aws

ユーザーガイド

AWS HealthOmics

Version latest

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

AWS HealthOmics: ユーザーガイド

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon の商標およびトレードドレスはAmazon 以外の製品およびサービスに使用することはできません。また、お客様に誤解を与える可能性がある形式で、または Amazon の信用を損なう形式で使用することもできません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

とは AWS HealthOmics

AWS HealthOmics は、バイオインフォマティクス、研究者、科学者などのユーザーが、ゲノミクスやその他の生体データを保存、クエリ、分析、およびインサイトを生成できるようにする AWS サービスです。研究組織や臨床組織のゲノム情報の保存と分析のプロセスを簡素化および高速化し、科学的発見とインサイトの生成を高速化します。

HealthOmics には 3 つの主要コンポーネントがあります。HealthOmics Storage を使用すると、ペタバイトのゲノミクスデータをギガベースあたり低コストで効率的に保存および共有できます。HealthOmics Analytics は、マルチオミクスおよびマルチモーダル分析用のゲノムデータを準備する方法を簡素化します。HealthOmics ワークフローは、バイオインフォマティクス計算の基盤となるインフラストラクチャを自動的にプロビジョニングおよびスケーリングします。

トピック

重要な注意点

HealthOmics は、専門的な医療上のアドバイス、診断、または治療に代わるものではなく、疾患や病状の修復、治療、緩和、予防、または診断を目的としていません。お客様は、臨床上の意思決定を通知することを目的としたサードパーティー製品に関連するものを含め AWS HealthOmics、 の使用の一環として人間によるレビューを代行する責任があります。

HealthOmics は、データの転送、保存、フォーマット、表示、およびワークフロー管理のためのインフラストラクチャと設定のサポートの提供のみを目的としています。 AWS HealthOmics は、バリアント呼び出しやゲノム分析と解釈を直接実行することを目的としたものではありません。 AWS HealthOmics は、臨床ラボテストやその他のデバイスデータ、結果、結果の解釈や分析を目的としたものではなく、ゲノム分析での使用を目的としたサードパーティー製ツールに代わるものではありません。

HealthOmics の概念

このトピックでは、このガイドで使用されている HealthOmics の用語を理解するのに役立つ、HealthOmics に固有の主要な概念と用語の定義について説明します。

トピック

[Storage (ストレージ)]

データストレージは、ゲノミクスシーケンスおよび関連情報についてはシーケンスストアに、すべての参照ゲノムについてはリファレンスストアに分割されます。以下の用語では、HealthOmics に固有の実装について説明します。

- シーケンスストア – ゲノミクスファイルを保存するためのデータストア。HealthOmics 内に 1 つ以上のシーケンスストアを持つことができます。アクセス許可と AWS KMS 暗号化をシーケンスストアに設定して、データにアクセスできるユーザーを制御できます。
- リードセット – リードセットは、ゲノム読み取りの抽象化であり、FASTQ、BAM、または CRAM 形式で保存されます。リードセットはシーケンスストアにインポートし、メタデータで注釈を付けることができます。属性ベースのアクセスコントロール (ABAC) を使用して、読み取りセットにアクセス許可を適用できます。
- リファレンス – ゲノムリファレンスは、ゲノム内の特定の読み取り、または読み取りグループがマッピングされている場所を特定するために、読み取りとともに使用されます。これらは FASTA 形式で、リファレンスストアに保存されます。
- リファレンスストア – リファレンスゲノムを保存するためのデータストア。各アカウントとリージョンに 1 つのリファレンスストアを持つことができます。

分析

HealthOmics Analytics を使用して、ゲノムデータを変換および分析できます。バリアントストアまたは注釈ストアを作成して、クエリの追加情報を含めます。

- バリアントストア – バリアントデータを母集団規模で保存するデータストア。バリアントストアは、ゲノムバリアントコール形式 (gVCF) と VCF 入力の両方をサポートします。

- 注釈ストア – TSV/CSV、VPC、一般機能形式 (GFF3) ファイルなどの注釈データベースを表すデータストア。Annotation Stores は、インポート中にバリアントストアと同じ座標系にマッピングされます。

ワークフロー

HealthOmics ワークフローを使用すると、ゲノミクスデータを処理および分析できます。

- ワークフロー – パラメータやツールへの参照を含むエンドツーエンドプロセスの全体的な定義。ワークフロー定義は、WDL、Nextflow、または CWL として表現できます。作成された各ワークフローには一意の識別子があります。
- 実行 — ワークフローの 1 回の呼び出し。個々の実行では、定義された入力データを使用して出力が生成されます。作成された各実行には一意の識別子があります。
- タスク – 実行内の個々のプロセス。HealthOmics ワークフローは、これらの定義されたコンピューティング仕様を使用してタスクを実行します。各タスクには一意の識別子があります。
- 実行グループ – 最大 vCPU、最大継続時間、または最大同時実行を設定して、実行ごとに使用されるコンピューティングリソースを制限するのに役立つ実行のグループ。実行グループ内で実行の優先順位を指定および設定できます。たとえば、優先度の低い実行の前に優先度の高い実行を実行し、優先度キューを作成するように指定できます。実行グループの使用はオプションであり、各実行グループには一意の識別子があります。

HealthOmics の機能

HealthOmics には以下の機能があります。

- HealthOmics Storage — ペタバイトの未加工のゲノムデータを、ギガベースあたり低コストで効率的に保存および共有できます。
- HealthOmics Analytics — マルチオミクスおよびマルチモーダル分析用のゲノミクスデータを準備する方法を簡素化します。
- HealthOmics ワークフロー — バイオインフォマティクスワークフローの基盤となるインフラストラクチャを自動的にプロビジョニングおよびスケーリングします。

各コンポーネントを個別に使用することも、統合されたend-to-endソリューションの一部として使用することもできます。

HealthOmics には以下の利点があります。

- ゲノムデータを安全に保存して組み合わせる — HealthOmics は、 や Amazon Athena などの AWS Lake Formation 他の AWS サービスと統合します。ゲノミクスデータを安全に保存し、クエリを実行したり、履歴データと組み合わせたりして、より良い診断とパーソナライズされた治療計画を実現できます。
- 患者のプライバシーを保護する — HealthOmics は HIPAA の対象となります。また、IAM および Amazon CloudWatch と統合されているため、データアクセスを制御およびログ記録し、分析でのデータの使用方法を追跡できます。
- スケーリング用に構築 — 請求を簡素化し、新しいコラボレーションツールを使用して、大規模な母集団データ分析をサポートします。
- 効率を最大化する — 自動化されたワークフローと統合されたツールを使用して、データ処理と分析を合理化します。

HealthOmics は、以下の生体医療アプリケーションに使用できます。

- 母集団シーケンス — 数千のゲノムを一度にクエリして、母集団全体でゲノムのバリエーションがどのような類似点にマッピングされるかを理解します。
- 臨床ゲノミクス — シーケンサーの出力から報告可能なデータまで、再現可能なゲノミクスワークフローを構築します。また、大量のスループットを最適化し、優先度の高い臨床サンプルのコンピューティング要件を設定して、ターンアラウンド時間を短縮することもできます。
- 臨床試験 — ゲノム分析を臨床試験に統合して、新しい薬剤候補の有効性をよりよく理解します。管理機関からの規制を満たすために、長期的なコスト削減とデータ出所により、臨床試験を簡素化および高速化します。
- 研究とイノベーションの強化 — 行と列ベースのアクセス制御を組み込み、匿名化されたゲノムデータのストレージ、アクセス、分析を合理化して制御します。

関連サービス

以下のサービスは HealthOmics で動作します。

- Amazon Elastic Container Registry – 各プライベートワークフローは、Amazon ECR イメージ (プライベート Amazon ECR リポジトリ内) を使用して、ワークフローの実行に必要なすべての実行可能ファイル、ライブラリ、スクリプトを含めます。
- Amazon Simple Storage Service – Amazon S3 は、ストアデータとワークフローデータ用のファイルストレージを提供します。
- AWS Lake Formation – Lake Formation は、分析データストアへのデータアクセスを管理します。

- Amazon Athena – Athena を使用してバリアントストアでクエリを実行します。
- Amazon SageMaker AI – SageMaker AI を使用して、Jupyter ノートブックを使用して HealthOmics タスクを実行します。

AWS HealthOmics のリージョンとエンドポイント

リージョンとエンドポイントの完全なリストについては、AWS 「 全般のリファレンス」を参照してください。

デフォルトでアクティブになっている AWS リージョンに加えて、アクティブ化する必要があるオプトインリージョンもあります。リージョンをアクティブ化または非アクティブ化する方法の詳細については、「 アカウント管理ガイド」の「アカウントで使用できる AWS リージョンを指定する」を参照してください。 AWS

HealthOmics にアクセスする方法

AWS HealthOmics 機能には、 マネジメントコンソール、 CLI、 SDKs、または API を使用してアクセスできます。

- AWS マネジメントコンソール – HealthOmics へのアクセスに使用できるウェブインターフェイスを提供します。
- AWS Command Line Interface （ AWS CLI) – Windows、macOS AWS HealthOmics、Linux など、さまざまな AWS サービス用のコマンドを提供します。のインストールの詳細については AWS CLI、「」を参照してくださいAWS Command Line Interface。
- AWS SDKs – さまざまなプログラミング言語とプラットフォーム (Java、Python、Ruby、.NET、iOS、Android を含む) のライブラリとサンプルコードで構成される SDKs (ソフトウェア開発キット) AWS を提供します。SDKs を使用すると、プログラムで HealthOmics を使用するのに便利です。詳細については、AWS 「 SDK デベロッパーセンター」を参照してください。
- AWS API – API オペレーションを使用して、プログラムで HealthOmics にアクセスして管理できます。詳細については、HealthOmics API リファレンス」を参照してください。

詳細はこちら

HealthOmics の詳細については、以下のワークショップとチュートリアルを参照してください。

- HealthOmics ワークショップ – HealthOmics エンドツーエンドワークショップ
- AWS ゲノミクスリソース – ゲノミクスに関連するパブリック Amazon ECR リポジトリ
- Python チュートリアル – HealthOmics ストレージ、分析、ワークフローをカバーする GitHub の Jupyter Notebook チュートリアル

AWS 以下を提供する追加の HealthOmics ツールに精通してください。

- WDL linter – WDL 用の HealthOmics linter
- Nextflow linter – Nextflow の HealthOmics linter
- HealthOmics Amazon ECR ヘルパーツール – HealthOmics 用の Amazon ECR ヘルパーツール
- GitHub の HealthOmics ツール – HealthOmics を操作するためのツール (転送マネージャー、URI パーサー、Omics 再実行、アナライザーの実行) 。

HealthOmics のセットアップ

をセットアップするには AWS HealthOmics、 にサインアップし AWS アカウント、管理ユーザーを作成し、追加のユーザーのアクセスを安全に管理します。

トピック

にサインアップする AWS アカウント

がない場合は AWS アカウント、次の手順を実行して作成します。

にサインアップするには AWS アカウント

1. https://portal.aws.amazon.com/billing/signup を開きます。
2. オンラインの手順に従います。

   サインアップ手順の一環として、電話またはテキストメッセージを受け取り、電話キーパッドで検証コードを入力します。

   にサインアップすると AWS アカウント、 AWS アカウントのルートユーザー が作成されます。ルートユーザーには、アカウントのすべての AWS のサービス とリソースへのアクセス権があります。セキュリティベストプラクティスとして、ユーザーに管理アクセス権を割り当て、ルートユーザーアクセスが必要なタスクの実行にはルートユーザーのみを使用するようにしてください。

AWS サインアッププロセスが完了すると、 から確認メールが送信されます。https://aws.amazon.com/ の [マイアカウント] をクリックして、いつでもアカウントの現在のアクティビティを表示し、アカウントを管理することができます。

管理アクセスを持つユーザーを作成する

にサインアップしたら AWS アカウント、日常的なタスクにルートユーザーを使用しないように AWS アカウントのルートユーザー、 のセキュリティを確保し AWS IAM Identity Center、 を有効にして管理ユーザーを作成します。

を保護する AWS アカウントのルートユーザー

1. ルートユーザーを選択し、 AWS アカウント E メールアドレスを入力して、アカウント所有者AWS Management Consoleとして にサインインします。次のページでパスワードを入力します。

   ルートユーザーを使用してサインインする方法については、AWS サインイン ユーザーガイドのルートユーザーとしてサインインするを参照してください。
2. ルートユーザーの多要素認証 (MFA) を有効にします。

   手順については、IAM ユーザーガイドの AWS アカウント「ルートユーザー (コンソール) の仮想 MFA デバイスを有効にする」を参照してください。

管理アクセスを持つユーザーを作成する

1. IAM アイデンティティセンターを有効にします。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「AWS IAM Identity Centerの有効化」を参照してください。
2. IAM アイデンティティセンターで、ユーザーに管理アクセスを付与します。

