aws

ユーザーガイド

EC2 イメージビルダー

EC2 イメージビルダー: ユーザーガイド

Table of Contents

Image Builder とは

EC2 Image Builder はフルマネージド AWS のサービス 型で、カスタマイズされ、安全で、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、 AWS Command Line Interface、または APIs を使用して、 に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。

ステップ 1: パイプラインの詳細を指定する

- パイプラインに名前を付け、タグを追加します。
- メタデータと脆弱性スキャンの設定を定義します。
- パイプラインのスケジュールを設定します。

ステップ 2: イメージをカスタマイズする

既存のレシピを選択するか、新しいレシピを作成できます。

- カスタマイズ用のベースイメージを選択します。
- 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。
- 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。
- 実行するテストコンポーネントを選択します。

ステップ 3: ワークフローを定義する

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 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。

ステップ 4: ビルドインフラストラクチャを設定する

- Image Builder がイメージ作成プロセス中に起動するインスタンスの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。
- 起動時に適用できる 1 つ以上のインスタンスタイプを選択します。
- Image Builder からの通知を受信する Amazon Simple Notification Service (SNS) トピックを選択します。
- イメージ作成プロセスに使用する VPC、サブネット、セキュリティグループを指定します。
- トラブルシューティング設定を選択します。Image Builder のログの書き込み先や、失敗時にビルドインスタンスを終了するか (デフォルト)、トラブルシューティングを続けるために実行を継続するかなどを選択できます。

ステップ 5: イメージディストリビューションを定義する

- Image Builder が Amazon マシンイメージ (AMI) またはコンテナイメージを配布 AWS リージョンする場所を選択します。
- Image Builder パイプラインが AMI を作成する場合は、次の設定もサポートされます。
  - 暗号化に使用する KMS キーを選択する。
  - AWS アカウント と Organizations 間で AMI 共有を設定します。
  - License Manager のセルフマネージドライセンスを配布イメージに関連付ける。
  - イメージの起動テンプレートを設定する。

Image Builder の機能

EC2 Image Builder は以下の機能を提供する：

コンプライアンスに準拠した最新のイメージを構築するための生産性の向上と運用の軽減

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、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、「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。

一元管理と系統追跡

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、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、「Image Builder リソースを と共有する AWS RAM」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
macOS	12.x (モントレー) 、13.x (ベンチュラ) 、14.x (ソノマ) 、15.x (セコイア)
Red Hat Enterprise Linux (RHEL)	7、8、9、10
SUSE Linux Enterprise Server (SUSE)	12 と 15
Ubuntu	18.04 LTS、20.04 LTS、22.04 LTS、24.04 LTS
Windows Server	2012 R2、2016、2019、2022、2025

対応イメージフォーマット

Amazon マシンイメージ (AMI) を作成するカスタムイメージでは、既存の AMI を開始点として選択できます。Docker コンテナイメージの場合は、DockerHub でホストされているパブリックイメージ、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。

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。

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レシピまたはコンテナレシピ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 イメージレシピは、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。

基本の 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、「Image Builder が AWS Task Orchestrator and Executor アプリケーションを使用してコンポーネントを管理する方法」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テスト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 AWS のサービス 、Image Builder レシピ設定に応じて、他の を使用してイメージを構築します。カスタムイメージへの製品とサービスの統合の詳細については、「Image Builder での製品とサービスの統合」を参照してください。

EC2 Image Builder の仕組み

EC2 Image Builder コンソールを使用してカスタムイメージパイプラインを作成すると、システムは次のステップをガイドします。

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。例えば、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

- コンポーネント管理
- 作成されたリソース
- ディストリビューション
- リソースの共有
- コンプライアンス

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。

コンポーネント管理

EC2 Image Builder は、複雑なワークフローの調整、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を使用した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[を使用してカスタムコンポーネントを開発するための手動セットアップ 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[コンポーネントを使用した Image Builder イメージのカスタマイズ]。 AWSTOE を使ったコンポーネントの作成については、[Image Builder が AWS Task Orchestrator and Executor アプリケーションを使用してコンポーネントを管理する方法]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、最終イメージを作成する前に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。

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
- 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 起動アクセス許可を定義して、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 を制御することができます。例えば、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、承認された準拠 AMIs でのみインスタンスを起動するための制限をメンバーアカウントに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、「コンソールからの新しいイメージレシピバージョンの作成」または「コンソールで新しいコンテナレシピの作成」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部と共有するには AWS Organizations、「」を参照してくださいImage Builder リソースを と共有する AWS RAM。

コンプライアンス

Center for Internet Security (CIS) ベンチマークに対して、EC2 Image Builder は 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どうかの評価を実行します。例えば、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。詳細については、「Center for Internet Security (CIS) ベンチマーク」を参照してください。

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。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、「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。

Image Builder での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>*.*<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. バージョン ARN には、<major>、<minor>、<patch> の最初の 3 つのノードしかありません。
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 つのノードでは、ノードごとに 2^30-1 または 1073741823 の上限を持つ任意の正の整数値またはゼロを割り当てることができます。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。

例えば、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、エラーになる
- `1.x.8` は無効で、エラーになる

Image Builder を使用してカスタムイメージをビルドするためのセットアップ

EC2 Image Builder でイメージをビルドするには、パイプラインを作成するための以下の前提条件が満たされていること確認してください。特に明記されていない限り、これらの前提条件はすべてのタイプのパイプラインに適用されます。

前提条件

- Image Builder サービスリンクロール
- 設定の要件
- コンテナイメージパイプラインのコンテナリポジトリ
- macOS イメージ用の専有ホスト
- IAM の前提条件
- Systems Manager エージェントの前提条件

前提条件を満たしたら、次のインターフェイスのいずれからでも EC2 Image Builder を管理できます。

- EC2 Image Builder コンソール
- の Image Builder コマンド AWS CLI
- EC2 Image Builder API リファレンス
- AWS SDKsとツール

Image Builder サービスリンクロール

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。作成するサービスリンクロールの詳細については、「Image Builder での IAM サービスリンクロールの使用」を参照してください。

設定の要件

- Image Builder は、AWS PrivateLink をサポートしています。Image Builder の VPC エンドポイントの設定の詳細については、「Image Builder と AWS PrivateLink インターフェイス VPC エンドポイント」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。

  macOS

  TBD - 前提条件について何かが必要です

コンテナイメージパイプラインのコンテナリポジトリ

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。

macOS イメージ用の専有ホスト

Amazon EC2 Mac インスタンスには、メタルインスタンスタイプで実行される専有ホストが必要です。カスタム macOS イメージを作成する前に、アカウントに専有ホストを割り当てる必要があります。Mac インスタンスの詳細と、macOS オペレーティングシステムをネイティブにサポートするインスタンスタイプの一覧については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 Mac インスタンス」を参照してください。

専有ホストを作成したら、イメージのインフラストラクチャ設定リソースの設定を行うことができます。インフラストラクチャ設定には、イメージからのインスタンスが起動する先のホスト、ホストプレイスメントグループ、またはアベイラビリティーゾーンを指定できるプレイスメントプロパティが含まれます。

IAM の前提条件

インスタンスプロファイルに関連付ける IAM ロールには、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:

JSON

```
{
"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ブレット記号を選択し、[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) を選択してください。
7. EC2InstanceProfileForImageBuilderECRContainerBuilds および AmazonSSMManagedInstanceCore ポリシーについて、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、「Image Builder リソースを と共有する AWS RAM」を参照してください。

Systems Manager エージェントの前提条件

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。

この情報の収集をオプトアウトするには、ご希望の環境に合った方法を選択してください。

- Image Builder コンソール —[拡張メタデータ収集を有効にする] チェックボックスの選択を解除します。
- AWS CLI – `--no-enhanced-image-metadata-enabled` オプションを指定します。
- Image Builder API または SDK — `enhancedImageMetadataEnabled` パラメータを `false` に設定します。

Image Builder は、`RunCommand` でイメージのビルドとテストのワークフローの一部として、ビルドインスタンスとテストインスタンスにアクションを送信します。`RunCommand` でビルドインスタンスとテストインスタンスへのアクション送信にの使用をオプトアウトすることはできません。

Image Builder チュートリアルを使用してカスタムイメージを作成する方法について説明します。

EC2 Image Builder を使用してカスタムイメージとコンポーネントをビルドするには、さまざまな方法があります。チュートリアルは、Image Builder の主要な概念を理解するために役立ちます。各チュートリアルでは、ユースケースを提示し、最初に従うことができるステップを説明します。全体的なプロセスを理解しやすくするために、手順では、可能であればデフォルト値を使用します。チュートリアルのいずれかを完了したら、その他の方法を調べて独自のイメージをカスタマイズできます。

最初のイメージのビルド

以下のチュートリアルでは、Image Builder コンソールウィザードを使用して最初のイメージをビルドする方法を説明します。チュートリアルを完了すると、以下の Image Builder リソースのセットが作成されます。チュートリアルの最後のステップは、作成したリソースをクリーンアップすることです。

- Amazon マシンイメージ (AMI) のイメージレシピまたはコンテナイメージのコンテナレシピ。
- デフォルト設定のインフラストラクチャ設定リソース。
- 出力をソースリージョン (コンソールウィザードの実行に使用したアカウントのリージョン) に配布する、デフォルト設定のディストリビューション設定リソース。
- リストされたリソースを使用して、デフォルトのイメージビルドワークフローで出力イメージをビルドするイメージパイプライン。
- 出力 AMI またはコンテナイメージ。

コンソールウィザードのチュートリアル

- パイプラインウィザード: AMI の作成
- パイプラインウィザード: コンテナイメージの作成

入力パラメータを持つカスタムコンポーネントの作成

以下のチュートリアルでは、入力パラメータを定義するカスタムコンポーネントを作成し、Image Builder レシピから値を設定する方法を示します。

パラメータを持つカスタムコンポーネント

Image Builder で Systems Manager パラメータを使用する

次のチュートリアルでは、Parameter Store AWS Systems Manager パラメータを作成し、イメージレシピで使用する方法を示します。

レシピでベースイメージパラメータを使用する

AMI ディストリビューション設定で Parameter Store パラメータを使用して、出力イメージ ID とカスタムコンポーネントを保存することもできます。詳細については、ディストリビューションAMI ディストリビューション設定の作成と更新については「」、カスタムコンポーネントSystems Manager パラメータストアパラメータを使用するについては「」を参照してください。

チュートリアル: Image Builder コンソールウィザードから AMI を出力するイメージパイプラインを作成する

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する

- 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。

> **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、Image Builder での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. [全般] 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 によって管理される Linux AMIs のリストが作成されます。このチュートリアルでは、`Amazon Linux 2 x86`イメージを選択します。

   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するようになります。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、Image Builder での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、これは、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。これを使用する場合は、システムマネージャーエージェントをベースイメージにあらかじめインストールするか、ユーザーデータにインストールを含めるようにしてください。

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パネルで、ビルドコンポーネントを追加を選択し、`Amazon managed`コンポーネント所有者フィルターリストから を選択します。これにより、コンソールインターフェイスの右側に選択パネルが開き、使用可能なコンポーネントを参照およびフィルタリングできます。

   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. バージョニングオプションのデフォルトのままにします (最新バージョンを使用できます)。

> **Note**
>
> この設定により、パイプラインは選択したコンポーネントに対して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します。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、Image Builder でのセマンティックバージョニングを参照してください。

d. Add to recipe を選択して、コンポーネントをレシピに追加します。これにより、コンポーネント選択パネルが閉じます。

e. ビルドコンポーネントパネルに戻ると、追加したコンポーネントが表示されます。

7. 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、ドラッグアンドドロップ操作を使用して、ビルドプロセス中に実行すべき順序にコンポーネントを再配置できます。

> **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。

a. 前のステップを繰り返して、update-linux-kernel-5コンポーネントをレシピに追加します。

b. 追加したコンポーネントには、カーネルバージョンの入力パラメータがあります。バージョニング管理オプション または [入力パラメータ] 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。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、「チュートリアル: 入力パラメータを持つカスタムコンポーネントを作成する」を参照してください。

c. 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。

d. update-linux-kernel-5 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。

こ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、選択したupdate-linuxコンポーネントのみを残します。

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、イメージを設定するために使用する EC2 ビルドとテストインスタンスの 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構成設定の詳細については、EC2 Image Builder API Reference の CreateInfrastructureConfiguration 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、既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、暗号化の特定のリージョン設定、起動許可 AWS アカウント、出力 AMI を起動できる組織、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構成の詳細については、「Image Builder の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。例えば、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。例えば、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する

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。

レシピを削除する

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する

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。

イメージを削除する

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。

チュートリアル: Image Builder コンソールウィザードから Docker コンテナイメージを出力するイメージパイプラインを作成する

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する

- 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 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、Image Builder での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. [全般] 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するようになります。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、Image Builder での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、パイプラインは選択したコンポーネントに対して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します。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、Image Builder での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、「チュートリアル: 入力パラメータを持つカスタムコンポーネントを作成する」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、ドラッグアンドドロップ操作を使用して、ビルドプロセス中に実行すべき順序にコンポーネントを再配置できます。

> **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 は次のコンテキスト変数を Dockerfile で使用します。

   parentImage (必須)

   この変数は、ビルド時にレシピのベースイメージに解決されます。

   例:

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

   environments (components 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 }}}
   ```

components (オプション)

Image Builder は、コンテナレシピに含まれているコンポーネントのビルドおよび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 内で environments 変数の後の任意の場所に配置できます。

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、既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構成設定の詳細については、EC2 Image Builder API Reference の CreateInfrastructureConfiguration 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。例えば、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。例えば、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する

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する

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する

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。

イメージを削除する

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。

チュートリアル: 入力パラメータを持つカスタムコンポーネントを作成する

コンポーネントパラメータの作成と設定を含む Image Builder コンポーネントは、EC2 Image Builder コンソールから直接、 から、 AWS CLIまたは Image Builder API または SDKs から管理できます。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、および実行時の 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

コンポーネントをビルドするには、YAML または JSON アプリケーションコンポーネントドキュメントを提供する必要があります。このドキュメントには、フェーズとステップの間に実行される、イメージをカスタマイズするために定義したコードが含まれています。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する

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する

ファイル編集ツールを使用して、コンポーネントドキュメントファイルを作成します。ドキュメントの例では、次のコンテンツを含む *hello-world-test.yaml* という名前のファイルを使用します。

```
# 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する

ファイル編集ツールを使用して、コンポーネントドキュメントファイルを作成します。ドキュメントの例では、次のコンテンツを含む *hello-world-test.yaml* という名前のファイルを使用します。

```
# 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。

macOS

YAML コンポーネントドキュメントを作成する

ファイル編集ツールを使用して、コンポーネントドキュメントファイルを作成します。ドキュメントの例では、次のコンテンツを含む *hello-world-test.yaml* という名前のファイルを使用します。

```
# Document Start
#
name: "HelloWorldTestingDocument-macOS"
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。

AWSTOE 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、「AWSTOEでのドキュメントの使用」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、「コンソールでのカスタムモデルコンポーネントの作成」を参照してください。
- コマンドラインから Image Builder の create-component コマンドを使用してコンポーネントを作成するには、「からカスタムコンポーネントを作成する AWS CLI」を参照してください。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。

コンソールでの Image Builder レシピのコンポーネントパラメータの設定

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には、以下のパラメータ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する場合、入力パラメータにデフォルト値が定義されていると、そのデフォルト値が [値] ボックスにグレーアウトされたテキストで表示されます。他の値を入力しない場合、Image Builder はデフォルト値を使用します。

レシピでベースイメージパラメータを使用する

イメージのカスタマイズのレシピを作成するときは、開始するベースイメージを識別する方法がいくつかあります。ベースイメージに Amazon マシンイメージ (AMI) ID を指定し、そのベースイメージが更新されると、その AMI ID が変更される可能性があります。それに合わせてレシピを更新する必要があります。

ベースイメージ ID が変更されるたびにレシピを変更する代わりに、 AWS Systems Manager Parameter Store パラメータ (SSM パラメータ) を定義してベースイメージ AMI ID の値を保存し、パラメータを使用してレシピにベースイメージを指定できます。 AWS マネージド AMIs では、最新バージョンのパブリックパラメータを使用できます。

このチュートリアルでは、AMI ID パラメータを作成し、イメージレシピで使用するプロセスについて説明します。このチュートリアルの Image Builder のステップはコンソールベースです。

内容

ステップ 1: Parameter Store パラメータを検索または作成する

このステップのプロセスは、ベースイメージに指定する AMI のタイプによって異なります。 AWS マネージド AMIs では、現在のバージョンを参照するパブリックパラメータを使用できます。一部のパラメータは、一部の で使用できない場合があります AWS リージョン。

開始するには、AMI に対応するタブを開きます。

AWS managed AMI

ベースイメージが AWS マネージド AMI の場合は、独自のパラメータを作成するのではなく、パブリックパラメータを使用して AMI ID を指定できます。AMI のパブリックパラメータを確認するには、AWS Systems Manager 「 ユーザーガイド」の「パブリックパラメータの検出」を参照してください。

Custom AMI

AMI ID パラメータを作成するには、コンソール AWS CLIまたは PowerShell を使用して Systems Manager でパラメータストアパラメータを作成する手順に従ってください。パラメータ値が AMI ID であることを確認するには、次の値を指定します。

パラメータ階層: `Standard`

タイプ: `String`

データ型: を選択しますaws:ec2:image。このタイプを指定すると、システムは入力された値を検証して、AMI ID であることを確認します。

値: 有効な AMI ID を入力します (例: ***ami-1234567890abcdef1***)。

ステップ 2: IAM アクセス許可を設定する

Systems Manager パラメータストアパラメータ (SSM パラメータ) を使用するには、パブリックかプライベートかにかかわらず、Image Builder 実行ロールで次の Systems Manager パラメータストアアクションを指定し、 パラメータをリソースとしてリストする必要があります。

- `ssm:GetParameter` – このアクションでは、SSM パラメータを使用してレシピのベースイメージを指定できます。
- `ssm:PutParameter` – このアクションにより、ディストリビューション中に出力 AMI ID を SSM パラメータに保存できます。ポリシー定義は同じように見えますが、このチュートリアルでは、ポリシー例に put アクションは含まれません。

カスタムコンポーネントで SSM パラメータを使用するには、代わりにインスタンスプロファイルロール`ssm:GetParameter`で を指定する必要があります。詳細については、「Systems Manager パラメータストアパラメータを使用する」を参照してください。

パイプラインを作成する場合、または で create-image コマンドを使用する場合は AWS CLI、Image Builder 実行ロールを 1 つだけ指定できます。Image Builder ワークフロー実行ロールを定義している場合は、そのロールに パラメータのアクセス許可を追加します。それ以外の場合は、SSM パラメータに必要なアクセス許可を含む新しいカスタムロールを作成します。

1. カスタムロールを作成する (オプション)

   Image Builder のアクセス許可用にカスタムロールが既に定義されている場合は、このステップをスキップできます。

   AWS Identity and Access Management 「 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」のプロセスに従います。
2. カスタムロールにアクセス許可を追加する

   カスタムロールに SSM パラメータのアクセス許可を追加するには、AWS Identity and Access Management 「 ユーザーガイド」の「ロールのアクセス許可ポリシーの更新」に従ってください。

   次のポリシー例は、アカウントで作成された パラメータを持つ `ssm:GetParameter`アクションを示しています。

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": "ssm:GetParameter",
   "Resource": "arn:aws:ssm:*:111122223333:parameter/ImageBuilder-*"
  }
 ]
}
```

パブリックパラメータリソースの詳細については、「 AWS Systems Manager ユーザーガイド」の「AMI パブリックパラメータの呼び出し」を参照してください。

ステップ 3: パラメータを使用するイメージレシピを作成する

1. https://console.aws.amazon.com/imagebuilder/ で、EC2 Image Builder コンソールを開きます。
2. Image recipes を選択し、リストページから Create image recipe を選択します。
3. ベースイメージセクションに次のように入力します。
   a. カスタム AMI の使用 オプションを選択します。これにより、AMI ID または AMI ID を含む SSM パラメータを入力できる追加のフィールドが表示されます。
   b. SSM パラメータオプションを選択します。
   c. SSM パラメータフィールドに、ステップ 1 で作成したパラメータのパラメータ名または Amazon リソースネーム (ARN) を入力します。名前を入力すると、コンソールに プレフィックスはありません。
4. 必要に応じて残りのレシピ設定を完了します。

> **Note**
>
> などの他のインターフェイスを介して親イメージを設定する場合 AWS CLI、パラメータ名のプレフィックスは である必要があります `ssm:` (例: `ssm:/ImageBuilder-Tutorial/BaseAMI`。

コンポーネントを使用した Image Builder イメージのカスタマイズ

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調整します。 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り AWSTOE 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。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、コンポーネントドキュメントから以下のフェーズが実行されます。

   - ビルド
   - 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。

   イメージビルドのテストステージでは、レシピに含まれている各コンポーネントに対して、コンポーネントドキュメントから次の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 アプリケーションフレームワークでは、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。イメージのビルドステージ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、`build` と `validate` の少なくとも 1 つのフェーズを定義する必要があります。イメージのテストステージ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` フェーズを定義し、他のフェーズは定義しないようにする必要があります。
>
> 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、「チュートリアル: 入力パラメータを持つカスタムコンポーネントを作成する」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、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するための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、「カスタム AWSTOE コンポーネントのコンポーネントドキュメントフレームワークを使用する」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 コマンドを使用したリソースのタグ付けの詳細については 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。

トピック

- コンポーネントの詳細を一覧表示して表示する
- AWS Marketplace コンポーネントを使用してイメージをカスタマイズする
- マネージドコンポーネントを使用した Image Builder イメージのカスタマイズ
- Image Builder イメージ用のカスタムコンポーネントの開発
- Image Builder が AWS Task Orchestrator and Executor アプリケーションを使用してコンポーネントを管理する方法

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

- Image Builder コンポーネントの一覧表示
- からコンポーネントビルドバージョンを一覧表示する AWS CLI
- からコンポーネントの詳細を取得する AWS CLI
- からコンポーネントポリシーの詳細を取得する AWS CLI

Image Builder コンポーネントの一覧表示

次のいずれかの方法を使用して、Image Builder コンポーネントの一覧を表示して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。

   - AWS Marketplace – AWS Marketplace 製品サブスクリプションに直接関連付けられているコンポーネント。
   - 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 – 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

以下の例は、list-components コマンドを使用して、アカウントが所有する Image Builder コンポーネントの一覧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

- AWSMarketplace
- Amazon
- Self
- Shared
- ThirdParty

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
```

からコンポーネントビルドバージョンを一覧表示する AWS CLI

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、Image Builder での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"
            }
        }
    ]
```

```
}
```

からコンポーネントの詳細を取得する 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": {}
    }
}
```

からコンポーネントポリシーの詳細を取得する 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
```

AWS Marketplace コンポーネントを使用してイメージをカスタマイズする

は、独立系ソフトウェアベンダー (ISVs) によって作成された多数のイメージに加えて、独自の Image Builder イメージをカスタマイズするために使用できるコンポーネント AWS Marketplace を提供します。これらの AWS Marketplace コンポーネントをイメージレシピで使用して新しいイメージを構築するには、事前にサブスクライブする必要があります。

イメージレシピで AWS Marketplace コンポーネントを指定すると、Image Builder はサブスクリプションを検証し、依存関係チェックを実行して、それを使用するために必要なリソースがあることを確認します。検証が成功すると、Image Builder は、イメージパイプラインビルドで使用するコンポーネントとそのアーティファクトの安全なダウンロードを作成します。

AWS Marketplace コンポーネントの検出

レシピで使用する AWS Marketplace ソフトウェアコンポーネントは、Image Builder コンソールの「製品の検出」ページから次のように検出できます。

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で、 AWS Marketplaceセクションの製品の検出を選択します。
3. [Components] (コンポーネント) タブを選択します。このタブには、 AWS Marketplace 関連するコンポーネントを含む配信オプションを使用するすべての製品が一覧表示されます AWS Marketplace。
4. コンポーネントを含む特定のソフトウェア製品を検索するには、検索バーに名前の一部を入力するか、`Status`、、`Operating System`、`Publisher`、または でフィルタリングしますCategories。検索バーには、結果のページ分割コントロールも含まれています。

結果

各 AWS Marketplace 製品には、以下の情報を含む独自の詳細パネルがあります。

AWS Marketplace 製品名と口ゴ

ソフトウェア製品名は、 の製品詳細にリンクされています AWS Marketplace。リンクを選択すると、 の製品の詳細を確認できます AWS Marketplace。または、すでに調査を行っている場合は、サブスクリプションオプションの概要を表示し、「サブスクリプションオプションの表示」ボタンを使用して検索結果から直接サブスクライブすることもできます。

バージョン

これには、コンポーネントのプライマリバージョンが含まれます。

オペレーティングシステム

コンポーネントを実行するように設計されたオペレーティングシステム。

パブリッシャー

コンポーネントのパブリッシャー。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。

Categories

コンポーネントに適用される 1 つ以上の AWS Marketplace 製品カテゴリ。

ステータス

この製品をサブスクライブしているかどうかを示します。サブスクライブしていない場合は、サブスクリプションオプションを表示を選択して AWS Marketplace 製品のサブスクリプションオプションの概要を確認し、オプションで Image Builder コンソールから直接サブスクライブできます。

関連付けられたコンポーネント

AWS Marketplace 製品にサブスクリプションに含まれているバージョンが 1 つ以上ある場合は、関連コンポーネントセクションに表示されます。セクションは最初に折りたたまれ、関連するコンポーネントの数が表示されます。セクションを展開すると、詳細を確認できます。

> **Note**
>
> AWS Marketplace イメージ製品に関連付けられている Center for Internet Security (CIS) コンポーネントは、Discover 製品の結果に表示されません。イメージ製品をサブスクライブすると、関連するコンポーネントがサブスクリプションページに表示され、イメージレシピの作成ダイアログにサードパーティーコンポーネントとして表示されます。コンポーネントの詳細については、「」を参照してくださいCIS Fardening のコンポーネント。

AWS Marketplace コンポーネントをサブスクライブする

レシピで使用するコンポーネントを含む AWS Marketplace 製品を見つけたら、次のように Image Builder コンソールから直接サブスクライブするか、 AWS Marketplace コンソールからサブスクライブできます。

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で、 AWS Marketplaceセクションの製品の検出を選択します。
3. [Components] (コンポーネント) タブを選択します。このタブには、 AWS Marketplace 関連するコンポーネントを含む配信オプションを使用するすべての製品が一覧表示されます AWS Marketplace。
4. 特定の AWS Marketplace 製品を検索するには、検索バーに名前の一部を入力します。パブリッシャーがわかっていても、正確な製品名やスペルがわからない場合は、 でフィルタリングPublisherして、パブリッシャーが利用可能な製品のリストを取得することもできます。
5. 結果リストからサブスクライブする製品を選択し、サブスクリプションオプションの表示を選択します。これは、 AWS Marketplace 製品のサブスクリプションオプションの概要を示しています。
6. Image Builder コンソールを離れずに製品をサブスクライブするには、サブスクライブを選択します。サブスクリプションが処理されていることが通知されます。サブスクライブすると、ステータスは に更新されますSubscribed。

現在サブスクライブしている AWS Marketplace 製品の詳細については、「」で説明されているコンソールの手順を参照してくださいAWS Marketplace Image Builder のサブスクリプション。

Image Builder イメージレシピで AWS Marketplace コンポーネントを使用する

Image Builder イメージレシピの AWS Marketplace コンポーネントは、他のタイプのコンポーネントを使用するのと同じ方法で使用できます。 AWS Marketplace イメージ製品に関連付けられているほとんどのコンポーネントでは、所有権力テゴリは ですAWS Marketplace。たとえば、サブスクライブした AWS Marketplace 製品のビルドコンポーネントを使用するには、ビルドコンポーネントを追加を選択し、AWS Marketplaceリストから を選択します。これにより、コンポーネントを一覧表示 AWS Marketplace する選択パネルがコンソールインターフェイスの右側に表示されます。

> **Note**
>
> CIS 強化コンポーネントをお探しの場合は、 ではなくThird party managed所有権リストから を選択しますAWS Marketplace。

コンポーネントのパラメータを選択、配置、設定する方法の詳細については、「」を参照してくださいイメージレシピの新しいバージョンを作成する。

マネージドコンポーネントを使用した Image Builder イメージのカスタマイズ

マネージドコンポーネントは AWS、例えば Center for Internet Security (CIS) などのサードパーティー組織と提携して作成されます。イメージまたはコンテナレシピでマネージドコンポーネントを使用する場合、Amazon はパッチやその他の更新が適用された最新のコンポーネントバージョンを提供します。コンポーネントのリストを取得したり、コンポーネント情報を取得したりするには、「」を参照してくださいコンポーネントの詳細を一覧表示して表示する。

次の注目の AWS マネージドコンポーネントのリストには、 を通じて CIS 強化 AMIs をサブスクライブするときに使用できるコンポーネントが含まれています AWS Marketplace。

主なコンポーネント

- Distributor パッケージ管理コンポーネントによる Image Builder Windows イメージのアプリケーションのインストール
- CIS Fardening のコンポーネント
- Image Builder 用の Amazon managed STIG 強化コンポーネント

