証

EC2 Image Builder は AWS コンプライアンスプログラムの対象ではありません。

特定のコンプライアンスプログラム AWS のサービス の対象となる のリストについては、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。一般的な情報については、AWS 「コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「アーティファクト での AWS レポートのダウンロード」を参照してください。

Image Builder を使用する際のコンプライアンス責任は、データの機密性、企業のコンプライアンス目的、および適用される法律と規制によって決定されます。 AWS では、コンプライアンスに役立つ以下のリソースを提供しています：

- 「セキュリティ＆コンプライアンスクイックリファレンスガイド」 – これらのデプロイガイドには、アーキテクチャ上の考慮事項の説明と、 AWSでセキュリティとコンプライアンスに重点を置いたベースライン環境をデプロイするための手順が記載されています。
- AWS コンプライアンスリソース – このワークブックとガイドのコレクションは、お客様の業界や地域に適用される場合があります。
- 「 デベロッパーガイド」の「 ルールによるリソースの評価」 – この AWS Config サービスは、リソース設定が社内プラクティス、業界ガイドライン、および規制にどの程度準拠しているかを評価します。 AWS Config
- AWS Security Hub – この AWS サービスは、 内のセキュリティ状態を包括的に把握 AWS し、セキュリティ業界標準とベストプラクティスへの準拠を確認するのに役立ちます。

AWS Task Orchestrator and Executor （ AWSTOE) の AWS Marketplace または のコンポーネントからのコンプライアンス製品を Image Builder イメージに組み込むことで、イメージが準拠していることを確認できます。詳細については、「Image Builder イメージ用のコンプライアンス製品」を参照してください。

EC2 Image Builder でのレジリエンス

AWS グローバルインフラストラクチャは、 AWS リージョンとアベイラビリティーゾーンを中心に構築されています。 AWS リージョンは、低レイテンシー、高スループット、および高度に冗長なネットワークで接続された、物理的に分離および分離された複数のアベイラビリティーゾーンを提供します。アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェイルオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性が高く、フォールトトレラントで、スケーラブルです。

EC2 Image Builder サービスを使用すると、あるリージョンで構築されたイメージを他のリージョンに配布できるため、AMI のマルチリージョン耐障害性が得られます。イメージパイプライン、レシピ、またはコンポーネントを「バックアップ」するメカニズムはありません。レシピとコンポーネントのドキュメントは、Amazon S3 バケットなどの Image Builder サービスの外部に保存できます。

EC2 Image Builder を高可用性 (HA) に設定することはできません。イメージを複数のリージョンに配信して、イメージの可用性を高めることができます。

AWS リージョンとアベイラビリティーゾーンの詳細については、AWS 「グローバルインフラストラクチャ」を参照してください。

Image Builder でのインフラストラクチャセキュリティ

AWS グローバルネットワークは、EC2 Image Builder などのサービスのセキュリティ機能を提供し、ネットワークアクセスを制御します。 AWS がそのサービスを支えるインフラストラクチャセキュリティの詳細については、「 AWS セキュリティ入門」ホワイトペーパーの「インフラストラクチャセキュリティ」セクションを参照してください。

Image Builder API アクションのグローバル AWS ネットワーク経由でリクエストを送信するには、クライアントソフトウェアが次のセキュリティガイドラインに準拠している必要があります。

- Image Builder API アクションのリクエストを送信するには、クライアントソフトウェアがサポートされているバTransport Layer Security (TLS)) を使用する必要があります。

  > **Note**
  >
  > AWS は、TLS バージョン 1.0 および 1.1 のサポートを段階的に廃止しています。引き続き接続できるように、ライアントソフトウェアをTLS バージョン 1.2 以上に更新することを強くお勧めします。詳細については、AWS セキュリティのブログ記事を参照してください。

- クライアントソフトウェアは、Ephemeral Diffie-Hellman (DHE)やElliptic Curve Ephemeral Diffie-Hellman (ECDHE)のような完全な前方秘匿性(PFS)を持つ暗号スイートをサポートしなければなりません。Java 7以降など、現在のシステムのほとんどはこれらのモードをサポートしています。
- （ AWS Identity and Access Management IAM) プリンシパルに関連付けられたアクセスキー ID とシークレットアクセスキーを使用して API リクエストに署名する必要があります。または、AWS Security Token Service (AWS STS) を使ってリクエストのための一時的なセキュリティ認証情報を生成することもできます。

さらに、Image Builder がイメージのビルドとテストに使用する EC2 インスタンスには、 AWS Systems Managerとアクセスする必要があります。

EC2 Image Builder でのパッチ管理

EC2 Image Builder は、最新の Amazon Linux 2、Amazon Linux 2023、Red Hat Enterprise Linux (RHEL)、CentOS、Ubuntu、SUSE Linux Enterprise Server、Windows 2012 R2 およびそれ以降の AMI を提供しています。責任分担モデルに従い、Amazon EC2 システムのパッチ適用責任はお客様が負います。アプリケーションワークロード内の EC2 インスタンスを簡単に交換できる場合は、ベース AMI を更新し、このイメージに基づいてすべてのコンピュートノードを再デプロイする方が効率的です。

Image Builder AMI を最新の状態に保つには、次の 2 つの方法があります。

- AWS-提供されているパッチコンポーネント — EC2 Image Builder にはとの 2 つのビルドコンポーネントupdate-linuxとupdate-windowsがあり、保留中のオペレーティングシステムの更新をすべてインストールします。これらのコンポーネントはUpdateOSのアクションモジュールを使用します。詳細については、「UpdateOS」を参照してください。コンポーネントは、 AWSの提供されているコンポーネントのリストから選択することで、イメージビルドパイプラインに追加できます。
- パッチ操作によるカスタムビルドコンポーネント — サポートされている AMI のオペレーティングシステムにパッチを選択的にインストールまたは更新するには、Image Builder コンポーネントを作成して必要なパッチをインストールできます。カスタムコンポーネントは、シェルスクリプト (Bash または PowerShell) を使用してパッチをインストールするか、 UpdateOSアクションモジュールを使用してインストールまたは除外するパッチを指定できます。詳細については、「AWSTOE コンポーネントマネージャーがサポートするアクションモジュール」を参照してください。