   を ID ソース IAM アイデンティティセンターディレクトリ として使用する方法のチュートリアルについては、AWS IAM Identity Center 「 ユーザーガイド」の「デフォルトを使用してユーザーアクセスを設定する IAM アイデンティティセンターディレクトリ」を参照してください。

管理アクセス権を持つユーザーとしてサインインする

- IAM アイデンティティセンターのユーザーとしてサインインするには、IAM アイデンティティセンターのユーザーの作成時に E メールアドレスに送信されたサインイン URL を使用します。

  IAM Identity Center ユーザーを使用してサインインする方法については、AWS サインイン 「 ユーザーガイド」の AWS 「 アクセスポータルにサインインする」を参照してください。

追加のユーザーにアクセス権を割り当てる

1. IAM アイデンティティセンターで、最小特権のアクセス許可を適用するというベストプラクティスに従ったアクセス許可セットを作成します。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「権限設定を作成する」を参照してください。
2. グループにユーザーを割り当て、そのグループにシングルサインオンアクセス権を割り当てます。

   手順については、「AWS IAM Identity Center ユーザーガイド」の「グループの結合」を参照してください。

HealthOmics の IAM アクセス許可を作成する

HealthOmics を使用するには、次の IAM アクセス許可を設定します。

- アカウントのユーザーが HealthOmics にアクセスするための IAM アイデンティティベースのポリシー。
- HealthOmics がユーザーに代わってリソースにアクセスするための IAM サービスロール。
- ユーザーと HealthOmics サービスが リソースにアクセスするための、他の サービス (Lake Formation や Amazon ECR など) のアクセス許可。

HealthOmics の IAM アクセス許可の設定の詳細については、「」を参照してくださいHealthOmics の IAM アクセス許可。

外部コードリポジトリに接続する

を使用すると AWS HealthOmics、 を介して Git ベースのリポジトリを使用してワークフローを管理できます AWS CodeConnections。HealthOmics は、この接続を使用してソースコードリポジトリにアクセスします。

外部コードリポジトリを使用する前に、接続のセットアップガイドに従って操作を開始します AWS CodeConnections。 AWS アカウントに適切な IAM ポリシーとアクセス許可が作成されていることを確認します。サポートされている Git プロバイダーのリストと詳細については、「どのサードパーティープロバイダーに接続を作成できますか？」を参照してください。

接続を作成する

任意のリポジトリプロバイダーとの接続を作成するには、「接続の作成」チュートリアルに従います。

HealthOmics での Amazon Q CLI の使用

Amazon Q CLI は との自然言語インタラクションを提供するため AWS HealthOmics、会話コマンドを使用して複雑なゲノムワークフローと分析タスクを実行できます。Amazon Q CLI を使用するには、Amazon Q がリソースにアクセスするための HealthOmics およびその他の サービス (CloudWatch、Amazon ECR、Amazon S3 など) の IAM アクセス許可を必ず設定してください。

HealthOmics エージェント生成 AI チュートリアルでは、コンテキストファイルを設定し、Amazon Q CLI で AWS HealthOmics ワークフローを作成、実行、最適化するためのstep-by-stepガイダンスを提供します。

HealthOmics の開始方法

HealthOmics の使用を開始するには、HealthOmics の IAM アクセス許可とロールが適切に設定されていることを確認します。

HealthOmics コンソールでの Ready2Run ワークフローの使用

次の演習では、Ready2Run ワークフローを使用する方法を示します。Ready2Run ワークフローには、ワークフローの実行に必要なパラメータとツールリファレンスが事前設定されています。ワークフローパブリッシャーはサンプルデータを提供するため、独自のデータを作成する必要はありません。

1. HealthOmics コンソールを開きます。
2. 左上のナビゲーションペイン (≡) を選択し、Ready2Run ワークフローを選択します。
3. Ready2Run ワークフローページで、ESMFold for up to 800 residuesワークフローを選択します。コンソールで、そのワークフローの詳細ページが開きます。
4. 詳細タブには、ワークフローに関する情報が表示されます。ワークフローを試すには、ページの右上で実行の開始を選択します。
5. 実行の詳細の指定 ページで、実行名を入力します。
6. 実行出力の Amazon S3 の場所を入力または選択します。
7. Run metadata retention mode で、runmeta data を保持するか削除するかを選択します。
8. サービスロールパネルで、新しいサービスロールを作成して使用します。
9. [次へ] を選択します。
10. パラメータ値の追加ページで、Ready2Run テストデータを使用してワークフローを実行するを選択します。
11. [次へ] を選択します。
12. 入力を確認し、実行の開始を選択します。

Amazon Q CLI のプロンプト例

Amazon Q CLI は、自然言語コマンド AWS HealthOmics を使用して、 でゲノムワークフローと分析タスクを実行できます。次のプロンプト例では、ワークフローの作成、実行の管理、ゲノムデータの分析を行うことができます。HealthOmics の詳細とプロンプトの例については、GitHub の HealthOmics エージェント生成 AI チュートリアルを参照してください。

- HealthOmics で実行される`main.wdl`ため、WDL 1.1 ワークフローファイルを作成します。ワークフローは、参照ゲノムを fastq ファイルの入力とペアとして受け取ります。BWA を使用して参照ゲノムのインデックスを作成し、fastq ファイルの各ペアをリファレンスにマッピングします。最後に、マッピングされた各 BAM を 1 つの BAM ファイルにマージし、このファイルと bai インデックスを出力します。」
- 「ワークフローをパッケージ化して HealthOmics で作成する」
- 「inputs.json ファイルを更新して、Amazon S3 バケットの実際のファイルを使用します`omics-my-bucket-with-genome-data`」(特定の Amazon S3 バケットの場所を指定するか、Amazon Q に調査させます)
- 「Amazon ECR リポジトリで適切なコンテナを検索し、これらを使用するように input.json を更新する」
- 「ワークフローの実行時に使用する適切な IAM ロールを検索または作成する」
- 「ワークフローの実行キャッシュを作成する」
- HealthOmics でワークフローを実行する」
- 「実行のステータスを確認する」

> **Warning**
>
> Amazon Q CLI を使用する場合は、先に進む前に、生成されたすべてのコンテンツと提案されたアクションを確認してください。応答品質を向上させ、ワークフローの要件に合わせてフィードバックを提供します。詳細については、「Amazon Q のセキュリティ上の考慮事項とベストプラクティス」を参照してください。

HealthOmics のプライベートワークフロー

独自のワークフロー定義を作成する場合は、プライベートワークフローを使用します。ワークフロー定義は、ワークフローに関する情報を指定し、ワークフロータスクを定義します。実行はワークフローの単一の呼び出しであり、タスクは実行内の単一のプロセスです。

HealthOmics は、Workflow Description Language (WDL)、Common Workflow Language (CWL)、または Nextflow で作成するワークフロー定義をサポートしています。

HealthOmics ワークフローには、次のオプション機能があります。

- Run groups – プライベートワークフローを実行グループに追加して、コンピューティングの使用状況を制御できます。実行グループは、最大同時実行数や最大実行期間など、一連のリソース制限を共有するワークフロー実行のコレクションです。これらの制限を設定して、実行グループが消費するコンピューティングリソースを制御します。
- Call caching – 呼び出しキャッシュを使用してタスク出力を保存および再利用できるため、実行時間が短縮され、コンピューティングコストが削減されます。
- Sharing workflows – プライベートワークフローは、同じリージョン AWS アカウント 内の他の と共有できます。
- Workflow versions – プライベートワークフローのバージョンを作成できます。ワークフローのバージョニングにより、ユーザーは更新された機能の使用を開始するタイミングを選択できます。ワークフローバージョンはイミュータブルであり、ワークフローと同じレベルのデータ出所を提供します。

ワークフローの IAM アクセス許可の設定については、「」を参照してくださいHealthOmics の IAM アクセス許可。

HealthOmics プライベートワークフローの使用方法の完全な例については、HealthOmics Github チュートリアル」またはHealthOmics」を参照してください。

トピック

- HealthOmics でのプライベートワークフローの作成
- README ファイルの使用
- HealthOmics でのワークフローのバージョニング
- HealthOmics の実行を開始する

- HealthOmics で実行グループと実行グループを削除する
- HealthOmics 実行グループの作成
- HealthOmics 実行のコールキャッシュ
- HealthOmics ワークフローの共有

HealthOmics でのプライベートワークフローの作成

プライベートワークフローは、ワークフローを作成する前に作成および設定するさまざまなリソースによって異なります。

- Workflow definition file: WDL、Nextflow、または で記述されたワークフロー定義ファイルCWL。ワークフロー定義は、ワークフローを使用する実行の入力と出力を指定します。また、コンピューティングやメモリの要件など、ワークフローの実行タスクと実行タスクの仕様も含まれています。ワークフロー定義ファイルは `.zip`形式である必要があります。詳細については、HealthOmics のワークフロー定義ファイル」を参照してください。
- Parameter template file (オプション): で記述されたパラメータテンプレートファイルJSON。ファイルを作成して実行パラメータを定義するか、HealthOmics がパラメータテンプレートを生成します。詳細については、HealthOmics ワークフローのパラメータテンプレートファイル」を参照してください。
- Amazon ECR container images: ワークフローのコンテナイメージを作成し、プライベート Amazon ECR リポジトリに保存します。
- Sentieon licenses (オプション): プライベートワークフローでSentieonソフトウェアを使用するSentieonライセンスをリクエストします。

必要に応じて、ワークフローの作成前または作成後に、ワークフロー定義で linter を実行できます。このlinterトピックでは、HealthOmics で使用できる linters について説明します。

トピック

- HealthOmics のワークフロー定義ファイル
- HealthOmics ワークフローのパラメータテンプレートファイル
- プライベートワークフロー用の Amazon ECR のコンテナイメージ
- プライベートワークフローの Sentieon ライセンスのリクエスト
- HealthOmics のワークフローlinters

- ワークフローの作成または更新

HealthOmics のワークフロー定義ファイル

ワークフロー定義を使用して、ワークフロー、実行、および実行内のタスクに関する情報を指定します。ワークフロー定義言語を使用して、1 つ以上のファイルにワークフロー定義を作成します。HealthOmics は、WDL、Nextflow、または CWL で記述されたワークフロー定義をサポートしています。これらの各言語の詳細については、「」を参照してください。 HealthOmics ワークフローのワークフロー定義の記述

ワークフロー定義では、次のタイプの情報を指定します。

- Language version – ワークフロー定義の言語とバージョン。
- Compute and memory – ワークフロー内のタスクのコンピューティングとメモリの要件。
- Inputs – ワークフロータスクへの入力の場所。詳細については、「HealthOmics 実行入力」を参照してください。
- Outputs – タスクが生成する出力を保存する場所。
- Task resources – 各タスクのコンピューティングとメモリの要件。
- Accelerators – アクセラレーターなど、タスクに必要なその他のリソース。

トピック

- HealthOmics ワークフロー定義の要件
- HealthOmics ワークフロー定義言語のバージョンサポート
- HealthOmics タスクのコンピューティングとメモリの要件
- HealthOmics ワークフロー定義のタスク出力
- HealthOmics ワークフロー定義のタスクリソース
- HealthOmics ワークフロー定義のタスクアクセラレーター
- HealthOmics ワークフローのワークフロー定義の記述

HealthOmics ワークフロー定義の要件

HealthOmics ワークフロー定義ファイルは、次の要件を満たしている必要があります。

- タスクでは、入出力パラメータ、Amazon ECR コンテナリポジトリ、メモリや CPU 割り当てなどのランタイム仕様を定義する必要があります。

- IAM ロールに必要なアクセス許可があることを確認します。
  - ワークフローは、Amazon S3 などの AWS リソースからの入力データにアクセスできます。
  - ワークフローは、必要に応じて外部リポジトリサービスにアクセスできます。
- ワークフロー定義で出力ファイルを宣言します。中間実行ファイルを出力場所にコピーするには、それらをワークフロー出力として宣言します。
- 入力場所と出力場所は、ワークフローと同じリージョンにある必要があります。
- HealthOmics ストレージワークフロー入力は `ACTIVE`ステータスである必要があります。HealthOmics は `ARCHIVED`ステータスの入力をインポートせず、ワークフローが失敗します。Amazon S3 オブジェクト入力の詳細については、「」を参照してくださいHealthOmics 実行入力。
- ZIP アーカイブに単一のワークフロー定義または「main」という名前のファイルが含まれている場合、ワークフローmainの場所はオプションです。
  - パスの例: `workflow-definition/main-file.wdl`
- Amazon S3 またはローカルドライブからワークフローを作成する前に、ワークフロー定義ファイルとサブワークフローなどの依存関係の zip アーカイブを作成します。
- ワークフロー内の Amazon ECR コンテナを Amazon ECR アクセス許可を検証するための入力パラメータとして宣言することをお勧めします。