Distributor パッケージ管理コンポーネントによる Image Builder Windows イメージのアプリケーションのインストール

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 AWS Systems Manager 、Distributor 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` を設定する」を参照してください。
- `aws-vss-components-windows` コンポーネントは AWS Systems Manager Distributor 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` を設定する」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する (イメージレシピ用) または 新しいコンテナ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 ユーザーガイド」の「アプリケーション整合性のある VSS スナップショットの作成」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。例えば、`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。

   JSON

```
{
 "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] (ポリシーの作成) 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選択します。(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] の横の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. [Next: Tags] (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) を選択してください。

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** を設定する

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 はデフォルトで推奨バージョンに設定されます。
- AdditionalArbutor (オプション) — パッケージのインストール、アンインストール、または更新を行うスクリプトに指定する追加パラメータを含む 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** を設定する

`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 とサードパーティーは、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。

で使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一覧表示の詳細については AWS CLI、AWS Systems Manager 「 ユーザーガイド」の「パッケージの表示 (コマンドライン) 」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、スクリプトを実行して設定に CIS ベンチマークレベル 1 のガイドラインを適用する、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。

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ーのパッケージをインストールしません。サードパーティーのパッケージがインスタンスに既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、「コンポーネントを使用した Image Builder イメージのカスタマイズ」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

- Windows 用 STIG バージョン履歴ログ
- Linux の STIG 強化コンポーネント
- Linux 用 STIG バージョン履歴ログ
- SCAP コンプライアンス検証ツール (コンポーネント)

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 準拠のセキュリティ強化コンポーネントにより、国防総省 (DoD) からの InstallRoot が Windows インフラストラクチャにインストールされます。これにより、DoD 証明書はダウンロード、インストール、および更新されます。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 2025.2.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 2 リリース 4

  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 3 リリース 4

  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 10

  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-225250, V-225318, V-225319, V-225324, V-225327, V-225328, V-225330, V-225331, V-225332, V-225333, V-225334, V-225335, V-225336, V-225342, V-225343, V-225355, V-225357, V-225358, V-225359, V-225360, V-225362, V-225363, V-225376, V-225392, V-225394, V-225412, V-225459, V-225460, V-225462, V-225468, V-225473, V-225476, V-225479, V-225480, V-225481, V-225482, V-225483, V-225484, V-225485, V-225487, V-225488, V-225489, V-225490, V-225511, V-225514, V-225525, V-225526, V-225536 V-2255373
- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 6

  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 Firewall STIG バージョン 2 リリース 2

  V-241994、V-241995、V-241996、V-241999、V-242000、V-242001、V-242006、V-242007、V-242008
- Internet Explorer 11 STIG バージョン 2 リリース 5

  V-223016, V-223056、および V-223078
- Microsoft Edge STIG バージョン 2 リリース 2 (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 2025.2.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 2 リリース 4

  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-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-254384、 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 3 リリース 4

  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-205842、
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、
V-236001、 および V-257503

- Windows Server 2016 STIG バージョン 2 リリース 10

  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-225059、 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、 V-236000、 および V-257502

- 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-225239、 V-225259、 V-225260、 V-225261、 V-225263、 V-225264、 V-225265、 V-225266、 V-225267、 V-225268、 V-225269、 V-225270、 V-225271、 V-225272、 V-225273、 V-225275、 V-225276、 V-225277、 V-225278、 V-225279、 V-225280、 V-225281、 V-225282、 V-225283、 V-225284、 V-225285、 V-225286、 V-225287、 V-225288、 V-225289、 V-225290、 V-225291、 V-225292、 V-225293、 V-225294、 V-225295、 V-225296、 V-225297、 V-225298、 V-225299、 V-225300、 V-225301、 V-225302、 V-225303、 V-225304、 V-225305、 V-225314、 V-225315、 V-225316、 V-225317、 V-225325、 V-225326、 V-225329、 V-225337、 V-225338、 V-225339、 V-225340、 V-225341、 V-225344、 V-225345、 V-225346、 V-225347、 V-225348、 V-225349、 V-225350、 V-225351、 V-225352、 V-225353、 V-225356、 V-225367、 V-225368、 V-225369、 V-225370、 V-225371、 V-225372、 V-225373、 V-225374、 V-225375、 V-225377、 V-225378、 V-225379、 V-225380、 V-225381、 V-225382、 V-225383、 V-225384、 V-225385、 V-225386、 V-225389、 V-225391、 V-225393、 V-225395、 V-225397、 V-225398、 V-225400、 V-225401、 V-225402、 V-225404、 V-225405、 V-225406、 V-225407、 V-225408、 V-225409、 V-225410、 V-225411、 V-225413、 V-225414、 V-225415、 V-225441、 V-225442、 V-225443、 V-225448、 V-225452、 V-225453、 V-225454、 V-225455、 V-225456、 V-225457、 V-225458、 V-225461、 V-225463、 V-225464、 V-225469、 V-225470、 V-225471、 V-225472、 V-225474、 V-225475、 V-225477、 V-225478、 V-225486、 V-225494、 V-225500、 V-225501、 V-225502、 V-225503、 V-225504、 V-225506、 V-225508、 V-225509、 V-225510、 V-225513、 V-225515、 V-225516、 V-225517、 V-225518、 V-225519、 V-225520、 V-225521、 V-225522、 V-225523、 V-225524、 V-225527、 V-225528、 V-225529、 V-225530、 V-225531、 V-225532、 V-225533、 V-225534、 V-225535、 V-225538、 V-225539、 V-225540、 V-225541、 V-225542、 V-225543、 V-225544、 V-225545、 V-225546、 V-225548、 V-225549、 V-225550、 V-225551、 V-225553、 V-225554、 V-225555、 V-225557、 V-225558、 V-225559、 V-225560、 V-225561、

V-225562、 V-225563、 V-225564、 V-225565、 V-225566、 V-225567、 V-225568、 V-225569、 V-225570、 V-225571、 V-225572、 V-225573、 および V-225574

- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 6

  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 Firewall 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-241993, V-241998、および V-242003

- Internet Explorer 11 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-223015、 V-223017、 V-223018、 V-223019、 V-223020、 V-223021、 V-223022、 V-223023、 V-223024、 V-223025、 V-223026、 V-223027、 V-223028、 V-223029、 V-223030、 V-223031、 V-223032、 V-223033、 V-223034、 V-223035、 V-223036、 V-223037、 V-223038、 V-223039、 V-223040、 V-223041、 V-223042、 V-223043、 V-223044、 V-223045、 V-223046、 V-223048、 V-223049、 V-223050、 V-223051、 V-223052、 V-223053、 V-223054、 V-223055、 V-223057、 V-223058、 V-223059、 V-223060、 V-223061、 V-223062、 V-223063、 V-223064、 V-223065、 V-223066、 V-223067、 V-223068、 V-223069、 V-223070、 V-223071、 V-223072、 V-223073、 V-223074、 V-223075、 V-223076、 V-223077、 V-223079、 V-223080、 V-223081、 V-223082、 V-223083、 V-223084、 V-223085、 V-223086、 V-223087、 V-223088、 V-223089、 V-223090、 V-223091、 V-223092、 V-223093、 V-223094、 V-223095、 V-223096、 V-223097、 V-223098、 V-223099、 V-223100、 V-223101、 V-223102、 V-223103、 V-223104、 V-223105、 V-223106、 V-223107、 V-223108、 V-223109、 V-223110、 V-223111、 V-223112、 V-223113、 V-223114、 V-223115、 V-223116、 V-223117、 V-223118、 V-223119、 V-223120、 V-223121、 V-223122、 V-223123、 V-223124、 V-223125、 V-223126、 V-223127、 V-223128、 V-223129、 V-223130、 V-223131、 V-223132、 V-223133、 V-223134、 V-223135、 V-223136、 V-223137、 V-223138、 V-223139、 V-223140、 V-223141、 V-223142、 V-223143、 V-223144、 V-223145、 V-223146、 V-223147、 V-223148、 V-223149、 V-250540、 および V-250541

- Microsoft Edge STIG バージョン 2 リリース 2 (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-235720、V-235721、V-235723、V-235724、V-235725、V-235726、V-235728、V-235729、V-235730 および V-246736

- Microsoft Defender STIG バージョン 2 リリース 4

  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 2025.2.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 2 リリース 4

  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 3 リリース 4

  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 10

  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-225274, V-225354, V-225364, V-225365, V-225366, V-225390, V-225396, V-225399, V-225444, V-225449, V-225491, V-225492, V-225493, V-225496, V-225497, V-225498, V-225505, V-225507, V-225547, V-225552、および V-225556

- Microsoft .NET Framework 4.0 STIG バージョン 2 リリース 6

  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 Firewall STIG バージョン 2 リリース 2

  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 5

  V-252910

- Microsoft Edge STIG バージョン 2 リリース 2 (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

- Microsoft Defender STIG バージョン 2 リリース 4

  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。

2025 年Q2 四半期の変更 - 06/26/2025:

STIG バージョンを更新し、2025 Q2 2 四半期リリースの STIGS を次のように適用しました。

STIG-Build-Windows-Low バージョン 2025.2.x

- Windows Server 2022 STIG バージョン 2 リリース 4
- Windows Server 2019 STIG バージョン 3 リリース 4
- Windows Server 2016 STIG バージョン 2 リリース 10
- 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 6
- Windows Firewall STIG バージョン 2 リリース 2
- Internet Explorer 11 STIG バージョン 2 リリース 5
- Microsoft Edge STIG バージョン 2 リリース 2 (Windows Server 2022 のみ)

STIG-Build-Windows-Medium バージョン 2025.2.x

- Windows Server 2022 STIG バージョン 2 リリース 4

- Windows Server 2019 STIG バージョン 3 リリース 4
- Windows Server 2016 STIG バージョン 2 リリース 10
- 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 6
- Windows Firewall STIG バージョン 2 リリース 2
- Internet Explorer 11 STIG バージョン 2 リリース 5
- Microsoft Edge STIG バージョン 2 リリース 2 (Windows Server 2022 のみ)
- Defender STIG バージョン 2 リリース 4

STIG-Build-Windows-High バージョン 2025.2.x

- Windows Server 2022 STIG バージョン 2 リリース 4
- Windows Server 2019 STIG バージョン 3 リリース 4
- Windows Server 2016 STIG バージョン 2 リリース 10
- 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 6
- Windows Firewall STIG バージョン 2 リリース 2
- Internet Explorer 11 STIG バージョン 2 リリース 5
- Microsoft Edge STIG バージョン 2 リリース 2 (Windows Server 2022 のみ)
- Defender STIG バージョン 2 リリース 4

2025 年Q1 四半期の変更 - 05/04/2025:

2025 年第 1 四半期リリースのすべての STIG コンポーネントについて、Internet Explorer 11 STIG バージョン 2 リリース 5 の STIGS を更新しました。

- STIG-Build-Windows-Low バージョン 2025.1.x
- STIG-Build-Windows-Medium バージョン 2025.1.x
- STIG-Build-Windows-High バージョン 2025.1.x

2024 年Q4 四半期の変更 - 02/04/2025:

STIG バージョンを更新し、2024 年Q4 四半期リリースの STIGS を次のように適用しました。

STIG-Build-Windows-Low バージョン 2024.4.0

- Windows Server 2022 STIG バージョン 2 リリース 2
- Windows Server 2019 STIG バージョン 3 リリース 2
- Windows Server 2016 STIG バージョン 2 リリース 9
- 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 2
- Internet Explorer 11 STIG バージョン 2 リリース 5
- Microsoft Edge STIG バージョン 2 リリース 2 (Windows Server 2022 のみ)

STIG-Build-Windows-Medium バージョン 2024.4.0

- Windows Server 2022 STIG バージョン 2 リリース 2
- Windows Server 2019 STIG バージョン 3 リリース 2
- Windows Server 2016 STIG バージョン 2 リリース 9
- 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 2
- Internet Explorer 11 STIG バージョン 2 リリース 5
- Microsoft Edge STIG バージョン 2 リリース 2 (Windows Server 2022 のみ)
- Defender STIG バージョン 2 リリース 4

STIG-Build-Windows-High バージョン 2024.4.0

- Windows Server 2022 STIG バージョン 2 リリース 2
- Windows Server 2019 STIG バージョン 3 リリース 2
- Windows Server 2016 STIG バージョン 2 リリース 9
- 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 2
- Internet Explorer 11 STIG バージョン 2 リリース 5

- Microsoft Edge STIG バージョン 2 リリース 2 (Windows Server 2022 のみ)
- Defender STIG バージョン 2 リリース 4

2024 年Q3 四半期の変更 - 10/04/2023 (変更なし):

2024 年第 3 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。

2024 年第 2 四半期の変更 - 2024 年 5 月 10 日 (変更なし):

2024 年第 2 四半期リリースの Windows コンポーネント STIG には変更はありませんでした。

2024 年第 1 四半期の変更 - 2024 年 2 月 6 日 (変更なし):

2024 年第 1 四半期リリースの Windows コンポーネント STIG には変更はありませんでした。

2023 年第 4 四半期の変更 - 2023 年 12 月 4 日 (変更なし):

2023 年第 4 四半期リリースの Windows コンポーネント STIG には変更はありませんでした。

2023 年第 3 四半期の変更-2023 年 4 月 10 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 には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 には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 には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 には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 には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 ファイアウォール 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 ファイアウォール 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 ファイアウォール 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。

Linux コンポーネントには、Linux インスタンスの以下の動作をカスタマイズするのに役立つオプションの入力パラメータがあります。

- InstallPackages (文字列) 値が の場合No、コンポーネントは追加のソフトウェアパッケージをインストールしません。値が の場合Yes、コンポーネントはコンプライアンスを最大化するために必要な追加のソフトウェアパッケージをインストールします。デフォルトは No です。

- SetDoDConsentBanner (文字列) 値が の場合No、STIG Linux コンポーネントのいずれかがインストールされているインスタンスにアタッチすると、DoD 同意バナーは表示されません。値がの場合Yes、STIG Linux コンポーネントのいずれかがインストールされているインスタンスにアタッチすると、ログイン前に DoD 同意バナーが表示されます。ログインする前にバナーを確認する必要があります。デフォルトは No です。

  同意バナーの例については、DLA ドキュメントサービスのウェブサイトにアクセスしたときに表示される Disclaimer Department of Defense Privacy and consent Notice を参照してください。

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)

RHEL 9 STIG 設定

- RHEL 9
- CentOS Stream 9

STIG-Build-Linux-Low バージョン 2025.2.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 15

- RHEL 7/CentOS 7/AL2

  V-204452、V-204576、および V-204605

RHEL 8 STIG バージョン 2 リリース 3

- RHEL 8/CentOS 8/AL 2023

  V-230241, V-230269, V-230270, V-230281, V-230285, V-230346, V-230381, V-230395, V-230468, V-230469, V-230485, V-230486, V-230491, V-230494, V-230495, V-230496, V-230497, V-230498, V-230499, V-244527V-274141

RHEL 9 STIG バージョン 2 リリース 4

- RHEL 9/CentOS Stream 9

  V-257782, V-257795, V-257796, V-257824, V-257880, V-257946, V-257947, V-258037, V-258067, V-258069, V-258076, V-258138、および V-258173

Ubuntu 18.04 STIG バージョン 2 リリース 15

V-219163, V-219164, V-219165, V-219172, V-219173, V-219174, V-219175, V-219178, V-219179, V-219180, V-219210, V-219301, V-219322, V-219327, V-219332V-219333

Ubuntu 20.04 STIG バージョン 2 リリース 2

V-238202, V-238203, V-238221, V-238222, V-238223, V-238224, V-238226, V-238234, V-238235, V-238237, V-238308, V-238323, V-238357, V-238362、および V-238373

Ubuntu 22.04 STIG バージョン 2 リリース 4

V-260472、V-260476、V-260479、V-260480、V-260481、V-260520、V-260521、V-260549、V-260550、

Ubuntu 24.04 STIG バージョン 1 リリース 1

V-270645, V-270646, V-270664, V-270677, V-270690, V-270695, V-270706, V-270710, V-270749, V-270752, V-270818、および V-270820

STIG-Build-Linux-Medium バージョン 2025.2.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 15

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/AL2

  V-204405、 V-204406、 V-204407、 V-204408、 V-204409、 V-204410、 V-204411、
  V-204412、 V-204413、 V-204414、 V-204415、 V-204416、 V-204417、 V-204418、
  V-204422、 V-204423、 V-204426、 V-204427、 V-204431、 V-204434、 V-204435、
  V-204437、 V-204449、 V-204450、 V-204451、 V-204457、 V-204466、 V-204490、
  V-204491、 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-204579、 V-204584、

V-204585、 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-204609、 V-204610、 V-204611、 V-204612、 V-204613、 V-204614、
V-204615、 V-204616、 V-204617、 V-204619、 V-204622、 V-204625、 V-204630、
V-204631、 V-204633、 V-233307、 V-237634、 V-237635、 V-251703、 V-255925、
V-255927、 V-255928、 および V-256970

RHEL 8 STIG バージョン 2 リリース 3

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-230228、 V-230231、 V-230233、 V-230236、 V-230237、 V-230238、 V-230239、
V-230240、 V-230244、 V-230245、 V-230246、 V-230247、 V-230248、 V-230249、
V-230250、 V-230255、 V-230257、 V-230258、 V-230259、 V-230261、 V-230262、
V-230266、 V-230267、 V-230268、 V-230273、 V-230275、 V-230277、 V-230278、
V-230279、 V-230280、 V-230282、 V-230286、 V-230287、 V-230288、 V-230290、
V-230291、 V-230296、 V-230298、 V-230310、 V-230311、 V-230312、 V-230313、
V-230314、 V-230315、 V-230316、 V-230324、 V-230325、 V-230326、 V-230327、
V-230330、 V-230332、 V-230333、 V-230335、 V-230337、 V-230339、 V-230341、
V-230343、 V-230345、 V-230348、 V-230353、 V-230354、 V-230356、 V-230357、
V-230358、 V-230359、 V-230360、 V-230361、 V-230362、 V-230363、 V-230365、
V-230366、 V-230368、 V-230369、 V-230370、 V-230373、 V-230375、 V-230376、
V-230377、 V-230378、 V-230382、 V-230383、 V-230386、 V-230387、 V-230390、
V-230392、 V-230393、 V-230394、 V-230396、 V-230397、 V-230398、 V-230399、
V-230400、 V-230401、 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-230478、 V-230480、 V-230482、 V-230483、
V-230488、 V-230489、 V-230502、 V-230503、 V-230505、 V-230507、 V-230523、
V-230525、 V-230526、 V-230527、 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-230550、 V-230555、 V-230556、 V-230557、 V-230559、 V-230560、 V-230561、 V-237640、 V-237642、 V-237643、 V-244523、 V-244524、 V-244525、 V-244526、 V-244528、 V-244533、 V-244538、 V-244542、 V-244543、 V-244544、 V-244547、 V-244550、 V-244551、 V-244552、 V-244553、 V-244554、 V-250315、 V-250316、 V-250317、 V-251710、 V-251711、 V-251713、 V-251714、 V-251715、 V-251716、 V-251717、 V-251718、 V-256974、 および V-257258

RHEL 9 STIG バージョン 2 リリース 4

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 9/CentOS Stream 9

V-257780、 V-257781、 V-257783、 V-257786、 V-257788、 V-257790、 V-257791、 V-257792、 V-257793、 V-257794、 V-257797、 V-257798、 V-257799、 V-257800、 V-257801、 V-257802、 V-257803、 V-257804、 V-257805、 V-257806、 V-257807、 V-257808、 V-257809、 V-257810、 V-257811、 V-257812、 V-257813、 V-257814、 V-257815、 V-257816、 V-257817、 V-257818、 V-257825、 V-257827、 V-257828、 V-257829、 V-257830、 V-257831、 V-257832、 V-257833、 V-257834、 V-257836、 V-257838、 V-257839、 V-257840、 V-257841、 V-257842、 V-257849、 V-257882、 V-257883、 V-257884、 V-257885、 V-257886、 V-257887、 V-257888、 V-257889、 V-257890、 V-257891、 V-257892、 V-257893、 V-257894、 V-257895、 V-257896、 V-257897、 V-257898、 V-257899、 V-257900、 V-257901、 V-257902、 V-257903、 V-257904、 V-257905、 V-257906、 V-257907、 V-257908、 V-257909、 V-257910、 V-257911、 V-257912、 V-257913、 V-257914、 V-257915、 V-257916、 V-257917、 V-257918、 V-257919、 V-257920、 V-257921、 V-257922、 V-257923、 V-257924、 V-257925、 V-257926、 V-257927、 V-257928、 V-257929、 V-257930、 V-257933、 V-257934、 V-257935、 V-257936、 V-257939、 V-257940、 V-257942、 V-257943、 V-257944、 V-257948、 V-257949、 V-257951、 V-257952、 V-257953、 V-257954、 V-257957、 V-257958、 V-257959、 V-257960、 V-257961、 V-257962、 V-257963、 V-257964、 V-257965、 V-257966、 V-257967、 V-257968、 V-257969、 V-257970、 V-257971、 V-257972、 V-257973、 V-257974、 V-257975、 V-257976、 V-257977、 V-257978、 V-257979、 V-257980、 V-257982、 V-257983、 V-257985、 V-257987、 V-257988、 V-257992、 V-257993、 V-257994、 V-257995、 V-257996、 V-257997、

V-257998、 V-257999、 V-258000、 V-258001、 V-258002、 V-258003、 V-258004、
V-258005、 V-258006、 V-258007、 V-258008、 V-258009、 V-258010、 V-258011、
V-258028、 V-258034、 V-258035、 V-258038、 V-258039、 V-258040、 V-258041、
V-258043、 V-258046、 V-258049、 V-258052、 V-258054、 V-258055、 V-258056、
V-258057、 V-258060、 V-258063、 V-258064、 V-258065、 V-258066、 V-258068、
V-258070、 V-258071、 V-258072、 V-258073、 V-258074、 V-258075、 V-258077、
V-258079、 V-258080、 V-258081、 V-258082、 V-258083、 V-258084、 V-258085、
V-258088、 V-258089、 V-258091、 V-258092、 V-258093、 V-258095、 V-258097、
V-258098、 V-258099、 V-258100、 V-258101、 V-258102、 V-258103、 V-258104、
V-258105、 V-258107、 V-258108、 V-258109、 V-258110、 V-258111、 V-258112、
V-258113、 V-258114、 V-258115、 V-258116、 V-258117、 V-258118、 V-258119、
V-258120、 V-258122、 V-258123、 V-258124、 V-258125、 V-258126、 V-258128、
V-258129、 V-258130、 V-258133、 V-258137、 V-258140、 V-258141、 V-258142、
V-258144、 V-258145、 V-258146、 V-258147、 V-258148、 V-258150、 V-258151、
V-258152、 V-258153、 V-258154、 V-258156、 V-258157、 V-258158、 V-258159、
V-258160、 V-258161、 V-258162、 V-258163、 V-258164、 V-258165、 V-258166、
V-258167、 V-258168、 V-258169、 V-258170、 V-258171、 V-258172、 V-258175、
V-258176、 V-258177、 V-258178、 V-258179、 V-258180、 V-258181、 V-258182、
V-258183、 V-258184、 V-258185、 V-258186、 V-258187、 V-258188、 V-258189、
V-258190、 V-258191、 V-258192、 V-258193、 V-258194、 V-258195、 V-258196、
V-258197、 V-258198、 V-258199、 V-258200、 V-258201、 V-258202、 V-258203、
V-258204、 V-258205、 V-258206、 V-258207、 V-258208、 V-258209、 V-258210、
V-258211、 V-258212、 V-258213、 V-258214、 V-258215、 V-258216、 V-258217、
V-258218、 V-258219、 V-258220、 V-258221、 V-258222、 V-258223、 V-258224、
V-258225、 V-258226、 V-258227、 V-258228、 V-258229、 V-258232、 V-258233、
V-258234、 V-258237、 V-258239、 V-258240、 および V-272488

Ubuntu 18.04 STIG バージョン 2 リリース 15

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-219149、 V-219155、 V-219156、 V-219160、 V-219166、 V-219168、 V-219176、 V-219181、
V-219184、 V-219186、 V-219188、 V-219189、 V-219190、 V-219191、 V-219192、 V-219193、
V-219194、 V-219195、 V-219196、 V-219197、 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-219213、 V-219214、 V-219215、 V-219216、 V-219217、 V-219218、 V-219219、 V-219220、

V-219221、 V-219222、 V-219223、 V-219224、 V-219225、 V-219226、 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-219303、 V-219304、 V-219306、 V-219309、 V-219310、 V-219311、 V-219315、 V-219318、 V-219319、 V-219323、 V-219326、 V-219328、 V-219330、 V-219331、 V-219335、 V-219336、 V-219337、 V-219338、 V-219339、 V-219342、 V-219344、 V-233779、 V-233780、 および V-255906

Ubuntu 20.04 STIG バージョン 2 リリース 2

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-238200、 V-238205、 V-238207、 V-238209、 V-238210、 V-238211、 V-238212、 V-238213、 V-238220、 V-238225、 V-238227、 V-238228、 V-238229、 V-238230、 V-238231、 V-238232、 V-238236、 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-238298、 V-238299、 V-238300、 V-238301、 V-238302、 V-238303、 V-238304、 V-238309、 V-238310、 V-238315、 V-238316、 V-238317、 V-238318、 V-238319、 V-238320、 V-238324、 V-238325、 V-238329、 V-238330、 V-238333、 V-238334、 V-238337、 V-238338、 V-238339、 V-238340、 V-238341、 V-238342、 V-238343、 V-238344、 V-238345、 V-238346、 V-238347、 V-238348、 V-238349、 V-238350、 V-238351、 V-238352、 V-238353、 V-238355、 V-238356、 V-238359、 V-238360、 V-238369、 V-238370、 V-238371、 V-238376、 V-238377、 V-238378、 V-251505、 および V-255912

Ubuntu 22.04 STIG バージョン 2 リリース 4

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-260471、 V-260473、 V-260474、 V-260475、 V-260477、 V-260478、 V-260485、 V-260486、 V-260487、 V-260488、 V-260489、 V-260490、 V-260491、 V-260492、 V-260493、 V-260494、 V-260495、 V-260496、 V-260497、 V-260498、 V-260499、 V-260500、 V-260505、 V-260506、

V-260507、 V-260508、 V-260509、 V-260510、 V-260511、 V-260512、 V-260513、 V-260514、 V-260522、 V-260527、 V-260528、 V-260530、 V-260533、 V-260534、 V-260535、 V-260537、 V-260538、 V-260540、 V-260542、 V-260543、 V-260545、 V-260546、 V-260547、 V-260553、 V-260554、 V-260555、 V-260556、 V-260557、 V-260560、 V-260561、 V-260562、 V-260563、 V-260564、 V-260565、 V-260566、 V-260567、 V-260569、 V-260572、 V-260573、 V-260574、 V-260575、 V-260576、 V-260582、 V-260584、 V-260585、 V-260586、 V-260588、 V-260589、 V-260590、 V-260591、 V-260594、 V-260597、 V-260598、 V-260599、 V-260600、 V-260601、 V-260602、 V-260603、 V-260604、 V-260605、 V-260606、 V-260607、 V-260608、 V-260609、 V-260610、 V-260611、 V-260612、 V-260613、 V-260614、 V-260615、 V-260616、 V-260617、 V-260618、 V-260619、 V-260620、 V-260621、 V-260622、 V-260623、 V-260624、 V-260625、 V-260626、 V-260627、 V-260628、 V-260629、 V-260630、 V-260631、 V-260632、 V-260633、 V-260634、 V-260635、 V-260636、 V-260637、 V-260638、 V-260639、 V-260640、 V-260641、 V-260642、 V-260643、 V-260644、 V-260645、 V-260646、 V-260647、 V-260648、 および V-260649

Ubuntu 24.04 STIG バージョン 1 リリース 1

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-270649、 V-270651、 V-270652、 V-270653、 V-270654、 V-270656、 V-270657、 V-270659、 V-270660、 V-270661、 V-270662、 V-270663、 V-270669、 V-270672、 V-270673、 V-270674、 V-270676、 V-270678、 V-270679、 V-270680、 V-270681、 V-270683、 V-270684、 V-270685、 V-270686、 V-270687、 V-270688、 V-270689、 V-270692、 V-270693、 V-270696、 V-270697、 V-270698、 V-270699、 V-270700、 V-270701、 V-270702、 V-270703、 V-270704、 V-270705、 V-270709、 V-270715、 V-270716、 V-270718、 V-270720、 V-270721、 V-270722、 V-270723、 V-270724、 V-270725、 V-270726、 V-270727、 V-270728、 V-270729、 V-270730、 V-270731、 V-270732、 V-270733、 V-270737、 V-270739、 V-270740、 V-270741、 V-270742、 V-270743、 V-270746、 V-270750、 V-270753、 V-270755、 V-270756、 V-270757、 V-270758、 V-270759、 V-270760、 V-270765、 V-270766、 V-270767、 V-270768、 V-270769、 V-270770、 V-270771、 V-270772、 V-270773、 V-270775、 V-270776、 V-270777、 V-270778、 V-270779、 V-270780、 V-270781、 V-270782、 V-270783、 V-270784、 V-270785、 V-270786、 V-270787、 V-270788、 V-270789、 V-270790、 V-270791、 V-270792、 V-270793、 V-270794、 V-270795、 V-270796、 V-270797、 V-270798、 V-270799、 V-270800、 V-270801、 V-270802、 V-270803、 V-270804、 V-270805、 V-270806、 V-270807、 V-270808、 V-270809、 V-270810、 V-270811、 V-270812、 V-270813、 V-270814、 V-270815、 V-270821、 V-270822、 V-270823、 V-270824、 V-270825、 V-270826、 V-270827、 V-270828、 V-270829、 V-270830、 V-270831、 および V-270832

STIG-Build-Linux-High バージョン 2025.2.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リストされたすべての STIG 設定が含まれます。

RHEL 7 STIG バージョン 3 リリース 15

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/AL2

  V-204424, V-204425, V-204442, V-204443, V-204447, V-204448, V-204455, V-204497, V-204502, V-204594, V-204620、および V-204621

RHEL 8 STIG バージョン 2 リリース 3

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-230223, V-230226, V-230264, V-230265, V-230487, V-230492, V-230529, V-230530, V-230531, V-230533, V-230558、および V-244540

RHEL 9 STIG バージョン 2 リリース 4

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 9/CentOS Stream 9

  V-257784, V-257785, V-257820, V-257821, V-257826, V-257835, V-257955, V-257956, V-257984, V-257986, V-258059, V-258078, V-258094, V-258230, V-258235、および V-258238

Ubuntu 18.04 STIG バージョン 2 リリース 15

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, V-251507、および V-264388

Ubuntu 20.04 STIG バージョン 2 リリース 2

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-238201, V-238218, V-238219, V-238326, V-238327, V-238380、および V-251504

Ubuntu 22.04 STIG バージョン 2 リリース 4

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-260469, V-260482, V-260483, V-260523, V-260524, V-260526, V-260529, V-260539, V-260570, V-260571、および V-260579

Ubuntu 24.04 STIG バージョン 1 リリース 1

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-270647, V-270648, V-270665, V-270666, V-270708, V-270711, V-270712, V-270713, V-270714、および V-270717

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。

2025 年Q2 四半期の変更 - 06/26/2025:

次の STIG バージョンを更新し、STIGS を 2025 年第 2 四半期リリースに適用しました。

STIG-Build-Linux-Low バージョン 2025.2.x

- RHEL 7 STIG バージョン 3 リリース 15
- RHEL 8 STIG バージョン 2 リリース 3
- RHEL 9 STIG バージョン 2 リリース 4
- Ubuntu 18.04 STIG バージョン 2 リリース 15
- Ubuntu 20.04 STIG バージョン 2 リリース 2
- Ubuntu 22.04 STIG バージョン 2 リリース 4
- Ubuntu 24.04 STIG バージョン 1 リリース 1

STIG-Build-Linux-Medium バージョン 2025.2.x

- RHEL 7 STIG バージョン 3 リリース 15
- RHEL 8 STIG バージョン 2 リリース 3
- RHEL 9 STIG バージョン 2 リリース 4
- Ubuntu 18.04 STIG バージョン 2 リリース 15
- Ubuntu 20.04 STIG バージョン 2 リリース 2
- Ubuntu 22.04 STIG バージョン 2 リリース 4
- Ubuntu 24.04 STIG バージョン 1 リリース 1

STIG-Build-Linux-High バージョン 2025.2.x

- RHEL 7 STIG バージョン 3 リリース 15
- RHEL 8 STIG バージョン 2 リリース 3
- RHEL 9 STIG バージョン 2 リリース 4
- Ubuntu 18.04 STIG バージョン 2 リリース 15
- Ubuntu 20.04 STIG バージョン 2 リリース 2
- Ubuntu 22.04 STIG バージョン 2 リリース 4
- Ubuntu 24.04 STIG バージョン 1 リリース 1

2025 年Q1 四半期の変更 - 04/11/2025:

次の STIG バージョンを更新し、2025 年第 1 四半期リリースに STIGS を適用し、Ubuntu 24.04 のサポートを追加しました。

STIG-Build-Linux-Low バージョン 2025.1.x

- RHEL 7 STIG バージョン 3 リリース 15
- RHEL 8 STIG バージョン 2 リリース 2
- RHEL 9 STIG バージョン 2 リリース 3
- Ubuntu 18.04 STIG バージョン 2 リリース 15
- Ubuntu 20.04 STIG バージョン 2 リリース 2
- Ubuntu 22.04 STIG バージョン 2 リリース 3
- Ubuntu 24.04 STIG バージョン 1 リリース 1

STIG-Build-Linux-Medium バージョン 2025.1.x

- RHEL 7 STIG バージョン 3 リリース 15
- RHEL 8 STIG バージョン 2 リリース 2
- RHEL 9 STIG バージョン 2 リリース 3
- Ubuntu 18.04 STIG バージョン 2 リリース 15
- Ubuntu 20.04 STIG バージョン 2 リリース 2
- Ubuntu 22.04 STIG バージョン 2 リリース 3
- Ubuntu 24.04 STIG バージョン 1 リリース 1

STIG-Build-Linux-High バージョン 2025.1.x

- RHEL 7 STIG バージョン 3 リリース 15
- RHEL 8 STIG バージョン 2 リリース 2
- RHEL 9 STIG バージョン 2 リリース 3
- Ubuntu 18.04 STIG バージョン 2 リリース 15
- Ubuntu 20.04 STIG バージョン 2 リリース 2
- Ubuntu 22.04 STIG バージョン 2 リリース 3
- Ubuntu 24.04 STIG バージョン 1 リリース 1

2024 年Q4 四半期の変更 - 12/10/2024:

次の STIG バージョンを更新し、2024 年第 4 四半期リリースに STIGS を適用し、Linux コンポーネントの 2 つの新しい入力パラメータに関する情報を追加しました。

STIG-Build-Linux-Low バージョン 2024.4.x

- RHEL 7 STIG バージョン 3 リリース 15
- RHEL 8 STIG バージョン 2 リリース 1
- RHEL 9 STIG バージョン 2 リリース 2
- Ubuntu 18.04 STIG バージョン 2 リリース 15
- Ubuntu 20.04 STIG バージョン 2 リリース 1
- Ubuntu 22.04 STIG バージョン 2 リリース 2

STIG-Build-Linux-Medium バージョン 2024.4.x

- RHEL 7 STIG バージョン 3 リリース 15
- RHEL 8 STIG バージョン 2 リリース 1
- RHEL 9 STIG バージョン 2 リリース 2
- Ubuntu 18.04 STIG バージョン 2 リリース 15
- Ubuntu 20.04 STIG バージョン 2 リリース 1
- Ubuntu 22.04 STIG バージョン 2 リリース 2

STIG-Build-Linux-High バージョン 2024.4.x

- RHEL 7 STIG バージョン 3 リリース 15
- RHEL 8 STIG バージョン 2 リリース 1
- RHEL 9 STIG バージョン 2 リリース 2
- Ubuntu 18.04 STIG バージョン 2 リリース 15
- Ubuntu 20.04 STIG バージョン 2 リリース 1
- Ubuntu 22.04 STIG バージョン 2 リリース 2

2024 年Q3 四半期の変更 - 10/04/2024 (変更なし):

2024 年第 3 四半期リリースでは、Linux コンポーネント STIGS に変更はありませんでした。

2024 年第 2 四半期の変更 - 2024 年 5 月 10 日:

STIG バージョンを更新し、2024 年第 2 四半期リリースの STIG を適用しました。また、RHEL 9、CentOS Stream 9、Ubuntu 22.04 のサポートを以下のように追加しました。

STIG-Build-Linux-Low バージョン 2024.2.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 14
- RHEL 9 STIG バージョン 1 リリース 3
- Ubuntu 18.04 STIG バージョン 2 リリース 14
- Ubuntu 20.04 STIG バージョン 1 リリース 12
- Ubuntu 22.04 STIG バージョン 1 リリース 1

STIG-Build-Linux-Medium バージョン 2024.2.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 14
- RHEL 9 STIG バージョン 1 リリース 3
- Ubuntu 18.04 STIG バージョン 2 リリース 14
- Ubuntu 20.04 STIG バージョン 1 リリース 12
- Ubuntu 22.04 STIG バージョン 1 リリース 1

STIG-Build-Linux-High バージョン 2024.2.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 14
- RHEL 9 STIG バージョン 1 リリース 3
- Ubuntu 18.04 STIG バージョン 2 リリース 14
- Ubuntu 20.04 STIG バージョン 1 リリース 12
- Ubuntu 22.04 STIG バージョン 1 リリース 1

2024 年第 1 四半期の変更 - 2024 年 2 月 6 日:

STIG バージョンを更新し、次のように 2024 年第 1 四半期リリースの STIG を適用しました。

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 年第 4 四半期の変更 - 2023 年 12 月 7 日:

STIG バージョンを更新し、次のように 2023 年第 4 四半期リリースの STIG を適用しました。

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 4 月 10 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 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 には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`scap-compliance-checker-linuxコンポーネントは、パイプラインビルドおよび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 ベンチマーク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、その情報をアプリケーションログにも書き込みます。

- 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-windows バージョン 2024.03.0

`scap-compliance-checker-windows` コンポーネントは、Image Builder が Image. AWSTOE logs を構築してテストするために作成する EC2 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. コンプライアンススコアをローカルレポートから 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 MS、2016、2019、2022 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: 5.10

2023 年Q4 四半期ベンチマーク：

- U_MS_Defender_Antivirus_V2R5_STIG_SCAP_1-2_Benchmark
- U_MS_DotNet_Framework_4-0_V2R2_STIG_SCAP_1-2_Benchmark
- U_MS_IE11_V2R6_STIG_SCAP_1-2_ベンチマーク
- U_MS_Windows_2012_and_2012_R2_DC_V3R5_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_2012_and_2012_R2_MS_V3R5_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Defender_Firewall_V2R3_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Server_2016_V2R7_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Server_2019_V3R2_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Server_2022_V2R2_STIG_SCAP_1-2_Benchmark
- U_CAN_Ubuntu_20-04_LTS_V1R10_STIG_SCAP_1-2_Benchmark
- U_RHEL_7_V3R15_STIG_SCAP_1-3_ベンチマーク
- U_RHEL_8_V1R13_STIG_SCAP_1-3_ベンチマーク
- U_RHEL_9_V2R1_STIG_SCAP_1-3_ベンチマーク

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 が作成した EC2 インスタンスで実行され、Image. AWSTOE log がレポートと SCC アプリケーションが生成するスコアの両方をビルドしてテストします。

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. コンプライアンススコアをローカルレポートから 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 18.04/20.04 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: 5.10

2023 年Q4 四半期ベンチマーク：

- U_CAN_Ubuntu_20-04_LTS_V1R10_STIG_SCAP_1-2_Benchmark
- U_RHEL_7_V3R15_STIG_SCAP_1-3_ベンチマーク
- U_RHEL_8_V1R13_STIG_SCAP_1-3_ベンチマーク
- U_RHEL_9_V2R1_STIG_SCAP_1-3_ベンチマーク
- U_MS_Defender_Antivirus_V2R5_STIG_SCAP_1-2_Benchmark
- U_MS_DotNet_Framework_4-0_V2R2_STIG_SCAP_1-2_Benchmark
- U_MS_IE11_V2R6_STIG_SCAP_1-2_ベンチマーク
- U_MS_Windows_2012_and_2012_R2_DC_V3R5_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_2012_and_2012_R2_MS_V3R5_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Defender_Firewall_V2R3_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Server_2016_V2R7_STIG_SCAP_1-2_Benchmark
- U_MS_Windows_Server_2019_V3R2_STIG_SCAP_1-2_Benchmark

- U_MS_Windows_Server_2022_V2R2_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
2025 年第 Q1 SCAP 更新	• scap-compliance-checker-windows バージョン 2024.03.0 (SCC バージョン: 5.10) • scap-compliance-checker-windows バージョン 2025.01.0 (SCC バージョン: 5.10	2025 年 4 月 11 日
2023 年第 Q4 SCAP 更新	• scap-compliance-checker-windows バージョン 2023.04.0 (SCC バージョン: 5.8) • scap-compliance-checker-windows バージョン 2023.04.0 (SCC バージョン: 5.8)	2021 年 12 月 20 日
2023 年第 Q3 SCAP 更新	• scap-compliance-checker-windows バージョン 2023.03.0 (SCC バージョン: 5.7.2) • scap-compliance-checker-linux バージョン 2023.03.0 (SCC バージョン: 5.7.2	2023 年 11 月 13 日
SCAP コンポーネントを追加	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 (SCC バージョン: 5.4.2) を作成しました • scap-compliance-checker-linux バージョン 2021.04.0 (SCC バージョン: 5.4.2) を作成しました	2021 年 12 月 20 日

Image Builder イメージ用のカスタムコンポーネントの開発

独自のコンポーネントを作成すると、独自の仕様に正確に従って Image Builder イメージをカスタマイズできます。Image Builder イメージまたはコンテナレシピ用のカスタムコンポーネントを開発するには、次の手順に従います。

1. コンポーネントドキュメントを開発してローカルで検証する場合は、 AWS Task Orchestrator and Executor （ AWSTOE) アプリケーションをインストールしてローカルマシンにセットアップできます。詳細については、「を使用してカスタムコンポーネントを開発するための手動セットアップ AWSTOE」を参照してください。
2. コンポーネントドキュメントフレームワークを使用する AWSTOE コンポーネントドキュメントを作成します。ドキュメントフレームワークの詳細については、「カスタム AWSTOE コンポーネントのコンポーネントドキュメントフレームワークを使用する」のドキュメントを参照してください。
3. カスタムコンポーネントの作成時には、コンポーネントドキュメントを指定します。詳細については、「Image Builder を使用したカスタムコンポーネントの作成」を参照してください。

トピック

- Image Builder でのカスタムコンポーネント用 YAML コンポーネントドキュメントの作成
- Image Builder を使用したカスタムコンポーネントの作成

Image Builder でのカスタムコンポーネント用 YAML コンポーネントドキュメントの作成

コンポーネントをビルドするには、YAML または JSON アプリケーションコンポーネントドキュメントを提供する必要があります。このドキュメントには、フェーズとステップの間に実行される、イメージをカスタマイズするために定義したコードが含まれています。

このセクションの例の一部は、コンポーネント管理アプリケーションの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。

macOS オペレーティングシステムの例では、ExecuteBash アクションモジュールを使用して、wget ユーティリティをインストールして検証します。UpdateOS アクションモジュールは macOS をサポートしていません。ExecuteBash アクションの使用方法の詳細について

は、ExecuteBashを参照してください。 AWSTOE 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、「AWSTOEでのドキュメントの使用」を参照してください。

> **Note**
>
> Image Builder は、コンポーネントドキュメントで定義されているフェーズから、次のようにコンポーネントタイプを決定します。
>
> - 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、コンポーネントを使用した Image Builder イメージのカスタマイズ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する

ファイル編集ツールを使用して、コンポーネントドキュメントを作成します。ドキュメントの例では、次のコンテンツを含む *update-linux-os.yaml* という名前のファイルを使用します。