  UpdateOSのアクションモジュールを使用するコンポーネント (Linux および Windows)

  ```
  schemaVersion: 1.0
  phases:
    - name: build
   steps:
     - name: UpdateOS
    action: UpdateOS
  ```

  Bash を使用して yum アップデートをインストールするコンポーネント。

```
schemaVersion: 1.0
phases:
  - name: build
 steps:
   - name: InstallYumUpdates
  action: ExecuteBash
  inputs:
    commands:
   - sudo yum update -y
```

EC2 Image Builder でのセキュリティのベストプラクティス

EC2 Image Builderには、独自のセキュリティポリシーを策定実装する際に考慮すべきセキュリティ機能が多数用意されています。以下のベストプラクティスは一般的なガイドラインであり、完全なセキュリティソリューションを説明するものではありません。これらのベストプラクティスはお客様の環境に必ずしも適切または十分でない可能性があるので、処方箋ではなく、あくまで有用な考慮事項とお考えください。

- Image Builder レシピでは、過度に制限の厳しいセキュリティグループを使用しないでください。
- 信頼できないアカウントとイメージを共有しないでください。
- プライベートまたは機密のデータを含むイメージを公開しないでください。
- イメージのビルド時には、Windows または Linux で使用可能なすべてのセキュリティパッチを適用してください。

セキュリティのポスチャーおよび該当するセキュリティコンプライアンスレベルを検証するために、イメージをテストすることを強くお勧めします。 Amazon Inspector などのソリューションは、イメージのセキュリティとコンプライアンス状態を検証するのに役立ちます。

Image Builder パイプライン用 IMDSv2

Image Builder パイプラインが実行されると、Image Builder がイメージの構築とテストに使用する EC2 インスタンスを起動するための HTTP リクエストが送信されます。パイプラインが起動リクエストに使用する IMDS のバージョンを設定するには、Image Builderインフラストラクチャ設定インスタンスのメタデータ設定で `httpTokens` パラメータを設定します。

> **Note**
>
> Image Builder がパイプラインビルドから起動するすべての EC2 インスタンスを IMDSv2 を使用するように設定して、インスタンスメタデータの取得リクエストに署名付きトークンヘッダーが必要になるようにすることをお勧めします。

Image Builder インフラストラクチャ設定の詳細については、「EC2 Image Builder インフラストラクチャ設定の管理」を参照してください。Linux Image の EC2 インスタンスメタデータオプションの詳細については、Linux インスタンス用 Amazon EC2 ユーザーガイドの「インスタンスメタデータオプションの設定」を参照してください。Windowsイメージについては、Amazon EC2 User Guide for Windows InstancesのConfigure instance metadata optionsを参照してください。

ビルド後のクリーンアップが必要です。

Image Builder がカスタムイメージのすべてのビルドステップを完了すると、Image Builder はテストとイメージ作成のためにビルドインスタンスを準備します。ビルドインスタンスをシャットダウンしてスナップショットを作成する前に、Image Builder は次のクリーンアップを実行してイメージのセキュリティを確保します。

Linux

Image Builder パイプラインはクリーンアップスクリプトを実行して、最終的なイメージがセキュリティのベストプラクティスに従っていることを確認し、スナップショットに引き継がれてはならないビルドアーティファクトや設定をすべて削除します。ただし、スクリプトのセクションをスキップしたり、ユーザーデータを完全に上書きしたりすることはできます。そして、Image Builder パイプラインによって生成されるイメージは、必ずしも特定の規制基準に準拠しているとは限りません。