Nextflow に関するその他の考慮事項：

- /bin

  Nextflow ワークフロー定義には、実行可能スクリプトを含む /bin フォルダを含めることができます。このパスには、タスクへの読み取り専用アクセスと実行可能アクセスがあります。これらのスクリプトに依存するタスクは、適切なスクリプトインタープリタで構築されたコンテナを使用する必要があります。ベストプラクティスは、インタプリタを直接呼び出すことです。例:

```
process my_bin_task {
   ...
   script:
      """
      python3 my_python_script.py
      """
}
```

- includeConfig

Nextflow ベースのワークフロー定義には、パラメータ定義の抽象化やリソースプロファイルの処理に役立つ nextflow.config ファイルを含めることができます。複数の環境で Nextflow パイプラインの開発と実行をサポートするには、includeConfig ディレクティブを使用してグローバル設定に追加する HealthOmics 固有の設定を使用します。移植性を維持するには、次のコードを使用して HealthOmics で実行されている場合にのみ ファイルを含めるようにワークフローを設定します。

```
// at the end of the nextflow.config file
if ("$AWS_WORKFLOW_RUN") {
    includeConfig 'conf/omics.config'
}
```

- Reports

  HealthOmics は、エンジン生成のダグ、トレース、実行レポートをサポートしていません。GetRun と GetRunTask API コールの組み合わせを使用して、トレースレポートと実行レポートに代わるものを生成できます。

CWL に関するその他の考慮事項：

- Container image uri interpolation

  HealthOmics では、DockerRequirement の dockerPull プロパティをインライン javascript 式にすることができます。 DockerRequirement 例:

```
requirements:
  DockerRequirement:
    dockerPull: "$(inputs.container_image)"
```

  これにより、ワークフローへの入力パラメータとしてコンテナイメージ URIs を指定できます。

- Javascript expressions

  Javascript 式は`strict mode`準拠している必要があります。

- Operation process

  HealthOmics は CWL オペレーションプロセスをサポートしていません。

HealthOmics ワークフロー定義言語のバージョンサポート

HealthOmics は、Nextflow、WDL、または CWL で記述されたワークフロー定義ファイルをサポートしています。以下のセクションでは、これらの言語の HealthOmics バージョンサポートについて説明します。

トピック

WDL バージョンのサポート

HealthOmics は、WDL 仕様のバージョン 1.0、1.1、および 開発バージョンをサポートしています。

すべての WDL ドキュメントには、準拠する仕様のバージョン (メジャーとマイナー) を指定するバージョンステートメントが含まれている必要があります。バージョンの詳細については、「WDL バージョニング」を参照してください。

WDL 仕様のバージョン 1.0 および 1.1 は、 `Directory`タイプをサポートしていません。入力または出力に `Directory`タイプを使用するには、ファイルの最初の行developmentでバージョンを に設定します。

```
version development  # first line of .wdl file
     ... remainder of the file ...
```

CWL バージョンのサポート

HealthOmics は、CWL 言語のバージョン 1.0、1.1、および 1.2 をサポートしています。

CWL ワークフロー定義ファイルで言語バージョンを指定できます。CWL の詳細については、「CWL ユーザーガイド」を参照してください。

Nextflow バージョンのサポート

HealthOmics は、3 つの Nextflow 安定バージョンをサポートしています。Nextflow は通常、6 か月ごとに安定したバージョンをリリースします。HealthOmics は毎月の「エッジ」リリースをサポートしていません。

HealthOmics は各バージョンでリリースされた機能をサポートしていますが、プレビュー機能はサポートしていません。

サポートバージョン

HealthOmics は、次の Nextflow バージョンをサポートしています。

- Nextflow v22.04.01 DSL 1 および DSL 2
- Nextflow v23.10.0 DSL 2 (デフォルト)
- Nextflow v24.10.8 DSL 2

ワークフローをサポートされている最新バージョン (v24.10.8) に移行するには、Nextflow アップグレードガイドに従います。

Nextflow 移行ガイドの以下のセクションで説明するように、Nextflow v23 から v24 に移行するときに重大な変更があります。

- 24.04 の重大な変更
- 24.10 の重大な変更

Nextflow バージョンを検出して処理する

HealthOmics は、指定した DSL バージョンと Nextflow バージョンを検出します。これらの入力に基づいて、実行する最適な Nextflow バージョンを自動的に決定します。

DSL バージョン

HealthOmics は、ワークフロー定義ファイルでリクエストされた DSL バージョンを検出します。たとえば、 を指定できます`nextflow.enable.dsl=2`。

HealthOmics はデフォルトで DSL 2 をサポートしています。ワークフロー定義ファイルで指定されている場合、DSL 1 との下位互換性があります。

- DSL 2 を指定した場合、Nextflow v22.04.0 または v24.10.8 を指定しない限り、HealthOmics は Nextflow v23.10.0 を実行します。
- DSL 1 を指定した場合、HealthOmics は Nextflow v22.04 DSL1 (DSL 1 を実行する唯一のサポートされているバージョン) を実行します。

- DSL バージョンを指定しない場合、または HealthOmics が何らかの理由で DSL 情報を解析できない場合 (ワークフロー定義ファイルの構文エラーなど)、HealthOmics はデフォルトで DSL 2 になり、Nextflow v23.10.0 を実行します。
- 最新の Nextflow バージョンとソフトウェア機能を利用するためにワークフローを DSL 1 から DSL 2 にアップグレードするには、「DSL 1 からの移行」を参照してください。

Nextflow バージョン

HealthOmics は、このファイルを指定すると、Nextflow 設定ファイル (nextflow.config) でリクエストされた Nextflow バージョンを検出します。含まれている設定による予期しない上書きを避けるため、ファイルの最後に `nextflowVersion`句を追加することをお勧めします。詳細については、「Nextflow 設定」を参照してください。

次の構文を使用して、Nextflow バージョンまたはバージョンの範囲を指定できます。

```
// exact match
manifest.nextflowVersion = '1.2.3'

// 1.2 or later (excluding 2 and later)
manifest.nextflowVersion = '1.2+'

// 1.2 or later
manifest.nextflowVersion = '>=1.2'

// any version in the range 1.2 to 1.5
manifest.nextflowVersion = '>=1.2, <=1.5'

// use the "!" prefix to stop execution if the current version
// doesn't match the required version.
manifest.nextflowVersion = '!>=1.2'
```

HealthOmics は Nextflow バージョン情報を次のように処理します。

- = を使用して HealthOmics がサポートする正確なバージョンを指定する場合、HealthOmics はそのバージョンを使用します。
- ! を使用して正確なバージョンまたはサポートされていないバージョンの範囲を指定すると、HealthOmics は例外を発生させ、実行に失敗します。バージョンリクエストに厳密に対応し、リクエストにサポートされていないバージョンが含まれている場合は、このオプションを使用することを検討してください。

- バージョンの範囲を指定すると、範囲に v24.10.8 が含まれていない限り、HealthOmics はその範囲内でサポートされている最新バージョンを使用します。この場合、HealthOmics は以前のバージョンを優先します。たとえば、範囲が v23.10.0 と v24.10.8 の両方をカバーする場合、HealthOmics は v23.10.0 を選択します。
- リクエストされたバージョンがない場合、またはリクエストされたバージョンが有効でないか、何らかの理由で解析できない場合：
  - DSL 1 を指定した場合、HealthOmics は Nextflow v22.04 を実行します。
  - それ以外の場合、HealthOmics は Nextflow v23.10.0 を実行します。

HealthOmics が各実行に使用した Nextflow バージョンに関する次の情報を取得できます。

- 実行ログには、HealthOmics が実行に使用した実際の Nextflow バージョンに関する情報が含まれています。
- HealthOmics は、リクエストされたバージョンと直接一致しない場合、または指定したバージョンとは異なるバージョンを使用する必要がある場合に、実行ログに警告を追加します。
- GetRun API オペレーションへのレスポンスには、HealthOmics が実行に使用した実際の Nextflow バージョンを含むフィールド (`engineVersion`) が含まれます。例：

```
"engineVersion":"22.04.0"
```

HealthOmics タスクのコンピューティングとメモリの要件

HealthOmics は、オミクスインスタンスでプライベートワークフロータスクを実行します。HealthOmics には、さまざまなタイプのタスクに対応するためのさまざまなインスタンスタイプが用意されています。各インスタンスタイプには、固定メモリと vCPU 設定 (および高速コンピューティングインスタンスタイプの固定 GPU 設定) があります。オミクスインスタンスの使用コストは、インスタンスタイプによって異なります。詳細については、HealthOmics の料金ページを参照してください。

ワークフロー内のタスクでは、ワークフロー定義ファイルに必要なメモリと vCPUsを指定します。ワークフロータスクを実行すると、HealthOmics は、リクエストされたメモリと vCPUs。たとえば、タスクに 64 GiB のメモリと 8 vCPUs が必要な場合、HealthOmics は を選択しますomics.r.2xlarge。

インスタンスタイプを確認し、ニーズに最適なインスタンスに合わせてリクエストされた vCPUs とメモリサイズを設定することをお勧めします。タスクコンテナは、インスタンスタイプに追加の

vCPUsとメモリがある場合でも、ワークフロー定義ファイルで指定した vCPUsの数とメモリサイズを使用します。

次のリストには、vCPU とメモリの割り当てに関する追加情報が含まれています。

- コンテナリソースの割り当てはハード制限です。タスクのメモリが不足した場合、または追加の vCPUsを使用しようとすると、タスクはエラーログを生成して終了します。
- コンピューティングまたはメモリの要件を指定しない場合、HealthOmics は 1 つの vCPU omics.c.large と 1 GiB のメモリを持つ設定を選択し、デフォルトに設定します。
- リクエストできる最小設定は、1 vCPU と 1 GiB のメモリです。
- サポートされているインスタンスタイプを超える vCPUs、メモリ、または GPUs を指定すると、HealthOmics はエラーメッセージをスローし、ワークフローは検証に失敗します。
- 小数単位を指定すると、HealthOmics は最も近い整数に切り上げられます。
- HealthOmics は、管理エージェントとログ記録エージェント用に少量のメモリ (5%) を予約するため、タスク内のアプリケーションが常にフルメモリ割り当てを利用できるとは限りません。
- HealthOmics は、指定したコンピューティング要件とメモリ要件に合わせてインスタンスタイプに一致し、ハードウェア世代が混在する場合があります。このため、同じタスクのタスク実行時間に若干の差異が生じる可能性があります。

これらのトピックでは、HealthOmics がサポートするインスタンスタイプについて詳しく説明します。

トピック

> **Note**
>
> 標準インスタンス、コンピューティングインスタンス、メモリ最適化インスタンスの場合、インスタンスのスループットを向上させる必要がある場合は、インスタンスの帯域幅サイズを増やします。vCPUs (サイズ 4xl 以下) の Amazon EC2 インスタンスでは、スループット

がバーストする可能性があります。Amazon EC2 インスタンスのスループットの詳細については、Amazon EC2 の使用可能なインスタンス帯域幅」を参照してください。

標準インスタンスタイプ

標準インスタンスタイプの場合、設定はコンピューティング能力とメモリのバランスを目指しています。

HealthOmics は、米国西部 (オレゴン) および米国東部 (バージニア北部) のリージョンで 32xlarge および 48xlarge インスタンスをサポートしています。

インスタンス	vCPUs の数	メモリ
omics.m.large	2	8 GiB
omics.m.xlarge	4	16 GiB
omics.m.2xlarge	8	32 GiB
omics.m.4xlarge	16	64 GiB
omics.m.8xlarge	32	128 GiB
omics.m.12xlarge	48	192 GiB
omics.m.16xlarge	64	256 GiB
omics.m.24xlarge	96	384 GiB
omics.m.32xlarge	128	512 GiB
omics.m.48xlarge	192	768 GiB

コンピューティング最適化インスタンス

コンピューティング最適化インスタンスタイプの場合、設定はコンピューティング能力が高く、メモリが少なくなります。

HealthOmics は、米国西部 (オレゴン) および米国東部 (バージニア北部) のリージョンで 32xlarge および 48xlarge インスタンスをサポートしています。

インスタンス	vCPUs の数	メモリ
omics.c.large	2	4 GiB
omics.c.xlarge	4	8 GiB
omics.c.2xlarge	8	16 GiB
omics.c.4xlarge	16	32 GiB
omics.c.8xlarge	32	64 GiB
omics.c.12xlarge	48	96 GiB
omics.c.16xlarge	64	128 GiB
omics.c.24xlarge	96	192 GiB
omics.c.32xlarge	128	256 GiB
omics.c.48xlarge	192	384 GiB

メモリ最適化インスタンス

メモリ最適化インスタンスタイプの場合、設定はコンピューティング能力が低く、メモリが多くなります。

HealthOmics は、米国西部 (オレゴン) および米国東部 (バージニア北部) のリージョンで 32xlarge および 48xlarge インスタンスをサポートしています。