```
# 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する

ファイル編集ツールを使用して、コンポーネントドキュメントを作成します。ドキュメントの例では、次のコンテンツを含む *update-windows-os.yaml* という名前のファイルを使用します。

```
# 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。

macOS

YAML コンポーネントファイルを作成する

ファイル編集ツールを使用して、コンポーネントドキュメントを作成します。ドキュメントの例では、次のコンテンツを含む *wget-macos.yaml* という名前のファイルを使用します。

```
name: WgetInstallDocument
description: This is wget installation document.
schemaVersion: 1.0

phases:
  - name: build
```

```
  steps:
    - name: WgetBuildStep
      action: ExecuteBash
      inputs:
        commands:
          - |
            PATH=/usr/local/bin:$PATH
            sudo -u ec2-user brew install wget


- name: validate
  steps:
    - name: WgetValidateStep
      action: ExecuteBash
      inputs:
        commands:
          - |
            function error_exit {
              echo $1
              echo "{\"failureMessage\":\"$2\"}"
              exit 1
            }

            type wget
            if [ $? -ne 0 ]; then
              error_exit "$stderr" "Wget installation failed!"
            fi

- name: test
  steps:
    - name: WgetTestStep
      action: ExecuteBash
      inputs:
        commands:
          - wget -h
```

> **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。

Image Builder を使用したカスタムコンポーネントの作成

コンポーネントドキュメントが完成したら、それを使用して、Image Builder レシピで使用できるカスタムコンポーネントを作成できます。カスタムコンポーネントは、Image Builder コンソール、API か SDK、またはコマンドラインから作成できます。入力パラメータを持つカスタムコンポーネントを作成してレシピで使用する方法の詳細については、「チュートリアル: 入力パラメータを持つカスタムコンポーネントを作成する」を参照してください。

以下のセクションでは、コンソールまたは AWS CLIからコンポーネントを作成する方法を示します。

内容

コンソールでのカスタムモデルコンポーネントの作成

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、Image Builder でのカスタムコンポーネント用 YAML コンポーネントドキュメントの作成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 フィルタを適用します。フィルタは、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。

コンポーネントの更新

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既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、Image Builder でのセマンティックバージョニングを参照してください。

からカスタム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を使用してコンポーネントのバージョンを一覧表示し、既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://amzn-s3-demo-destination-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する

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。前の手順で作成した *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 オペレーションリクエストパラメータに指定された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://amzn-s3-demo-destination-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"
 }
}
```

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://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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://amzn-s3-demo-destination-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する

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。前の手順で作成した *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 オペレーションリクエストパラメータに指定された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://amzn-s3-demo-destination-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://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS ではフォーワードスラッシュ (/) が使用されます。

macOS

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 wget-macos.yaml s3://amzn-s3-demo-destination-bucket/my-path/wget-macos.yaml
```

YAML ドキュメントからコンポーネントを作成する

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。前の手順で作成した *wget-macos.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 オペレーションリクエストパラメータに指定された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、*install-wget-macos-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": "install install-wget-macos-component",
 "semanticVersion": "1.1.2",
 "description": "An example component that installs and verifies the wget
 utility on macOS.",
 "changeDescription": "Initial version.",
 "platform": "macOS",
 "uri": "s3://amzn-s3-demo-destination-bucket/my-path/wget-macos.yaml",
 "kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/98765432-
b123-456b-7f89-0123456f789c",
 "tags": {
  "MyTagKey-purpose": "install-software"
 }
}
```

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://install-wget-macos-
component.json
```

> **Note**
>
> - JSON ファイルパスの先頭に `file://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS ではフォーワードスラッシュ (/) が使用されます。

AWSTOE からの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、Image Builder での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。で list-components コマンドを使用し 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。

スクリプトをインポートして からコンポーネントを作成する AWS CLI

シナリオによっては、既存のスクリプトから始める方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。このファイルは、既に *amzn-s3-demo-source-bucket* にアップロードされている AdminConfig.ps1 というPowerShell スクリプトを直接に参照していることを注意してください。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://amzn-s3-demo-source-bucket/AdminConfig.ps1",
"kmsKeyId": "arn:aws:kms:us-west-2:123456789012:key/60763706-
b131-418b-8f85-3420912f020c"
}
```

インポートされたスクリプトからコンポーネントを作成するには、次のコマンドを実行します。

```
aws imagebuilder import-component --cli-input-json file://import-component.json
```

> **Note**
>
> 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、「未使用または古くなった Image Builder リソースの削除」を参照してください。

Image Builder が AWS Task Orchestrator and Executor アプリケーションを使用してコンポーネントを管理する方法

EC2 Image Builder は AWS Task Orchestrator and Executor （ AWSTOE) アプリケーションを使用して、追加の devops スクリプトやコードを使用せずに、複雑なワークフローの調整、システム設定の変更、イメージのテストを行います。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。

AWSTOE は、イメージの作成時に Image Builder がビルドインスタンスとテストインスタンスにインストールするスタンドアロンアプリケーションです。これを手動で EC2 インスタンスにインストールして、独自のカスタムコンポーネントを作成することもできます。追加のセットアップは必要なく、オンプレミスでも実行できます。

内容

AWSTOE ダウンロード

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、 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 ダウンロード用に 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、8、および 9 Ubuntu 16.04、18.04、20.04、22.04、24.04 CentOS 7 および 8 スーズ 12 と 15	`https://awstoe-<region>.s3.amazonaws.com/latest/linux/386/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/386/awstoe
AMD64	AL 2 と 2023 年 RHEL 7、8、および 9 Ubuntu 16.04、18.04、20.04、22.04、24.04 CentOS 7 および 8 CentOS Stream 8 スーズ 12 と 15	`https://awstoe-<region>.s3.amazonaws.com/latest/linux/amd64/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/linux/amd64/awstoe
AMD64	macOS 10.14.x (Mojave)、10.15.x (Catalina)、11.x (Big Sur)、12.x (Monterey)	`https://awstoe-region.s3.amazonaws.com/latest/darwin/amd64/awstoe`	https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/darwin/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
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
ARM64	AL 2 と 2023 年 RHEL 7、8、および 9 Ubuntu 16.04、18.04、20.04、22.04、24.04 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

サポート対象の リージョン

AWSTOE は、次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

AWS リージョン 名前	AWS リージョン
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
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

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

AWSTOE コマンドリファレンス

AWSTOE は、Amazon EC2 インスタンスで実行されるコマンドラインコンポーネント管理アプリケーションです。Image Builder が EC2 ビルドインスタンスまたはテストインスタンスを起動すると、インスタンス AWSTOE に がインストールされます。次に、 で AWSTOE コマンドを実行して AWS CLI 、イメージまたはコンテナレシピで指定されたコンポーネントをインストールまたは検証します。

> **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の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 run コマンド

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 パラメータで指定された設定ファイル、または --documents パラメータ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
> --documents

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とおり:

- 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 Logs からのロギングエラーは無視してください。

--cw-log-group

ショートフォーム:N/A

CloudWatch Logs の LogGroup の名前。

--cw-log-region

ショートフォーム:N/A

CloudWatch Logs 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 ファイルのストリーミング AWSTOE 先を指定する CloudWatch Logs LogStreamの名前。

--document-s3-bucket-owner

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。

--documents ***./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 パラメータを指定してはなりません。

有効な場所は以下のとおり:

- ローカルファイルパス (*/component-doc-example.yaml*)。
- S3 URI (s3://*bucket/key*)。
- *Image Builder ############### ARN (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。

--execution-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。

--log-directory

短縮形： -l

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親ディレクトリ `TOE_<DATETIME>_<EXECUTIONID>` の中にあります。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)、 はこのパラメータで という名前の S3 バケットにコンポーネントアプリケーションログ AWSTOE をアップロードします。

--log-s3-bucket-owner

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-key-prefix

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。

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

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-directory

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-bucket-owner

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。

--documents ***./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とおり:

- 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。

を使用してカスタムコンポーネントを開発するための手動セットアップ 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 のサービスでは、 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手動でインストールし、カスタムコンポーネントを開発するためのスタンドアロンアプリケーションとして使用するには、次の手順に従います。Image Builder コンソールまたは AWS CLI コマンドを使用してカスタムコンポーネントを作成する場合、Image Builder はこれらのステップを自動的に処理します。詳細については、「Image Builder を使用したカスタムコンポーネントの作成」を参照してください。

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、macOS、Windows ベースのオペレーティングシステム AWSTOE で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 または macOS 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 または macOS 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または macOS 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 アプリケーションのソフトウェアが変更されているか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。サポートオプションの詳細については、 サポート 「」を参照してくださいサポート。

AWSTOE Linux ベースおよび macOS 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、デジタル署名による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

使用しているオペレーティングシステムが Linux、Unix、または macOS の場合、GPG ツールが既にインストールされている可能性があります。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。

GPG ツールをインストールするには、インスタンスに一致するオペレーティングシステムを選択します。

Debian-based Linux

ターミナルから、次のコマンドを実行します。

```
apt-get install gnupg
```

Red Hat–based Linux

ターミナルから、次のコマンドを実行します。

```
yum install gnupg
```

macOS

ターミナルから、次のコマンドを実行します。

```
brew install gnupg
```

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、GPGキーリングに信頼されたキーとして追加します。

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。

   - 次の場所からキーをダウンロードします。

     https://awstoe-**<region>**.s3.**<region>**.amazonaws.com/assets/awstoe.gpg。例えば、https://awstoe-us-east-1.s3.us-east-1.amazonaws.com/latest/assets/awstoe.gpg。
   - 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 サポート。

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 キー接頭辞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、 サポート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。 AWS オフィスを訪問し、直接キーを受け取るのが理想的です。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 アプリケーションのソフトウェアが変更されているか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 サポート。

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：

BA 81 25 EE AC 64 2E A9 F3 C5 93 CA 6D 3E B7 93 7D 68 75 74

> **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、サムプリントが次の値である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

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、

      BA 81 25 EE AC 64 2E A9 F3 C5 93 CA 6D 3E B7 93 7D 68 75 74

      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。
    - 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

   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 ダウンロード用に 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

   ダウンロードを検証する手順は、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 サーバーでダウンロードを確認する方法については、「Linux または macOS での署名の検証」を参照してください。Windows サーバー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。

> **Note**
>
> 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、Amazon S3 や Amazon CloudWatch 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します。
- `S3Download` または `S3Upload` アクションモジュールを実行する。
- CloudWatch にログをストリーミングします (有効になっている場合)。

EC2 インスタンス AWSTOE で実行している場合、 の実行は 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 を使用します。

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
```

Windows で PowerShell を使ってこれらの変数を設定するには、`$env`を使う。

```
$env:AWS_ACCESS_KEY_ID=your_access_key_id
```

```
$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。

```
set AWS_ACCESS_KEY_ID=your_access_key_id
```

```
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

コンポーネントは、プレーンテキストの YAML ドキュメントを使用して作成されます。ドキュメントの構文については[カスタム AWSTOE コンポーネントのコンポーネントドキュメントフレームワークを使用する]を参照。

以下は、開発の開始点として役立つ Hello World コンポーネントドキュメントの例です。

Linux

このガイドの Linux 用のコンポーネント例の一部は、`hello-world-linux.yml` という名前のコンポーネントドキュメントファイルを参照しています。これらの例を試すには、次のドキュメントを使用できます。

```
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.'
```

Windows

このガイドの Windows 用のコンポーネント例の一部は、hello-world-windows.yml という名前のコンポーネントドキュメントファイルを参照しています。これらの例を試すには、次のドキュメントを使用できます。

```
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.'
```

macOS

このガイドの macOS 用のコンポーネント例の一部は、hello-world-macos.yml という名前のコンポーネントドキュメントファイルを参照しています。これらの例を試すには、次のドキュメントを使用できます。

```
name: Hello World
description: This is hello world testing document for macOS.
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

アプリケーションの AWSTOE コンポーネント構文を AWSTOE ローカルで検証できます。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。例えば、`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。

Linux または macOS

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

Windows

```
awstoe.exe validate --documents C:\Users\user\Documents\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。たとえば、 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、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 amzn-s3-demo-destination-bucket --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する

```
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 build,test
```

のヘルプを取得する AWSTOE

```
awstoe --help
```

カスタム AWSTOE コンポーネントのコンポーネントドキュメントフレームワークを使用する

AWS Task Orchestrator and Executor （ AWSTOE) コンポーネントフレームワーク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
- コンポーネントロギング

- 入力チェーンと出力連鎖
- 文書スキーマと定義
- ドキュメントの例
- カスタムコンポーネントドキュメントでの変数の使用
- AWSTOEでの条件構造の使用
- AWSTOE コンポーネントドキュメントでの比較演算子の使用
- AWSTOE コンポーネントドキュメントでの論理演算子の使用
- AWSTOEでループ構文を使用する

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。例えば、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た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 および macOS: `/var/lib/amazon/toe/`
- Windows: `$env:ProgramFiles\Amazon\TaskOrchestratorAndExecutor\`

2. ファイルプレフィックス — "TOE_" 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。
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 がコンソールに AWSTOE 書き込むすべての標準出力 (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、連鎖式の解決 AWSTOE に適用された最適化を表します。

> **Note**
>
> ログフォルダには、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記述することで、入出力を連鎖させる機能を提供します。

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

or

```
{{ 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:
  - 無効な表現: `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。例えば、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
名前	文書の名前。	String	いいえ
説明	文書の説明。	String	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。	String	はい
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
名前	フェーズの名前。	String	はい
ステップ	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
名前	ステップのユーザー定義名。	String		
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。	String		
timeoutSeconds	ステップが失敗または再試行さ	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
	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	String	いいえ	中止

フィールド	説明	タイプ	必須	デフォルト値
	を超えた後 IgnoredFailure にステップを設定し、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	

ドキュメントの例

次の例は、ターゲットオペレーティングシステムのタスクを実行する AWSTOE コンポーネントドキュメントを示しています。

Linux

例 1: カスタムバイナリファイルを実行する

以下は、Linux インスタンスでカスタムバイナリファイルをダウンロードして実行するドキュメントの例です。

```
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>amzn-s3-demo-source-bucket</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 }}'
```

Windows

例 1: Windows 更新プログラムをインストールする

次のドキュメントの例では、利用可能な Windows 更新プログラムをすべてインストールし、設定スクリプトを実行し、AMI の作成前に変更を検証し、AMI の作成後に変更をテストします。

```
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}}'
```

例 2: Windows インスタンス AWS CLI に をインストールする

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 }}'
```

例 3: MSI インストーラ AWS CLI を使用して をインストールする

以下は、MSI インストーラ AWS CLI で をインストールするドキュメントの例です。

```
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 }}'
```

macOS

例 1: カスタム macOS バイナリファイルを実行する

次のドキュメントの例では、macOS インスタンスにカスタムバイナリファイルをダウンロードして実行します。