パイプラインがビルド段階とテスト段階を完了すると、Image Builder は出力イメージを作成する直前に次のクリーンアップスクリプトを自動的に実行します。

> **Important**
>
> レシピのユーザーデータをオーバーライドすると、スクリプトは実行されません。その場合は、`perform_cleanup`という名前の空のファイルを作成するコマンドをユーザーデータに含めてください。Image Builder はこのファイルを検出し、新しいイメージを作成する前にクリーンアップスクリプトを実行します。

```
#!/bin/bash
if [[ ! -f {{workingDirectory}}/perform_cleanup ]]; then
    echo "Skipping cleanup"
    exit 0
else
    sudo rm -f {{workingDirectory}}/perform_cleanup
fi

function cleanup() {
    FILES=("$@")
    for FILE in "${FILES[@]}"; do
        if [[ -f "$FILE" ]]; then
            echo "Deleting $FILE";
            sudo shred -zuf $FILE;
        fi;
        if [[ -f $FILE ]]; then
            echo "Failed to delete '$FILE'. Failing."
            exit 1
        fi;
    done
};


# Clean up for cloud-init files
CLOUD_INIT_FILES=(
    "/etc/sudoers.d/90-cloud-init-users"
    "/etc/locale.conf"
    "/var/log/cloud-init.log"
    "/var/log/cloud-init-output.log"
)
if [[ -f {{workingDirectory}}/skip_cleanup_cloudinit_files ]]; then
    echo "Skipping cleanup of cloud init files"
else
    echo "Cleaning up cloud init files"
    cleanup "${CLOUD_INIT_FILES[@]}"
    if [[ $( sudo find /var/lib/cloud -type f | sudo wc -l ) -gt 0 ]]; then
        echo "Deleting files within /var/lib/cloud/*"
        sudo find /var/lib/cloud -type f -exec shred -zuf {} \;
    fi;

    if [[ $( sudo ls /var/lib/cloud | sudo wc -l ) -gt 0 ]]; then
        echo "Deleting /var/lib/cloud/*"
        sudo rm -rf /var/lib/cloud/* || true
```

```
    fi;
fi;


# Clean up for temporary instance files
INSTANCE_FILES=(
    "/etc/.updated"
    "/etc/aliases.db"
    "/etc/hostname"
    "/var/lib/misc/postfix.aliasesdb-stamp"
    "/var/lib/postfix/master.lock"
    "/var/spool/postfix/pid/master.pid"
    "/var/.updated"
    "/var/cache/yum/x86_64/2/.gpgkeyschecked.yum"
)
if [[ -f {{workingDirectory}}/skip_cleanup_instance_files ]]; then
    echo "Skipping cleanup of instance files"
else
    echo "Cleaning up instance files"
    cleanup "${INSTANCE_FILES[@]}"
fi;


# Clean up for ssh files
SSH_FILES=(
    "/etc/ssh/ssh_host_rsa_key"
    "/etc/ssh/ssh_host_rsa_key.pub"
    "/etc/ssh/ssh_host_ecdsa_key"
    "/etc/ssh/ssh_host_ecdsa_key.pub"
    "/etc/ssh/ssh_host_ed25519_key"
    "/etc/ssh/ssh_host_ed25519_key.pub"
    "/root/.ssh/authorized_keys"
)
if [[ -f {{workingDirectory}}/skip_cleanup_ssh_files ]]; then
    echo "Skipping cleanup of ssh files"
else
    echo "Cleaning up ssh files"
    cleanup "${SSH_FILES[@]}"
    USERS=$(ls /home/)
    for user in $USERS; do
        echo Deleting /home/"$user"/.ssh/authorized_keys;
        sudo find /home/"$user"/.ssh/authorized_keys -type f -exec shred -zuf {} \;
    done
    for user in $USERS; do
```

```
        if [[ -f /home/"$user"/.ssh/authorized_keys ]]; then
            echo Failed to delete /home/"$user"/.ssh/authorized_keys;
            exit 1
        fi;
    done;
fi;


# Clean up for instance log files
INSTANCE_LOG_FILES=(
    "/var/log/audit/audit.log"
    "/var/log/boot.log"
    "/var/log/dmesg"
    "/var/log/cron"
)
if [[ -f {{workingDirectory}}/skip_cleanup_instance_log_files ]]; then
    echo "Skipping cleanup of instance log files"
else
    echo "Cleaning up instance log files"
    cleanup "${INSTANCE_LOG_FILES[@]}"
fi;

# Clean up for TOE files
if [[ -f {{workingDirectory}}/skip_cleanup_toe_files ]]; then
    echo "Skipping cleanup of TOE files"
else
    echo "Cleaning TOE files"
    if [[ $( sudo find {{workingDirectory}}/TOE_* -type f | sudo wc -l) -gt 0 ]];
 then
        echo "Deleting files within {{workingDirectory}}/TOE_*"
        sudo find {{workingDirectory}}/TOE_* -type f -exec shred -zuf {} \;
    fi
    if [[ $( sudo find {{workingDirectory}}/TOE_* -type f | sudo wc -l) -gt 0 ]];
 then
        echo "Failed to delete {{workingDirectory}}/TOE_*"
        exit 1
    fi
    if [[ $( sudo find {{workingDirectory}}/TOE_* -type d | sudo wc -l) -gt 0 ]];
 then
        echo "Deleting {{workingDirectory}}/TOE_*"
        sudo rm -rf {{workingDirectory}}/TOE_*
    fi
    if [[ $( sudo find {{workingDirectory}}/TOE_* -type d | sudo wc -l) -gt 0 ]];
 then
```

```
        echo "Failed to delete {{workingDirectory}}/TOE_*"
        exit 1
    fi
fi

# Clean up for ssm log files
if [[ -f {{workingDirectory}}/skip_cleanup_ssm_log_files ]]; then
    echo "Skipping cleanup of ssm log files"
else
    echo "Cleaning up ssm log files"
    if [[ $( sudo find /var/log/amazon/ssm -type f | sudo wc -l) -gt 0 ]]; then
        echo "Deleting files within /var/log/amazon/ssm/*"
        sudo find /var/log/amazon/ssm -type f -exec shred -zuf {} \;
    fi
    if [[ $( sudo find /var/log/amazon/ssm -type f | sudo wc -l) -gt 0 ]]; then
        echo "Failed to delete /var/log/amazon/ssm"
        exit 1
    fi
    if [[ -d "/var/log/amazon/ssm" ]]; then
        echo "Deleting /var/log/amazon/ssm/*"
        sudo rm -rf /var/log/amazon/ssm
    fi
    if [[ -d "/var/log/amazon/ssm" ]]; then
        echo "Failed to delete /var/log/amazon/ssm"
        exit 1
    fi
fi


if [[ $( sudo find /var/log/sa/sa* -type f | sudo wc -l ) -gt 0 ]]; then
    echo "Deleting /var/log/sa/sa*"
    sudo shred -zuf /var/log/sa/sa*
fi
if [[ $( sudo find /var/log/sa/sa* -type f | sudo wc -l ) -gt 0 ]]; then
    echo "Failed to delete /var/log/sa/sa*"
    exit 1
fi

if [[ $( sudo find /var/lib/dhclient/dhclient*.lease -type f | sudo wc -l ) -gt
 0 ]]; then
        echo "Deleting /var/lib/dhclient/dhclient*.lease"
        sudo shred -zuf /var/lib/dhclient/dhclient*.lease
fi
```

```
if [[ $( sudo find /var/lib/dhclient/dhclient*.lease -type f | sudo wc -l ) -gt
 0 ]]; then
        echo "Failed to delete /var/lib/dhclient/dhclient*.lease"
        exit 1
fi

if [[ $( sudo find /var/tmp -type f | sudo wc -l) -gt 0 ]]; then
        echo "Deleting files within /var/tmp/*"
        sudo find /var/tmp -type f -exec shred -zuf {} \;
fi
if [[ $( sudo find /var/tmp -type f | sudo wc -l) -gt 0 ]]; then
        echo "Failed to delete /var/tmp"
        exit 1
fi
if [[ $( sudo ls /var/tmp | sudo wc -l ) -gt 0 ]]; then
        echo "Deleting /var/tmp/*"
        sudo rm -rf /var/tmp/*
fi

# Shredding is not guaranteed to work well on rolling logs

if [[ -f "/var/lib/rsyslog/imjournal.state" ]]; then
        echo "Deleting /var/lib/rsyslog/imjournal.state"
        sudo shred -zuf /var/lib/rsyslog/imjournal.state
        sudo rm -f /var/lib/rsyslog/imjournal.state
fi

if [[ $( sudo ls /var/log/journal/ | sudo wc -l ) -gt 0 ]]; then
        echo "Deleting /var/log/journal/*"
        sudo find /var/log/journal/ -type f -exec shred -zuf {} \;
        sudo rm -rf /var/log/journal/*
fi

sudo touch /etc/machine-id
```