インスタンス	vCPUs の数	メモリ
omics.r.large	2	16 GiB
omics.r.xlarge	4	32 GiB
omics.r.2xlarge	8	64 GiB
omics.r.4xlarge	16	128 GiB

インスタンス	vCPUs の数	メモリ
omics.r.8xlarge	32	256 GiB
omics.r.12xlarge	48	384 GiB
omics.r.16xlarge	64	512 GiB
omics.r.24xlarge	96	768 GiB
omics.r.32xlarge	128	1024 GiB
omics.r.48xlarge	192	1536 GiB

高速コンピューティングインスタンス

必要に応じて、ワークフロー内のタスクごとに GPU リソースを指定して、HealthOmics がタスクに高速コンピューティングインスタンスを割り当てるようにすることができます。ワークフロー定義ファイルで GPU 情報を指定する方法については、「」を参照してくださいHealthOmics ワークフロー定義のタスクアクセラレーター。

複数のインスタンスタイプをサポートする GPU を指定すると、HealthOmics は可用性に基づいてインスタンスタイプを選択します。両方のインスタンスタイプが使用可能な場合、HealthOmics は低コストのインスタンスを優先します。

G4 インスタンスは、イスラエル (テルアビブ) リージョンではサポートされていません。G5 インスタンスは、アジアパシフィック (シンガポール) リージョンではサポートされていません。

トピック

- G6 および G6e インスタンスタイプ
- G4 および G5 インスタンス

G6 および G6e インスタンスタイプ

HealthOmics は、次の G6 高速コンピューティングインスタンス設定をサポートしています。すべての omics.g6 インスタンスは、Nvidia L4 または Nvidia L4 A10G GPUs を使用します。

HealthOmics は、米国西部 (オレゴン) および米国東部 (バージニア北部) のリージョンで G6 および G6e インスタンスをサポートしています。

インスタンス	vCPUs の数	メモリ	GPUs の数	GPU メモリ	
omics.g6.xlarge	4	16 GiB	1	48 GiB	
omics.g6.2xlarge	8	32 GiB	1	48 GiB	
omics.g6.4xlarge	16	64 GiB	1	48 GiB	
omics.g6.8xlarge	32	128 GiB	1	48 GiB	
omics.g6.12xlarge	48	192 GiB	4	192 GiB	
omics.g6.16xlarge	64	256 GiB	1	48 GiB	
omics.g6.24xlarge	96	192 GiB	4	192 GiB	

すべての omics.g6e インスタンスは Nvidia L40s GPUs を使用します。

インスタンス	vCPUs の数	メモリ	GPUs の数	GPU メモリ	
omics.g6e.xlarge	4	32 GiB	1	24 GiB	
omics.g6e.2xlarge	8	64 GiB	1	24 GiB	
omics.g6e.4xlarge	16	128 GiB	1	24 GiB	
omics.g6e.8xlarge	32	256 GiB	1	24 GiB	

インスタンス	vCPUs の数	メモリ	GPUs の数	GPU メモリ	
omics.g6e.12xlarge	48	384 GiB	4	96 GiB	
omics.g6e.16xlarge	64	512 GiB	1	96 GiB	
omics.g6e.24xlarge	96	768 GiB	4	96 GiB	

G4 および G5 インスタンス

HealthOmics は、次の G4 および G5 高速コンピューティングインスタンス設定をサポートしています。

すべての omics.g5 インスタンスは、Nvidia L4 A10G、Nvidia Tesla A10G、または Nvidia Tesla T4 A10G GPUs を使用します。

インスタンス	vCPUs の数	メモリ	GPUs の数	GPU メモリ	
omics.g5.xlarge	4	16 GiB	1	24 GiB	
omics.g5.2xlarge	8	32 GiB	1	24 GiB	
omics.g5.4xlarge	16	64 GiB	1	24 GiB	
omics.g5.8xlarge	32	128 GiB	1	24 GiB	
omics.g5.12xlarge	48	192 GiB	4	96 GiB	
omics.g5.16xlarge	64	256 GiB	1	24 GiB	

インスタンス	vCPUs の数	メモリ	GPUs の数	GPU メモリ	
omics.g5.24xlarge	96	384 GiB	4	96 GiB	

すべての omics.g4dn インスタンスは、Nvidia Tesla T4 または Nvidia Tesla T4 A10G GPUs を使用します。

インスタンス	vCPUs の数	メモリ	GPUs の数	GPU メモリ	
omics.g4dn.xlarge	4	16 GiB	1	16 GiB	
omics.g4dn.2xlarge	8	32 GiB	1	16 GiB	
omics.g4dn.4xlarge	16	64 GiB	1	16 GiB	
omics.g4dn.8xlarge	32	128 GiB	1	16 GiB	
omics.g4dn.12xlarge	48	192 GiB	4	64 GiB	
omics.g4dn.16xlarge	64	256 GiB	1	24 GiB	

HealthOmics ワークフロー定義のタスク出力

ワークフロー定義でタスク出力を指定します。デフォルトでは、ワークフローが完了すると HealthOmics はすべての中間タスクファイルを破棄します。中間ファイルをエクスポートするには、出力として定義します。

コールキャッシュを使用する場合、HealthOmics は出力として定義した中間ファイルを含め、タスク出力をキャッシュに保存します。

以下のトピックでは、各ワークフロー定義言語のタスク定義の例を示します。

トピック

WDL のタスク出力

WDL で記述されたワークフロー定義については、最上位のワークフローoutputsセクションで出力を定義します。

Omics

トピック

STDOUT のタスク出力

この例では、STDOUT コンテンツをタスク出力ファイルにエコーSayHelloする という名前のタスクを作成します。WDL stdout関数は、STDOUT コンテンツ (この例では、入力文字列 Hello World!) をファイル にキャプチャしますstdout_file。

HealthOmics はすべての STDOUT コンテンツのログを作成するため、出力はタスクの他の STDERR ログ情報とともに CloudWatch Logs にも表示されます。

```
version 1.0
 workflow HelloWorld {
    input {
        String message = "Hello, World!"
        String ubuntu_container = "123456789012.dkr.ecr.us-east-1.amazonaws.com/
dockerhub/library/ubuntu:20.04"
    }

    call SayHello {
        input:
            message = message,
```

```
            container = ubuntu_container
    }

    output {
        File stdout_file = SayHello.stdout_file
    }
}

task SayHello {
    input {
        String message
        String container
    }

    command <<<
        echo "~{message}"
        echo "Current date: $(date)"
        echo "This message was printed to STDOUT"
    >>>

    runtime {
        docker: container
        cpu: 1
        memory: "2 GB"
    }

    output {
        File stdout_file = stdout()
    }
}
```

STDERR のタスク出力

この例では、STDERR コンテンツをタスク出力ファイルにエコーSayHelloする という名前のタスクを作成します。WDL stderr関数は、STDERR コンテンツ (この例では、入力文字列 Hello World!) をファイル にキャプチャしますstderr_file。

HealthOmics はすべての STDERR コンテンツのログを作成するため、出力はタスクの他の STDERR ログ情報とともに CloudWatch Logs に表示されます。

```
version 1.0
 workflow HelloWorld {
    input {
```

```
        String message = "Hello, World!"
        String ubuntu_container = "123456789012.dkr.ecr.us-east-1.amazonaws.com/
dockerhub/library/ubuntu:20.04"
    }

    call SayHello {
        input:
            message = message,
            container = ubuntu_container
    }

    output {
        File stderr_file = SayHello.stderr_file
    }
}

task SayHello {
    input {
        String message
        String container
    }

    command <<<
        echo "~{message}" >&2
        echo "Current date: $(date)" >&2
        echo "This message was printed to STDERR" >&2
    >>>

    runtime {
        docker: container
        cpu: 1
        memory: "2 GB"
    }

    output {
        File stderr_file = stderr()
    }
}
```

ファイルへのタスク出力

この例では、SayHello タスクは 2 つのファイル (message.txt と info.txt) を作成し、これらのファイルを名前付き出力 (message_file と info_file) として明示的に宣言します。

```
version 1.0
workflow HelloWorld {
    input {
        String message = "Hello, World!"
        String ubuntu_container = "123456789012.dkr.ecr.us-east-1.amazonaws.com/
dockerhub/library/ubuntu:20.04"
    }

    call SayHello {
        input:
            message = message,
            container = ubuntu_container
    }

    output {
        File message_file = SayHello.message_file
        File info_file = SayHello.info_file
    }
}

task SayHello {
    input {
        String message
        String container
    }

    command <<<
        # Create message file
        echo "~{message}" > message.txt

        # Create info file with date and additional information
        echo "Current date: $(date)" > info.txt
        echo "This message was saved to a file" >> info.txt
    >>>

    runtime {
        docker: container
        cpu: 1
        memory: "2 GB"
    }

    output {
        File message_file = "message.txt"
```

```
        File info_file = "info.txt"
    }
}
```

ファイルの配列へのタスク出力

この例では、GenerateGreetingsタスクはファイルの配列をタスク出力として生成します。タスクは、入力配列 のメンバーごとに 1 つの挨拶ファイルを動的に生成しますnames。ファイル名はランタイムまでわからないため、出力定義は WDL glob() 関数を使用してパターン に一致するすべてのファイルを出力します*_greeting.txt。

```
version 1.0
 workflow HelloArray {
    input {
        Array[String] names = ["World", "Friend", "Developer"]
        String ubuntu_container = "123456789012.dkr.ecr.us-east-1.amazonaws.com/
dockerhub/library/ubuntu:20.04"
    }

    call GenerateGreetings {
        input:
            names = names,
            container = ubuntu_container
    }

    output {
        Array[File] greeting_files = GenerateGreetings.greeting_files
    }
}

task GenerateGreetings {
    input {
        Array[String] names
        String container
    }

    command  <<<
        # Create a greeting file for each name
        for name in ~{sep=" " names}; do
            echo "Hello, $name!" > ${name}_greeting.txt
        done
    >>>
```

```
    runtime {
        docker: container
        cpu: 1
        memory: "2 GB"
    }

    output {
        Array[File] greeting_files = glob("*_greeting.txt")
    }
}
```

Nextflow のタスク出力

Nextflow で記述されたワークフロー定義の場合、タスクコンテンツを出力 Amazon S3 バケットにエクスポートする publishDir ディレクティブを定義します。publishDir 値を に設定します`/mnt/workflow/pubdir`。

HealthOmics がファイルを Amazon S3 にエクスポートするには、ファイルがこのディレクトリにある必要があります。

タスクが後続のタスクへの入力として使用する出力ファイルのグループを生成する場合は、これらのファイルをディレクトリにグループ化し、そのディレクトリをタスク出力として出力することをお勧めします。個々のファイルを列挙すると、基盤となるファイルシステムで I/O ボトルネックが発生する可能性があります。例:

```
process my_task {
      ...
      // recommended
      output "output-folder/", emit: output

      // not recommended
      // output "output-folder/**", emit: output
      ...
  }
```

CWL のタスク出力

CWL で記述されたワークフロー定義では、タスクを使用して`CommandLineTool`タスク出力を指定できます。以下のセクションでは、さまざまなタイプの出力を定義する`CommandLineTool`タスクの例を示します。

トピック

STDOUT のタスク出力

この例では、STDOUT コンテンツを という名前のテキスト出力ファイルにエコーするCommandLineToolタスクを作成しますoutput.txt。たとえば、次の入力を指定すると、結果のタスク出力は output.txt ファイル内の Hello World! になります。

```
{
    "message": "Hello World!"
}
```

outputs ディレクティブは、出力名を example_outに、タイプを に指定しますstdout。ダウンストリームタスクがこのタスクの出力を消費するには、出力を と呼びますexample_out。

HealthOmics はすべての STDERR および STDOUT コンテンツのログを作成するため、出力はタスクの他の STDERR ログ情報とともに CloudWatch Logs にも表示されます。

```
cwlVersion: v1.2
class: CommandLineTool
baseCommand: echo
stdout: output.txt
inputs:
  message:
    type: string
    inputBinding:
      position: 1
outputs:
  example_out:
    type: stdout

requirements:
    DockerRequirement:
        dockerPull: 123456789012.dkr.ecr.us-east-1.amazonaws.com/dockerhub/library/
ubuntu:20.04
    ResourceRequirement:
        ramMin: 2048
```

```
        coresMin: 1
```

STDERR のタスク出力

この例では、STDERR コンテンツを という名前のテキスト出力ファイルにエコーするCommandLineToolタスクを作成しますstderr.txt。タスクは、 が (STDOUT ではなく) STDERR にecho書き込むbaseCommandように を変更します。

outputs ディレクティブは、出力名を stderr_outに、タイプを に指定しますstderr。

HealthOmics はすべての STDERR および STDOUT コンテンツのログを作成するため、出力はタスクの他の STDERR ログ情報とともに CloudWatch Logs に表示されます。

```
cwlVersion: v1.2
class: CommandLineTool
baseCommand: [bash, -c]
stderr: stderr.txt
inputs:
  message:
    type: string
    inputBinding:
      position: 1
      shellQuote: true
      valueFrom: "echo $(self) >&2"
outputs:
  stderr_out:
    type: stderr

requirements:
    DockerRequirement:
        dockerPull: 123456789012.dkr.ecr.us-east-1.amazonaws.com/dockerhub/library/
ubuntu:20.04
    ResourceRequirement:
        ramMin: 2048
        coresMin: 1
```