```
name: macOSBin
description: Download and run a binary file on macOS.
schemaVersion: 1.0
phases:
  - name: build
    steps:
      - name: Download
        action: S3Download
        inputs:
          - source: s3://<replaceable>amzn-s3-demo-source-bucket</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 }}'
```

カスタムコンポーネントドキュメントでの変数の使用

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。

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 {{ *MyConstant* }} で囲む必要があります。
- 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。例えば、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、参照先 - '{{ *MyConstant* }}' と - "{{ *MyConstant* }}" はどちらも有効です。

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 }}
```

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)。
type	はい	パラメータのデータ型。対応するタイプは string を含みます。
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 {{ *MyParameter* }} で囲む必要があります。
- 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。例えば、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、参照先 - '{{ *MyParameter* }}' と - "{{ *MyParameter* }}" はどちらも有効です。

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。

- ステップ入力のパラメータ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をオーバーライドする

の --parameters オプションをキーと値のペア AWS CLI とともに使用して、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
--parameters *name*=*value*,...	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
```

Systems Manager パラメータストアパラメータを使用する

変数の前に を付けることで、コンポーネントドキュメントで AWS Systems Manager Parameter Store パラメータ (SSM パラメータ) を参照できます`aws:ssm`。例えば、 などです

`{{ aws:ssm:/my/param }}` は SSM パラメータ の値に解決されます`/my/param`。

この機能は、次の SSM パラメータタイプをサポートしています。

- 文字列 – AWSTOE 文字列タイプにマッピングされます。
- StringList – タイプにマッピングします AWSTOE `stringList`。
- SecureString – AWSTOE 文字列タイプにマッピングします。

パラメータストアの詳細については、 AWS Systems Manager ユーザーガイドのAWS Systems Manager 「パラメータストア」を参照してください。

SSM パラメータ を使用してシー AWS Secrets Manager クレットを参照することもできますSecureString。例: {{ aws:ssm:/aws/reference/secretsmanager/test/test-secret }}。詳細については、「Parameter Store パラメータからのシークレットの参照 AWS Secrets Manager」を参照してください。

> **Important**
>
> Image Builder は、ログからSecureStringパラメータ解決を除外します。ただし、コンポーネントドキュメントで発行されたコマンドによって機密情報がログに記録されないようにする責任もあります。たとえば、安全な文字列で echo コマンドを使用すると、コマンドはプレーンテキストの値をログに書き込みます。

必要な IAM 許可

コンポーネントで Systems Manager パラメータを使用するには、インスタンスロールにパラメータリソース ARN のアクセスssm:GetParameter許可が必要です。例:

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": "ssm:GetParameter",
   "Resource": "arn:aws:ssm:*:111122223333:parameter/ImageBuilder-*"
  }
 ]
}
```

暗号化された値にアクセスするには、次のアクセス許可も必要です。

- カスタマーマネージドで暗号化されたSecureStringパラメータまたは AWS Secrets Manager 値kms:Decryptに を追加します AWS KMS key。

- Secrets Manager シークレットを参照`secretsmanager:GetSecretValue`する場合は、 を追加します。

コンポーネントドキュメントで SSM パラメータを参照する

次の例は、コンポーネント内の Systems Manager パラメータの Systems Manager パラメータストアパラメータを参照する方法を示しています。

```
name: UseSSMParameterVariable
description: This is a sample component document that prints out the value of an SSM
 Parameter. Never do this for a SecureString parameter.
schemaVersion: 1.0

phases:
  - name: verify
    steps:
      - name: EchoParameterValue
        action: ExecuteBash
        inputs:
          commands:
            - echo "Log SSM parameter name: /my/test/param, value {{ aws:ssm:/my/test/
param }}."
```

SSM パラメータの動的ランタイム変数解決

AWSTOE には、変数参照内で実行時に値を操作または変換するために使用できる以下の組み込み関数が用意されています。

関数の解決

`resolve` 関数は、別の変数参照内の変数参照を解決し、動的変数名参照を可能にします。これは、パラメータパスの一部が可変で、ドキュメントパラメータとして渡される可能性がある SSM パラメータを使用する場合に便利です。

`resolve` 関数は、SSM パラメータの名前部分の動的解決のみをサポートします。

構文

`dynamic_variable` 次の例では、 は SSM パラメータの名前を表し、次のいずれかである必要があります。

- SSM パラメータリファレンス (例: `aws:ssm:/my/param`)

- コンポーネントドキュメントのパラメータリファレンス (例: *parameter-name*)

```
{{ aws:ssm:resolve(dynamic_variable) }}
```

例: 実行時に SSM パラメータを解決する

次の例は、YAML コンポーネントドキュメントで resolve関数を使用する方法を示しています。

```
name: SsmParameterTest
description: This component verifies an SSM parameter variable reference with the echo
 command.
schemaVersion: 1.0

parameters:
  - parameter-name:
      type: string
      description: "test"

phases:
  - name: validate
    steps:
      - name: PrintDynamicVariable
        action: ExecuteBash
        inputs:
          commands:
            - echo "{{ aws:ssm:resolve(parameter-name) }}"
```

AWSTOEでの条件構造の使用

条件構造は、指定された条件式が true と false のどちらに評価されるかに基づいて、異なるアクションをコンポーネントドキュメントで実行します。if 構造を使用すると、コンポーネントドキュメントの実行フローを制御できます。

if 構造

if 構造を使用すると、ステップを実行する必要があるかどうかを評価できます。デフォルトでは、if 条件式が true に評価されると AWSTOE はステップを実行し、条件が false に評価されると AWSTOE はステップをスキップします。ステップがスキップされた場合でも、フェーズとドキュメントが正常に実行されたかどうかを AWSTOE が評価するときは、成功したステップとして扱われます。

> **Note**
>
> if ステートメントが評価されるのは 1 回だけです。これは、ステップが再起動をトリガーした場合でも同様です。再起動したステップは、その if ステートメントが既に評価されたことを認識し、中断した箇所から続行します。

構文

```
if:
  - <conditional expression>:
      [then: <step action>]
      [else: <step action>]
```

キー名	必要	説明
条件式	はい	条件式の最上位には、次のタイプの演算子を 1 つだけ含めることができます。 • 比較演算子 – 比較演算子のリスト、およびそれらが AWSTOE コンポーネントドキュメントでどのように機能するかについては、「」を参照してください比較演算子。 • 論理演算子 - 論理演算子には and、or、not があり、1 つ以上の比較演算子と共に動作します。AWSTOE コンポーネントドキュメントで論理演算子がどのように機能するかの詳

キー名	必要	説明
		細については、「論理演算子」を参照してください。 式が複数の条件を満たす必要がある場合は、論理演算子を使用して条件を指定します。
次に	いいえ	条件式が `true` に評価された場合に実行するアクションを定義します。
else	いいえ	条件式が `false` に評価された場合に実行するアクションを定義します。
ステップアクション	条件付き	`then` または `else` を使用するときは、次のステップアクションのいずれかを指定する必要があります。 • Abort - AWSTOE はステップを失敗としてマークします。 • Execute - AWSTOE はステップを実行します。 • Skip - AWSTOE はステップをスキップします。

例 1: パッケージをインストールする

AWSTOE コンポーネントドキュメントの以下のステップ例では、論理演算子を使用してパラメータ値をテストし、パッケージが解凍されている場合は、適切なパッケージマネージャーコマンドを実行してアプリケーションをインストールします。

```
- name: InstallUnzipAptGet
    action: ExecuteBash
    if:
      and:
          - binaryExists: 'apt-get'
          - not:
              binaryExists: 'unzip'
    inputs:
      commands:
          - sudo apt-get update
          - sudo apt-get install -y unzip

  - name: InstallUnzipYum
    action: ExecuteBash
    if:
      and:
          - binaryExists: 'yum'
          - not:
              binaryExists: 'unzip'
    inputs:
      commands:
          - sudo yum install -y unzip

  - name: InstallUnzipZypper
    action: ExecuteBash
    if:
      and:
          - binaryExists: 'zypper'
          - not:
              binaryExists: 'unzip'
    inputs:
      commands:
          - sudo zypper refresh
          - sudo zypper install -y unzip
```

例 2: ステップをスキップする

次の例は、ステップをスキップする 2 つの方法を示しています。1 つは論理演算子を使用し、もう 1 つは比較演算子と Skip ステップアクションを使用します。

```
# Creates a file if it does not exist using not
- name: CreateMyConfigFile-1
  action: ExecuteBash
  if:
    not:
      fileExists: '/etc/my_config'
  inputs:
    commands:
      - echo "Hello world" > '/etc/my_config'

# Creates a file if it does not exist using then and else
- name: CreateMyConfigFile-2
  action: ExecuteBash
  if:
    fileExists: '/etc/my_config'
    then: Skip
    else: Execute
  inputs:
    commands:
      - echo "Hello world" > '/etc/my_config'
```

AWSTOE コンポーネントドキュメントでの比較演算子の使用

Assert アクションモジュールや if 構造 を使用する条件式では、以下の比較演算子を使用できます。比較演算子には、stringIsEmpty など、単一の値に対して動作するものもあれば、ベースライン値を 2 番目の値 (変数値) と比較して、条件式が true と false のどちらに評価されるかを判定するものもあります。

比較が 2 つの値で動作する場合、2 番目の値は連鎖変数にすることができます。

異なるタイプの値を比較すると、次の値変換が比較前に発生する可能性があります。

- 数値比較の場合、変数値が文字列の場合、 は文字列を評価前の数値 AWSTOE に変換します。変換が不可能な場合、比較は false を返します。例えば、変数値が "1.0" の場合は変換が機能しますが、変数値が "a10" の場合は変換に失敗します。
- 文字列比較の場合、変数値が数値の場合、 は評価の前に文字列 AWSTOE に変換します。

文字列の比較

以下の比較演算子は文字列を対象として、値の比較、スペースや空の文字列かどうかのテスト、入力値と正規表現パターンの比較を行います。文字列比較では、大文字と小文字は区別されず、入力文字列の先頭または末尾のスペースはトリミングされません。

文字列比較演算子

- stringIsEmpty
- stringIsWhitespace
- stringEquals
- stringLessThan
- stringLessThanEquals
- stringGreaterThan
- stringGreaterThanEquals
- patternMatches

stringIsEmpty

`stringIsEmpty` 演算子は、指定された文字列に文字が含まれていない場合に `true` を返します。例:

```
# Evaluates to true
stringIsEmpty: ""

# Evaluates to false
stringIsEmpty: " "

# Evaluates to false
stringIsEmpty: "Hello."
```

stringIsWhitespace

`stringIsWhitespace` に指定された文字列にスペースのみが含まれているかどうかをテストします。例:

```
# Evaluates to true
stringIsWhitespace: "   "
```

```
# Evaluates to false
stringIsWhitespace: ""

# Evaluates to false
stringIsWhitespace: " Hello?"
```

stringEquals

`stringEquals` に指定された文字列が、`value` パラメータに指定された文字列と完全に一致するかどうかをテストします。例:

```
# Evaluates to true
stringEquals: 'Testing, testing...'
value: 'Testing, testing...'

# Evaluates to false
stringEquals: 'Testing, testing...'
value: 'Hello again.'

# Evaluates to false
stringEquals: 'Testing, testing...'
value: 'TESTING, TESTING....'

# Evaluates to false
stringEquals: 'Testing, testing...'
value: '   Testing, testing...'

# Evaluates to false
stringEquals: 'Testing, testing...'
value: 'Testing, testing...   '
```

stringLessThan

`stringLessThan` に指定された文字列が、`value` パラメータに指定された文字列より小さいかどうかをテストします。例:

```
# Evaluates to true
# This comparison operator isn't case sensitive
stringlessThan: 'A'
value: 'a'

# Evaluates to true - 'a' is less than 'b'
stringlessThan: 'b'
```

```
value: 'a'

# Evaluates to true
# Numeric strings compare as less than alphabetic strings
stringlessThan: 'a'
value: '0'

# Evaluates to false
stringlessThan: '0'
value: 'a'
```

stringLessThanEquals

`stringLessThanEquals` に指定された文字列が、`value` パラメータに指定された文字列以下であるかどうかをテストします。例:

```
# Evaluates to true - 'a' is equal to 'a'
stringLessThanEquals: 'a'
value: 'a'

# Evaluates to true - since the comparison isn't case sensitive, 'a' is equal to 'A'
stringLessThanEquals: 'A'
value: 'a'

# Evaluates to true - 'a' is less than 'b'
stringLessThanEquals: 'b'
value: 'a'

# Evaluates to true - '0' is less than 'a'
stringLessThanEquals: 'a'
value: '0'

# Evaluates to false - 'a' is greater than '0'
stringLessThanEquals: '0'
value: 'a'
```

stringGreaterThan

`stringGreaterThan` に指定された文字列が、`value` パラメータに指定された文字列より大きいかどうかをテストします。例:

```
# Evaluates to false - since the comparison isn't case sensitive, 'A' is equal to
 'a'
```

```
stringGreaterThan: 'a'
value: 'A'

# Evaluates to true - 'b' is greater than 'a'
stringGreaterThan: 'a'
value: 'b'

# Evaluates to true - 'a' is greater than '0'
stringGreaterThan: '0'
value: 'a'

# Evaluates to false - '0' is less than 'a'
stringGreaterThan: 'a'
value: '0'
```

stringGreaterThanEquals

`stringGreaterThanEquals` に指定された文字列が、`value` パラメータに指定された文字列以上であるかどうかをテストします。例:

```
# Evaluates to true - 'a' is equal to 'A'
stringGreaterThanEquals: 'A'
value: 'a'

# Evaluates to true - 'b' is greater than 'a'
stringGreaterThanEquals: 'a'
value: 'b'

# Evaluates to true - 'a' is greater than '0'
stringGreaterThanEquals: '0'
value: 'a'

# Evaluates to false - '0' is less than 'a'
stringGreaterThanEquals: 'a'
value: '0'
```

patternMatches

`value` パラメータで指定された文字列が、`patternMatches` で指定された正規表現パターンと一致するかどうかをテストします。比較には、RE2 構文に準拠する Golang regexp パッケージが使用されます。RE2 のルールの詳細については、GitHub の google / re2 リポジトリを参照してください。

次の例は、true を返すパターン一致を示しています。

```
patternMatches: '^[a-z]+$'
value: 'ThisIsValue'
```

数値の比較

以下の比較演算子は数値を対象として動作します。これらの演算子に指定する値は、YAML 仕様に従って、次のいずれかのタイプである必要があります。数値比較のサポートでは、golang の big パッケージの比較演算子 (func (*Float) Cmp など) が使用されます。

- 整数
- 浮動小数点数 (float64 に基づき、-1.7e+308 ~ +1.7e+308 の数値をサポート)
- 正規表現パターン ^[-+]?([0-9]+[.])?[0-9]+$ に一致する文字列。

数値比較演算子

- numberEquals
- numberLessThan
- numberLessThanEquals
- numberGreaterThan
- numberGreaterThanEquals

numberEquals

numberEquals に指定された数値が、value パラメータに指定された数値と等しいかどうかをテストします。次の比較の例はすべて true を返します。

```
# Values provided as a positive number
numberEquals: 1
value: 1

# Comparison value provided as a string
numberEquals: '1'
value: 1

# Value provided as a string
numberEquals: 1
```

```
value: '1'

# Values provided as floats
numberEquals: 5.0
value: 5.0

# Values provided as a negative number
numberEquals: -1
value: -1
```

numberLessThan

numberLessThan に指定された数値が、value パラメータに指定された数値より小さいかどうかをテストします。例:

```
# Evaluates to true
numberLessThan: 2
value: 1

# Evaluates to true
numberLessThan: 2
value: 1.9

# Evaluates to false
numberLessThan: 2
value: '2'
```

numberLessThanEquals

numberLessThanEquals に指定された数値が、value パラメータに指定された数値以下であるかどうかをテストします。例:

```
# Evaluates to true
numberLessThanEquals: 2
value: 1

# Evaluates to true
numberLessThanEquals: 2
value: 1.9

# Evaluates to true
numberLessThanEquals: 2
value: '2'
```

```
# Evaluates to false
numberLessThanEquals: 2
value: 2.1
```

numberGreaterThan

numberGreaterThan に指定された数値が、value パラメータに指定された数値より大きいかどうかをテストします。例:

```
# Evaluates to true
numberGreaterThan: 1
value: 2

# Evaluates to true
numberGreaterThan: 1
value: 1.1

# Evaluates to false
numberGreaterThan: 1
value: '1'
```

numberGreaterThanEquals

numberGreaterThanEquals に指定された数値が、value パラメータに指定された数値以上であるかどうかをテストします。例:

```
# Evaluates to true
numberGreaterThanEquals: 1
value: 2

# Evaluates to true
numberGreaterThanEquals: 1
value: 1.1

# Evaluates to true
numberGreaterThanEquals: 1
value: '1'

# Evaluates to false
numberGreaterThanEquals: 1
value: 0.8
```

ファイルのチェック

以下の比較演算子は、ファイルハッシュのチェックや、ファイルまたはフォルダが存在するかどうかの確認を行います。

ファイルとフォルダの演算子

- binaryExists
- fileExists
- folderExists
- fileMD5Equals
- fileSHA1Equals
- fileSHA256Equals
- fileSHA512Equals

binaryExists

現在のパスでアプリケーションが利用可能かどうかをテストします。例:

```
binaryExists: 'foo'
```

> **Note**
>
> Linux および macOS システムでは、アプリケーションの名前を *foo* とした場合、これは bash コマンド type ***foo*** >/dev/null 2>&1 と同じ動作になり、$? == 0 が比較の成功を示します。
> Windows システムでは、アプリケーションの名前を *foo* とした場合、これは PowerShell コマンド & C:\Windows\System32\where.exe /Q ***foo*** と同じ動作になり、$LASTEXITCODE = 0 が比較の成功を示します。

fileExists

指定されたパスにファイルが存在するかどうかをテストします。絶対パスまたは相対パスを指定できます。指定された場所が存在し、ファイルである場合、比較は `true` に評価されます。例:

```
fileExists: '/path/to/file'
```

> **Note**
>
> Linux および macOS システムでは、これは bash コマンド -d */path/to/file* と同じ動作になり、$? == 0 が比較の成功を示します。
> Windows システムでは、これは PowerShell コマンド Test-Path -Path '*C:\path\to\file*' -PathType 'Leaf' と同じ動作になります。

folderExists

指定されたパスにフォルダが存在するかどうかをテストします。絶対パスまたは相対パスを指定できます。指定された場所が存在し、フォルダである場合、比較は `true` に評価されます。例:

```
folderExists: '/path/to/folder'
```

> **Note**
>
> Linux および macOS システムでは、これは bash コマンド -d */path/to/folder* と同じ動作になり、$? == 0 が比較の成功を示します。
> Windows システムでは、これは PowerShell コマンド Test-Path -Path '*C:\path\to\folder*' -PathType 'Container' と同じ動作になります。

fileMD5Equals

ファイルの MD5 ハッシュが指定された値に等しいかどうかをテストします。例:

```
fileMD5Equals: '<MD5Hash>'
path: '/path/to/file'
```

fileSHA1Equals

ファイルの SHA1 ハッシュが指定された値に等しいかどうかをテストします。例:

```
fileSHA1Equals: '<SHA1Hash>'
path: '/path/to/file'
```

fileSHA256Equals

ファイルの SHA256 ハッシュが指定された値に等しいかどうかをテストします。例:

```
fileSHA256Equals: '<SHA256Hash>'
path: '/path/to/file'
```

fileSHA512Equals

ファイルの SHA512 ハッシュが指定された値に等しいかどうかをテストします。例：

```
fileSHA512Equals: '<SHA512Hash>'
path: '/path/to/file'
```

AWSTOE コンポーネントドキュメントでの論理演算子の使用

次の論理演算子を使用して、コンポーネントドキュメントの条件式を追加または変更できます。AWSTOE は、条件式を条件が指定された順序で評価します。コンポーネントドキュメントの比較演算子の詳細については、「AWSTOE コンポーネントドキュメントでの比較演算子の使用」を参照してください。

and

and 演算子を使用すると、2 つ以上の比較を 1 つの式として評価できます。リスト内のすべての条件が true になる場合、式は `true` に評価されます。それ以外の場合、式は `false` に評価されます。

例:

次の例では、文字列と数値の 2 つの比較を実行します。どちらの比較も true になるため、式は true に評価されます。

```
and:
  - stringEquals: 'test_string'
    value: 'test_string'
  - numberEquals: 1
    value: 1
```

次の例も 2 つの比較を実行します。最初の比較は false になるため、その時点で評価は停止し、2 番目の比較はスキップされます。式は `false` に評価されます。

```
and:
  - stringEquals: 'test_string'
    value: 'Hello world!'
```

```
  - numberEquals: 1
    value: 1
```

or

or 演算子を使用すると、2 つ以上の比較を 1 つの式として評価できます。指定された比較のいずれかが true になる場合、式は `true` に評価されます。指定された比較のいずれも `true` に評価されない場合、式は `false` に評価されます。

例:

次の例では、文字列と数値の 2 つの比較を実行します。最初の比較は true になるため、式は `true` に評価され、2 番目の比較はスキップされます。

```
or:
  - stringEquals: 'test_string'
    value: 'test_string'
  - numberEquals: 1
    value: 3
```

次の例も 2 つの比較を実行します。最初の比較は false になり、評価は続行されます。2 番目の比較は true になるため、式は `true` に評価されます。

```
or:
  - stringEquals: 'test_string'
    value: 'Hello world!'
  - numberEquals: 1
    value: 1
```

最後の例では、両方の比較が false になるため、式は `false` に評価されます。

```
or:
  - stringEquals: 'test_string'
    value: 'Hello world!'
  - numberEquals: 1
    value: 3
```

not

not 演算子を使用すると、1 つの比較を否定できます。比較が false になる場合、式は `true` に評価されます。比較が true になる場合、式は `false` に評価されます。

例:

次の例では、文字列比較を実行します。比較は false になるため、式は `true` に評価されます。

```
not:
  - stringEquals: 'test_string'
    value: 'Hello world!'
```

次の例も文字列比較を実行します。比較は true になるため、式は `false` に評価されます。

```
not:
  - stringEquals: 'test_string'
    value: 'test_string'
```

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コンストラクトを使用して式を AWSTOE 解決します。

参照式がループを含まないステップ内で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、end、および updateByの値を検証して、組み合わせが無限ループにならないようにします。

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	[Default] (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。	String	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	[Default] (デフォルト)
name	ループの一意の名前。同じ	String	いいえ	""

フィールド	説明	タイプ	必須	[Default] (デフォルト)
	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。			
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ために、 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	[Default] (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。	String	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。	String	はい	該当なし

フィールド	説明	タイプ	必須	[Default] (デフォルト)
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。	String	いいえ	カンマ: ","

> **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

次の例では、<phase_name>.<step_name>.[inputs | outputs].<var_name> という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。

```
- 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、 はステップの再試行ポリシー AWSTOE 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。再試行の最大回数が過ぎると、 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。再試行の最大回数が過ぎると、 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みなされますが、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"
}
```

例えば、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、失敗した反復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に役立ちます。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機能と、それらの設定方法について説明します。

コンポーネントは、プレーンテキストの YAML ドキュメントを使用して作成されます。ドキュメントの構文についてはカスタム AWSTOE コンポーネントのコンポーネントドキュメントフレームワークを使用するを参照。

> **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。

以下のクロスリファレンスは、実行されるアクションのタイプ別にアクションモジュールを分類したものです。

一般実行

- Assert (Linux、Windows、macOS)
- ExecuteBash (Linux、macOS)
- ExecuteBinary (Linux、Windows、macOS)
- ExecuteDocument (Linux、Windows、macOS)
- ExecutePowerShell (Windows)

ファイルダウンロードとアップロード

- S3Download (Linux、Windows、macOS)
- S3Upload (Linux、Windows、macOS)
- WebDownload (Linux、Windows、macOS)

ファイルシステムの操作

- AppendFile (Linux、Windows、macOS)
- CopyFile (Linux、Windows、macOS)
- CopyFolder (Linux、Windows、macOS)
- CreateFile (Linux、Windows、macOS)
- CreateFolder (Linux、Windows、macOS)
- CreateSymlink (Linux、Windows、macOS)
- DeleteFile (Linux、Windows、macOS)
- DeleteFolder (Linux、Windows、macOS)
- ListFiles (Linux、Windows、macOS)
- MoveFile (Linux、Windows、macOS)
- MoveFolder (Linux、Windows、macOS)
- ReadFile (Linux、Windows、macOS)
- SetFileEncoding (Linux、Windows、macOS)
- SetFileOwner (Linux、Windows、macOS)
- SetFolderOwner (Linux、Windows、macOS)
- SetFilePermissions (Linux、Windows、macOS)
- SetFolderPermissions (Linux、Windows、macOS)

ソフトウェアインストールアクション

- InstallMSI (Windows)
- UninstallMSI (Windows)

システムアクション

- Reboot (Linux、Windows)
- SetRegistry (Windows)

- UpdateOS (Linux、Windows)

汎用実行モジュール

以下のセクションでは、コマンドを実行したり、実行ワークフローを制御したりするアクションモジュールの詳細について説明します。

一般的な実行アクションモジュール

- Assert (Linux、Windows、macOS)
- ExecuteBash (Linux、macOS)
- ExecuteBinary (Linux、Windows、macOS)
- ExecuteDocument (Linux、Windows、macOS)
- ExecutePowerShell (Windows)

Assert (Linux、Windows、macOS)

Assert アクションモジュールは、比較演算子または論理演算子を入力として使用して、値の比較を実行します。演算子式の結果 (true または false) が、そのステップの全体的な成功または失敗ステータスを示します。

比較演算子または論理演算子式が `true` に評価された場合、ステップは `Success` としてマークされます。それ以外の場合、ステップは `Failed` としてマークされます。ステップが失敗した場合は、`onFailure` パラメータによってステップの結果が決定されます。

Input

キー名	説明	タイプ	必須
`input`	比較演算子または論理演算子を 1 つ含みます。論理演算子には複数の比較演算子を含めることができます。	演算子に応じて可変	はい

入力の例: **`stringEquals`** 比較演算子を使用した単純な比較

この例は true に評価されます。

```
- name: StringComparison
  action: Assert
  inputs:
    stringEquals: '2.1.1'
    value: '{{ validate.ApplicationVersion.outputs.stdout }}'
```

入力の例: **patternMatches** 比較演算子を使用した正規表現による比較

これらの例はすべて true に評価されます。

```
- name: Letters only
  action: Assert
  inputs:
    patternMatches: '^[a-zA-Z]+$'
    value: 'ThisIsOnlyLetters'

- name: Letters and spaces only
  action: Assert
  inputs:
    patternMatches: '^[a-zA-Z\s]+$'
    value: 'This text contains spaces'

- name: Numbers only
  action: Assert
  inputs:
    patternMatches: '^[0-9]+$'
    value: '1234567890'
```

入力例: 論理演算子と連鎖変数を使用してネストされた比較

次の例は、連鎖変数との比較を使用する論理演算子を含む、ネストされた比較を示しています。Assert は、次のいずれかが true の場合に true に評価されます。

- ApplicationVersion は 2.0 より大きく、CPUArchitecture は arm64 に等しくなります。
- CPUArchitecture は x86_64 に等しくなります。

```
- name: NestedComparisons
  action: Assert
  inputs:
```

```
    or: # <- first level deep
      - and: # <- second level deep
          - numberGreaterThan: 2.0 # <- third level deep
            value: '{{ validate.ApplicationVersion.outputs.stdout }}'
          - stringEquals: 'arm64'
            value: '{{ validate.CPUArchitecture.outputs.stdout }}'
      - stringEquals: 'x86_64'
        value: '{{ validate.CPUArchitecture.outputs.stdout }}'
```

出力:

Assert の出力は、ステップの成功または失敗を示します。

ExecuteBash (Linux、macOS)

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/コマンドで 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。

ExecuteBash モジュールは、スクリプトが終了コード 194 で終了した場合、システムの再起動を処理します。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

キー名	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	リスト	はい

キー名	説明	タイプ	必須
	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。		

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
```

Output

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

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 (Linux、Windows、macOS)

ExecuteBinary アクションモジュールを使うと、コマンドライン引数のリストを使ってバイナリファイルを実行することができます。

ExecuteBinary モジュールは、バイナリファイルが終了コード 194 (Linux) または 3010 (Windows) で終了した場合に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

キー名	説明	タイプ	必須
path	実行用のバイナリファイルへのパス。	String	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
```

Output

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

出力例

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

ExecuteDocument (Linux、Windows、macOS)