Windows

Image Builder パイプラインが Windows イメージをカスタマイズすると、Microsoft Sysprep ユーティリティが実行されます。これらのアクションはAWS 、イメージの強化とクリーニングのベストプラクティスに従います。

Linux クリーンアップスクリプトをオーバーライドしてください。

Image Builder は、デフォルトで安全で、セキュリティのベストプラクティスに従ったイメージを作成します。ただし、より高度なユースケースでは、組み込みのクリーンアップスクリプトの 1 つ以上のセクションをスキップする必要がある場合があります。クリーンアップの一部を省略する必要がある場合は、出力した AMI をテストしてイメージのセキュリティを確認することを強くお勧めします。

> **Important**
>
> クリーンアップスクリプトのセクションをスキップすると、所有者アカウントの詳細や SSH キーなどの機密情報が最終的なイメージや、そのイメージから起動されるすべてのインスタンスに含まれる可能性があります。また、異なるアベイラビリティーゾーン、リージョン、またはアカウントで起動すると問題が発生する可能性があります。

次の表は、クリーンアップスクリプトのセクション、そのセクションで削除されるファイル、および Image Builder がスキップすべきセクションにフラグを立てるために使用できるファイル名の概要を示しています。クリーンアップスクリプトの特定のセクションをスキップするには、CreateFileのコンポーネントアクションモジュールまたはユーザーデータ内のコマンド (オーバーライドする場合) を使用して、「セクションをスキップ」列で指定した名前の空のファイルを作成します。

> **Note**
>
> クリーンアップスクリプトのセクションをスキップするために作成するファイルには、ファイル拡張子を付けないでください。たとえば、スクリプトのCLOUD_INIT_FILESセクションをスキップしたいが、skip_cleanup_cloudinit_files.txtという名前のファイルを作成した場合、Image Builder はスキップファイルを認識できません。

入力

クリーンアップセクション	ファイルが削除されました。	セクションファイル名をスキップする
CLOUD_INIT_FILES	/etc/sudoers.d/90-cloud-init-users /etc/locale.conf	skip_cleanup_cloudinit_files

クリーンアップセクション	ファイルが削除されました。	セクションファイル名をスキップする
	/var/log/cloud-init.log /var/log/cloud-init-output.log	
INSTANCE_FILES	/etc/.updated /etc/aliases.db /etc/hostname /var/lib/misc/postfix.aliasesdb-stamp /var/lib/postfix/master.lock /var/spool/postfix/pid/master.pid /var/.updated /var/cache/yum/x86_64/2/.gpgkeyschecked.yum	skip_cleanup_instance_files

クリーンアップセクション	ファイルが削除されました。	セクションファイル名をスキップする
SSH_FILES	/etc/ssh/ssh_host_rsa_key /etc/ssh/ssh_host_rsa_key.pub /etc/ssh/ssh_host_ecdsa_key /etc/ssh/ssh_host_ecdsa_key.pub /etc/ssh/ssh_host_ed25519_key /etc/ssh/ssh_host_ed25519_key.pub /root/.ssh/authorized_keys /home/<all users>/.ssh/authorized_keys;	skip_cleanup_ssh_files
INSTANCE_LOG_FILES	/var/log/audit/audit.log /var/log/boot.log /var/log/dmesg /var/log/cron	skip_cleanup_instance_log_files
TOE_FILES	{{workingDirectory}}/TOE_*	skip_cleanup_toe_files

クリーンアップセクション	ファイルが削除されました。	セクションファイル名をスキップする
SSM_LOG_FILES	/var/log/amazon/ssm/*	skip_cleanup_ssm_log_files

EC2Image Builderのトラブルシューティング

EC2 Image Builder は と統合され、イメージビルドの問題 AWS のサービス のトラブルシューティングに役立つモニタリングとトラブルシューティングを行います。Image Builder は、イメージ構築プロセスの各ステップの進行状況を追跡して表示します。さらに、Image Builder は、指定した Amazon S3 の場所にログをエクスポートできます。

高度なトラブルシューティングを行う場合は、「AWS Systems Manager コマンドを実行」を使用して定義済みのコマンドとスクリプトを実行できます。

内容

パイプラインビルドのトラブルシューティング

Image Builder パイプラインのビルドが失敗した場合、Image Builder は失敗を説明するエラーメッセージを返します。Image Builder は、次の出力例のようなworkflow execution IDを含む失敗メッセージでもを返します。

```
Workflow Execution ID: wf-12345abc-6789-0123-abc4-567890123abc failed with reason: …
```