ファイルへのタスク出力

この例では、入力ファイルから圧縮された tar アーカイブを作成するCommandLineToolタスクを作成します。アーカイブの名前を入力パラメータ (archive_name) として指定します。

outputs ディレクティブは、archive_file出力タイプが であることを指定しFile、入力パラメータへの参照を使用して出力ファイルにarchive_nameバインドします。

```
cwlVersion: v1.2
class: CommandLineTool
baseCommand: [tar, cfz]
inputs:
  archive_name:
    type: string
    inputBinding:
      position: 1
  input_files:
    type: File[]
    inputBinding:
      position: 2

outputs:
  archive_file:
    type: File
    outputBinding:
      glob: "$(inputs.archive_name)"

requirements:
    DockerRequirement:
        dockerPull: 123456789012.dkr.ecr.us-east-1.amazonaws.com/dockerhub/library/
ubuntu:20.04
    ResourceRequirement:
        ramMin: 2048
        coresMin: 1
```

ファイルの配列へのタスク出力

この例では、CommandLineToolタスクは touch コマンドを使用してファイルの配列を作成します。コマンドは、files-to-create入力パラメータの文字列を使用してファイルに名前を付けます。コマンドはファイルの配列を出力します。配列には、globパターンに一致する作業ディレクトリ内のすべてのファイルが含まれます。この例では、すべてのファイルに一致するワイルドカードパターン (「*」) を使用しています。

```
cwlVersion: v1.2
class: CommandLineTool
baseCommand: touch
inputs:
  files-to-create:
    type:
      type: array
```

```
      items: string
    inputBinding:
      position: 1
outputs:
  output-files:
    type:
      type: array
      items: File
    outputBinding:
      glob: "*"

requirements:
    DockerRequirement:
        dockerPull: 123456789012.dkr.ecr.us-east-1.amazonaws.com/dockerhub/library/
ubuntu:20.04
    ResourceRequirement:
        ramMin: 2048
        coresMin: 1
```

HealthOmics ワークフロー定義のタスクリソース

ワークフロー定義で、タスクごとに以下を定義します。

- タスクのコンテナイメージ。詳細については、「プライベートワークフロー用の Amazon ECR のコンテナイメージ」を参照してください。
- タスクに必要な CPUs とメモリの数。詳細については、「HealthOmics タスクのコンピューティングとメモリの要件」を参照してください。

HealthOmics は、タスクごとのストレージ仕様を無視します。HealthOmics は、実行内のすべてのタスクがアクセスできる実行ストレージを提供します。詳細については、「HealthOmics ワークフローでストレージタイプを実行する」を参照してください。

WDL

```
task my_task {
   runtime {
      container: "<aws-account-id>.dkr.ecr.<aws-region>.amazonaws.com/<image-name>"
      cpu: 2
      memory: "4 GB"
   }
   ...
```

```
}
```

WDL ワークフローの場合、HealthOmics はサービスエラーのために失敗したタスクに対して最大 2 回の再試行を試みます (API リクエストは 5XX HTTP ステータスコードを返します)。タスクの再試行の詳細については、「」を参照してくださいタスクの再試行。

WDL 定義ファイルでタスクに次の設定を指定することで、再試行動作をオプトアウトできます。

```
runtime {
   preemptible: 0
}
```

NextFlow

```
process my_task {
   container "<aws-account-id>.dkr.ecr.<aws-region>.amazonaws.com/<image-name>"
   cpus 2
   memory "4 GiB"
   ...
}
```

CWL

```
cwlVersion: v1.2
class: CommandLineTool
requirements:
    DockerRequirement:
        dockerPull: "<aws-account-id>.dkr.ecr.<aws-region>.amazonaws.com/<image-
name>"
    ResourceRequirement:
        coresMax: 2
        ramMax: 4000 # specified in mebibytes
```

HealthOmics ワークフロー定義のタスクアクセラレーター

ワークフロー定義では、オプションでタスクの GPU アクセラレーター仕様を指定できます。HealthOmics は、サポートされているインスタンスタイプとともに、次のアクセラレーター仕様の値をサポートしています。

アクセラレーターの仕様	Healthomics インスタンスタイプ				
nvidia-tesla-t4	G4				
nvidia-tesla-t4-a10g	G4 と G5				
nvidia-tesla-a10g	G5				
nvidia-l4-a10g	G5 と G6				
nvidia-l4	G6				
nvidia-l40s	G6e				

複数のインスタンスタイプをサポートするアクセラレータータイプを指定すると、HealthOmics は使用可能な容量に基づいてインスタンスタイプを選択します。両方のインスタンスタイプが使用可能な場合、HealthOmics は低コストのインスタンスを優先します。

インスタンスタイプの詳細については、「」を参照してください高速コンピューティングインスタンス。

次の例では、ワークフロー定義で をアクセラレーターnvidia-l4として指定します。

WDL

```
task my_task {
 runtime {
    ...
    acceleratorCount: 1
    acceleratorType: "nvidia-l4"
 }
 ...
}
```

NextFlow

```
process my_task {
 ...
 accelerator 1, type: "nvidia-l4"
 ...
}
```

CWL

```
cwlVersion: v1.2
class: CommandLineTool
requirements:
  ...
  cwltool:CUDARequirement:
      cudaDeviceCountMin: 1
      cudaComputeCapability: "nvidia-l4"
      cudaVersionMin: "1.0"
```

HealthOmics ワークフローのワークフロー定義の記述

HealthOmics は、WDL、Nextflow、または CWL で記述されたワークフロー定義をサポートしています。これらのワークフロー言語の詳細については、WDL、Nextflow、または CWL の仕様を参照してください。

HealthOmics は、3 つのワークフロー定義言語のバージョン管理をサポートしています。詳細については、「HealthOmics ワークフロー定義言語のバージョンサポート」を参照してください。

トピック

- WDL でのワークフローの記述
- Nextflow でのワークフローの記述
- CWL でのワークフローの記述
- ワークフロー定義の例
- WDL ワークフロー定義の例

WDL でのワークフローの記述

次の表は、WDL の入力が一致するプリミティブ型または複雑な JSON 型にどのようにマッピングされるかを示しています。型強制は制限されており、可能な限り型は明示的である必要があります。

プリミティブ型

WDL タイプ	JSON タイプ	WDL の例	JSON キーと値の例	メモ
`Boolean`	`boolean`	`Boolean b`	`"b": true`	値は小文字で、引用符で囲まない必要があります。
`Int`	`integer`	`Int i`	`"i": 7`	引用符で囲まない必要があります。
`Float`	`number`	`Float f`	`"f": 42.2`	引用符で囲まない必要があります。
`String`	`string`	`String s`	`"s": "characters"`	URI である JSON 文字列は、インポートする WDL ファイルにマッピングする必要があります。
`File`	`string`	`File f`	`"f": "s3://amzn-s3-demo-bucket1/path/to/file"`	Amazon S3 および HealthOmics ストレージ URIs は、ワークフローに提供された IAM ロールがこれらのオブジェクトへの読

WDL タイプ	JSON タイプ	WDL の例	JSON キーと値の例	メモ
				み取りアクセス権を持っている限り、インポートされます。他の URI スキーム (`file://`、、など) `https://`はサポートされていません`ftp://`。URI は オブジェクトを指定する必要があります。ディレクトリにすることはできません。つまり、 で終わることはできません`/`。

WDL タイプ	JSON タイプ	WDL の例	JSON キーと値の例	メモ
`Directory`	`string`	`Directory d`	`"d": "s3://bucket/path/"`	`Directory` タイプは WDL 1.0 または 1.1 に含まれていないため、WDL ファイルの ヘッダー`version development` に を追加する必要があります。URI は Amazon S3 URI で、プレフィックスが '/' で終わる必要があります。ディレクトリのすべてのコンテンツは、1 回のダウンロードとしてワークフローに再帰的にコピーされます。には、ワークフローに関連するファイルのみを含める `Directory` 必要があります。

WDL の複雑な型は、プリミティブ型で構成されるデータ構造です。リストなどのデータ構造は配列に変換されます。

複合型

WDL タイプ	JSON タイプ	WDL の例	JSON キーと値の例	メモ
Array	array	Array[Int] nums	“nums": [1, 2, 3]	配列のメンバーは、WDL 配列タイプの形式に従う必要があります。
Pair	object	Pair[String, Int] str_to_i	“str_to_i": {"left": "0", "right": 1}	ペアの各値は、一致する WDL タイプの JSON 形式を使用する必要があります。
Map	object	Map[Int, String] int_to_string	"int_to_string": { 2: "hello", 1: "goodbye" }	マップの各エントリは、一致する WDL タイプの JSON 形式を使用する必要があります。
Struct	object	struct SampleBamAndIndex { String sample_name File bam File bam_index } SampleBamAndIndex b_and_i	"b_and_i": { "sample_name": "NA12878", "bam": "s3://amzn-s3-demo-bucket1/NA12878.bam", "bam_index": "s3://amzn-	構造体メンバーの名前は、JSON オブジェクトキーの名前と完全に一致する必要があります。各値は、一致する WDL タイプの JSON 形式を使用する必要があります。

WDL タイプ	JSON タイプ	WDL の例	JSON キーと値の例	メモ
			`s3-demo-bucket1/NA12878.bam.bai" }`	
`Object`	該当なし	該当なし	該当なし	WDL `Object`タイプは古いため、いずれの場合`Struct`も に置き換える必要があります。

HealthOmics ワークフローエンジンは、修飾入力パラメータまたは名前空間入力パラメータをサポートしていません。修飾パラメータの処理と WDL パラメータへのマッピングは、WDL 言語では指定されておらず、あいまいである可能性があります。このような理由から、ベストプラクティスは、最上位 (メイン) ワークフロー定義ファイル内のすべての入力パラメータを宣言し、標準の WDL メカニズムを使用してサブワークフロー呼び出しに渡すことです。

Nextflow でのワークフローの記述

HealthOmics は Nextflow DSL1 と DSL2 をサポートしています。詳細については、「Nextflow バージョンのサポート」を参照してください。

Nextflow DSL2 は Groovy プログラミング言語に基づいているため、パラメータは動的であり、型強制は Groovy と同じルールを使用して可能です。入力 JSON によって提供されるパラメータと値は、ワークフローのパラメータ (`params`) マップで使用できます。

> **Note**
>
> HealthOmics は、Nextflow バージョン v23.10 の `nf-schema`および `nf-validation`プラグインをサポートしています (v22.04 はサポートしていません) 。

次の情報は、Nextflow v23.10 ワークフローでのこれらのプラグインの使用に関連しています。

- HealthOmics は、nf-schema@2.3.0 および nf-validation@1.1.1 プラグインをプリインストールします。HealthOmics は、 `nextflow.config` ファイルで指定した他のプラグインバージョンを無視します。
- ワークフローの実行中に追加のプラグインを取得することはできません。
- Nextflow v24.04 以降では、`nf-validation`プラグインの名前は に変更されます`nf-schema`。詳細については、Nextflow GitHub リポジトリの「nf-schema」を参照してください。

Amazon S3 または HealthOmics URI を使用して Nextflow ファイルまたはパスオブジェクトを作成すると、読み取りアクセスが付与されている限り、一致するオブジェクトがワークフローで使用できるようになります。Amazon S3 URIs では、プレフィックスまたはディレクトリを使用できます。例については「Amazon S3 入力パラメータ形式」を参照してください。

HealthOmics は、Amazon S3 URI または HealthOmics Storage URIs での glob パターンの使用をサポートしています。 URIs HealthOmics `path` または `file`チャネルの作成には、ワークフロー定義で Glob パターンを使用します。