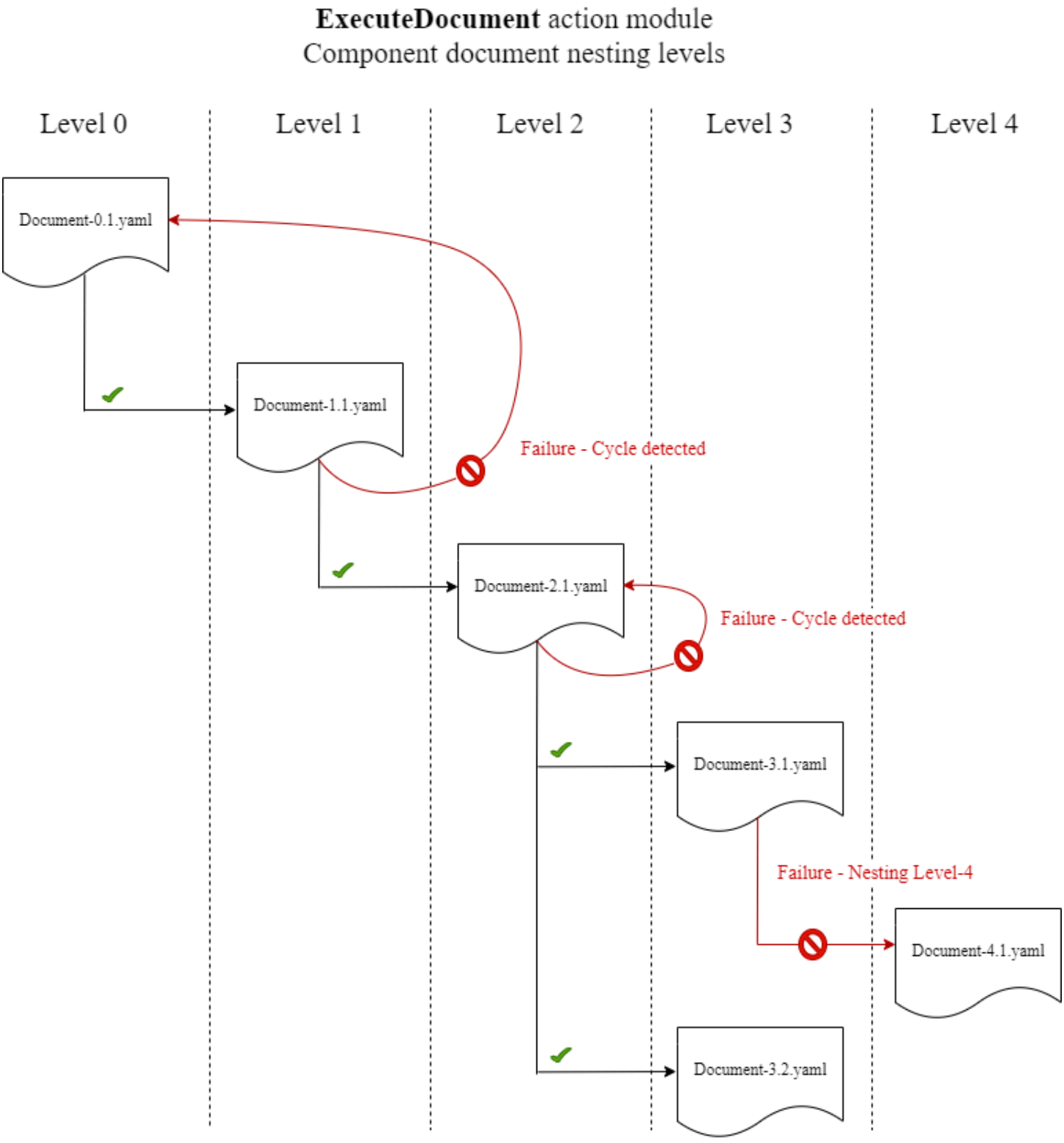
ExecuteDocument アクションモジュールは、ネストされたコンポーネントドキュメントをサポートし、1 つのドキュメントから複数のコンポーネントドキュメントを実行できるようにします。AWSTOE 実行時に入力パラメータで渡されたドキュメントを検証します。

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しておくと、 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)

キー名	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	String	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勧めです。)	String	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	String	いいえ

キー名	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

キー名	説明	タイプ	必須
name	ExecuteDocument アクションモジュールが実行している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。	String	はい
value	入力パラメータの値。	String	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: '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
```

Output

AWSTOE は、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では、outputs で実行時の簡単な概要がフィールドに表示され、detailedOutput でアクションモジュールで実行されるフェーズ、ステップ、ドキュメントの詳細も表示されます。

```
{
 \"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":""
- "ignoredFailedStepCount":0
- "logUrl":""
- "status":"success"

出力例

次の例は、ネストされた実行が発生した場合の 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 (Windows)

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 PowerShell をサポートしています。

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 (例えば、`input.ps1`) に変換され、Windows PowerShell を使用して実行されます。シェルファイルを実行した結果が終了コードです。

ExecutePowerShell モジュールは、シェルコマンドが終了コードが `3010` で終了した場合、システムの再起動を処理します。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、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

キー名	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ています。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ています。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。	String	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)
```

Output

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

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

以下のセクションでは、ファイルをアップロードまたはダウンロードするアクションモジュールの詳細について説明します。

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

- S3Download (Linux、Windows、macOS)

- S3Upload (Linux、Windows、macOS)
- WebDownload (Linux、Windows、macOS)

S3Download (Linux、Windows、macOS)

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 がそのファイルを上書きします。

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。キー接頭辞がフォー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/**。

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 バケット/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

キー	説明	タイプ	必須	[Default] (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。	String	はい	該当なし
`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	String	はい	該当なし

キー	説明	タイプ	必須	[Default] (デフォルト)
	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* 。			
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。	String	いいえ	該当なし

キー	説明	タイプ	必須	[Default] (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既に存在する場合、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: `Error: S3Download: File already exists and "overwrite" property for "destination" file is set`	ブール値	いいえ	真

キー	説明	タイプ	必須	[Default] (デフォルト)
	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。例えば、*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://amzn-s3-demo-source-bucket/path/to/package.zip
      destination: C:\myfolder\package.zip
      expectedBucketOwner: 123456789022
      overwrite: false
    - source: s3://amzn-s3-demo-source-bucket/path/to/package.zip
      destination: C:\myfolder\package.zip
      expectedBucketOwner: 123456789022
      overwrite: true
    - source: s3://amzn-s3-demo-source-bucket/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する

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://amzn-s3-demo-source-bucket/path/to/*
      destination: C:\myfolder\
      expectedBucketOwner: 123456789022
      overwrite: false
    - source: s3://amzn-s3-demo-source-bucket/path/to/*
      destination: C:\myfolder\
      expectedBucketOwner: 123456789022
      overwrite: true
    - source: s3://amzn-s3-demo-source-bucket/path/to/*
      destination: C:\myfolder\
      expectedBucketOwner: 123456789022
```

Output

なし。

S3Upload (Linux、Windows、macOS)

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

キー	説明	タイプ	必須	[Default] (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。	String	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。	String	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。	String	いいえ	`false`
`expectedBucketOwner`	宛先パスで指定されている Amazon S3 バケットの予想所有者アカウント	String	いいえ	該当なし

キー	説明	タイプ	必須	[Default] (デフォルト)
	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: ローカルファイルをAmazon S3 オブジェクトにコピーする

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://amzn-s3-demo-destination-bucket/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 バケットにコピーする

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://amzn-s3-demo-destination-bucket/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://amzn-s3-demo-destination-bucket/path/to/
      recurse: true
      expectedBucketOwner: 123456789022
```

Output

なし。

WebDownload (Linux、Windows、macOS)

WebDownload アクションモジュールを使用すると、HTTP/HTTPS プロトコル (HTTPS を推奨) を介してリモートの場所からファイルやリソースをダウンロードできます。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エクスポネンシャル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

キー名	説明	タイプ	必須	[Default] (デフォルト)
`source`	RFC 3986 標準に準拠した有効な HTTP/HTTPS	String	はい	該当なし

キー名	説明	タイプ	必須	[Default] (デフォルト)
	URL (HTTPS を推奨)。式を連鎖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の絶対パスまたは相対パス。フォルダパスは / 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。	String	はい	該当なし

キー名	説明	タイプ	必須	[Default] (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

キー名	説明	タイプ	必須	[Default] (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。	String	いいえ	該当なし

キー名	説明	タイプ	必須	[Default] (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。	String	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

Output

キー名	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	String

キー名	説明	[Type] (タイプ)				
	改行文字で区切られた文字列。					

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

以下のセクションでは、ファイルシステム操作を実行するアクションモジュールの詳細について説明します。

ファイルシステム操作アクションモジュール

- AppendFile (Linux、Windows、macOS)
- CopyFile (Linux、Windows、macOS)
- CopyFolder (Linux、Windows、macOS)
- CreateFile (Linux、Windows、macOS)
- CreateFolder (Linux、Windows、macOS)
- CreateSymlink (Linux、Windows、macOS)
- DeleteFile (Linux、Windows、macOS)
- DeleteFolder (Linux、Windows、macOS)
- ListFiles (Linux、Windows、macOS)
- MoveFile (Linux、Windows、macOS)
- MoveFolder (Linux、Windows、macOS)
- ReadFile (Linux、Windows、macOS)
- SetFileEncoding (Linux、Windows、macOS)
- SetFileOwner (Linux、Windows、macOS)
- SetFolderOwner (Linux、Windows、macOS)
- SetFilePermissions (Linux、Windows、macOS)
- SetFolderPermissions (Linux、Windows、macOS)

AppendFile (Linux、Windows、macOS)

AppendFile アクションモジュールは、指定された内容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エンコーディング(`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

キー名	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。	String	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。	String	いいえ	空の文字列	該当なし	はい
encoding	エンコード形式です。	String	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	はい

キー名	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
					文字と小文字を区別しません。	

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
```

Output

なし。

CopyFile (Linux、Windows、macOS)

CopyFile 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。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

キー名	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。	String	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。	String	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

キー名	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
	既にある場合でも、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
```

Output

なし。

CopyFolder (Linux、Windows、macOS)

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内に指定され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、デフォルトで、既存のフォルダを上書きします。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。

コピー先フォルダ名がソースフォルダ名と異なる場合は、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名として扱われます。ソースフォルダと同じフォルダ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

キー名	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	ソースフォルダのパス。	String	はい	該当なし	該当なし	はい
destination	宛先フォルダのパス。	String	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既にある場合、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: フォルダのコピー (Linux)

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

入力例: フォルダのコピー (Windows)

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

入力例: ソースフォルダ名を使用してフォルダをコピーする (Linux)

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

入力例: ソースフォルダ名を使用してフォルダをコピーする (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
```

Output

なし。

CreateFile (Linux、Windows、macOS)

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に既に存在する場合、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。

ファイルエンコーディング値がデフォルトのエンコーディング (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

キー名	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。	String	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。	String	いいえ	該当なし	該当なし	はい
encoding	エンコード形式です。	String	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	はい

キー名	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。	String	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。	String	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。	String	いいえ	0666	該当なし	Windows ではサポートされていません。

キー名	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 false に設定すると、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するを参照してください。

Output

なし。

CreateFolder (Linux、Windows、macOS)

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に既にフォルダが存在する場合、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

キー名	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。	String	はい	該当なし	該当なし	はい
`owner`	ユーザー名または ID。	String	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。

キー名	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
group	グループ名または ID。	String	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。	String	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 false に設定すると、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: フォルダの作成 (Linux)

```
- name: CreatingFolderLinux
  action: CreateFolder
```

```
    inputs:
      - path: /Sample/MyFolder/
```

入力例: フォルダの作成 (Windows)

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

入力例: フォルダのプロパティを指定してフォルダの作成

```
- 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 (ある場合) を上書きするフォルダを作成します。

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

Output

なし。

CreateSymlink (Linux、Windows、macOS)

CreateSymLink アクションモジュールは、シンボリックリンク、つまり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既に存在する場合、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既に存在する。
- 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

キー名	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。	String	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が指すターゲットファイルのパス。	String	はい	該当なし	該当なし	Windows ではサポートされていません。
force	同じ名前のリンクが既に存在する場合、リ	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。

キー名	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
	ンクの作成を強制します。					

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
```

Output

なし。

DeleteFile (Linux、Windows、macOS)

DeleteFile アクションモジュールは、指定された場所にある 1 つまたは複数の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

キー名	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。	String	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\*
```

Output

なし。

DeleteFolder (Linux、Windows、macOS)

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

キー名	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。	String	はい	該当なし	該当なし	はい
force	フォルダが空であろうとなかろうと、フォルダを削除します。	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: フォルダの削除 (Linux)

```
- name: DeletingFolderWithOutForceLinux
  action: DeleteFolder
  inputs:
```

```
      - path: /Sample/MyFolder/Sample/
```

入力例: フォルダの削除 (Windows)

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

Output

なし。

ListFiles (Linux、Windows、macOS)

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 `true` に設定すると、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

キー名	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。	String	はい	該当なし	該当なし	はい

キー名	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
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。	String	いいえ	該当なし	該当なし	はい
recursive	フォルダ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
```

Output

キー名	説明	[Type] (タイプ)
`files`	ファイルのリストです。	String

出力例

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

MoveFile (Linux、Windows、macOS)

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。

指定したフォルダーに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。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、以下の場合に工ラーを返します。

- 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

キー名	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。	String	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。	String	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既にある場合でも、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
```

Output

なし。

MoveFolder (Linux、Windows、macOS)

MoveFolder 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を再帰的に作成します。

ソースフォルダと同じフォルダが宛先フォルダに既に存在する場合、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を移動する場合、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。

コピー先フォルダ名がソースフォルダ名と異なる場合は、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名として扱われます。ソースフォルダと同じフォルダ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

キー名	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	ソースフォルダのパス。	String	はい	該当なし	該当なし	はい
destination	宛先フォルダのパス。	String	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既にある場合、保存先フォル	ブール値	いいえ	true	該当なし	はい

キー名	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
	ダは置き換えられません。					

入力例: フォルダの移動 (Linux)

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

入力例: フォルダの移動 (Windows)

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

入力例: ソースフォルダ名を使用してフォルダを移動する (Linux)

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

入力例: ソースフォルダ名を使用してフォルダを移動する (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
```

Output

なし。

ReadFile (Linux、Windows、macOS)

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エンコーディング (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

キー名	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。	String	はい	該当なし	該当なし	はい	
encoding	エンコード形式です。	String	いいえ	utf8	utf8、utf-LE、utf-16LE 、utf16-	はい	tf-16、

キー名	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、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。	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
```

Output

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

出力例

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

SetFileEncoding (Linux、Windows、macOS)

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

キー名	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。	String	はい	該当なし	該当なし	はい
encoding	エンコード形式です。	String	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
```

Output

なし。

SetFileOwner (Linux、Windows、macOS)

SetFileOwner アクションモジュールは、owner と group 既存のファイルのプロパティと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ー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

キー名	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。	String	はい	該当なし	該当なし	Windows ではサポー

キー名	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。	String	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
```

Output

なし。

SetFolderOwner (Linux、Windows、macOS)

SetFolderOwner アクションモジュールは、既存のフォルダのプロパティと `owner` 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ー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

キー名	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。	String	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。	String	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

キー名	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
```

Output

なし。

SetFilePermissions (Linux、Windows、macOS)

SetFilePermissions アクションモジュールは、既存のファイルの内容の `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

キー名	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。	String	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。	String	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
```

Output

なし。

SetFolderPermissions (Linux、Windows、macOS)

SetFolderPermissions アクションモジュールは、permissions の既存のフォルダとそのすべてのサブファイルおよびサブフォルダ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

キー名	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。	String	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。	String	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。

キー名	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
```

Output

なし。

ソフトウェアインストールアクション

以下のセクションでは、ソフトウェアをインストールまたはアンインストール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

- InstallMSI (Windows)
- UninstallMSI (Windows)

InstallMSI (Windows)

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

キー名	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可 • 有効な S3 オブジェクト URI。 • RFC 3986 標準に従った有効なウェ	String	はい	該当なし	該当なし

キー名	説明	タイプ	必須	デフォルト値	許容値
	ブ HTTP/ HTTPS URL (HTTPS を推奨) 。 チェーン式は許可されています。				

キー名	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 — 再起動がスキップされ	String	いいえ	Allow	Allow, Force, Skip

キー名	説明	タイプ	必須	デフォルト値	許容値
	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。				

キー名	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。	String	いいえ	*VX	i,w,e,a,r,u,c,m,o,p,v,x,+,!,*

キー名	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。	String	いいえ	該当なし	該当なし

キー名	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

キー名	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み実行され	ブール値	いいえ	false	true, false

キー名	説明	タイプ	必須	デフォルト値	許容値
	ます。これがデフォルトの動作です。				

キー名	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名が必要で	ブール値	いいえ	false	true, false

キー名	説明	タイプ	必須	デフォルト値	許容値
	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
```

Output

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": ""
}
```

UninstallMSI (Windows)

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

キー名	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)。	String	はい	該当なし	該当なし

キー名	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。				

キー名	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 — 再起動がスキップされ	String	いいえ	Allow	Allow, Force, Skip

キー名	説明	タイプ	必須	デフォルト値	許容値
	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。				

キー名	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。	String	いいえ	*VX	i,w,e,a,r,u,c,m,o,p,v,x,+,!,*

キー名	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。	String	いいえ	該当なし	該当なし

キー名	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

キー名	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み実行され	ブール値	いいえ	false	true, false

キー名	説明	タイプ	必須	デフォルト値	許容値
	ます。これがデフォルトの動作です。				

キー名	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名が必要で	ブール値	いいえ	false	true, false

キー名	説明	タイプ	必須	デフォルト値	許容値
	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
```

Output

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

以下のセクションでは、システムアクションを実行したり、システム設定を更新したり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

- Reboot (Linux、Windows)
- SetRegistry (Windows)
- UpdateOS (Linux、Windows)

Reboot (Linux、Windows)

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ーエージェントは、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
- 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、reboot exitcode 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

キー名	説明	タイプ	必須	[Default] (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 (Windows)

SetRegistry アクションモジュールは入力のリストを受け入れ、指定し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

キー名	説明	タイプ	必須
path	レジストリキーのパス。	String	はい
name	レジストリキーの名前。	String	はい

キー名	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。	String	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 (Linux、Windows)

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

キー名	説明	タイプ	必須
include		文字列リスト	いいえ

キー名	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。		

キー名	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効率化するには、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すべての値を上書きします。
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 Logs からのロギング障害を無視します。
- CWLogGroup (文字列) — CloudWatch Logs LogGroup 名。
- cwLogRegion (文字列) – CloudWatch Logs に適用される AWS リージョン。
- cwLogStream (文字列) – CloudWatch Logs LogStreamの名前。console.logファイルをストリーミングする AWSTOE 場所を指定します。
- 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 コマンドが実行されている 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 のコンポーネントビルドバージョン 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、カスタムコンポーネントドキュメントでの変数の使用 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親ディレクトリ `TOE_<DATETIME>_<EXECUTIONID>` の中にあります。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 という名前の S3 バケット AWSTOE 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>"
}
```

Image Builder の出力イメージリソース

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 コマンドを使用したリソースのタグ付けの詳細については 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、イメージを一覧表示、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、「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の管理」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 Reference の ListImages 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。

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、Linux、macOS) 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。

- Filter source を使用して、仮想マシン (VMIE) または ISO ディスクイメージ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。

Type

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

イメージリソースのオペレーティングシステムプラットフォーム。例えば、「Windows」、「Linux」、または「macOS」になります。

[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、Linux、macOS) 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。
- Filter source を使用して、仮想マシン (VMIE) または ISO ディスクイメージ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。

Type

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

イメージリソースのオペレーティングシステムプラットフォームのバージョン。通常は <major>.<minor>.<patch> という形式の数値フィールドです。

[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

イメージリソースのオペレーティングシステムプラットフォーム。例えば、「Windows」、「Linux」、または「macOS」になります。

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、Linux、macOS) 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。
- Filter source を使用して、仮想マシン (VMIE) または ISO ディスクイメージ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。

Type

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

イメージリソースのオペレーティングシステムプラットフォームのバージョン。通常は <major>.<minor>.<patch> という形式の数値フィールドです。

プラットフォーム

イメージリソースのオペレーティングシステムプラットフォーム。例えば、「Windows」、「Linux」、または「macOS」になります。

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"
  },
```

```
  {
    "arn": "arn:aws:imagebuilder:us-west-2:123456789012:image/image-recipe-
macos/1.1.1",
    "name": "image-recipe-macos",
    "type": "AMI",
    "version": "1.1.1",
    "platform": "macOS",
    "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。タブでは、コンソールのステップと AWS CLI コマンドについて説明します。

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」リファレンスの「ListWaitingWorkflowSteps」 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 Reference の ListImageBuildVersions を参照すること。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。例えば、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参照バージョンの 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、指定され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。

イメージリソースの詳細の表示

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。

別の が AWS Resource Access Manager （ AWS RAM) リソース共有を通じて 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
- からイメージポリシーの詳細を取得する 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。タブがイメージに適用されない場合、タブは非アクティブになり、データが表示されません。

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。例えば、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。例えば、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、「Image Builder の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 キーペア
- 通知用の Amazon SNS トピック

詳細については、「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する AWS アカウントのカンマ区切り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です。

> **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に課金されます。

- 高速起動設定のターゲットアカウント – EC2 Fast Launch AWS アカウント が起動用に事前プロビジョニングされたスナップショットを配布する 。
- 関連付けられたライセンス設定 – 指定されたリージョンの AMI に関連付けられている License Manager ライセンス設定 ARNs。
- 起動テンプレート設定 – 特定のアカウントに使用する Amazon EC2 起動テンプレートを識別します。
- 起動テンプレートのデフォルトバージョンを設定する – 指定された Amazon EC2 起動テンプレートを、指定された のデフォルトの起動テンプレートとして設定します AWS アカウント。

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、「Image Builder の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、最初にビルドまたはインポートワークフローの概要とワークフローステップをデフォルトで表示します。ただし、ワークフローフィルタには、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、リストから を選択します。

AMI 出力を生成するイメージワークフローには、ビルド、インポート、またはテストのワークフローを含めることができます。コンテナ出力を生成するコンテナワークフローには、ビルド、テスト、またはディストリビューションワークフローを含めることができます。

> **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、ビルドワークフローとテストワークフローの両方が設定されたイメージビルドが開始したばかりの場合、ビルドワークフローはリストに表示される唯一のワークフロータイプです。テストワークフローが開始されると、Image Builder はそれをリストに追加します。

ワークフローフィルタに続いて、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
- スキップ済み
- 実行中
- 完了
- 失敗
- ロールバック中

- 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。

成功したステップ

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。

失敗したステップ

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
- Medium
- 高
- [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 検出結果の脆弱性情報のソース。

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。

からイメージポリシーの詳細を取得する 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
```

Image Builder を使用したカスタムイメージの作成

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。たとえば、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か、ビルドパイプラインを実行してイメージを作成することもできます。API アクションに関連する SDK リクエストについては、「EC2 Image Builder API」リファレンス」で、目的のコマンドの「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、パイプラインウィザード: AMI の作成 または パイプラインウィザード: コンテナイメージの作成 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

  イメージレシピリソースの 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 - 出力イメージリソースにタグを追加するには、`--tags` パラメータを使用します。
- 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",
        ...
    }
]
}
```

からイメージの作成をキャンセルする 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
```

Image Builder での仮想マシン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、そのプロセスは VM の環境、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を作成します。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。エクスポートするには、カスタムイメージビルドからの出力として VM ディスクファイルを作成し、ファイルを配布します。

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。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。

Image Builder への VM 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。

Console

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. インポートダイアログを開くには、イメージのインポートを選択します。
4. 次の一般情報を入力します。
   - イメージに一意の名前を指定します。
   - 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*
5. インポートタイプを選択します: VM import。
6. [イメージをインポート] ページで、以下の各セクションに詳細を入力します。完了したら、イメージのインポート] を選択します。

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

1. 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。

   a. 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。
   b. 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。
   c. 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。

2. 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。

3. 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。

Encryption

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。

AWS CLI

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 コマンドを合理化するために 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 オペレーションリクエストパラメータに指定されたパターンに従います。API オペレーション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 オペレーション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、Image Builder import-vm-image コマンドに指定します。

   以下に、こ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 (<major>.<minor>.<patch>) を示す数値が付いています。例えば、`1.0.0`。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、Image Builder での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。
- 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",
    "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://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS ではフォーワードスラッシュ (/) が使用されます。

からイメージビルドから VM ディスクを配布する AWS CLI

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、「から出力 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。

Image Builder を使用して検証済みの Windows ISO ディスクイメージをインポートする

Windows オペレーティングシステム ISO ファイルは、Windows オペレーティングシステムの特定のバージョンの完全なインストールパッケージを含むディスクイメージファイルです。Microsoft では、公式の Windows オペレーティングシステム ISO ファイルをウェブサイトから直接、または認定リセラーを通じてダウンロードできます。潜在的なマルウェアや不正なバージョンを避けるため、信頼できる正当なソースから ISO ファイルを取得することが重要です。

EC2 Image Builder は、`build-image-from-iso`インポートワークフローを使用して ISO ディスクファイルをインポートし、そこからセカンダリボリュームを作成します。設定が完了すると、Image Builder はインポートから作成したボリュームのスナップショットを取得し、それを使用して Amazon マシンイメージ (AMI) を作成します。

ISO ディスクイメージのインポートでサポートされているオペレーティングシステム

Image Builder は、次の Windows オペレーティングシステムの ISO ディスクイメージをサポートしています。

- Windows 11 Enterprise バージョン 24H2
- Windows 11 Enterprise バージョン 23H2
- Windows 11 Enterprise バージョン 22H2

Image Builder は、次の Windows オペレーティングシステムの ISO ディスクイメージをサポートしていません。

- 長期サービスチャネル (LTSC) イメージ
- Windows Media Creation Tool から作成された ISO ディスクイメージ
- 評価イメージ

ISO ディスクイメージをインポートするための前提条件

ISO ディスクイメージをインポートするには、まず以下の前提条件を満たす必要があります。

- ディスクイメージのオペレーティングシステムは、Image Builder がサポートするオペレーティングシステムである必要があります。サポートされているオペレーティングシステムのリストについては、「」を参照してくださいISO ディスクイメージのインポートでサポートされているオペレーティングシステム。
- ISO イメージをインポートできるようにするには、Microsoft 365 管理センターからダウンロードします。
- インポートプロセスを実行する前に、インポートを実行する同じ AWS アカウント と AWS リージョン の Amazon S3 に ISO ディスクファイルをアップロードする必要があります。
- ファイル拡張子はインポートプロセスでは大文字と小文字が区別され、 である必要があります.ISO。ファイル拡張子が小文字の場合は、次のいずれかのコマンドを実行して名前を変更できます。

  Command

  ```
  aws s3 cp s3://amzn-s3-demo-bucket/Win11_24H2_English.iso s3://amzn-s3-demo-bucket/Win11_24H2_English.ISO
  ```

  PowerShell

  ```
  Copy-S3Object -BucketName amzn-s3-demo-bucket -Key Win11_24H2_English.iso -DestinationKey Win11_24H2_English.ISO
  ```

- Microsoft ライセンスはインポートに自動的に含まれません。独自のライセンス (BYOL) を持参する必要があります。Microsoft ソフトウェアのライセンスの詳細については、「Amazon Web Services のライセンス」および「Microsoft のよくある質問」ページを参照してください。
- インポートプロセスでは、次のように 2 つの異なる IAM ロールを使用します。

  実行ロール

  このロールは、Image Builder が AWS のサービス ユーザーに代わって を呼び出すアクセス許可を付与します。実行ロールに必要なアクセス許可を含むAWSServiceRoleForImageBuilderサービスにリンクされたロールを指定するか、独自のロールを作成できます。

  インスタンスプロファイルロール

  このロールは、サービスが EC2 インスタンスで実行するアクションのアクセス許可を付与します。インフラストラクチャ設定リソースでインスタンスプロファイルロールを指定できます。次の管理ポリシーをインスタンスプロファイルロールにアタッチして、インポートプロセスに必要なすべてのアクセス許可があることを確認します。

  - EC2InstanceProfileForImageBuilder
  - AmazonSSMManagedInstanceCore

  詳細については、「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。

ISO ディスクイメージを Image Builder にインポートする

インポートプロセスを開始する前に、すべての を満たしていることを確認してください前提条件。

インポートプロセスでは、イメージに次のソフトウェアとドライバーが追加されます。

- EC2Launch v2
- AWS Systems Manager エージェント
- AWS NVMe ドライバー
- AWS ENA ネットワークドライバー
- AWS PCI シリアルドライバー
- EC2 Windows ユーティリティ

Console

Image Builder コンソールで ISO ディスクイメージ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. インポートダイアログを開くには、イメージのインポートを選択します。
4. 次の一般情報を入力します。

   - イメージに一意の名前を指定します。
   - 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*
5. インポートタイプを選択します: ISO import。
6. 次の ISO インポート設定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。

   - S3 URI – ISO ディスクファイルが保存されている場所を入力します。ファイルを参照するには、S3 を参照する」を選択します。
   - IAM ロール – IAM ロールを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、新しい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。

     AWSServiceRoleForImageBuilder サービスにリンクされたロールを指定するか、サービスアクセス用の独自のカスタムロールを指定できます。
7. オプションで、Image Builder イメージリソースにタグを追加できます。これにより、AMI にタグは追加されません。
8. ISO インフラストラクチャ設定は、Image Builder がインポートプロセスをホストするために起動するインスタンスの設定を定義します。Image Builder が作成するインフラストラクチャ設定は、サービスのデフォルトに基づいて使用することも、既存のインフラストラクチャ設定を使用することもできます。詳細については、「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。

   新しいインフラストラクチャ設定 を作成するには、「インフラストラクチャ設定の作成」を選択します。これにより、別のタブで開きます。新しいリソースの作成が完了したら、インポート設定に戻り、既存のインフラストラクチャ設定を使用するを選択します。

9. インポートプロセスを開始するには、イメージのインポートを選択します。

インポートが完了すると、所有しているイメージのリストにイメージが表示されます。詳細については、「イメージを一覧表示する」を参照してください。

AWS CLI

この例では、ISO ディスクファイルからイメージをインポートし、 を使用してそのファイルから AMI を作成する方法を示します AWS CLI。

以下に、こ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 (<major>.<minor>.<patch>) を示す数値が付いています。例えば、`1.0.0`。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、Image Builder での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。
- executionRole (文字列) – Image Builder に Microsoft ISO ファイルからイメージをインポートするワークフローアクションを実行するためのアクセス権を付与する IAM ロールの名前または Amazon リソースネーム (ARN)。AWSServiceRoleForImageBuilder サービスにリンクされたロールを指定するか、サービスアクセス用の独自のカスタムロールを指定できます。
- platform (文字列、必須) — ISO ディスク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。有効な値には `Windows` が含まれます。
- osVersion (文字列、必須) — ISO ディスクイメージのオペレーティングシステムのバージョン。有効な値には `Microsoft Windows 11` が含まれます。
- infrastructureConfigurationArn (文字列、必須) – ISO イメージが構築されている EC2 インスタンスの起動に使用されるインフラストラクチャ設定リソースの Amazon リソースネーム (ARN)。
- uri (文字列、必須) – Amazon S3 に保存されている ISO ディスクファイルの URI。