Image Builder は、標準のイメージ作成プロセスのランタイムステージ用に定義された一連のステップを通じて、イメージビルドアクションを整理、指示します。プロセスのビルド段階とテスト段階にはそれぞれワークフローが関連付けられています。Image Builder がワークフローを実行して新しいイメージを構築またはテストすると、ランタイムの詳細を追跡するワークフローメタデータリソースが生成されます。

コンテナイメージには、配信中に実行される追加のワークフローがあります。

ワークフローのランタイムインスタンス障害の詳細を調べてください。

ワークフローのランタイム障害をトラブルシューティングするには、 GetWorkflowExecution と ListWorkflowStepExecutions API アクションを で呼び出しますworkflow execution ID。

ワークフローのランタイムログを確認してください。

- Amazon CloudWatch Logs

Image Builder は、詳細なワークフロー実行ログを次の Image Builder CloudWatch Logs グループとストリームに発行します。

LogGroup:

```
/aws/imagebuilder/ImageName
```

LogStream (x.x.x/x):

```
ImageVersion/ImageBuildVersion
```

CloudWatch Logs を使用すると、フィルターパターンでログデータを検索できます。詳細については、「Amazon Logs ユーザーガイド」の「フィルターパターンを使用してログデータを検索する」を参照してください。 CloudWatch

- AWS CloudTrail

  アカウントでアクティブ化 CloudTrail されている場合、すべてのビルドアクティビティもログインされます。ソース で CloudTrail イベントをフィルタリングできますimagebuilder.amazonaws.com。または、実行ログで返される Amazon EC2 インスタンス ID を検索して、パイプライン実行に関する詳細を確認することもできます。

- Amazon Simple Storage Service (S3)

  インフラストラクチャ設定で S3 バケット名とkey prefix 指定した場合、ワークフローステップのランタイムログパスは次のパターンに従います。

  ```
  S3://S3BucketName/KeyPrefix/ImageName/ImageVersion/ImageBuildVersion/
  WorkflowExecutionId/StepName
  ```

  S3 バケットに送信するログには、イメージビルドプロセス中の EC2 インスタンスでのアクティビティのステップとエラーメッセージが表示されます。ログには、コンポーネントマネージャーからのログ出力、実行されたコンポーネントの定義、インスタンスで実行されたすべてのステップの詳細な出力 (JSON) が含まれます。問題が発生した場合は、application.logから始めてこれらのファイルを確認し、インスタンスの問題の原因を診断する必要があります。

デフォルトでは、パイプラインに障害が発生すると、Image Builder は実行中の Amazon EC2 ビルドまたはテストインスタンスをシャットダウンします。パイプラインが使用するインフラストラクチャ

設定リソースのインスタンス設定を変更して、ビルドインスタンスまたはテストインスタンスをトラブルシューティングのために保持することができます。

コンソールでインスタンス設定を変更するには、インフラストラクチャー設定リソースの「トラブルシューティング設定」セクションにある「Terminate instance on failure」チェックボックスをオフにする必要があります。

update-infrastructure-configurationのコマンドを使用して AWS CLIでインスタンス設定を変更することもできます。コマンドが`--cli-input-json`パラメーターで参照するJSONファイルの`terminateInstanceOnFailure`の値を`false`に設定する。詳細については、「インフラストラクチャ設定を更新します。」を参照してください。

トラブルシューティングシナリオ

このセクションには、以下の詳細なトラブルシューティングシナリオが記載されています。

- アクセス拒否 — ステータスコード 403
- ビルドインスタンスで Systems Manager エージェントの可用性を確認中にビルドがタイムアウトする
- Windows セカンダリディスクは起動時にオフラインです。
- CIS で強化されたベースイメージではビルドが失敗します。
- AssertInventoryCollection 失敗 (Systems Manager Automation)

シナリオの詳細を表示するには、シナリオのタイトルを選択して展開します。複数のタイトルを同時に展開できます。

アクセス拒否 — ステータスコード 403

説明

パイプラインの構築がAccessDenied「: アクセス拒否ステータスコード: 403」で失敗します。

原因

エラーの原因として以下が考えられます。

- インスタンスプロファイルには API やコンポーネントリソースにアクセスするのに必要な権限がありません。

- インスタンスプロファイルロールには、Amazon S3 へのロギングに必要な権限がありません。通常、これはインスタンスプロファイルロールに S3 バケットに対するPutObjectアクセス許可がない場合に発生します。

ソリューション

原因に応じて、この問題は次のように解決できます。

- インスタンスプロファイルに管理ポリシーがない — 不足しているポリシーをインスタンスプロファイルロールに追加してください。次に、パイプラインを再度実行します。
- インスタンスプロファイルに S3 バケットへの書き込みアクセス許可がない — S3 バケットへの書き込みPutObjectアクセス許可を付与するポリシーをインスタンスプロファイルロールに追加します。次に、パイプラインを再度実行します。

ビルドインスタンスで Systems Manager エージェントの可用性を確認中にビルドがタイムアウトする

説明

パイプラインビルドは「ステータス = 'TimedOut'」および「失敗メッセージ = 'ステップがターゲットインスタンスで Systems Manager Agent の可用性を検証している間にタイムアウトしました」で失敗します。

原因

エラーの原因として以下が考えられます。

- ビルド操作を実行し、コンポーネントを実行するために起動されたインスタンスは、Systems Manager エンドポイントにアクセスできませんでした。
- インスタンスプロファイルに必要な権限がありません。

ソリューション

考えられる原因に応じて、この問題は次のように解決できます。

- アクセスの問題、プライベートサブネット – プライベートサブネットで構築する場合は、Systems Manager、Image Builder、およびログ記録が必要な場合は Amazon S3/ のPrivateLink エンドポイントが設定されていることを確認してくださいCloudWatch。 PrivateLink エンドポイントの設定の詳細については、「VPC エンドポイントの概念 (AWS PrivateLink)」を参照してください。