Nextflow で記述されたワークフローの場合、タスクコンテンツを出力 Amazon S3 バケットにエクスポートする publishDir ディレクティブを定義します。次の例に示すように、publishDir 値を に設定します`/mnt/workflow/pubdir`。Amazon S3 にファイルをエクスポートするには、ファイルがこのディレクトリにある必要があります。

```
 nextflow.enable.dsl=2

workflow {
  CramToBamTask(params.ref_fasta, params.ref_fasta_index, params.ref_dict,
 params.input_cram, params.sample_name)
  ValidateSamFile(CramToBamTask.out.outputBam)
}

process CramToBamTask {
  container "<account>.dkr.ecr.us-west-2.amazonaws.com/genomes-in-the-cloud"

  publishDir "/mnt/workflow/pubdir"

  input:
      path ref_fasta
      path ref_fasta_index
      path ref_dict
      path input_cram
      val sample_name
```

```
  output:
      path "${sample_name}.bam", emit: outputBam
      path "${sample_name}.bai", emit: outputBai

  script:
  """
      set -eo pipefail

      samtools view -h -T $ref_fasta $input_cram |
      samtools view -b -o ${sample_name}.bam -
      samtools index -b ${sample_name}.bam
      mv ${sample_name}.bam.bai ${sample_name}.bai
  """
}

process ValidateSamFile {
  container "<account>.dkr.ecr.us-west-2.amazonaws.com/genomes-in-the-cloud"

  publishDir "/mnt/workflow/pubdir"

  input:
      file input_bam

  output:
      path "validation_report"

  script:
  """
      java -Xmx3G -jar /usr/gitc/picard.jar \
      ValidateSamFile \
      INPUT=${input_bam} \
      OUTPUT=validation_report \
      MODE=SUMMARY \
      IS_BISULFITE_SEQUENCED=false
  """
}
```

CWL でのワークフローの記述

Common Workflow Language または CWL で記述されたワークフローは、WDL および Nextflow で記述されたワークフローと同様の機能を提供します。Amazon S3 または HealthOmics ストレージ URIs を入力パラメータとして使用できます。

サブワークフローの secondaryFile で入力を定義する場合は、メインワークフローに同じ定義を追加します。

HealthOmics ワークフローはオペレーションプロセスをサポートしていません。CWL ワークフローのオペレーションプロセスの詳細については、CWL ドキュメントを参照してください。

既存の CWL ワークフロー定義ファイルを HealthOmics を使用するように変換するには、次の変更を行います。

- すべての Docker コンテナ URIs Amazon ECR URIs。
- すべてのワークフローファイルがメインワークフローで入力として宣言され、すべての変数が明示的に定義されていることを確認します。
- すべての JavaScript コードが厳格モードの苦情であることを確認します。

CWL ワークフローは、使用するコンテナごとに定義する必要があります。固定 Amazon ECR URI を使用して dockerPull エントリをハードコードすることはお勧めしません。

以下は、CWL で記述されたワークフローの例です。

```
cwlVersion: v1.2
class: Workflow

inputs:
in_file:
  type: File
  secondaryFiles: [.fai]

out_filename: string
docker_image: string


outputs:
copied_file:
  type: File
  outputSource: copy_step/copied_file

steps:
copy_step:
  in:
    in_file: in_file
```

```
    out_filename: out_filename
    docker_image: docker_image
  out: [copied_file]
  run: copy.cwl
```

次の ファイルは、copy.cwlタスクを定義します。

```
cwlVersion: v1.2
class: CommandLineTool
baseCommand: cp

inputs:
in_file:
  type: File
  secondaryFiles: [.fai]
  inputBinding:
    position: 1

out_filename:
  type: string
  inputBinding:
    position: 2
docker_image:
  type: string

outputs:
copied_file:
  type: File
  outputBinding:
      glob: $(inputs.out_filename)

requirements:
InlineJavascriptRequirement: {}
DockerRequirement:
  dockerPull: "$(inputs.docker_image)"
```

以下は、GPU 要件を使用して CWL で記述されたワークフローの例です。

```
cwlVersion: v1.2
class: CommandLineTool
baseCommand: ["/bin/bash", "docm_haplotypeCaller.sh"]
```

```
$namespaces:
cwltool: http://commonwl.org/cwltool#
requirements:
cwltool:CUDARequirement:
  cudaDeviceCountMin: 1
  cudaComputeCapability: "nvidia-tesla-t4"
  cudaVersionMin: "1.0"
InlineJavascriptRequirement: {}
InitialWorkDirRequirement:
  listing:
  - entryname: 'docm_haplotypeCaller.sh'
    entry: |
            nvidia-smi --query-gpu=gpu_name,gpu_bus_id,vbios_version --format=csv

inputs: []
outputs: []
```

ワークフロー定義の例

次の例は、WDL、Nextflow、CWL で同じワークフロー定義を示しています。

WDL

```
version 1.1

task my_task {
   runtime { ... }
   inputs {
       File input_file
       String name
       Int threshold
   }

   command <<<
   my_tool --name ~{name} --threshold ~{threshold} ~{input_file}
   >>>

   output {
       File results = "results.txt"
   }
}

workflow my_workflow {
   inputs {
```

```
        File input_file
        String name
        Int threshold = 50
    }

    call my_task {
        input:
           input_file = input_file,
           name = name,
           threshold = threshold
    }
    outputs {
        File results = my_task.results
    }
}
```

Nextflow

```
nextflow.enable.dsl = 2

params.input_file = null
params.name = null
params.threshold = 50

process my_task {
   // <directives>

   input:
     path input_file
     val name
     val threshold

   output:
     path 'results.txt', emit: results

   script:
     """
     my_tool --name ${name} --threshold ${threshold} ${input_file}
     """


}
```

```
workflow MY_WORKFLOW {
   my_task(
       params.input_file,
       params.name,
       params.threshold
   )
}

workflow {
   MY_WORKFLOW()
}
```

CWL

```
cwlVersion: v1.2
class: Workflow

requirements:
    InlineJavascriptRequirement: {}

inputs:
   input_file: File
   name: string
   threshold: int

outputs:
    result:
        type: ...
        outputSource: ...

steps:
    my_task:
        run:
            class: CommandLineTool
            baseCommand: my_tool
            requirements:
                ...
            inputs:
                name:
                    type: string
                    inputBinding:
```

```
                    prefix: "--name"
            threshold:
                type: int
                inputBinding:
                    prefix: "--threshold"
            input_file:
                type: File
                inputBinding: {}
        outputs:
            results:
                type: File
                outputBinding:
                    glob: results.txt
```

WDL ワークフロー定義の例

次の例は、WDL BAMで から CRAM に変換するためのプライベートワークフロー定義を示しています。CRAM から へのBAMワークフローでは、2 つのタスクを定義し、genomes-in-the-cloudコンテナのツールを使用します。この例は、一般公開されています。

次の例は、Amazon ECR コンテナをパラメータとして含める方法を示しています。これにより、HealthOmics は実行を開始する前にコンテナへのアクセス許可を検証できます。

```
{
     ...
     "gotc_docker":"<account_id>.dkr.ecr.<region>.amazonaws.com/genomes-in-the-
cloud:2.4.7-1603303710"
  }
```

次の例は、ファイルが Amazon S3 バケットにある場合に、実行で使用するファイルを指定する方法を示しています。

```
{
      "input_cram": "s3://amzn-s3-demo-bucket1/inputs/NA12878.cram",
      "ref_dict": "s3://amzn-s3-demo-bucket1/inputs/Homo_sapiens_assembly38.dict",
      "ref_fasta": "s3://amzn-s3-demo-bucket1/inputs/Homo_sapiens_assembly38.fasta",
      "ref_fasta_index": "s3://amzn-s3-demo-bucket1/inputs/
Homo_sapiens_assembly38.fasta.fai",
      "sample_name": "NA12878"
  }
```

シーケンスストアからファイルを指定する場合は、次の例に示すように、シーケンスストアの URI を使用して を指定します。

```
{
      "input_cram": "omics://429915189008.storage.us-west-2.amazonaws.com/111122223333/
readSet/4500843795/source1",
      "ref_dict": "s3://amzn-s3-demo-bucket1/inputs/Homo_sapiens_assembly38.dict",
      "ref_fasta": "s3://amzn-s3-demo-bucket1/inputs/Homo_sapiens_assembly38.fasta",
      "ref_fasta_index": "s3://amzn-s3-demo-bucket1/inputs/
Homo_sapiens_assembly38.fasta.fai",
      "sample_name": "NA12878"
  }
```

その後、次に示すように、WDL でワークフローを定義できます。

```
 version 1.0
  workflow CramToBamFlow {
      input {
          File ref_fasta
          File ref_fasta_index
          File ref_dict
          File input_cram
          String sample_name
          String gotc_docker = "<account>.dkr.ecr.us-west-2.amazonaws.com/genomes-in-
the-
  cloud:latest"
      }
      #Converts CRAM to SAM to BAM and makes BAI.
      call CramToBamTask{
           input:
              ref_fasta = ref_fasta,
              ref_fasta_index = ref_fasta_index,
              ref_dict = ref_dict,
              input_cram = input_cram,
              sample_name = sample_name,
              docker_image = gotc_docker,
       }
       #Validates Bam.
       call ValidateSamFile{
          input:
             input_bam = CramToBamTask.outputBam,
             docker_image = gotc_docker,
       }
```

```
      #Outputs Bam, Bai, and validation report to the FireCloud data model.
      output {
          File outputBam = CramToBamTask.outputBam
          File outputBai = CramToBamTask.outputBai
          File validation_report = ValidateSamFile.report
       }
}
#Task definitions.
task CramToBamTask {
    input {
       # Command parameters
       File ref_fasta
       File ref_fasta_index
       File ref_dict
       File input_cram
       String sample_name
       # Runtime parameters
       String docker_image
    }
   #Calls samtools view to do the conversion.
   command {
       set -eo pipefail

       samtools view -h -T ~{ref_fasta} ~{input_cram} |
       samtools view -b -o ~{sample_name}.bam -
       samtools index -b ~{sample_name}.bam
       mv ~{sample_name}.bam.bai ~{sample_name}.bai
    }

    #Runtime attributes:
    runtime {
        docker: docker_image
    }

    #Outputs a BAM and BAI with the same sample name
     output {
         File outputBam = "~{sample_name}.bam"
         File outputBai = "~{sample_name}.bai"
    }
}

#Validates BAM output to ensure it wasn't corrupted during the file conversion.
task ValidateSamFile {
   input {
```

```
        File input_bam
        Int machine_mem_size = 4
        String docker_image
    }
    String output_name = basename(input_bam, ".bam") + ".validation_report"
    Int command_mem_size = machine_mem_size - 1
    command {
        java -Xmx~{command_mem_size}G -jar /usr/gitc/picard.jar \
        ValidateSamFile \
        INPUT=~{input_bam} \
        OUTPUT=~{output_name} \
        MODE=SUMMARY \
        IS_BISULFITE_SEQUENCED=false
     }
     runtime {
     docker: docker_image
     }
    #A text file is generated that lists errors or warnings that apply.
     output {
         File report = "~{output_name}"
     }
 }
```

HealthOmics ワークフローのパラメータテンプレートファイル

パラメータテンプレートは、ワークフローの入力パラメータを定義します。入力パラメータを定義して、ワークフローの柔軟性と汎用性を高めることができます。たとえば、参照ゲノムファイルの Amazon S3 の場所のパラメータを定義できます。パラメータテンプレートは、Git ベースのリポジトリサービスまたはローカルドライブを通じて提供できます。その後、ユーザーはさまざまなデータセットを使用してワークフローを実行できます。

ワークフローのパラメータテンプレートを作成するか、HealthOmics がパラメータテンプレートを自動的に生成できます。

パラメータテンプレートは JSON ファイルです。ファイルでは、各入力パラメータは、ワークフロー入力の名前と一致する名前付きオブジェクトです。実行を開始するときに、すべての必須パラメータの値を指定しない場合、実行は失敗します。

入力パラメータオブジェクトには、次の属性が含まれます。

- description – この必須属性は、コンソールが実行開始ページに表示する文字列です。この説明は実行メタデータとしても保持されます。

- optional – このオプション属性は、入力パラメータがオプションかどうかを示します。optional フィールドを指定しない場合、入力パラメータは必須です。

次のパラメータテンプレートの例は、入力パラメータを指定する方法を示しています。

```
{
  "myRequiredParameter1": {
     "description": "this parameter is required",
  },
  "myRequiredParameter2": {
     "description": "this parameter is also required",
     "optional": false
  },
  "myOptionalParameter": {
     "description": "this parameter is optional",
     "optional": true
  }
}
```

パラメータテンプレートの生成

HealthOmics は、ワークフロー定義を解析して入力パラメータを検出することでパラメータテンプレートを生成します。ワークフローのパラメータテンプレートファイルを指定すると、ファイルのパラメータによって、ワークフロー定義で検出されたパラメータが上書きされます。

以下のセクションで説明するように、CWL、WDL、Nextflow エンジンの解析ロジックには若干の違いがあります。

トピック

- CWL のパラメータ検出
- WDL のパラメータ検出
- Nextflow のパラメータ検出

CWL のパラメータ検出

CWL ワークフローエンジンでは、解析ロジックは次の仮定を行います。

- NULL 対応タイプは、オプションの入力パラメータとしてマークされます。
- NULL でサポートされていないタイプは、必須の入力パラメータとしてマークされます。

- デフォルト値を持つパラメータは、オプションの入力パラメータとしてマークされます。
- 説明は、mainワークフロー定義の labelセクションから抽出されます。を指定labelしない場合、説明は空白になります (空の文字列) 。