```
aws imagebuilder import-disk-image \
    --name "example-iso-disk-import" \
    --semantic-version "1.0.0" \
    --description "Import an ISO disk image" \
```

```
    --execution-role "AWSServiceRoleForImageBuilder" \
    --platform "Windows" \
    --os-version "Microsoft Windows 11" \
    --infrastructure-configuration-arn "arn:aws:imagebuilder:us-
east-1:111122223333:infrastructure-configuration/example-infrastructure-
configuration-123456789abc",
    --uri: "s3://amzn-s3-demo-source-bucket/examplefile.iso"
```

インポートが完了すると、所有しているイメージのリストにイメージが表示されます。詳細については、「イメージを一覧表示する」を参照してください。

PowerShell

この例では、ISO ディスクファイルからイメージをインポートし、PowerShell を使用してそこから AMI を作成する方法を示します。

以下に、こ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 (<major>.<minor>.<patch>) を示す数値が付いています。例えば、`1.0.0`。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、Image Builder での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。
- executionRole (文字列) – Image Builder に Microsoft ISO ファイルからイメージをインポートするワークフローアクションを実行するためのアクセス権を付与する IAM ロールの名前または Amazon リソースネーム (ARN)。AWSServiceRoleForImageBuilder サービスにリンクされたロールを指定するか、サービスアクセス用の独自のカスタムロールを指定できます。
- platform (文字列、必須) — ISO ディスク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。有効な値には `Windows` が含まれます。
- osVersion (文字列、必須) — ISO ディスクイメージのオペレーティングシステムのバージョン。有効な値には `Microsoft Windows 11` が含まれます。
- infrastructureConfigurationArn (文字列、必須) – ISO イメージが構築されている EC2 インスタンスの起動に使用されるインフラストラクチャ設定リソースの Amazon リソースネーム (ARN)。
- uri (文字列、必須) – Amazon S3 に保存されている ISO ディスクファイルの URI。

```
Import-EC2IBDiskImage `
    -Name "example-iso-disk-import" `
    -SemanticVersion "1.0.0" `
    -Description "Import an ISO disk image" `
    -ExecutionRole "AWSServiceRoleForImageBuilder" `
    -Platform "Windows" `
    -OsVersion "Microsoft Windows 11" `
    -InfrastructureConfigurationArn "arn:aws:imagebuilder:us-
east-1:111122223333:infrastructure-configuration/example-infrastructure-
configuration-123456789abc" `
    -Uri "s3://amzn-s3-demo-source-bucket/examplefile.ISO"
```

インポートが完了すると、所有しているイメージのリストにイメージが表示されます。詳細については、「イメージを一覧表示する」を参照してください。

出力 AMI からインスタンスを起動する

インポートプロセスが作成する AMI からインスタンスを起動すると、Windows オペレーティングシステムは Sysprep Specialize を実行します。これにより、パブリック S3 エンドポイントから EC2Launch v2 と Systems Manager Agent が自動的にダウンロードおよびインストールされます。これらのエンドポイントには、パブリックインターネットアクセスが必要です。プライベートサブネットからインスタンスを起動すると、Sysprep Specialize プロセスは S3 エンドポイントにアクセスできず、起動は失敗します。

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 は有料サービスです。

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。
- 検出結果の重要度を評価します。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
- Medium
- 高
- [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 がアカウントで既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 と Amazon Inspector のスコアが並べて比較されるため、該当する場合は Amazon Inspector がスコアを変更した箇所を確認できます。
- [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、「イメージリソースの詳細の表示」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、関連するイメージビルド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
- Medium
- 高
- [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。

Image Builder リソースのクリーンアップ

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、「未使用または古くなった Image Builder リソースの削除」を参照してください。

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。関連付けられたリソースには、他の 、組織 AWS アカウント、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。未使用または古いイメージと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

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. [ロールの作成] を選択してください。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。

   JSON

```
{
    "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. [ロールの作成] を選択してください。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 として設定 AWS アカウントする 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。

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 の 「ListLifecyclePolicies」 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、 コマンド AWS CLI 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**
>
> デフォルトを使用するには AWS リージョン、 `--region`パラメータなしで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。

Type

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

次のいずれかの方法を使用して、 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。また、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。

ライフサイクルポリシーが選択するリソースに適用するには、次のライフサイクルルールを 1 つ以上オンにします。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

次のいずれかの方法を使用して、 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。また、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。

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。たとえば、 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

AMI ライフサイクルの除外ルール

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 AWS アカウント リソースを指定します。

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。

Image Builder でのカスタムイメージの設定

設定リソースは、イメージパイプラインを構成する要素であり、それらのパイプラインから生成され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 コマンドを使用したリソースのタグ付けの詳細については 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 のレシピの管理

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 コマンドを使用したリソースのタグ付けの詳細については 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**
>
> Amazon マネージドコンポーネントをレシピで使用することも、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。詳細については、「Image Builder イメージ用のカスタムコンポーネントの開発」を参照してください。出力 AMIs を作成するイメージレシピでは、 AWS Marketplace イメージ製品とコンポーネントを使用することもできます。 AWS Marketplace 製品との統合の詳細については、「」を参照してくださいAWS Marketplace Image Builder での統合。

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

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

コンソールからのイメージレシピの一覧表示

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。

から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
```

コンソールからのイメージレシピの詳細の表示

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、コンソールからのイメージレシピの一覧表示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次のことができます。

- レシピを削除する。Image Builder でのリソースの削除については、「未使用または古くなった Image Builder リソースの削除」を参照してください。
- 新しいバージョンを作成する。
- 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、「チュートリアル: Image Builder コンソールウィザードから AMI を出力するイメージパイプラインを作成する」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。

からイメージレシピの詳細を取得する AWS CLI

次の例は、imagebuilder CLI コマンドを使用して 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
```

からイメージ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で一覧表示する
- コンソールでのコンテナレシピ詳細の表示
- AWS CLIでコンテナレシピの詳細を取得する
- 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で一覧表示する

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。

- レシピを削除する。Image Builder でリソースを削除する方法の詳細については、「未使用または古くなった Image Builder リソースの削除」を参照してください。
- 新しいバージョンを作成する。
- 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、チュートリアル: Image Builder コンソールウィザードから AMI を出力するイメージパイプラインを作成する 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。

AWS CLIでコンテナレシピの詳細を取得する

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する

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

- コンソールからの新しいイメージレシピバージョンの作成
- を使用して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する

コンソールからの新しいイメージレシピバージョンの作成

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、Image Builder での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 – このタブは開き、 のサブスクライブされたイメージをベースレシピに合わせて事前に選択 AWS Marketplace する必要があります。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 ソース (必須) - ベースイメージとして使用する AMI ID を含む AMI ID または AWS Systems Manager (SSM) パラメータストアパラメータを入力します。SSM エージェントは、選択した AMI にプリインストールされている必要があります。

- AMI ID – この設定には、元のエントリが事前に入力されていません。ベースイメージの AMI ID を入力します。例えば、*ami-1234567890abcdef1* などです。
- SSM パラメータ – ベースイメージの AMI ID を含む SSM パラメータストアパラメータの名前または ARN を入力します。例: */ib/test/param* または`arn:aws:ssm:`*us-east-1*`:`*111122223333*`:parameter`*/ib/test/param*。

- 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。

    Image Builder がインスタンスを起動すると、ユーザーデータスクリプトは、コンポーネントの実行が開始される前に、クラウド開始フェーズ中に実行されます。このステップは、インスタンスの ファイルに記録されます`var/log/cloud-init.log`。

    > **Note**
    >
    > - ユーザーデータを入力する場合は、システムマネージャーエージェントをベースイメージにあらかじめインストールするか、ユーザーデータにインストールを含めるようにしてください。
    > - 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「Image Builder 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。

- 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。コンポーネントを追加するには、ビルドコンポーネントの追加を選択し、適用する所有権フィルターを選択します。たとえば、 AWS Marketplace 製品に関連付けられているビルドコンポーネントを追加するには、 を選択しますAWS Marketplace。これにより、コンポーネントを一覧表示 AWS Marketplace する選択パネルがコンソールインターフェイスの右側に表示されます。CIS コンポーネントで、 を選択しますThird party managed。

  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勧めします。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 — コンポーネントが受け付ける入力パラメータを表示します。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する場合、入力パラメータにデフォルト値が定義されていると、そのデフォルト値が [値] ボックスにグレーアウトされたテキストで表示されます。他の値を入力しない場合、Image Builder はデフォルト値を使用します。

入力パラメータが必須で、コンポーネントに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。必須パラメータのいずれかが不足していてデフォルト値も定義されていない場合、Image Builder は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] 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、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。これは、そのキーがルートボリュームに使用されるものと同じ場合でも同様です。

新しいイメージレシピバージョンを作成するには：

1. レシピ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、レシピの作成を選択します。

   最終イメージには、 AWS Marketplace イメージ製品とコンポーネントから最大 4 つの製品コードを含めることができます。選択したベースイメージとコンポーネントに 4 つ以上の製品コードが含まれている場合、Image Builder はレシピを作成しようとするとエラーを返します。

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。

を使用して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 からカスタム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、レシピに含める 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 オペレーションリクエストパラメータに指定されたパターンに従います。API オペレーション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。親イメージは、次のいずれかのオプションを使用して指定できます。
  - AMI ID
  - Image Builder イメージリソース ARN
  - AWS Systems Manager (SSM) パラメータストアパラメータ。プレフィックスは `ssm:` で、パラメータ名または ARN が続きます。
  - AWS Marketplace 製品 ID

> **Note**
>
> Linux および macOS の例では Image Builder AMI を使用し、Windows の例では ARN を使用します。

- 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、Image Builder での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、常に最後に実行します。

  - 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。入力パラメータが必須で、コンポーネントに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。必須パラメータのいずれかが不足していてデフォルト値も定義されていない場合、Image Builder は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。

- 追加のインスタンス設定 (オブジェクト) — ビルドインスタンス用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・インスタンス上の Systems Manager エージェントの設定が含まれます。
    - uninstallAfterBuild (Boolean) - 新しい AMI を作成する前に、Systems Manager エージェントを最終ビルド・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。

- ユーザーデータの上書き (文字列) — 構築インスタンスを起動するとき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-1234567890abcdef1",
```

```
"semanticVersion": "1.0.0",
"components": [
 {
  "componentArn": "arn:aws:imagebuilder:us-west-2:111122223333: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-2:111122223333:component/my-
example-component/2019.12.02/1"
 },
 {
  "componentArn": "arn:aws:imagebuilder:us-west-2:111122223333: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、Image Builder でのセマンティックバージョニングを参照してください。

macOS

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": "macOS Catalina Image recipe",
"description": "Hello World image recipe for macOS.",
"parentImage": "ami-1234567890abcdef1",
"semanticVersion": "1.0.0",
"components": [
 {
  "componentArn": "arn:aws:imagebuilder:us-
west-2:111122223333:component/catalina$"
 }
],
"additionalInstanceConfiguration": {
 "systemsManagerAgent": {
   "uninstallAfterBuild": true
 },
 "userDataOverride": "IyEvYmluL2Jhc2gKbWtkaXIgLXAgL3Zhci9iYi8KdG91Y2ggL3Zhcg=="
}
}
```

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://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS ではフォーワードスラッシュ (/) が使用されます。

最終イメージには、 AWS Marketplace イメージ製品とコンポーネントから最大 4 つの製品コードを含めることができます。選択したベースイメージとコンポーネントに 4 つ以上の製品コードが含まれている場合、Image Builder は`create-image-recipe`コマンドの実行時にエラーを返します。

VM をベースイメージとしてコンソールにインポートする

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、「パイプラインウィザード: AMI の作成」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する」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レシピ名が継承されます。
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。

Encryption

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。

新しいコンテナレシピのバージョンを作成

このセクションでは、コンテナレシピ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レシピの作成

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。

基本の 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。

  Docker コンテナイメージについては、DockerHub でホストされているパブリックイメージ、Amazon ECR 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。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。自動バージョン管理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、Image Builder での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 ソース (必須) – コンテナビルド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カスタム AMI を特定します。これは、AMI ID または AMI ID を含む AWS Systems Manager (SSM) パラメータストアパラメータです。
  - AMI ID – この設定には、元のエントリが事前に入力されていません。ベースイメージの AMI ID を入力します。例えば、*ami-1234567890abcdef1* などです。
  - SSM パラメータ – ベースイメージの AMI ID を含む SSM パラメータストアパラメータの名前または ARN を入力します。例: */ib/test/param* または`arn:aws:ssm:`*us-east-1*`:`*111122223333*`:parameter`*/ib/test/param*。
- 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。コンポーネントを追加するには、ビルドコンポーネントの追加を選択し、適用する所有権フィルターを選択します。たとえば、 AWS Marketplace 製品に関連付けられているビルドコンポーネントを追加するには、 を選択しますAWS Marketplace。これにより、コンポーネントを一覧表示 AWS Marketplace する選択パネルがコンソールインターフェイスの右側に表示されます。CIS コンポーネントで、 を選択しますThird party managed。

  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勧めします。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 — コンポーネントが受け付ける入力パラメータを表示します。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する場合、入力パラメータにデフォルト値が定義されていると、そのデフォルト値が [値] ボックスにグレーアウトされたテキストで表示されます。他の値を入力しない場合、Image Builder はデフォルト値を使用します。

  入力パラメータが必須で、コンポーネントに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。必須パラメータのいずれかが不足していてデフォルト値も定義されていない場合、Image Builder は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] 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。次のコンテキスト変数を指定できます。これらは、実行時に Image Builder によってビルド情報に置き換えられます。

  parentImage (必須)

  この変数は、ビルド時にレシピのベースイメージに解決されます。

  例:

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

environments (components 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 }}}
```

components (オプション)

Image Builder は、コンテナレシピに含まれているコンポーネントのビルドおよび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 内で environments 変数の後の任意の場所に配置できます。

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。

新しいコンテナレシピのバージョンを作成するには:

1. コンテナレシピ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、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。

AWS CLIでコンテナレシピを作成する

で `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 からカスタム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、レシピに含める 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、代わりにすべてのコンテナレシピ設定を含む JSON ファイルを提供できる

   > **Note**
   >
   > JSON ファイル内のデータ値の命名規則は、Image Builder API オペレーションリクエストパラメータに指定されたパターンに従います。API オペレーション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、この例のパラメータの概要を示します。

   - コンポーネント (オブジェクトの配列、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、常に最後に実行します。

     - 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コンテナ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。入力パラメータが必須で、コンポーネントに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。必須パラメータのいずれかが不足していてデフォルト値も定義されていない場合、Image Builder は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 `DOCKER` が含まれます。
- DockerFileTemplateData (文字列) — イメージの構築に使用される Dockerfile テンプレート。インラインデータブロブとして表現されます。
- 名前 (文字列、必須) — コンテナレシピの名前。
- 説明 (文字列) — コンテナレシピの説明。
- parentImage (文字列、必須) – カスタマイズされたイメージのベースラインとしてコンテナレシピで使用する Docker コンテナイメージ。
  - DockerHub でホストされているパブリックイメージ

- Amazon ECR の既存のコンテナイメージ
- Amazon が管理するコンテナイメージ

- 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、Image Builder での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)コンテナレシピ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) – コンテナビルドおよびテストインスタンスのベースイメージ。これには、AMI ID を含めることも、 AWS Systems Manager (SSM) パラメータストアパラメータを指定することもできます。プレフィックスは で、パラメータ名または ARN が`ssm:`続きます。SSM パラメータを使用する場合は、パラメータ値に AMI ID が含まれている必要があります。ベースイメージを指定しない場合、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パラメータで指定した 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 (Boolean) –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します。
      - 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-west-2:111122223333: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-1234567890abcdef1",
   "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://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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、「未使用または古くなった Image Builder リソースの削除」を参照してください。

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。

  Mac イメージでは、macOS オペレーティングシステムをネイティブにサポートするインスタンスタイプを選択することもできます。詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 Mac インスタンス」を参照してください。

- インスタンス配置の設定。イメージからのインスタンスが起動する先のホスト、ホストプレイスメントグループ、またはアベイラビリティーゾーンを指定できます。
- 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。例えば、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、「Image Builder を使用してカスタムイメージをビルドするためのセットアップ」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

> **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 コマンドを使用したリソースのタグ付けの詳細については 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する

このセクションでは、EC2 Image Builder インフラストラクチャ構成の情報を検索し、詳細を表示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

- からインフラストラクチャ設定を一覧表示する AWS CLI
- からインフラストラクチャ設定の詳細を取得する AWS CLI

からインフラストラクチャ設定を一覧表示する 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
```

からインフラストラクチャ設定の詳細を取得する AWS CLI

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. 全般 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 はこれらの権限を使用して、コンポーネントのダウンロードと実行、CloudWatch へのログのアップロード、およびレシピ内のコンポーネントで指定されている追加アクションの実行を行います。
5. [AWS インフラストラクチャ] パネルでは、利用可能な残りのすべてのインフラストラクチャ設定を構成できます。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。

     > **Note**
     >
     > Mac インスタンスは、専有ホストの `.metal` インスタンスタイプで実行されます。指定するインスタンスタイプは、実行するホストに定義されているタイプのいずれかと一致する必要があります。Mac インスタンスの詳細と、macOS オペレーティングシステムをネイティブにサポートするインスタンスタイプの一覧については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 Mac インスタンス」を参照してください。

   - SNS トピック (オプション) - EC2 Image Builder からの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、「Image Builder と AWS PrivateLink インターフェイス VPC エンドポイント」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、キーペアを作成するか、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。
- [インスタンス配置の設定] 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。
  - イメージの作成中に Image Builder がインスタンスを起動する [アベイラビリティーゾーン] を選択できます。

- 必要に応じて、起動したインスタンスを実行するサーバーの [テナンシー] を選択します。デフォルトでは、EC2 インスタンスは共有テナンシーハードウェアで実行されます。つまり、複数の AWS アカウント が同じ物理ハードウェアを共有する可能性があります。dedicated テナンシーのインスタンスは、シングルテナントのハードウェアで実行されます。host テナンシーのインスタンスは、専有ホストで実行されます。

  Mac インスタンスには、カスタムイメージのビルド前に前提条件として作成された専有ホストが必要です。macOS イメージ用の host を選択します。その後、インスタンスを起動するターゲットホストまたはホストリソースグループを選択できますが、専有ホストで自動配置が有効になっている場合は必須ではありません。詳細については、「Amazon EC2 ユーザーガイド」の「自動配置」を参照してください。

  - [テナンシーのホスト ID] - インスタンスが実行される専有ホストの ID。
  - [テナンシーのホストリソースグループ] - インスタンスを起動するホストリソースグループの Amazon リソースネーム (ARN)。

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します。
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します。

AWS CLI

以下の手順では、 AWS CLIで Image Builder の 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ます。手順 2 のコマンドには、手順 1 で作成したファイルを指定します。これらの例では、手順 1 のファイルを create-infrastructure-configuration.json として参照しています。

1. CLI 入力 JSON ファイルの作成

   次の例は、インフラストラクチャ設定用に作成できる JSON ファイルのバリエーションを示しています。ファイル編集ツールを使用して独自の JSON ファイルを作成してください。

   例 1: 失敗したビルドのインスタンスを保持する設定

   この例では、m5.large と m5.xlarge の 2 つのインスタンスタイプを指定します。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。例えば、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、「Image Builder を使用してカスタムイメージをビルドするためのセットアップ」を参照してください。

```
{
    "name": "ExampleInfraConfigDontTerminate",
    "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"
}
```

例 2: 自動配置が有効な macOS 設定

この例では、専有ホストで自動配置が有効になっている Mac インスタンスのインスタンスタイプとプレイスメントを指定します。

```
{
   "name": "macOSInfraConfigAutoPlacement",
   "description": "An example infrastructure configuration for macOS.",
```

```
   "instanceProfileName": "EC2InstanceProfileForImageBuilder",
   "instanceTypes": ["mac1.metal, mac2.metal"],
   "terminateInstanceOnFailure": false,
   "placement": {
      "tenancy": "host"
   }
}
```

例 3: ホスト ID が指定された macOS 設定

この例では、特定の専有ホストをターゲットとする Mac インスタンスのインスタンスタイプとプレイスメントを指定します。

```
{
   "name": "macOSInfraConfigHostPlacement",
   "description": "An example infrastructure configuration for macOS.",
   "instanceProfileName": "EC2InstanceProfileForImageBuilder",
   "instanceTypes": ["mac2-m1ultra.metal"],
   "terminateInstanceOnFailure": false,
   "placement": {
      "tenancy": "host",
      "hostId" : "h-1234567890abcdef0"
   }
}
```

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する

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できます。タグはキーと値のペアとして入力します。

- 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。
- インスタンスタイプや通知用の SNS トピックを含む AWS インフラストラクチャ。
- 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、 AWS CLIで Image Builder の 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。

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
   ```

Image Builder と AWS PrivateLink インターフェイス VPC エンドポイント

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、または AWS Direct Connect 接続なしで 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。
- 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が適用されると、 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 CloudWatch に口グ記録されない場合があります。

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、別の S3 バケットから AWSTOE コンポーネント管理アプリケーションを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。

JSON

```
{
    "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する

次のエンドポイントポリシーの例は、組織に属している ID とリソースにアクセスを限定し、Amazon が管理する Image Builder 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*、*resource-org-id* を組織の値に置き換えます。

JSON

```
{
  "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 イメージでは、イメージレシピにユーザーデータが指定されていない場合、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ために、Image Builder は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、次のリソースを S3 エンドポイントポリシーに追加します。
>
> `"arn:aws:s3:::amazon-ssm-region/*"`

JSON

```
{
  "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/*"
      ]
    }
  ]
}
```

Image Builder のディストリビューション設定の管理

出力イメージのディストリビューション設定を構成する前に、ディストリビューションのターゲットリージョンで、出力イメージから起動されるインスタンスの基盤となるインフラストラクチャやその他の要件が利用可能であるかどうかを確認することをお勧めします。例えば、macOS イメージからインスタンスを起動するために必要な EC2 Mac 専有ホストは、すべてのリージョンでサポートされているわけではありません。専有ホストのインスタンスタイプと料金の詳細については、「Amazon EC2 Dedicated Hosts の料金」を参照してください。

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。

AMI 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。

- 他の AWS アカウント、組織、OUs が所有者のアカウントから AMI を起動することを許可します。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。

  > **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。情報と例については、「Amazon EC2 API リファレンス」の「ModifyImageAttribute」を参照してください。

- 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。 AWS Organizations および OUs への AMI の配布の詳細については、「組織または OUs」を参照してください。
- 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、「から出力 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) にイメージを 1 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 コマンドを使用したリソースのタグ付けの詳細については 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

ディストリビューション設定の一覧と詳細の表示

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に関する情報を特定し、詳細を確認するためのさまざまな方法について説明します。

ディストリビューション設定の詳細

コンソールでのディストリビューション設定の一覧表示

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。

コンソールでのディストリビューション設定の詳細の表示

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、「コンソールでのディストリビューション設定の一覧表示」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、「未使用または古くなった Image Builder リソースの削除」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。

からディストリビューションを一覧表示する AWS CLI

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
```

からディストリビューション設定の詳細を取得する 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

SSM 出力パラメータの前提条件

パラメータストアパラメータ (SSM AWS Systems Manager パラメータ) を設定する新しい AMI ディストリビューション設定を作成する前に、次の前提条件を満たしていることを確認してください。

実行ロール

パイプラインを作成する場合、または で create-image コマンドを使用する場合は AWS CLI、Image Builder 実行ロールを 1 つだけ指定できます。Image Builder ワークフロー実行ロールを定義している場合は、そのロールに機能アクセス許可を追加します。それ以外の場合は、必要なアクセス許可を含む新しいカスタムロールを作成します。

- ディストリビューション中に出力 AMI ID を SSM パラメータに保存するには、Image Builder 実行ロールで `ssm:PutParameter`アクションを指定し、 パラメータをリソースとしてリストする必要があります。
- パラメータデータ型を に設定`AWS EC2 Image`して、パラメータ値を AMI ID として検証するように Systems Manager にシグナルを送信する場合は、 `ec2:DescribeImages`アクションも追加する必要があります。

AMI ディストリビューション設定を作成する

ディストリビューション設定には、出力 AMI 名、暗号化の特定のリージョン設定、起動許可 AWS アカウント、出力 AMI を起動できる組織、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て、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 ModifyImageAttribute で AMI を公開する例を参照されたい。

Console

で新しい AMI 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 AWS Management Console。

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. 全般セクションで、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 AWS KMS key は、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、すべてのディストリビューションターゲットアカウントに `EC2ImageBuilderDistributionCrossAccountRole` ロールを作成し、そのロールに Ec2ImageBuilderCrossAccountDistributionAccess ポリシー マネージドポリシーを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、指定された 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。任意の AWS プリンシパルは、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する
     - [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。
     - 組織 – 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、 で作成され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。

iv. SSM パラメータ設定セクションを展開して、送信先リージョンに配信されるイメージの出力 AMI ID を保存する SSM パラメータを設定できます。必要に応じて、リージョンでディストリビューションアカウントを指定できます。

パラメータ名 – パラメータの名前を入力します。例: `/output/image/param`。

データ型 – デフォルト値 () のままにします`AWS EC2 Image`。これにより、Systems Manager にパラメータ値を検証して、有効な AMI ID であることを確認するように指示します。

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。

AWS CLI

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 ファイルを作成します。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 組織単位 (OUs) を定義します。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。

   例 1: に配布する AWS アカウント

   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"
     ]
    }
   }
  }
 ]
}
```

例 2: Organizations と 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"
     ]
    }
   }
  }
 ]
}
```

例 3: 出力 AMI ID を SSM パラメータに保存する

この例では、出力 AMI ID をディストリビューションリージョンの Parameter Store AWS Systems Manager パラメータに保存します。

```
{
 "name": "SSMParameterOutputAMI",
```

```
 "description": "Updates an SSM parameter with the output AMI ID for the
 distribution.",
 "distributions": [
  {
   "region": "us-west-2",
   "amiDistributionConfiguration": {
    "name": "Name {{ imagebuilder:buildDate }}"
   },
   "ssmParameterConfigurations": [
    {
     "amiAccountId": "111122223333",
     "parameterName": "/output/image/param",
     "dataType": "aws:ec2:image"
    }
   ]
  }
 ]
}
```

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://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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を更新する

AMI ディストリビューション設定を変更できます。ただし、行った変更は、Image Builder が既に配布しているリソースには適用されません。例えば、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、既に配布されていた AMI は、手動で削除するまでそのリージョンに残ります。

AWS Management Console

で AMI ディストリビューション設定を行うには、次の手順に従います AWS Management Console。

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設定を作成する」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。

AWS CLI

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

   ファイル編集ツールを使用して、次の例に示すキーと、環境に対して有効な値を含む 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://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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。

出力 AMI の EC2 高速起動を有効にするディストリビューション設定の作成

次の例は、 AWS CLIから create-distribution-configuration コマンドを使用して、AMI 用に EC2 高速起動が構成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。

> **Note**
>
> Image Builder では、EC2 高速起動を事前に有効にした AMI のクロスアカウントディストリビューションはサポートされていません。EC2 高速起動は配布先の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 ファイルを作成します。

この例では、並列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://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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。

から出力 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 オペレーションリクエストパラメータに指定されたパターンに従います。API オペレーション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 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、create-distribution-configuration コマンドに指定します。

以下は、この例の `s3ExportConfiguration` JSON オブジェクトに指定するパラメータの概要であります。

- 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以下のサポートされているフォーマットのいずれか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://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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。

コンテナイメージのディストリビューション設定を作成および更新する

このセクションでは、Image Builder コンテナイメージ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

- から Image Builder 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する AWS CLI
- から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 AWS CLI

から Image Builder 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する AWS CLI

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://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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。

から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 AWS CLI

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://` 表記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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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 コマンド例では、イメージレシピとインフラストラクチャ設定 JSON ファイルを作成済みである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、「を使用して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。

クロスアカウント AMI ディストリビューションの前提条件

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。次の例は、送信元アカウントのアカウント ID を指定する送信先アカウントの信頼ポリシーを示しています。

   JSON

   ```
   {
       "Version": "2012-10-17",
       "Statement": [{
           "Effect": "Allow",
           "Principal": {
               "AWS": "arn:aws:iam::444455556666:root"
           },
           "Action": "sts:AssumeRole"
       }]
   }
   ```

   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 AMI を各リージョンの適切なキーで暗号化および復号するときに、ユーザーに代わって動作できるようになります。

JSON

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

6. AWS Systems Manager Parameter Store パラメータを使用してディストリビューション
アカウントとリージョンの出力 AMI の AMI ID を保存する場合は、各送信先アカウントの
EC2ImageBuilderDistributionCrossAccountRole に次のポリシーを追加する必要があ
ります。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ssm:PutParameter"
            ],
            "Resource": "arn:aws:ssm:*:111122223333:parameter/ImageBuilder-*"
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeImages"
            ],
            "Resource": "*"
        }
    ]
}
```