- 権限がない — 以下の管理ポリシーを Image Builder の IAM サービスにリンクされたロールに追加します。
  - EC2InstanceProfileForImageBuilder
  - EC2InstanceProfileForImageBuilderECRContainerBuilds
  - AmazonSSMManagedInstanceCore

  Image Builderサービス連携ロールの詳細については、EC2 Image Builder サービスにリンクされたロールを使用するを参照してください。

Windows セカンダリディスクは起動時にオフラインです。

説明

Image Builder Windows AMI の構築に使用されるインスタンスタイプが AMI からの起動に使用されるインスタンスタイプと一致しない場合、起動時にルート以外のボリュームがオフラインになるという問題が発生する可能性があります。これは主に、ビルドインスタンスが起動インスタンスよりも新しいアーキテクチャを使用している場合に発生します。

次の例は、Image Builder AMI が EC2 Nitro インスタンスタイプで構築され、EC2 Xen インスタンスで起動された場合に何が起こるかを示しています。

ビルドインスタンスタイプ:m5.large (ニトロ)

起動インスタンスタイプ:t2.medium (Xen)

```
PS C:\Users\Administrator> get-disk
Number  Friendly Name  Serial Number         Health Status  Operational Status  Total
 Size  Partition Style
------  -------------  -------------         -------------  ------------------
 ----------  ---------------
0       AWS PVDISK     vol0abc12d34e567f8a9  Healthy        Online                  30
 GB  MBR
1       AWS PVDISK     vol1bcd23e45f678a9b0  Healthy        Offline                  8
 GB  MBR
```

原因

Windows のデフォルト設定により、新しく検出されたディスクは自動的にオンラインになり、フォーマットされません。EC2 でインスタンスタイプが変更されると、Windows はこれを新しいディスクが検出されたものとして扱います。これは、基礎となるドライバーが変更されたためです。

ソリューション

Windows AMI を構築するときには、起動元と同じインスタンスタイプのシステムを使用することをお勧めします。異なるシステムで構築されたインスタンスタイプをインフラストラクチャ設定に含めないでください。指定するインスタンスタイプに Nitro システムを使用しているものがあれば、それらはすべて Nitro システムを使用する必要があります。

Nitroシステム上に構築されたインスタンスの詳細については、Amazon EC2 User Guide for Windows InstancesのNitroシステム上に構築されたインスタンスを参照してください。

CIS で強化されたベースイメージではビルドが失敗します。

説明

CIS 強化ベースイメージを使用していて、ビルドが失敗する。

原因

/tmpディレクトリがnoexecとして分類されると、Image Builder が失敗する可能性があります。

ソリューション

イメージレシピのworkingDirectoryフィールドで、作業ディレクトリの別の場所を選択してください。詳細については、ImageRecipe「データ型の説明」を参照してください。

AssertInventoryCollection 失敗 (Systems Manager Automation)

説明

Systems Manager オートメーションでは、AssertInventoryCollectionオートメーションステップが失敗したと表示されます。

原因

あなたまたはあなたの組織は、EC2 インスタンスのインベントリ情報を収集する Systems Manager ステートマネージャーアソシエーションを作成したかもしれません。Image Builder パイプラインで拡張イメージメタデータ収集が有効になっている場合 (これがデフォルト)、Image Builder はビルドインスタンスの新しいインベントリ関連付けを作成しようとします。ただし、Systems Manager では、管理対象インスタンスに複数のインベントリを関連付けることはできません。また、関連付けがすでに存在する場合も、新しい関連付けは許可されません。これにより操作は失敗し、パイプラインの構築も失敗します。

ソリューション

この問題を解決するには、次のいずれかのメソッドを使って、「拡張イメージメタデータ収集」をオフにします。

- コンソールのイメージパイプラインを更新して、「拡張メタデータ収集を有効にする」チェックボックスをオフにします。変更を保存し、パイプラインビルドを実行します。

  EC2 Image Builderコンソールを使用したAMIイメージパイプラインの更新の詳細については、AMIイメージパイプラインの更新 (コンソール)を参照してください。EC2 Image Builder コンソールを使用してコンテナイメージパイプラインを更新する方法の詳細については、コンテナイメージパイプラインの更新 (コンソール)を参照してください。

- update-image-pipelineのコマンドを使用して AWS CLIでイメージパイプラインを更新することもできます。これを行うには、JSON ファイルに EnhancedImageMetadataEnabledプロパティを含めて、falseに設定します。以下の例では、プロパティがfalseに設定されている。

```
{
    "name": "MyWindows2019Pipeline",
    "description": "Builds Windows 2019 Images",
    "enhancedImageMetadataEnabled": false,
    "imageRecipeArn": "arn:aws:imagebuilder:us-west-2:123456789012:image-recipe/my-example-recipe/2020.12.03",
    "infrastructureConfigurationArn": "arn:aws:imagebuilder:us-west-2:123456789012:infrastructure-configuration/my-example-infrastructure-configuration",
    "distributionConfigurationArn": "arn:aws:imagebuilder:us-west-2:123456789012:distribution-configuration/my-example-distribution-configuration",
    "imageTestsConfiguration": {
        "imageTestsEnabled": true,
        "timeoutMinutes": 60
    },
    "schedule": {
        "scheduleExpression": "cron(0 0 * * SUN *)",
        "pipelineExecutionStartCondition":
 "EXPRESSION_MATCH_AND_DEPENDENCY_UPDATES_AVAILABLE"
    },
    "status": "ENABLED"
}
```

新しいパイプラインでこの現象が発生しないようにするには、EC2 Image Builderコンソールを使用して新しいパイプラインを作成するときにEnable enhanced metadata collectionチェックボックスをオフにするか、 AWS CLIを使用してパイプラインを作成するときにJSONファイルのEnhancedImageMetadataEnabledプロパティの値をfalseに設定します。