次の表は、CWL 補間の例を示しています。各例のパラメータ名は ですx。パラメータが必要な場合は、 パラメータの値を指定する必要があります。パラメータがオプションの場合、値を指定する必要はありません。

この表は、プリミティブ型の CWL 補間の例を示しています。

Input	入出力の例	必須
`x:` `  type: int`	1 または 2 または ...	あり

`x:`
`  type: int`
`  default: 2`
	デフォルト値は 2 です。有効な入力は 1 または 2 または ...	なし

`x:` `  type: int?`	有効な入力は None または 1 または 2 または ... です。	なし

`x:`
`  type: int?`
`  default: 2`
	デフォルト値は 2 です。有効な入力は None または 1 または 2 または ... です。	なし

次の表は、複雑なタイプの CWL 補間の例を示しています。複合型はプリミティブ型のコレクションです。

Input	入出力の例	必須

`x:`
`  type: array`
`  items: int`
	[] または [1,2,3]	あり

Input	入出力の例	必須

`x:`
`  type: array?`
`  items: int`
	なしまたは [] または [1,2,3]	なし

`x:`
`  type: array`
`  items: int?`
	[] または [なし、3、なし〕	あり

`x:`
`  type: array?`
`  items: int?`
	[None] または None または [1,2,3] または [None, 3] ですが [] ではありません	なし

WDL のパラメータ検出

WDL ワークフローエンジンでは、解析ロジックは次の前提になります。

- NULL 対応タイプは、オプションの入力パラメータとしてマークされます。
- nullable 以外のサポート対象タイプの場合：
  - リテラルまたは式が割り当てられた入力変数は、オプションパラメータとしてマークされます。例:

    ```
     Int x = 2
    Float f0 = 1.0 + f1
    ```

  - 入力パラメータに値または式が割り当てられていない場合は、必須パラメータとしてマークされます。
- 説明は、`main`ワークフロー定義`parameter_meta`の から抽出されます。を指定`parameter_meta`しない場合、説明は空白になります (空の文字列) 。詳細については、パラメータメタデータの WDL 仕様を参照してください。

次の表は、WDL 補間の例を示しています。各例のパラメータ名は ですx。パラメータが必要な場合は、 パラメータの値を指定する必要があります。パラメータがオプションの場合、値を指定する必要はありません。

この表は、プリミティブ型の WDL 補間の例を示しています。

Input	入出力の例	必須
Int x	1 または 2 または ...	あり
Int x = 2	2	なし
Int x = 1+2	3	なし
Int x = y+z	y+z	なし
Int? x	なし、1 または 2 または ...	あり
Int? x = 2	なしまたは 2	なし
Int? x = 1+2	なしまたは 3	なし
Int? x = y+z	なしまたは y+z	なし

次の表は、複雑なタイプの WDL 補間の例を示しています。複合型はプリミティブ型のコレクションです。

Input	入出力の例	必須		
配列[Int] x	[1,2,3] または []	あり		
配列[Int]+ x	[1]、ただし [] ではない	あり		
Array[Int]? x	なしまたは [] または [1,2,3]	なし		
配列[Int?] x	[] または [なし、3、なし〕	あり		
Array[Int?]=? x	[None] または None または [1,2,3] または	なし		

Input	入出力の例	必須		
	[None, 3] ですが [] ではありません			
構造体サンプル {文字列 a、Int y} 後続の入力: サンプル mySample	`String a = mySample.a Int y = mySample.y`	あり		
構造体サンプル {文字列 a、Int y} 後の入力: Sample? mySample	`if (defined(mySample)) { String a = mySample.a Int y = mySample.y }`	なし		

Nextflow のパラメータ検出

Nextflow の場合、HealthOmics はnextflow_schema.jsonファイルを解析してパラメータテンプレートを生成します。ワークフロー定義にスキーマファイルが含まれていない場合、HealthOmics はメインワークフロー定義ファイルを解析します。

トピック

- スキーマファイルの解析
- メインファイルの解析
- ネストされたパラメータ
- Nextflow 補間の例

スキーマファイルの解析

解析を正しく機能させるには、スキーマファイルが次の要件を満たしていることを確認してください。

- スキーマファイルは という名前nextflow_schema.jsonで、メインワークフローファイルと同じディレクトリにあります。
- スキーマファイルは、次のいずれかのスキーマで定義されている有効な JSON です。
  - json-schema.org/draft/2020-12/schema。
  - json-schema.org/draft-07/schema。

HealthOmics はnextflow_schema.jsonファイルを解析してパラメータテンプレートを生成します。

- スキーマで定義されているすべての properties を抽出します。
- プロパティで使用可能なdescription場合は、 プロパティを含めます。
- プロパティの requiredフィールドに基づいて、各パラメータがオプションか必須かを識別します。

次の例は、定義ファイルと生成されたパラメータファイルを示しています。

```
{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "type": "object",
    "$defs": {
        "input_options": {
            "title": "Input options",
            "type": "object",
            "required": ["input_file"],
            "properties": {
                "input_file": {
                    "type": "string",
                    "format": "file-path",
                    "pattern": "^s3://[a-z0-9.-]{3,63}(?:/\\S*)?$",
                    "description": "description for input_file"
                },
                "input_num": {
                    "type": "integer",
                    "default": 42,
                    "description": "description for input_num"
                }
            }
        },
        "output_options": {
            "title": "Output options",
```

```
            "type": "object",
            "required": ["output_dir"],
            "properties": {
                "output_dir": {
                    "type": "string",
                    "format": "file-path",
                    "description": "description for output_dir",
                }
            }
        }
    },
    "properties": {
        "ungrouped_input_bool": {
            "type": "boolean",
            "default": true
        }
    },
    "required": ["ungrouped_input_bool"],
    "allOf": [
        { "$ref": "#/$defs/input_options" },
        { "$ref": "#/$defs/output_options" }
    ]
}
```

生成されたパラメータテンプレート：

```
{
    "input_file": {
        "description": "description for input_file",
        "optional": False
    },
    "input_num": {
        "description": "description for input_num",
        "optional": True
    },
    "output_dir": {
        "description": "description for output_dir",
        "optional": False
    },
    "ungrouped_input_bool": {
        "description": None,
        "optional": False
    }
```

```
}
```

メインファイルの解析

ワークフロー定義に nextflow_schema.json ファイルが含まれていない場合、HealthOmics はメインワークフロー定義ファイルを解析します。

HealthOmics は、メインワークフロー定義ファイルと nextflow.config ファイルにあるparams式を分析します。デフォルト値paramsを持つすべての はオプションとしてマークされます。

解析を正しく機能させるには、次の要件に注意してください。

- HealthOmics は、メインワークフロー定義ファイルのみを解析します。すべてのパラメータを確実にキャプチャするには、すべてのサブモジュールとインポートされたワークフローにparamsワイヤスルーすることをお勧めします。
- 設定ファイルはオプションです。定義する場合は、名前を付けnextflow.config、メインワークフロー定義ファイルと同じディレクトリに配置します。

次の例は、定義ファイルと生成されたパラメータテンプレートを示しています。

```
params.input_file = "default.txt"
params.threads = 4
params.memory = "8GB"

workflow {
    if (params.version) {
        println "Using version: ${params.version}"
    }
}
```

生成されたパラメータテンプレート：

```
{
    "input_file": {
        "description": None,
        "optional": True
    },
    "threads": {
        "description": None,
```

```
        "optional": True
    },
    "memory": {
        "description": None,
        "optional": True
    },
    "version": {
        "description": None,
        "optional": False
    }
}
```

nextflow.config で定義されているデフォルト値の場合、HealthOmics は、次の例に示すようにparams {}、 内で宣言されたparams割り当てとパラメータを収集します。割り当てステートメントでは、 はステートメントの左側に表示されるparams必要があります。

```
params.alpha = "alpha"
params.beta = "beta"

params {
    gamma = "gamma"
    delta = "delta"
}

env {
   // ignored, as this assignment isn't in the params block
   VERSION = "TEST"
}

// ignored, as params is not on the left side
interpolated_image = "${params.cli_image}"
```

生成されたパラメータテンプレート：

```
{
    // other params in your main workflow defintion
    "alpha": {
        "description": None,
        "optional": True
    },
    "beta": {
        "description": None,
```

```
        "optional": True
    },
    "gamma": {
        "description": None,
        "optional": True
    },
    "delta": {
        "description": None,
        "optional": True
    }
}
```

ネストされたパラメータ

nextflow_schema.json と の両方がネストされたパラメータnextflow.configを許可します。ただし、HealthOmics パラメータテンプレートでは、最上位のパラメータのみが必要です。ワークフローでネストされたパラメータを使用する場合は、そのパラメータの入力として JSON オブジェクトを指定する必要があります。

スキーマファイルにネストされたパラメータ

HealthOmics は、nextflow_schema.jsonファイルの解析params時にネストされた をスキップします。たとえば、次のnextflow_schema.jsonファイルを定義します。

```
{
    "properties": {
        "input": {
            "properties": {
                "input_file": { ... },
                "input_num": { ... }
            }
        },
        "input_bool": { ... }
    }
}
```

HealthOmics は、パラメータテンプレートを生成するinput_numときに input_fileと を無視します。

```
{
    "input": {
```

```
        "description": None,
        "optional": True
    },
    "input_bool": {
        "description": None,
        "optional": True
    }
}
```

このワークフローを実行すると、HealthOmics は次のような input.json ファイルを想定します。

```
{
   "input": {
       "input_file": "s3://bucket/obj",
       "input_num": 2
   },
   "input_bool": false
}
```

設定ファイルにネストされたパラメータ

HealthOmics は、nextflow.configファイルにネストされた params を収集せず、解析中にスキップします。たとえば、次のnextflow.configファイルを定義します。

```
params.alpha = "alpha"
  params.nested.beta = "beta"

  params {
      gamma = "gamma"
      group {
          delta = "delta"
      }
  }
```

HealthOmics は、パラメータテンプレートを生成するparams.group.deltaときに params.nested.betaと を無視します。

```
{
    "alpha": {
        "description": None,
        "optional": True
```

```
    },
    "gamma": {
        "description": None,
        "optional": True
    }
}
```

Nextflow 補間の例

次の表は、メインファイル内のパラメータの Nextflow 補間の例を示しています。

パラメータ	必須
params.input_file	あり
params.input_file = "s3://bucket/data.json"	なし
params.nested.input_file	該当なし
params.nested.input_file = "s3://bucket/data.json"	該当なし

次の表は、 `nextflow.config` ファイル内のパラメータの Nextflow 補間の例を示しています。

パラメータ	必須
`params.input_file = "s3://bucket/data.json"`	なし
`params { input_file = "s3://bucket/data.json" }`	いいえ
`params { nested { input_file = "s3://bucket/data.json"`	該当なし

パラメータ	必須
`    }` `}`	
`input_file = params.input_file`	該当なし

プライベートワークフロー用の Amazon ECR のコンテナイメージ

プライベートワークフローを作成する前に、ワークフローのコンテナイメージを作成します。Amazon Elastic Container Registry (Amazon ECR) のプライベートイメージリポジトリにイメージをアップロードします。ワークフローを実行すると、HealthOmics サービスは指定したコンテナにアクセスします。

コンテナイメージ Amazon ECR リポジトリは、サービスを呼び出すアカウントと同じ AWS リージョンに存在する必要があります。ソースイメージリポジトリが適切なアクセス許可を提供している限り、別の がコンテナイメージを所有 AWS アカウント できます。詳細については、「共有ワークフローの Amazon Elastic Container Registry リポジトリポリシー」を参照してください。

実行を開始する前にアクセスを検証できるように、Amazon ECR コンテナイメージ URIs をワークフローのパラメータとして定義することをお勧めします。また、リージョンパラメータを変更することで、新しいリージョンでワークフローを簡単に実行できます。

> **Note**
>
> HealthOmics は ARM コンテナをサポートしておらず、パブリックリポジトリへのアクセスもサポートしていません。

HealthOmics が Amazon ECR にアクセスするための IAM アクセス許可の設定については、「」を参照してくださいリソースのアクセス許可。

トピック

- Amazon ECR コンテナイメージの一般的な考慮事項
- HealthOmics ワークフローの環境変数
- Amazon ECR コンテナイメージでの Java の使用

- ECR コンテナイメージにタスク入力を追加する

Amazon ECR コンテナイメージの一般的な考慮事項

- アーキテクチャ

  HealthOmics は x86_64 コンテナをサポートしています。ローカルマシンが Apple Mac などの ARM ベースである場合は、次のようなコマンドを使用して x86_64 コンテナイメージを構築します。

  ```
  docker build --platform amd64 -t my_tool:latest .
  ```

- エントリポイントとシェル

  HealthOmics ワークフローエンジンは、ワークフロータスクで使用されるコンテナイメージへのコマンドオーバーライドとして bash スクリプトを挿入します。したがって、コンテナイメージは、bash シェルがデフォルトになるように、指定された ENTRYPOINT なしで構築する必要があります。