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 リファレンスの「ディストリビューション設定の作成」を参照してください。

コンソールでの Image Builder AMI のクロスアカウント配信の設定

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「クロスアカウント AMI ディストリビューションの前提条件」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。

から Image Builder AMI のクロスアカウントディストリビューションを設定する AWS CLI

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、アカウント間で Image Builder AMI AWS CLI を構築および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 クロスアカウント AMI ディストリビューションの前提条件 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する

   で 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する

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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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 起動テンプレートを使用する AMI ディストリビューションの設定

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**
>
> Windows 高速起動を有効にする起動テンプレートをイメージに含める場合は、起動テンプレートに次のタグを含める必要があります。これにより、Image Builder がユーザーに代わって Windows 高速起動を有効にできます。
>
> `CreatedBy: EC2 Image Builder`

コンソールでの AMI ディストリビューション設定への EC2 起動テンプレートの追加

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。

から EC2 起動テンプレートを AMI ディストリビューション設定に追加する AWS CLI

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する

   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する

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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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。

Image Builder リソースを と共有する AWS RAM

EC2 Image Builder は AWS Resource Access Manager （ AWS RAM) と統合されるため、以下のタイプの Image Builder 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。

- コンポーネント
- イメージ
- recipe

を介してリソースを共有するには AWS RAM、リソース共有を作成する必要があります。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、個人 AWS アカウント、組織単位、または組織全体にすることができます AWS Organizations。次の一覧は、共有先として指定できるアカウントと組織のタイプが含まれています。

- の組織 AWS アカウント 内外に固有です 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。

このモデルでは、リソースを所有 AWS アカウント する (所有者) は、同じリージョン内の他の AWS アカウント または を介して AWS Organizations （ コンシューマー) リソースを共有します。共有リソースが更新されると、コンシューマーはそれらの更新を自動的に取得します。

> **Note**
>
> 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がコンシューマーのリソース制限に影響することはありません。

トピック

- リソース所有者
- リソースコンシューマー
- Image Builder AWS RAM リソースのリソース共有を作成する

- から Image Builder リソースの共有を解除する AWS RAM

リソース所有者

Image Builder リソースは、作成された 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。

自分が所有していて共有できる Image Builder リソースの一覧を取得するには、コンソールで所有権フィルターを指定するか、または AWS CLIでコマンドを実行するときに指定します。

- Image Builder コンポーネントの一覧表示
- イメージを一覧表示する
- 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

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。

Image Builder リソースを共有するための前提条件

コンポーネント、イメージ、レシピなどの Image Builder リソースを共有するには、次の前提条件が満たされている必要があります。

- は、共有する Image Builder リソースを所有している AWS アカウント 必要があります。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、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する」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。

リソースコンシューマー

コンシューマーは共有リソースを使用できますが、どのような方法でも変更することはできません。コンシューマーは、Image Builder レシピを作成するときに、共有イメージをベースイメージとして指定したり、共有コンポーネントを追加したりできます。また、Image Builder イメージパイプラインを作成するときや、 AWS CLIで create-image コマンドを使用するときに、共有レシピを指定することもできます。

の組織に属していて AWS Organizations、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リソースへのアクセス権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。

Image Builder AWS RAM リソースのリソース共有を作成する

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、 AWS Resource Access Manager リソース共有に追加する必要があります。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。

リソースを共有するには、以下のオプションを使用できます。

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、「 AWS RAM ユーザーガイド」の「ユーザーが所有する AWS リソースの共有」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既存の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 つのステップがあります。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がリソースを表示できるようにします。

1. リソースポリシーを適用する

   リソースポリシーを正常に適用するには、共有しているアカウントに、基礎的な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。

   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 プリンシパルを識別します。

```
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、共有しているアカウントに、プライベートリポジトリでホストされているファイルなど、共有コンポーネントによって参照される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-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 プリンシパルを識別します。

```
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、共有しているアカウントに、ベースイメージや選択したコンポーネントなど、レシピが参照する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コンテナレシピに適用して、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、共有しているアカウントに、ベースイメージや選択したコンポーネントなど、レシピが参照する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 プリンシパルを識別します。

```
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がリソースを表示できるようにするには、 で promote-resource-share-created-from-policy AWS RAM コマンドを実行します AWS CLI。

から Image Builder リソースの共有を解除する AWS RAM

共有コンポーネント、イメージ、レシピなど、所有している Image Builder リソースの共有を解除するには、 AWS Resource Access Manager リソース共有から削除する必要があります。これを行うには、 AWS RAM コンソールまたは AWS CLIを使用できます。

> **Note**
>
> 所有者は、共有リソースが共有されなくなるまで、その共有リソースを削除することはできません。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。

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するには

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するには

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。

Image Builder 出力リソースのタグ付け

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}}`

  ビルドバージョンに解決されます。これは、Image Builder Amazon リソースネーム (ARN) の末尾に位置する番号です。例えば、`"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 (AMI) を追跡できるようにするために、Image Builder は、次のタグを自動的に出力 AMI 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

からリソースにタグを付ける AWS CLI

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の内容は以下のとおり:

```
{
    "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
```

からリソースのタグを解除する 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の内容は以下のとおり:

```
{
    "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
```

から特定のリソースのすべてのタグを一覧表示する AWS CLI

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
```

未使用または古くなった Image Builder リソースの削除

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

からリソースを削除する AWS Management Console

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する

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。

レシピを削除する

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する

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。

から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

からイメージパイプラインを削除する 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
```

からイメージレシピを削除する AWS CLI

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。

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の管理

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。

ワークフロートピック

- ワークフローフレームワーク: ステージ
- サービスアクセス
- イメージにマネージドワークフローを使用する
- イメージワークフローを一覧表示する
- イメージワークフローの作成
- YAML ワークフロードキュメントを作成する

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、次の個別のステージが含まれます。

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、サービスリンクロールまたは独自のカスタムロールを `executionRole` パラメータの値として指定します。

サービスロールの作成方法の詳細については、「 AWS Identity and Access Management 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」を参照してください。

イメージにマネージドワークフローを使用する

マネージドワークフローは によって作成および管理されます AWS。イメージパイプラインまたは 1 回限りのイメージ作成でマネージドワークフローを使用する場合は、使用するマネージドワークフローの Amazon リソースネーム (ARN) を選択できます。Amazon は、パッチやその他の更新が適用された最新バージョンを提供します。マネージドワークフローのリストを取得するには、「」を参照しイメージワークフローを一覧表示する、所有者 = Amazon (コンソール) でフィルタリングします。

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」リファレンスの「ListWorkflows」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。

次の詳細をフィルタリングして、結果のリストを合理化できます。たとえば、コンソールで Owner = Amazon でフィルタリングすると、リストには AWS マネージドワークフローのみが表示されます。

- ワークフロー
- バージョン
- タイプ
- 所有者

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、「Image Builder での暗号化とキーの管理」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。

はじめに

- 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。

- ワークフロードキュメントの識別
- ワークフロードキュメントの入力パラメータ
- ワークフロードキュメントのステップ
- ワークフロードキュメントの出力

ワークフロードキュメントの識別

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
名前	ワークフロードキュメントの名前。	String	いいえ
説明	ドキュメントの説明。	String	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。	String	はい

例

```
---
name: sample-test-image
description: Workflow for a sample image, with extra configuration options exposed
 through workflow parameters.
schemaVersion: 1.0
```

ワークフロードキュメントの入力パラメータ

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
名前	パラメータの名前。	String	はい	
description	パラメータの説明。	String	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	String	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
```

ワークフロードキュメントのステップ

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。	String	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 、numberLsThan 、numberLsThanEquals 、stringEquals 。
description	手順の説明。	String	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		
名前	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。	String	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。 行動 • 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	String	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"
```

ワークフロードキュメントの出力

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
名前	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。	String	はい
値	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。	String	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	[Default] (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。	String	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。	String

出力名	説明	[Type] (タイプ)
ステータス	Systems Manager sendCommand から返されたステータス。	String
output	Systems Manager sendCommand から返された出力。	String

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	[Default] (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。	String	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。	String
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。	String

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	[Default] (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。	String	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。	String
ステータス	Systems Manager sendCommand から返されたステータス。	String
output	Systems Manager sendCommand から返された出力。	String

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	[Default] (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。	String	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。	String

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	[Default] (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。	String	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。	String

出力名	説明	[Type] (タイプ)
ステータス	Systems Manager sendCommand から返されたステータス。	String
output	Systems Manager sendCommand から返された出力。	String

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	[Default] (デフォルト)	制約
imageIdOverride	インスタンスの起動に使用するイメージ	String	いいえ	ビルドステージ: イメージレシピベースイメージ テストステージ: ビルドステージから AMI を出力する	有効な AMI ID である必要があります
instanceTypesOverride	Image Builder は、正常に起動するインスタンスタイプが見つかるまで、リスト内の各インスタンスタイプを試行します。	文字列のリスト	いいえ	インフラストラクチャ設定で指定されたインスタンスタイプ	有効なインスタンスタイプである必要があります
waitFor	ワークフローステップを完了し、次のステップに進む前に待機する条件	String	はい		Image Builder は、`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。	String

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	[Default] (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。	String	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

入力名	説明	タイプ	必須	[Default] (デフォルト)	制約
					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。	String	はい		
パラメータ	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。	String	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。	String

出力名	説明	[Type] (タイプ)
ステータス	Systems Manager sendCommand から返されたステータス。	String
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 、イメージを強化およびクリーンアップするため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	[Default] (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。	String	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。	String
ステータス	Systems Manager sendCommand から返されたステータス。	String
output	Systems Manager sendCommand から返された出力。	String

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」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	[Default] (デフォルト)	制約
instanceId	サニタイズするインスタンスの ID。	String	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

入力名	説明	タイプ	必須	[Default] (デフォルト)	制約
					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。	String
ステータス	Systems Manager sendCommand から返されたステータス。	String
output	Systems Manager sendCommand から返された出力。	String

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	[Default] (デフォルト)	制約
instanceId	終了するインスタンスの ID。	String	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。このステップは、EventBridge イベントをデフォルトの EventBridge イベントバスに詳細タイプ `EC2 Image Builder Workflow Step Waiting` で公開します。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	[Default] (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。	String	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。	String

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
```

入力パラメータが動的変数を参照する場合、次の例に示すように、パラメータ名の末尾に連鎖インジケータ (.$) をアタッチする必要があります。

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
cloudWatchLogGroup	出力ログの CloudWatch Logs グループの名前。 形式: `/aws/imagebuilder/`*<recipe-name>*	String	`/aws/imagebuilder/`*sampleImageRecipe*
cloudWatchLogStream	出力ログの CloudWatch Logs ストリームの名前。	String	*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`
collectImageScanFindings	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*

変数名	説明	[Type] (タイプ)	値の例
imageId	ベースイメージの AMI ID。	String	*ami-12345 67890abcdef1*
imageName	イメージの名前。	String	*sampleImage*
imageType	画像出力タイプ。	String	`AMI | Docker`
imageVersionNumber	イメージのバージョン番号。	String	*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。	String	*SampleImageBuilderInstanceProfileRole*
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。	String	`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)。	String	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。	String	*subnet-12 34567890a bcdef1*
terminateInstanceO nFailure	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。	String	`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。	String	/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、データ型 (文字列または数値) に基づいて値の比較 (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、「条件ステートメントでのネストレベル」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。

条件ステートメントでのネストレベル

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。

条件ステートメントの例

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]+$。

次の比較の例はすべて true に評価されます。

```
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
```

反復可能なパイプラインプロセスによる Image Builder でのカスタムイメージ作成の管理

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。

- 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 EventBridge を通じてルールベースの自動化を有効にします。

> **Note**
>
> イベントブリッジ 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 EventBridge events コマンドを使用して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、AWS CLI 「 コマンドリファレンス」の「イベント」を参照してください。

内容

パイプライン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 イメージパイプラインの情報を検索し、詳細を表示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

- からイメージパイプラインを一覧表示する AWS CLI
- からイメージパイプラインの詳細を取得する AWS CLI

からイメージパイプラインを一覧表示する AWS CLI

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
```

からイメージパイプラインの詳細を取得する AWS CLI

次の例は、 で 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 Amazon マネージドイメージ、作成したイメージ、共有されたイメージ、または を通じてサブスクライブしたイメージのベースイメージを含む AMI イメージレシピを選択します AWS Marketplace。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。

デフォルト値が指定されている場合の [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ステップバイステップのチュートリアルについては、「チュートリアル: Image Builder コンソールウィザードから AMI を出力するイメージパイプラインを作成する」を参照してください。

内容

から AMI イメージパイプラインを作成する 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、「Image Builder での cron 式の使用」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表すためにバックスプラッシュ (\) が使用され、Linux と macOS ではフォーワードスラッシュ (/) が使用されます。

2. 入力として作成した ファイルを使用して、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
```

コンソールでの AMI イメージパイプラインの更新

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「から AMI イメージパイプラインを更新する 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イメージ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、「Image Builder のレシピの管理」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 ルール — 選択したイベントバスについて、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 コンソールにリンクする イベントバスの作成 と ルールの作成 アクションが含まれます。詳細については、「EventBridge ルール」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する

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。
- インスタンスタイプや通知用の SNS トピックを含む AWS インフラストラクチャ。
- 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する

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する

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. パイプラインの詳細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、Image Builder での cron 式の使用 を参照してください。

から AMI イメージパイプラインを更新する 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 はパイプラインの部分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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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する
- 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。
- AWS イメージをリージョンに配布する

イメージパイプラインを作成する コンソールウィザードの使用に関する詳細とステップバイステップのチュートリアルについては、「チュートリアル: Image Builder コンソールウィザードから Docker コンテナイメージを出力するイメージパイプラインを作成する」を参照してください。

内容

からコンテナイメージパイプラインを作成する AWS CLI

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、「Image Builder での cron 式の使用」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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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
```

コンソールでのコンテナイメージパイプラインの更新

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「からコンテナイメージパイプラインを更新する AWS CLI」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する

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イメージ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、「新しいコンテナ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 ルール — 選択したイベントバスについて、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 コンソールにリンクする イベントバスの作成 と ルールの作成 アクションが含まれます。詳細については、「EventBridge ルール」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する

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。
- インスタンスタイプや通知用の SNS トピックを含む AWS インフラストラクチャ。
- 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する

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する

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. パイプラインの詳細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、Image Builder での cron 式の使用 を参照してください。

からコンテナイメージパイプラインを更新する 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 はパイプラインの部分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表すためにバックスプラッシュ (\) が使用され、Linux と macOS 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
```

Image Builder でのパイプラインワークフローの設定

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、「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の管理」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

コンソールでの Image Builder パイプライン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。

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、サービスリンクロールまたは独自のカスタムロールを `executionRole` パラメータの値として指定します。

サービスロールの作成方法の詳細については、「 AWS Identity and Access Management 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」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。例えば、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. パイプラインの詳細ページの左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。

Image Builder での cron 式の使用

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 式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 式ではデフォルトで協定世界時 (UTC) タイムゾーンが使用されますが、タイムゾーンを指定することもできます。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 日までが含まれます。
*	* (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。日付フィールドと曜日フィールドで、一方が指定されるか、可能なすべての値 (*) を含む場合、もう一方は ? でなければなりません。両方を指定することはできません。例えば、「日付」フィールドに 7 を入力する (その月の 7 日にビルドを実行する) 場合、曜日位置には ? 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。例えば、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、ワイルドカード W は、平日を指定します。「日付」フィールドに、より前の数字を入力すると W、その日に最も近い平日がターゲットになるということです。例えば、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。例えば、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

- cron 式の日フィールドと曜日フィールドを同時に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。

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 式を示しています。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。

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)
```

値

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たり、独自のアプリケーションから EventBridge にイベントを送信したりすることもできます。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に基づいて、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 から EventBridge に送信されるシステム生成イベントの概要については、「Image Builder が送信するイベントメッセージ」を参照してください。

> **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

イベントブリッジの用語

このセクションでは、EventBridge と Image Builder パイプラインの統合方法を理解するための用語の概要を説明します。

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します。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。

ターゲット

EventBridge がルールに一致したときに呼び出し、イベントからターゲットにデータを配信するリソースまたはエンドポイント。

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 — 例えば、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 の イメージパイプライン 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のEventBridge コンソールにも直接リンクしています。

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

EventBridge の詳細については、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 EventBridge とは
- 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イメージ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 の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。Amazon EventBridge コンソールのルールを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実行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 rate 式ルールの設定について詳しくは、Amazon EventBridge User Guide のレート表現を参照してください。

Image Builder での製品とサービスの統合

EC2 Image Builder は、 AWS Marketplace 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、堅牢で安全なカスタムマシンイメージの作成に役立ちます。

製品

Image Builder レシピは、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 などの組織標準を満たすために、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。
- Image Builder コンポーネント - レシピに指定したコンポーネントは、ソフトウェアのインストールやコンプライアンス検証の実行など、ビルドアクションやテストアクションを実行することができます。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 して、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 Organizations – AWS Organizations 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。Image Builder は、他のすべての AWS アーティファクトやサービスと同様に、「」で定義されているポリシーを

尊重します AWS Organizations。 は、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強制するなど、一般的なシナリオのためにテンプレート SCP AWS を提供します。 SCPs

- AWS CloudTrail - CloudTrail に送信される Image Builder イベントを監視します。CloudTrail と Image Builder の統合については CloudTrail を使用した Image Builder API コールのログ記録 を参照してください。

  CloudTrail の詳細 (有効にする方法、ログファイルを検索する方法を含む) については、「AWS CloudTrail ユーザーガイド」を参照してください。

- Amazon CloudWatch Logs - CloudWatch を使用して Image Builder ログファイルをモニタリングし、保存してアクセスできます。オプションで、ログを S3 バケットに保存できます。CloudWatch と Image Builder の統合の詳細については、「Amazon CloudWatch Logs を使用した Image Builder ログのモニタリング」を参照してください。

  CloudWatch Logs の詳細については、Amazon CloudWatch Logs User Guide の Amazon CloudWatch Logs とはを参照してください。

- Amazon Elastic Container Registry (Amazon ECR) - Amazon ECR は、セキュリティ、スケーラビリティ、信頼性を備えた AWS マネージド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。

- Amazon EventBridge - アカウント内の Image Builder アクティビティからリアルタイムのイベントデータのストリームに接続します。EventBridge の詳細については、Amazon EventBridge User Guide の Amazon EventBridge とはを参照してください。

- 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。スキャンと料金の詳細については、「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 License Manager - ディストリビューションプロセス中に、License Manager のセルフマネージドライセンスを出力 AMI にアタッチできます。宛先リージョンに指定するライセンスは、そのリージョンに既に存在している必要があります。セルフマネージドライセンスの詳細については、「License Manager のセルフマネージドライセンス」を参照してください。
- 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」の「製品の購入」を参照してください。
- AWS Resource Access Manager （ AWS RAM ） – を使用すると AWS RAM、 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。複数の がある場合は 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。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、「Image Builder リソースを と共有する AWS RAM」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

- Image Builder における 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

Image Builder における Amazon EventBridge の統合

Amazon EventBridge はサーバーレスのイベントバスサービスで、Image Builder アプリケーションと他の AWS のサービスからの関連データを接続するために使用できます。EventBridge では、ルールが受信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でき、これらのイベントは並行して実行されます。

EventBridge を使用すると、 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。 AWS のサービス からのイベントは、ほぼリアルタイムに EventBridge に提供されます。受信イベントに反応してアクションを開始するルールを設定できます。例えば、EC2 インスタンスが保留から実行に変わったときに、イベントを Lambda 関数に送信します。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 EventBridge User Guide の Creating Amazon EventBridge rules that react to events を参照してください。

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 Run Command 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 * * ? *)
  ```

  EventBridge で cron 式を使用する詳細については、Amazon EventBridge User Guide 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)
  ```

EventBridge で rate 式を使用する詳細については、Amazon EventBridge User Guide の rate 式を参照してください。

EventBridge ルールが Image Builder のイメージパイプラインとどのように統合されるかの詳細については、「Image Builder パイプラインで EventBridge ルールを使用する」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

Image Builder は、Image Builder リソースのステータスに大きい変更があったときに EventBridge にイベントメッセージを送信します。例えば、イメージの状態が変更された場合が該当します。以下の例では、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ます。

EC2 Image Builder Image State Change

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ると、Image Builder からこのイベントが送信されます。ある状態から別の状態にイメージのステータスが変わる例には、次のようなものがあります。

- `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 Detected

イメージで CVE 検出が有効になっていると、イメージスキャンが完了するたびに Image Builder から結果を含むメッセージが送信されます。

```
{
    "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 Workflow Step Waiting

非同期アクションの完了を待機するために `WaitForAction` ワークフローステップが一時停止すると、Image Builder からこのメッセージが送信されます。

```
{
    "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 は有料サービスです。

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。
- 検出結果の重要度を評価します。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
  - Medium
  - 高
  - [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わせてソリューションを構築するの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、認証された購入者と登録された販売者を、セキュリティ、ネットワーキング、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。販売者は、 で製品を送信するときに AWS Marketplace、製品の価格と利用規約を定義します。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。詳細については AWS Marketplace、「 とは」を参照してください AWS Marketplace。

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。
- コンポーネントを配信する AWS Marketplace イメージ製品を検索します。
- 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。
- Image Builder レシピのベースイメージとして、サブスクライブしたイメージ AWS Marketplace 製品を使用します。
- Image Builder レシピでサブスクライブした AWS Marketplace コンポーネントを使用します。

Image Builder は と統合 AWS Marketplace して、サブスクライブしたイメージ製品とコンポーネントを表示します。 AWS Marketplace Image Builder コンソールを離れることなく、Discover products ページからイメージ製品とコンポーネントを検索することもできます。

Image Builder が作成する出力 AMI には、 AWS Marketplace イメージ製品とコンポーネントの製品コードが含まれています。最終的なカスタマイズイメージには、最大 4 つの製品コードを含めることができます。

AWS Marketplace Image Builder のサブスクリプション

Image Builder コンソールの AWS Marketplace セクションのサブスクリプションページには、現在サブスクライブしている AWS Marketplace 製品のリストが表示されます。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。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。
- サブスクライブしたバージョン。
- サブスクライブした製品に含まれている関連コンポーネントがある場合、Image Builder は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。新しいレシピでサブスクライブされたイメージ製品を使用するには、サブスクライブされた製品を選択し、新しいレシピを作成するを選択します。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 コンソールから AWS Marketplace イメージ製品を検出する

このセクションでは、レシピのベース AWS Marketplace イメージとして使用するイメージ製品に焦点を当てます。関連するソフトウェア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、「Image Builder コンポーネントの一覧表示」を参照してください。 AWS Marketplace コンポーネントの検索、サブスクライブ、使用の詳細については、「」を参照してくださいAWS Marketplace コンポーネントを使用してイメージをカスタマイズする。

製品を検出する

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で、 AWS Marketplaceセクションの製品の検出を選択します。
3. イメージ製品は、Discover products ページの Image products タブで検索できます。

   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ます。

   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。例えば、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。例えば 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。

- 製品名と口ゴ。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 タブに切り替え、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。

Image Builder レシピで AWS Marketplace イメージ製品を使用する

レシピの作成ページを開き、次のようにベース 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、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。これにより、サブスクリプションタブでサブスクライブした 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 イメージ製品を検出する。
6. 通常どおり、残りの詳細を入力します。または製品サブスクリプションに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。`AWS Marketplace` コンポーネント所有者タイプリストから を選択してそれらを表示するか、CIS コンポーネント`Third party managed`に を選択します。
7. [レシピを作成する] を選択します。

最終イメージには、 AWS Marketplace イメージ製品とコンポーネントから最大 4 つの製品コードを含めることができます。選択したベースイメージとコンポーネントに 4 つ以上の製品コードが含まれている場合、Image Builder はレシピを作成しようとするとエラーを返します。

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 サービスロールが次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 サービスリンクロールには Amazon リソースネーム (ARN) を使用してください。Image Builder のサービスリンクロールについては、Image Builder での IAM サービスリンクロールの使用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] を選択してください。
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。例えば、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": 1237940039285380274899124227,
  "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": "amzn-s3-demo-bucket"
      }
    },
    "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準拠していることを確実にし、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 Image Builder 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-managed 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Image Builder 用の Amazon managed STIG 強化コンポーネント。

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 コール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。

イベントとログのモニタリング

- CloudTrail を使用した Image Builder API コールのログ記録
- Amazon CloudWatch Logs を使用した Image Builder ログのモニタリング

CloudTrail を使用した Image Builder API コールのログ記録

EC2 Image Builder は AWS CloudTrail、Image Builder のユーザー、ロール、または のサービスによって実行された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 のすべての API コール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。証跡を作成すると、Image Builder のイベントを含む CloudTrail イベントを Amazon S3 バケットに継続的に配信できます。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。CloudTrail によって収集された情報を使用して、Image Builder に対して行われたリクエスト、リクエストが行われた IP アドレス、リクエストが行われた人、リクエストが行われた日時、およびその他の詳細を特定することができます。

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。

CloudTrail のImage Builder 情報

CloudTrail は、アカウントの作成 AWS アカウント 時に で有効になります。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イベント履歴の他の AWS サービスイベントとともに CloudTrail 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、「CloudTrail イベント履歴でのイベントの表示」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、CloudTrail はログファイルを Amazon 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 Amazon S3 バケットにログファイルを配信します。さらに、CloudTrail ログで収集されたイベントデータをさらに分析して処理するように他の AWS サービスを設定できます。詳細については、次を参照してください:

- 追跡を作成するための概要
- 「CloudTrail がサポートされているサービスと統合」
- 「CloudTrail の Amazon SNS 通知の設定」
- 複数のリージョンから CloudTrail ログファイルを受け取るおよび複数のアカウントから CloudTrail ログファイルを受け取る

すべての Image Builder アクションは CloudTrail によってログに記録され、「EC2 Image Builder API リファレンス」で説明されています。例えば、`CreateImagePipeline`、`UpdateInfrastructureConfiguration`、`StartImagePipelineExec`の各アクションを呼び出すと、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、リクエストの生成者に関する情報が含まれます。アイデンティティ情報は、以下を判別するのに役立ちます。

- リクエストがルートまたは AWS Identity and Access Management (IAM) ユーザー認証情報を使用して行われたかどうか。
- 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テンポラリ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。

詳細については、「CloudTrail userIdentity エレメント」を参照してください。

Amazon CloudWatch Logs を使用した Image Builder ログのモニタリング

CloudWatch Logs 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 Logs 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、以下のImage Builder CloudWatch ロググループにストリーミングされ、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以下の権限を削除することで、CloudWatch Logs 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、「Image Builder の問題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。お客様は AWS 、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を活用できます。

セキュリティは、 AWS とお客様の間の責任共有です。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 AWS 実行されるインフラストラクチャを保護する AWS 責任があります。 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。サードパーティーの監査者は、AWS コンプライアンスプログラムコンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。EC2 Image Builder に適用されるコンプライアンスプログラムについては、AWS のサービス コンプライアンスプログ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、ユーザーは、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と保護 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

Image Builder でのデータ保護と責任 AWS 共有モデル

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。ユーザーは、このインフラストラクチャでホストされ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に関する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 AWS 責任共有モデルおよび GDPR のブログ記事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 が必須で、TLS 1.3 をお勧めします。
- 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。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「 AWS CloudTrail ユーザーガイド」のCloudTrail 証跡の使用」を参照してください。
- AWS 暗号化ソリューションと、その中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 などの高度な管理された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 140-3 検証済み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) 140-3」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を、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含め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 SDK を使用して Image Builder AWS CLI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 タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

Image Builder での暗号化とキーの管理

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、「 デベロッパーガイド」の「開始方法」を参照してください。 AWS Key Management Service

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、 AWS KMS は暗号化コンテキストを暗号化テキストに暗号化バインドします。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。

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。

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に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"
}
```


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。

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。

Identity and Access Management と Image Builder の統合

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。

サービスユーザー - 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すると、管理者に適切なアクセス許可をリクエストするのに役に立ちます。Image Builder の機能にアクセスできない場合は、Image Builder での IAM 問題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、Image Builder と IAM ポリシーおよびロールとの連携 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「アイデンティティ」を参照してください。

Image Builder と IAM ポリシーおよびロールとの連携

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については、「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のポリシーのサポート: あり

アイデンティティベースポリシーは、IAM ユーザーグループ、ユーザーのグループ、ロールなど、アイデンティティにアタッチできる JSON 許可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。ID ベースのポリシーの作成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」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学ぶには、「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: あり