EC2 Image Builder ユーザーガイドのドキュメント履歴

以下の表は のドキュメントの重要な変更点をまとめたものです。このドキュメントの更新に関する通知を受け取るには、RSS フィードにサブスクライブできます。

- API バージョン: 2023-12-12

変更	説明	日付
STIG Q1 の更新	Linux STIG バージョンを更新し、2024 年第 1 四半期リリース用の STIGS を適用しました。Windows バージョンに変更はありませんでした。	2024 年 2 月 23 日
機能リリース: イメージワークフロー管理	イメージワークフローを使用すると、イメージ作成プロセスの柔軟性、可視性、制御性が向上します。ワークフローに合わせてビルドとテストのステップをカスタマイズすることも、Image Builder のデフォルトワークフローを使用することもできます。	2023 年 12 月 12 日
STIG Q4 の更新	Linux STIG バージョンを更新し、2023 年第 4 四半期リリース用の STIGS を適用しました。Windows バージョンに変更はありませんでした。新しいコンポーネント、ソフトウェア、ベンチマーク番号の Linux および Windows SCAP も更新されました。	2023 年 12 月 7 日
機能リリース: イメージライフサイクル管理	イメージライフサイクル管理のポリシーとルールを使用す	2023 年 11 月 17 日

	ると、古くなったイメージやそれに関連するリソースに対してタグ付けと削除のプロセスを確実に実施するためのリソース管理戦略を定義できます。	
STIG の Q3 アップデート	STIGのバージョンを更新し、2023年第3四半期リリース用のSTIGSを適用。メッセージを追加更新し、ごくわずかな例外を除いて、サードパーティ製パッケージは自動的にインストールされないことを明記しました。スキップされた STIG はすべてログに記録されます。	2023 年 10 月 5 日
STIG の新しいバージョン	STIGのバージョンを更新し、2023年第2四半期リリース用のSTIGSを適用。	2023 年 5 月 3 日
STIG の新しいバージョン	STIGのバージョンを更新し、2023年第1四半期リリース用のSTIGSを適用。とのサポートが追加されました。	2023 年 4 月 14 日
でサポートされているリージョンを更新する AWSTOE	アジアパシフィック (ハイデラバード)、 AWS リージョンアジアパシフィック (ジャカルタ)、欧州 (チューリッヒ)、欧州 (スペイン)、中東 (アラブ首長国連邦) AWSTOE のサポートが追加されました。	2023 年 4 月 13 日

AWSTOE アプリケーションダウンロードの更新	Windows で AWSTOE のインストールダウンロードの署名を更新しました。TLS も更新されました。S3 バケットからのアプリケーションのダウンロードには、TLS バージョン 1.2 以降が必要になりました。	2023 年 3 月 31 日
特徴量リリース:ビルドワークフローの強化	イメージビルドのバージョン詳細の新しいワークフロータブに、イメージビルドのランタイム詳細を追加しました。ビルドのトラブルシューティングに関する情報を改善しました。	2023 年 3 月 30 日
機能リリース:CVE 検出とレポート	Amazon Inspector のスキャンを有効にしているアカウントの場合、Image Builder は、Amazon ECR に保存されているコンテナイメージを含む新しいイメージのビルドプロセスのテストステージ中に Amazon Inspector から共通脆弱性識別子 (CVE) の検出結果をキャプチャできます。Image Builder は結果のスナップショットを作成して、詳細な分析をサポートします。Image Builder では、アカウント、パイプライン、またはイメージごとにフィルタリングできる結果の数もレポートされ、詳細を掘り下げることができます。	2023 年 3 月 30 日

バージョン履歴を追加	Windows と Linux のセクションにバージョン履歴を追加しました。	2023 年 2 月 17 日
STIG の新しいバージョン	STIGのバージョンを更新し、2022年第4四半期リリース用のSTIGSを適用。	2023 年 2 月 1 日
機能リリース： AWS Marketplace 統合と CIS 強化	CIS 強化イメージや Center for Internet Security の新しい CIS 強化コンポーネントなど、サブスクライブされたイメージを新しいカスタムイメージのベースラインとして簡単に検索して使用するための AWS Marketplace 統合が追加されました。	2023 年 1 月 13 日
CIS Fardening のコンポーネント	CIS が所有および管理する CIS 強化コンポーネントを追加しました。	2023 年 1 月 13 日
STIG の新しいバージョン	Ubuntu サポートを導入し、STIG バージョンを更新し、2022 年第 2 四半期のリリースに向けて STIGS を適用しました。	2022 年 7 月 20 日
ドキュメント更新:YAML コンポーネントの作成ドキュメントページのナビゲーション	Create YAML コンポーネントドキュメントの内容を独自のページに移動し、他のページもそれを参照するように更新しました。	2022 年 6 月 7 日
STIG の新しいバージョン	STIGのバージョンを更新し、2022年第1四半期リリース用のSTIGSを適用。	2022 年 4 月 25 日

ExecuteDocument アクションモジュールを追加	ExecuteDocument アクションモジュールのドキュメントを General execution の下に追加しました。	2022 年 3 月 28 日
特徴量リリース:Windows AMI の高速起動のSupport	Windows AMI の高速起動をサポートするためのディストリビューション設定が追加されました。	2022 年 2 月 21 日
メンテナンスリリース: AWSTOE バイナリサムプリントの更新	AWSTOE 署名者証明書のバイナリサムプリントを更新しました。	2022 年 2 月 18 日
機能リリース: の入力を設定する AWSTOE	AWSTOE run コマンドの入力として JSON 設定ファイルを使用するためのサポートが追加されました。	2022 年 2 月 3 日
STIG の新しいバージョン	STIGのバージョンを更新し、2021年第4四半期リリース用のSTIGSを適用。また、新しい SCAP コンプライアンスチェッカー (SCC) コンポーネントに関するセクションも追加されました。	2021 年 12 月 22 日
特徴量リリース:VM Import/Export (VMIE) 統合	すべてのチャネル (コンソール、API/CLI など) によるVMのインポート、およびAPI/CLIによるVMのエクスポートのサポートを追加しました。現在、Image Builder コンソールから VM をエクスポートすることはできません。	2021 年 12 月 20 日