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さらに付与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 でのクロスアカウントリソースアクセス」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のサポート:あり

管理者は JSON AWS 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対してどのような条件下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許可のみのアクションなど、いくつかの例外があります。また、ポリシーに複数の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アクセス許可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

ポリシーリソースのサポート: あり

管理者は JSON AWS 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対してどのような条件下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、リソースレベルの許可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: あり

管理者は JSON AWS 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対してどのような条件下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して、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 要素を指定する場合、または 1 つの Condition 要素に複数のキーを指定する場合、 AWS では 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ています。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: なし

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許可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: 一部

属性ベースのアクセス制御 (ABAC) は、属性に基づいてアクセス許可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オペレーションを許可するように ABAC ポリシーをします。

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ありです。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 認可でアクセス許可を定義する」を参照してください。ABAC をセットアップする手順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「属性ベースの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: あり

一部の AWS のサービス は、一時的な認証情報を使用してサインインすると機能しません。一時的な認証情報 AWS のサービス を使用する場合などの詳細については、IAM ユーザーガイドAWS のサービス の「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は、一時的な認証情報を使用します。たとえば、会社のシングルサインオン (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「ユーザーから IAM ロールに切り替える (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 アクセスできます 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 AWS 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

転送アクセスセッション (FAS) のサポート: あり

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 へのリクエストをリクエストする 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 のサービスリンクロール

サービスロールのサポート: 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 のサービスに許可を委任するロールを作成する」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 のサービスリンクロール

サービスリンクロールのサポート: あり

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、サービスリンクロールのアクセス許可を表示できますが、編集することはできません。

Image Builder サービスリンクロールの詳細については、Image Builder での IAM サービスリンクロールの使用 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。

単一のステートメントに複数のアクションを指定するには次のようにコンマで区切ります。

```
"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 AWS のサービスのアクション、リソース、および条件キーを参照してください。

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 JSON AWS 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対してどのような条件下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、リソースレベルの許可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": "*"
```

ARN は、リソースを識別し、名前が一意であることを確認するのに役立つ複数のノードで構成されます。名前の最後のノードには、リソースタイプ、名前、および ID の書式設定のいくつかのバリエーションが含まれます。Image Builder がリソースを作成するときは、次の形式を使用します。

```
arn:aws:imagebuilder:region:owner:resource-type/resource-name/version/
build-version
```

> **Note**
>
> ビルドバージョンは、必ずしもリソース ARN に含まれているわけではありません。ただし、GetComponent などの一部の API オペレーションでは、取得するリソースを一意に識別するためにビルドバージョンが必要です。

ベースイメージやコンポーネントなど、Image Builder がレシピで使用するリソースの場合、所有者ノードは次のいずれかになります。

- リソース所有者のアカウント番号
- Amazon マネージドリソースの場合: aws
- AWS Marketplace リソースの場合: aws-marketplace

次の例は、Linux に Amazon CloudWatch エージェントをインストールするためのマネージドコンポーネントの ARN を示しています。

```
arn:aws:imagebuilder:us-east-1:aws:component/amazon-cloudwatch-agent-linux/1.0.1/1
```

この例では、 からの架空のマネージドコンポーネントの ARN を示しています AWS Marketplace。

```
arn:aws:imagebuilder:us-east-1:aws-marketplace:component/example-linux-software-component/1.0.1
```

所有権フィルターの使用など、コンポーネントのリストを取得する方法の詳細については、「」を参照してくださいImage Builder コンポーネントの一覧表示。

ARN の例

以下は、IAM ポリシーで指定できるリソース ARNs の例です。

- インスタンス ARN

```
"Resource": "arn:aws:imagebuilder:us-east-1:111122223333:instance/i-1234567890abcdef0"
```

- 特定のアカウントのすべてのインスタンスを指定するワイルドカード (*) の例

```
"Resource": "arn:aws:imagebuilder:us-east-1:111122223333:instance/*"
```

- マネージドイメージワークフローのすべてのバージョンを指定するワイルドカード (*) の例

```
"Resource": "arn:aws:imagebuilder:us-east-1:aws:workflow/build/build-image/*"
```

- マネージドイメージ ARN

```
"Resource": "arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2-
arm64/2024.12.17/1"
```

- マネージドイメージのすべてのバージョンを指定するワイルドカード (*) の例

```
"Resource": "arn:aws:imagebuilder:us-east-1:aws:image/amazon-linux-2-arm64/x.x.x"
```

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。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:LifecyclePolicyResourceType

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ライフサイクルリソースタイプでアクセスをフィルタリングします。

このキーの値は、 AMI_IMAGEまたは のいずれかですCONTAINER_IMAGE。

利用可能性 - このキーはCreateLifecyclePolicyとUpdateLifecyclePolicy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、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は、指定されたプリンシパルが Image Builder リソースに対して、どのようなアクションをどのような条件で実行できるかを指定するものです。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、「Image Builder リソースを と共有する AWS RAM」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制御するには`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、からリソースにタグを付ける AWS CLI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 役割を引き受ける、またはクロスアカウント役割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 GetFederationToken などの AWS STS API オペレーションを呼び出します。

サービスにリンクされた役割

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ーは、サービスリンク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 はサービスリンクロールをサポートします。Image Builder サービスリンクロールの作成または管理については、Image Builder での IAM サービスリンクロールの使用を参照してください。

サービス役割

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役割を引き受けることが許可されます。この役割により、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。サービス役割は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ー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。

Image Builder で S3 バケットダウンロードアクセスのデータ境界を管理する

EC2 Image Builder は、アカウントで Image Builder ワークロードを実行するために必要なダウンロード可能なリソースを含む AWS サービス所有の S3 バケットの 2 つのクラスを維持します。データ境界を使用して環境内の Amazon S3 へのアクセスを制御する場合は、これらのバケットへのアクセスを明示的に許可する必要がある場合があります。これらのバケットを許可リストに登録するには、Amazon S3 へのアクセス制御の方法に応じて、バケット ARN またはバケット URL を使用します。

コンポーネント管理ブートストラップスクリプト (必須)

この S3 バケットには、イメージの作成に使用される EC2 インスタンスで AWSTOE アプリケーションをセットアップするためのブートストラップスクリプトが含まれています。Image Builder では、新しいイメージのビルドとテストをサポートするために、スクリプトをダウンロードするためのアクセスが必要です。

- S3 バケット ARN: `arn:<AWS partition>:s3:::ec2imagebuilder-managed-resources-<AWS Region>-prod`
- S3 バケット URL: `https://ec2imagebuilder-managed-resources-<AWS Region>.s3.<AWS Region>.<AWS partition-specific domain name>`

マネージドコンポーネント

この S3 バケットには、Amazon マネージドコンポーネントのパッケージペイロードが含まれています。Image Builder には、レシピで設定されているマネージドコンポーネントをダウンロードするためのアクセスが必要です。

- S3 バケット ARN: `arn:<AWS partition>:s3:::ec2imagebuilder-toe-<AWS Region>-prod`
- S3 バケット URL: `https://ec2imagebuilder-toe-<AWS Region>.s3.<AWS Region>.<AWS partition-specific domain name>`

Image Builder のアイデンティティベースのポリシー

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の付与を開始するには、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することで、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「ジョブ機能のAWS マネージドポリシー」を参照してください。
- 最小特権を適用する – IAM ポリシーで許可を設定する場合は、タスクの実行に必要な許可のみを付与します。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、最小特権アクセス許可とも呼ばれています。IAM を使用して許可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 でのポリシーとアクセス許可」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、サービスアクションが などの特定の を通じて使用されている場合に 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 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 でポリシーを検証する」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、MFA をオンにしてセキュリティを強化します。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 を使用した安全な API アクセス」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、次のいずれかの管理 AWS 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 の AWS マネージドポリシーを使用する を参照してください。

> **Important**
>
> Image Builder サービスリンク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。

JSON

```
{
    "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。

Image Builder 内の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、コンポーネントを使用した Image Builder イメージのカスタマイズ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ーに付与しています。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 NotIpAddress条件と aws:SourceIp条件キーを使用します。これは、 AWS全体の条件キーです。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。

JSON

```
{
  "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 の AWS マネージドポリシーを使用する

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、ロールに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。ユースケースに固有のカスタマー管理ポリシーを定義して、アクセス許可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 マネージド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) に影響します。 AWS は、新しい が起動されるか、新しい API オペレーション AWS のサービス が既存のサービスで使用できるようになったときに、 AWS マネージド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。このポリシーは、リソースの検証や、 のアカウントの現在のリソースの表示など、 AWS のサービス 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。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- 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 サービスリンクロールの存在の確認、および Image Builder サービスリンクロールの作成を行うためのアクセス権が付与されます。
- License Manager — 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。

このポリシーのアクセス許可を確認するにはAWS マネージドポリシーリファレンスの「AWSImageBuilderFullAccess」を参照してください。

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 サービスリンクロールが存在することを確認する権限が付与されます。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- 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 サービスリンク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。

このポリシーのアクセス許可を確認するにはAWS マネージドポリシーリファレンスの「AWSImageBuilderReadOnlyAccess」を参照してください。

AWSServiceRoleForImageBuilder ポリシー

このAWSServiceRoleForImageBuilderポリシーにより、Image Builder AWS のサービス はユーザーに代わって を呼び出すことができます。

アクセス許可の詳細

このポリシーは、ロールが Systems Manager で作成されたときに、Image Builder サービスリンクロールにアタッチされます。Image Builder サービスリンクロールの詳細については、Image Builder での IAM サービスリンクロールの使用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 Logs — 名前が `/aws/imagebuilder/` で始まる任意のロググループに CloudWatch Logs を作成してアップロードするためのアクセス権が付与されます。
- Amazon EC2 – Image Builder には、アカウントで EC2 インスタンスを作成および起動するイメージ (AMIs) を作成、スナップショット作成、登録するためのアクセス許可が付与されます。Image Builder は、作成または使用されているイメージ、インスタンス、ボリュームが `CreatedBy: EC2 Image Builder`または でタグ付けされている限り、必要に応じて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、キーペアを使用します`CreatedBy: EC2 Fast Launch`。

  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 EventBridge ルールを作成および管理するためのアクセス権限が付与されます。
- 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 がビルドインスタンス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。
- License Manager — Image Builder には、`license-manager:UpdateLicenseSpecificationsForResource` を介して License Manager 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 のコマンドとその呼び出しの一覧の取得、インベントリエントリの一覧の取得、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記述、インスタンス配置のサポートに必要なホストの記述、およびコマンド呼び出しの詳細の取得を行うための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、`AWS-RunPowerShellScript`、`AWS-RunShellScript`、または `AWSEC2-RunSysprep` のスクリプトファイルについて、`"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。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 サービスリンク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、クロスアカウントのイメージ配信に使用されます。

このポリシーのアクセス許可を確認するにはAWS マネージドポリシーリファレンスの「AWSServiceRoleForImageBuilder」を参照してください。

Ec2ImageBuilderCrossAccountDistributionAccess ポリシー

Ec2ImageBuilderCrossAccountDistributionAccess ポリシーは、Image Builder がターゲットリージョンのアカウントにイメージを配信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。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- 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。

このポリシーのアクセス許可を確認するにはAWS マネージドポリシーリファレンスの「Ec2ImageBuilderCrossAccountDistributionAccess」を参照してください。

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。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- 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 が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。

このポリシーのアクセス許可を確認するにはAWS マネージドポリシーリファレンスの「EC2ImageBuilderLifecycleExecutionPolicy」を参照してください。

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。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- CloudWatch Logs — 名前が `/aws/imagebuilder/` で始まる任意のロググループに CloudWatch Logs を作成してアップロードするためのアクセス権が付与されます。
- Amazon EC2 – ボリュームとスナップショットを記述し、Image Builder が作成したボリュームまたはスナップショットリソースのスナップショットを作成し、Image Builder リソースのタグを作成するアクセス許可が付与されます。
- Image Builder – Image Builder または AWS Marketplace 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 で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 で始まる Amazon S3 バケットに保存されているオブジェクト`ec2imagebuilder-`、または ISO ファイル拡張子を持つ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。

このポリシーのアクセス許可を確認するにはAWS マネージドポリシーリファレンスの「EC2InstanceProfileForImageBuilder」を参照してください。

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。

アクセス許可の詳細

このポリシーには、以下のアクセス許可が含まれています。

- CloudWatch Logs — 名前が `/aws/imagebuilder/` で始まる任意のロググループに CloudWatch Logs を作成してアップロードするためのアクセス権が付与されます。
- Amazon ECR - Amazon ECR には、コンテナイメージの取得、登録、保存、および認可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 を介して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。

このポリシーのアクセス許可を確認するにはAWS マネージドポリシーリファレンスの「EC2InstanceProfileForImageBuilderECRContainerBuilds」を参照してください。

Image Builder の AWS 管理ポリシーの更新

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ついて説明します。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
AWSServiceRoleForImageBuilder - 既存ポリシーへの更新	Image Builder は、Microsoft クライアント OS ISO ファイルをベースイメージとしてインポートできるように、サービスロールに次の変更を加えました。 • Image Builder がスナップショットを作成し、検証済みの ISO ディスクファイルからベースラインオペレーティングシステムがインポートされた AMI を作成および登録できるようにする ec2:RegisterImage を追加しました。	2024 年 12 月 30 日
EC2InstanceProfileForImageBuilder – 既存ポリシーへの更新	Image Builder は、ディスクイメージファイルからのイメージ作成をサポートするために、インスタンスプロファイルポリシーに次の変更を加えました。 • 次の ec2 アクションを追加しました: DescribeVolumes と DescribeSnapshots on all resources to retrieve details。Image Builder によって作成されたリソースに、ボリュームリソースとスナップショットリソースの CreateSnapshot、CreateTags のアクションも	2024 年 12 月 30 日

変更	説明	日付
	追加されました。「ISO」ファイル拡張子を持つリソースに s3:GetObject を追加しました。	
EC2InstanceProfileForImageBuilder - ポリシーを更新	Image Builder は、Image Builder が AWS Marketplace コンポーネントを取得できるようにEC2InstanceProfileForImageBuilder ポリシーを更新しました。	2024 年 12 月 2 日
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 でインスタンス配置のサポートを提供するために、サービスロールに次の変更が加えられました。 • ec2:DescribeHosts を追加しました。これにより、Image Builder は hostId をポーリングして、インスタンスを起動するためにいつ有効な状態にあるかを判断できます。 • 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ための API アクションである ssm:GetCommandInvocation が追加されました。	2023 年 10 月 19 日

変更	説明	日付
AWSServiceRoleForImageBuilder – 既存ポリシーへの更新	Image Builder でインスタンス配置のサポートを提供するために、サービスロールに次の変更が加えられました。 • ec2:DescribeHosts が追加されました。これにより、Image Builder は hostId をポーリングして、インスタンスを起動するのに有効な状態になったかどうかを判断できます。 • 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ための API アクションである ssm:GetCommandInvocation が追加されました。	2023 年 9 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 と inspector2: ListFindings が追加されました。これにより、Amazon Inspector がテストインスタンスのスキャンをいつ完了したかを Image Builder が判断し、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できるようになりました。 • ecr: CreateRepository が追加されました。これには、Image Builder がリポジトリに `CreatedBy: EC2 Image Builder` (タグオン作成) のタグを付ける必要がありました。また、同じ CreatedBy タグ制約と、名前が `image-builder-*` で始まるリポジトリを要求する制約が追加された ecr: TagResource (作成時のタグ付けに必要) も追加されまし	2023 年 3 月 30 日

変更	説明	日付
	た。名前の制約は、権限の昇格を防ぎ、Image Builder が作成しなかったリポジトリへの変更を防ぎます。 • `CreatedBy: EC2 Image Builder` でタグ付けされた ECR リポジトリ用に ECR: BatchDeleteImage が追加されました。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。 • 名前に `ImageBuilder-*` が含まれる Amazon EventBridge マネージドルールを作成および管理するための Image Builder のイベント権限を追加しました。	
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 呼び出しのリソースとして License Manager ライセンスを追加し、お客様がライセンス設定に関連付けられているベースイメージ AMI を使用できるようにしました。	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権限を追加しました。 • ec2: CreateTags アクションとリソースタグ条件の範囲をさらに厳しくしました。	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 アクションとリソースタグ条件の範囲をさらに厳しくしました。	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、フルアクセ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 は AWS 、管理ポリシーの変更の追跡を開始しました。	2021 年 4 月 02 日

Image Builder での IAM サービスリンクロールの使用

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。サービスリンクロールは、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。

サービスリンク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。サービスリンク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、リンクで [はい] を選択します。

Image Builder のサービスリンクロールの権限

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。サービスリンクロールは、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 はサービスリンクロールを再度作成します。

AWSServiceRoleForImageBuilder の権限を確認するには、AWSServiceRoleForImageBuilder ポリシー ページを参照してください。サービスリンク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「サービスリンクロールの権限」を参照してください。

Image Builder サービスリンクロール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する

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、未使用または古くなった Image Builder リソースの削除を参照してください。

IAM を使用して、サービスリンクロールを削除する

IAM コンソール、 AWS CLI、または AWS API を使用して、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。

Image Builder サービスリンク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。

Image Builder での IAM 問題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

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は、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。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がありません

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、既存のロールをそのサービスに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。メアリーには、ロールをサービスに渡す許可がありません。

```
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、Image Builder と IAM ポリシーおよびロールとの連携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「サードパーティーが所有する へのアクセスを提供する AWS アカウント」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) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

Image Builder のコンプライアンス検証リソース

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AWS プログラムによる対象範囲内のサービスコンプライアンス」を参照してください。一般的な情報については、AWS 「 Compliance ProgramsAssurance」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、「Artifact 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、リソース設定が内部プラクティス、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。

Image Builder イメージに AWS Marketplace または AWS Task Orchestrator and Executor (AWSTOE) のコンポーネントからのコンプライアンス製品を組み込むと、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。

Image Builder のデータの冗長性と回復性

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 ネットワークを介して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、ク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。

Image Builder イメージのパッチ管理

AWS は、毎月更新されたマネージド AMIs を提供し、最新の更新とセキュリティパッチが次のオペレーティングシステムに適用されます。これらの AMI をカスタマイズのベースイメージとして使用できます。詳細については、「サポートされるオペレーティングシステム」を参照してください。

- Amazon Linux 2、AL2023、Red Hat Enterprise Linux (RHEL)、CentOS、Ubuntu、SUSE Linux Enterprise Server を含む Linux ディストリビューション
- Windows Server 2016 以降
- macOS 10.14.x 以降

カスタムイメージの作成後は、責任共有モデルに従って、自身の責任で Amazon EC2 システムのパッチ適用を行う必要があります。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差し替えることができる場合は、ベース AMI を更新し、そのイメージに基づいてすべてのコンピューティングノードを再デプロイする方法が最も効率的と考えられます。

> **Note**
>
> macOS へのパッチ適用には、最新のマネージド AMI をベースイメージとして使用する新しいバージョンのレシピを作成し、そのレシピと他のイメージビルドリソースから、更新されたカスタムイメージをビルドすることをお勧めします。Mac インスタンスを簡単に差し替えることができない場合は、「Amazon EC2 ユーザーガイド」の「Mac インスタンス上のオペレーティングシステムとソフトウェアの更新」ページを参照してください。

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が提供するパッチコンポーネント - EC2 Image Builder には、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する次のビルドコンポーネントがあります。
  - `update-linux`
  - `update-windows`

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ことも、UpdateOSの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することもできます。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 アクションモジュールを使用するコンポーネント (Linux と Windows のみ。UpdateOS アクションモジュールは macOS ではサポートされません)

```
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
```

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適切ではないか、十分ではない場合があるため、これらは指示ではなく、有用な考慮事項と見なしてください。

- 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。
- macOS レシピに定期的にマネージド AMI の更新を適用し、最新のセキュリティパッチが適用されたインスタンスを起動する新しいイメージを作成してください。

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、「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 イメージの 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、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータサービスのオプションを設定する」を参照してください。Windows イメージについては、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータサービスのオプションを設定する」を参照してください。

ビルド後のクリーンアップが必要

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 、イメージを強化およびクリーンアップするためのベストプラクティスに従います。

macOS

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 rm -f $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 rm -f {} \;
  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 rm -f {} \;
  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"
  "/var/log/amazon/ec2/ec2-macos-init.log"
  "/var/log/amazon/ec2/ena-ethernet.log"
  "/var/log/amazon/ec2/system-monitoring.log"
)
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 rm -f {} \;
  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 rm -f {} \;
  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 rm -f /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 rm -f /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 rm -f {} \;
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 rm -f /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 rm -f {} \;
      sudo rm -rf /var/log/journal/*
fi

sudo touch /etc/machine-id
```

Linux クリーンアップスクリプトをオーバーライドする

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。例えば、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。

Input

クリーンアップセクション	削除されたファイル	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 /var/log/cloud-init.log	skip_cleanup_cloudinit_files

クリーンアップセクション	削除されたファイル	セクションファイル名をスキップする
	/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	削除されたファイル	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	削除されたファイル	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

Image Builder の問題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 AWS のサービス 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る

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、`workflow execution ID` を使用して GetWorkflowExecution と ListWorkflowStepExecutions API アクションを呼び出すことができます。

ワークフローのランタイムログを確認する

- Amazon CloudWatch Logs

  Image Builder は、詳細なワークフロー実行ログを以下の Image Builder CloudWatch Logs グループとストリームに公開します。

  LogGroup:

  ```
  /aws/imagebuilder/ImageName
  ```

  LogStream (x.x.x/x):

  ```
  ImageVersion/ImageBuildVersion
  ```

  CloudWatch Logs では、フィルタパターンを使用してログデータを検索できます。詳細については、Amazon CloudWatch Logs ユーザーガイドのフィルターパターンを使用したログデータの検索を参照してください。

- AWS CloudTrail

  アカウントで有効化されている場合、すべてのビルドアクティビティは CloudTrail にも記録されます。CloudTrail イベントは `imagebuilder.amazonaws.com` ソースでフィルタリングできます。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 バケット名とキー接頭辞を指定した場合、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する」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が起動時にオフライン
- CIS で強化されたベースイメージではビルドが失敗する
- アサートインベントリ収集が失敗する (Systems Manager 自動化)

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

パイプラインのビルドが「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。最も一般的には、インスタンスプロファイルロールに S3 バケットの PutObject 権限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 バケットの書き込み権限がありません — S3 バケットに書き込むための PutObject 権限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

パイプラインのビルドは、「status = '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上の Systems Manager エージェント の可用性を確認している間にステップが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、およびロギングが必要な場合は Amazon S3/CloudWatch 用の PrivateLink エンドポイントをセットアップしていることを確認してください。PrivateLink エンドポイントの設定の詳細については、「 を通じて AWS サービスにアクセスする 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 サービスリンクロールに追加します。
  - EC2InstanceProfileForImageBuilder
  - EC2InstanceProfileForImageBuilderECRContainerBuilds
  - AmazonSSMManagedInstanceCore

  Image Builder サービスリンクロールの詳細については、Image Builder での IAM サービスリンクロールの使用を参照してください。

Windows セカンダリディスクが起動時にオフライン

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 システムに構築されたインスタンスの詳細については、「Amazon EC2 ユーザーガイド」の「Nitro System 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する

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。

アサートインベントリ収集が失敗する (Systems Manager 自動化)

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、関連付けが既に存在する場合も、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、コンソールでの AMI イメージパイプラインの更新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、コンソールでのコンテナイメージパイプラインの更新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。

Image Builder ユーザーガイドのドキュメントの変更履歴

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: 2025-04-30

変更	説明	日付
STIG 2025 年Q2 2 四半期の更新	Windows STIG バージョンを更新し、2025 年第 2 四半期リリースの STIGS を適用しました。	2025 年 6 月 26 日
STIG 2025 年Q1 四半期の更新	Windows STIG バージョンを更新し、2025 年第 1 四半期リリースの STIGS および SCAP コンポーネントバージョンを適用しました。	2025 年 5 月 5 日
機能リリース: SSM パラメータのサポート	Image Builder は、レシピ内およびイメージ配布中の AWS Systems Manager(SSM) パラメータストアパラメータの使用をサポートするようになりました。	2025 年 4 月 30 日
STIG の Q4 アップデート	Windows STIG バージョンを更新し、2024 年第 4 四半期リリースの STIGS を適用しました。	2025 年 2 月 4 日
IAM ポリシーの更新: サービスロールポリシー	インポートされた公式 Microsoft クライアント OS ISO ファイルからのイメージ作成をサポートするように、サービスにリンクされたロー	2024 年 12 月 30 日

	ルポリシーを更新しました。詳細については、AWSServiceRoleForImageBuilder ポリシーを参照してください。	
IAM ポリシーの更新: インスタンスプロファイルポリシー	ディスクイメージファイルからのイメージ作成をサポートするようにインスタンスプロファイルロールポリシーを更新しました。詳細については、EC2InstanceProfileForImageBuilder ポリシーを参照してください。	2024 年 12 月 30 日
機能リリース: ISO から AMI	Image Builder は、Microsoft Windows 11 以降のクライアントオペレーティングシステム用の公式 ISO ディスクファイルからイメージを作成できるようになりました。	2024 年 12 月 30 日
STIG の Q4 アップデート	Linux STIG バージョンを更新し、STIGS を 2024 年第 4 四半期リリースに適用しました。Linux コンポーネントの 2 つの新しい入力パラメータに関する情報を追加しました。	2024 年 12 月 10 日
IAM ポリシーの更新: インスタンスプロファイルポリシー	インスタンスプロファイルポリシーを更新して、 AWS Marketplace コンポーネントを取得するアクセスを許可しました。詳細については、EC2InstanceProfileForImageBuilder ポリシーを参照してください。	2024 年 12 月 2 日

機能リリース: AWS Marketplace コンポーネント	AWS Marketplace ソフトウェアコンポーネントのサポートを追加しました。	2024 年 12 月 2 日
機能リリース: macOS サポート	macOS イメージのサポートが追加されました。	2024 年 10 月 22 日
での論理演算子のドキュメントサポート AWS Task Orchestrator and Executor	コンポーネントドキュメントで条件式を追加または変更するには、論理演算子を使用します。	2024 年 8 月 16 日
の条件付きコンストラクトのドキュメントサポート AWS Task Orchestrator and Executor	コンポーネントドキュメントでコンポーネントアクションの制御フローを指示するには、「if」ステートメントなどの条件構造を使用します。	2024 年 8 月 16 日
での比較演算子のドキュメントサポート AWS Task Orchestrator and Executor	コンポーネントドキュメントで値を比較するには、比較演算子を使用します。	2024 年 8 月 16 日
Assert 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。	2024 年 8 月 14 日
オペレーティングシステムのサポート	RHEL 9 および Ubuntu 24.04 LTS のサポートを追加しました。	2024 年 8 月 2 日
ドキュメントの更新: コンテンツの再編成	ドキュメントを再編成して表現とナビゲーションを改善しました。	2024 年 6 月 21 日

STIG の Q2 アップデート	Linux の STIG バージョンを更新し、2024 年第 2 四半期リリースの STIG を適用しました。RHEL 9、CentOS Stream 9、Ubuntu 22.04 のサポートを追加しました。Windows バージョンに変更はありません。	2024 年 5 月 10 日
STIG の Q1 アップデート	Linux の STIG バージョンを更新し、2024 年第 1 四半期リリースの STIG を適用しました。Windows バージョンに変更はありません。	2024 年 2 月 23 日
STIG の Q1 アップデート	Linux の STIG バージョンを更新し、2024 年第 1 四半期リリースの STIG を適用しました。Windows バージョンに変更はありません。	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 の Q4 アップデート	Linux の STIG バージョンを更新し、2023 年第 4 四半期リリースの STIG を適用しました。Windows バージョンに変更はありません。また、Linux と Windows の SCAP が、新しいコンポーネント、ソフトウェア、ベンチマーク番号に更新されました。	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。	2023 年 11 月 17 日
IAM ポリシーの更新: サービスロール	インスタンス配置のサポートのためにサービスリンクロールポリシーを更新しました。詳細については、AWSServiceRoleForImageBuilder ポリシーを参照してください。	2023 年 10 月 19 日
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、サードパーティー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。AL2023 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	AWS リージョンアジアパシフィック (ハイデラバード) 、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 アプリケーションの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
機能リリース: 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簡単に検索して新しいカスタムイメージのベースラインとして使用する AWS Marketplace 統合を追加しました。	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
「ドキュメントを実行」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
機能リリース: 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
機能リリース: 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、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と、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 インテグレーションコンテンツの追加	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 との統合	Image Builder を関連する AWS のサービスイベントに接続し、EventBridge で定義されたルールに基づいてイベントを開始できるようにする EventBridge サポートが追加されました。	2021 年 8 月 18 日
ドキュメントの更新: AWSTOE ページを並べ替える	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
機能リリース: 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

機能リリース: 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
機能リリース: 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
cron 式の更新	Image Builder の cron 処理が更新され、cron 式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
機能リリース: 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、今後の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: パイプライン編集の更新	「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するための AutoScaling の使用は廃止されました。	2020 年 6 月 15 日
AWS PrivateLink を介した接続のサポート	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、NAT デバイス、VPN 接続、または AWS Direct Connect 接続なしで 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 AWS PrivateLink 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 STIG AWSTOE 準拠イメージを作成できます。	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。