機能リリース: AWS Organizations および OUs の AMI 共有	ディストリビューション設定を更新して、出力 AMIs AWS Organizations および OUs。	2021 年 11 月 24 日
ドキュメントの更新:コンポーネントのステージとフェーズを更新	Image Builder のコンポーネントステージのコンテンツ、およびそれらが AWSTOE コンポーネントフェーズとどのように相互作用するかを拡張しました。	2021 年 9 月 22 日
ドキュメントの更新: CloudTrail 統合コンテンツの追加	モニタリングの概要と CloudTrail 統合コンテンツを追加しました。	2021 年 9 月 17 日
STIG の新しいバージョン	STIGのバージョンを更新し、2021年第3四半期リリース用のSTIGSを適用。	2021 年 9 月 10 日
機能リリース: Amazon EventBridge 統合	Image Builder を関連する からのイベントに接続し AWS のサービス、 で定義されたルールに基づいてイベントを開始できる EventBridge サポートが追加されました EventBridge。	2021 年 8 月 18 日
ドキュメントの更新: AWSTOE ページの順序を変更する	わかりやすくするために AWSTOE ページを再配置しました。	2021 年 8 月 11 日

特徴量リリース:パラメータ化されたコンポーネントと追加のインスタンス設定	レシピのコンポーネントをカスタマイズするためのパラメータ指定のサポートが追加されました。イメージの構築とテストに使用する EC2 インスタンスの構成が拡張されました。これには、起動時に実行するコマンドを指定する機能や、Systems Manager エージェントのインストールと削除をより細かく制御できる機能が含まれます。	2021 年 7 月 7 日
STIG の新しいバージョン	STIGのバージョンを更新し、2021年第2四半期リリース用のSTIGSを適用。	2021 年 6 月 30 日
機能強化:タグ付けの強化	リソースのタグ付けに関するメッセージが改善されました。	2021 年 6 月 25 日
特徴量リリース:テンプレート統合を起動	ディストリビューション設定に AMI ディストリビューション用の Amazon EC2 起動テンプレートを使用するためのサポートが追加されました。	2021 年 4 月 7 日
特徴量リリース:コンテナビルドの強化	ブロックデバイスマッピングの設定と、コンテナの構築のベースイメージとして使用する AMI の指定のサポートが追加されました。	2021 年 4 月 7 日
STIG の新しいバージョン	STIGのバージョンを更新し、STIGSを適用。	2021 年 3 月 5 日

更cron 式	Image Builder の cron 処理が更新され、cron 表現の細分性が大幅に向上し、標準の cron スケジューリングエンジンが使用されるようになりました。例は新しいフォーマットで更新されました。	2021 年 2 月 8 日
特徴量リリース:コンテナサポート	Image Builder を使用した Docker コンテナイメージの作成と、生成されたイメージを Amazon Elastic Container Registry (Amazon ECR) に登録して保存する、というサポートが追加されました。コンテンツは、新しい機能を反映し、future 成長に対応するように再編成されました。	2020 年 12 月 17 日
cron ドキュメントの再構築	このページでは、cron が Image Builder パイプラインビルドでどのように機能するかについての詳細と、UTC 時間に関する詳細が含まれるようになりました。特定のフィールドで使用できないワイルドカードは削除されました。例には、コンソールと CLI の両方のエクスプレッションサンプルが含まれるようになりました。	2020 年 11 月 13 日

コンソールバージョン 2.0: パイプライン編集が更新されました。	「はじめに」と「パイプラインの作成」チュートリアルの内容が変更され、「イメージパイプラインの管理」ページも変更され、コンソールの新しい機能とフローが組み込まれました。	2020 年 11 月 13 日
STIG の新しいバージョン	STIGのバージョンを更新し、STIGSを適用。注-デフォルトで適用される STIG を表示するようにリスト形式が変更されました。	2020 年 10 月 15 日
でのループコンストラクトのサポート AWSTOE	AWSTOE のアプリケーションで、繰り返される命令シーケンスを定義するループ構文を作成します。	2020 年 7 月 29 日
AWSTOE コンポーネントのローカル開発のサポート	AWSTOE アプリケーションを使用してイメージコンポーネントをローカルで開発およびテストします。	2020 年 7 月 28 日
暗号化された AMI	EC2 Image Builder は、暗号化された AMI ディストリビューションのサポートを追加します。	2020 年 7 月 1 日
AutoScaling 非推奨	インスタンスを起動 AutoScaling するための の使用の廃止。	2023 年 6 月 15 日

による接続のサポート AWS PrivateLink	VPCとEC2 Image Builderの間にプライベート接続を確立するには、インターフェイス VPC エンドポイントを作成します。インターフェイスエンドポイントは、インターネットゲートウェイ AWS PrivateLink、NAT デバイス、VPN 接続、 AWS Direct Connect 接続のいずれも必要とせずに、Image Builder APIs にプライベートにアクセスできるテクノロジーである を利用しています。VPC のインスタンスは、パブリック IP アドレスがなくても API と通信できます。VPCとImage Builder間のトラフィックは、Amazonのネットワークから出ることはありません。	2020 年 6 月 10 日
STIG の新しいバージョン	STIGのバージョンを更新し、STIGSを適用。	2020 年 1 月 23 日
トラブルシューティング	一般的なトラブルシューティングのシナリオ。	2020 年 1 月 22 日
STIG コンポーネント	STIG コンポーネントを使用して AWSTOE STIG 準拠の画像を作成できます。	2020 年 1 月 22 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。