- マウントされたパス

  共有ファイルシステムは /tmp でコンテナタスクにマウントされます。この場所にコンテナイメージに組み込まれているデータまたはツールはすべて上書きされます。

  ワークフロー定義は、/mnt/workflow の読み取り専用マウントを介してタスクで使用できます。

- イメージのサイズ

  コンテナイメージの最大サイズHealthOmics ワークフローの固定サイズクォータについては、「」を参照してください。

HealthOmics ワークフローの環境変数

HealthOmics は、コンテナで実行されているワークフローに関する情報を含む環境変数を提供します。これらの変数の値は、ワークフロータスクのロジックで使用できます。

すべての HealthOmics ワークフロー変数はAWS_WORKFLOW_、 プレフィックスで始まります。このプレフィックスは、保護された環境変数プレフィックスです。ワークフローコンテナの独自の変数にこのプレフィックスを使用しないでください。

HealthOmics には、次のワークフロー環境変数が用意されています。

AWS_REGION

この変数は、コンテナが実行されているリージョンです。

AWS_WORKFLOW_RUN

この変数は、現在の実行の名前です。

AWS_WORKFLOW_RUN_ID

この変数は、現在の実行の実行識別子です。

AWS_WORKFLOW_RUN_UUID

この変数は、現在の実行の実行 UUID です。

AWS_WORKFLOW_TASK

この変数は現在のタスクの名前です。

AWS_WORKFLOW_TASK_ID

この変数は、現在のタスクのタスク識別子です。

AWS_WORKFLOW_TASK_UUID

この変数は、現在のタスクのタスク UUID です。

次の例は、各環境変数の一般的な値を示しています。

```
AWS Region: us-east-1
Workflow Run: arn:aws:omics:us-east-1:123456789012:run/6470304
Workflow Run ID: 6470304
Workflow Run UUID: f4d9ed47-192e-760e-f3a8-13afedbd4937
Workflow Task: arn:aws:omics:us-east-1:123456789012:task/4192063
Workflow Task ID: 4192063
Workflow Task UUID: f0c9ed49-652c-4a38-7646-60ad835e0a2e
```

Amazon ECR コンテナイメージでの Java の使用

ワークフロータスクで GATK などの Java アプリケーションを使用する場合は、コンテナの次のメモリ要件を考慮してください。

- Java アプリケーションはスタックメモリとヒープメモリを使用します。デフォルトでは、最大ヒープメモリはコンテナで使用可能なメモリの合計に対する割合です。デフォルトでは、特定の

JVM ディストリビューションと JVM バージョンによって異なるため、JVM の関連ドキュメントを参照するか、Java コマンドラインオプション (「-Xmx」など) を使用してヒープメモリの最大数を明示的に設定してください。

- JVM スタックにもメモリが必要なため、最大ヒープメモリをコンテナのメモリ割り当ての 100% に設定しないでください。メモリは、JVM ガベージコレクターおよびコンテナで実行されているその他のオペレーティングシステムプロセスにも必要です。
- GATK などの一部の Java アプリケーションでは、ネイティブメソッド呼び出しや、メモリマッピングファイルなどのその他の最適化を使用できます。これらの手法には、JVM 最大ヒープパラメータによって制御されない「オフヒープ」で実行されるメモリ割り当てが必要です。

  Java アプリケーションがオフヒープメモリを割り当てることを知っている (または疑っている) 場合は、タスクメモリ割り当てにオフヒープメモリ要件が含まれていることを確認してください。

  これらのオフヒープ割り当てによりコンテナのメモリが不足した場合、JVM はこのメモリを制御しないため、通常 Java OutOfMemory エラーは表示されません。

ECR コンテナイメージにタスク入力を追加する

ワークフロータスクの実行に必要なすべての実行可能ファイル、ライブラリ、スクリプトを、タスクの実行に使用される Amazon ECR イメージに追加します。

タスクコンテナイメージの外部にあるスクリプト、バイナリ、ライブラリを使用しないことがベストプラクティスです。これは、`nf-core`ワークフローパッケージの一部として`bin`ディレクトリを使用するワークフローを使用する場合に特に重要です。このディレクトリはワークフロータスクで使用できますが、読み取り専用ディレクトリとしてマウントされます。このディレクトリに必要なリソースはタスクイメージにコピーし、実行時またはタスクに使用されるコンテナイメージの構築時に使用可能にする必要があります。

HealthOmics がサポートするコンテナイメージの最大サイズ[HealthOmics ワークフローの固定サイズクォータ]については、「」を参照してください。

プライベートワークフローの Sentieon ライセンスのリクエスト

プライベートワークフローで Sentieon ソフトウェアを使用している場合は、Senieon ライセンスが必要です。Sentieon ソフトウェアのライセンスをリクエストしてセットアップするには、次の手順に従います。

- Sentieon ライセンスをリクエストする

  - ソフトウェアライセンスをリクエストするには、Sentieon サポートグループ (support@sentieon.com) に E メールを送信します。
    - E メールに AWS 正規ユーザー ID を入力します。
    - 以下の手順に従って、 AWS 正規ユーザー ID を見つけます。
- HealthOmics サービスロールを更新して、リージョン内の Sentieon ライセンスサーバープロキシと Sentieon Omics バケットへのアクセスを許可します。次の例では、 へのアクセスを許可しますus-east-1。必要に応じて、このテキストを自分のリージョンに置き換えます。

  JSON

```
{
          "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObjectAcl",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::omics-ap-us-east-1/*",
                "arn:aws:s3:::sentieon-omics-license-us-east-1/*"
            ]
        }
    ]
}
```

- Sentieon ライセンスサーバープロキシにアクセスするための AWS サポートケースを生成します。
  - サポートケースを作成するには、support.console.aws.amazon.com に移動します。
  - サポートケースで AWS アカウント と リージョンを指定します。アカウントがライセンスサーバープロキシの許可リストに追加されます。
- Sentieon コンテナと Sentieon ライセンススクリプトを使用してプライベートワークフローを構築します。
  - プライベートワークフロー内で Sentieon ツールを使用するその他の手順については、GitHub のSentieon-Amazon-Omics」を参照してください。

- Sentieon ソフトウェアバージョン 202112.07 以降では、HealthOmics ライセンスサーバープロキシがサポートされています。202112.07 より前のバージョンの Sentieon ソフトウェアを使用するには、Sentieon サポートにお問い合わせください。

HealthOmics のワークフローlinters

ワークフローを作成したら、最初の実行を開始する前に、ワークフローで linter を実行することをお勧めします。linter は、実行が失敗する原因となるエラーを検出します。

WDL の場合、ワークフローを作成すると HealthOmics は自動的に linter を実行します。linter 出力は、get-workflowレスポンスの `statusMessage`フィールドで使用できます。次の CLI コマンドを使用してステータス出力を取得します (作成した WDL ワークフローのワークフロー ID を使用します)。

```
aws omics get-workflow
   –id 123456
   –query 'statusMessage'
```

HealthOmics には、ワークフローを作成する前にワークフロー定義で実行できる linter が用意されています。HealthOmics に移行する既存のパイプラインでこれらの linters を実行します。

- WDL – WDL linter を実行するためのパブリック Amazon ECR イメージ。
- Nextflow – Nextflow の Linter ルールを実行するためのパブリック Amazon ECR イメージ。この linter のソースコードには、GitHub からアクセスできます。
- CWL – 利用できません

ワークフローの作成または更新

プライベートワークフローを作成するには、以下が必要です。

- Workflow definition file: WDL、Nextflow、または で記述されたワークフロー定義ファイルCWL。ワークフロー定義は、ワークフローを使用する実行の入力と出力を指定します。また、コンピューティングやメモリの要件など、ワークフローの実行タスクと実行タスクの仕様も含まれています。ワークフロー定義ファイルは `.zip`形式である必要があります。詳細については、HealthOmics のワークフロー定義ファイル」を参照してください。

- Parameter template file (オプション): で記述されたパラメータテンプレートファイルJSON。ファイルを作成して実行パラメータを定義するか、HealthOmics がパラメータテンプレートを生成します。詳細については、HealthOmics ワークフローのパラメータテンプレートファイル」を参照してください。
- Amazon ECR container images: ワークフローのコンテナイメージを作成し、プライベート Amazon ECR リポジトリに保存します。
- Sentieon licenses (オプション): プライベートワークフローでSentieonソフトウェアを使用する Sentieonライセンスをリクエストします。

4 MiB (圧縮) を超えるワークフロー定義ファイルの場合、ワークフローの作成時に次のいずれかのオプションを選択します。

- Amazon Simple Storage Service フォルダにアップロードし、場所を指定します。
- 外部リポジトリ (最大サイズ 1 GiB) にアップロードし、リポジトリの詳細を指定します。

ワークフローを作成したら、 `UpdateWorkflow`オペレーションを使用して次のワークフロー情報を更新できます。

- 名前
- 説明
- デフォルトのストレージタイプ
- デフォルトのストレージ容量 (ワークフロー ID を使用)
- README.md ファイル

ワークフロー内のその他の情報を変更するには、新しいワークフローまたはワークフローバージョンを作成します。

ワークフローのバージョニングを使用して、ワークフローを整理および構造化します。バージョンは、反復ワークフロー更新の導入を管理するのにも役立ちます。バージョンの詳細については、「ワークフローバージョンを作成する」を参照してください。

トピック

- プライベートワークフローを作成する
- プライベートワークフローを更新する
- プライベートワークフローを削除する

- ワークフローのステータスを確認する
- ワークフロー定義からゲノムファイルを参照する

プライベートワークフローを作成する

HealthOmics コンソール、 AWS CLI コマンド、またはいずれかの AWS SDKs を使用してワークフローを作成します。

> **Note**
>
> ワークフロー名に個人を特定できる情報 (PII) を含めないでください。これらの名前は CloudWatch ログに表示されます。

ワークフローを作成すると、HealthOmics は汎用一意識別子 (UUID) をワークフローに割り当てます。ワークフロー UUID は、ワークフローとワークフローバージョン間で一意なグローバル一意識別子 (ガイド) です。データ出所の目的で、ワークフロー UUID を使用してワークフローを一意に識別することをお勧めします。

トピック

- コンソールを使用したワークフローの作成
- CLI を使用したワークフローの作成
- SDK を使用したワークフローの作成

コンソールを使用したワークフローの作成

ワークフローを作成する手順

1. HealthOmics コンソールを開きます。
2. 左上のナビゲーションペイン (≡) を選択し、プライベートワークフローを選択します。
3. プライベートワークフローページで、ワークフローの作成を選択します。
4. ワークフローの定義ページで、次の情報を入力します。
    1. ワークフロー名: このワークフローの固有の名前。ワークフロー名を設定して、 AWS HealthOmics コンソールと CloudWatch ログで実行を整理することをお勧めします。
    2. 説明 (オプション): このワークフローの説明。

5. ワークフロー定義パネルで、次の情報を指定します。

   1. ワークフロー言語 (オプション): ワークフローの仕様言語を選択します。それ以外の場合、HealthOmics はワークフロー定義から言語を決定します。
   2. ワークフロー定義ソースでは、Git ベースのリポジトリ、Amazon S3 の場所、またはローカルドライブから定義フォルダをインポートすることを選択します。

      a. リポジトリサービスからのインポートの場合：

      > **Note**
      >
      > HealthOmics は、、GitHub、GitLab、、 のパブリックリポジトリとプライベートリポジトリをサポートしていますBitbucketGitHub self-managedGitLab self-managed。

      i. 接続を選択して、 AWS リソースを外部リポジトリに接続します。接続を作成するには、「」を参照してください外部コードリポジトリに接続する。

      > **Note**
      >
      > TLV リージョンのお客様は、ワークフローを作成するにはIAD、 (us-east-1) リージョンで接続を作成する必要があります。

      ii. 完全なリポジトリ ID で、リポジトリ ID を user-name/repo-name として入力します。このリポジトリ内のファイルにアクセスできることを確認します。

      iii. ソースリファレンス (オプション) に、リポジトリソースリファレンス (ブランチ、タグ、またはコミット ID) を入力します。ソース参照が指定されていない場合、HealthOmics はデフォルトのブランチを使用します。

      iv. ファイルパターンを除外 で、特定のフォルダ、ファイル、または拡張子を除外するファイルパターンを入力します。これにより、リポジトリファイルをインポートする際のデータサイズを管理できます。最大 50 パターンがあり、パターンは glob パターン構文に従う必要があります。例:

      A. `tests/`

      B. `*.jpeg`

      C. `large_data.zip`

      b. S3 から定義フォルダを選択する：

i. zip 形式のワークフロー定義フォルダを含む Amazon S3 の場所を入力します。Amazon S3 バケットは、ワークフローと同じリージョンにある必要があります。

ii. アカウントが Amazon S3 バケットを所有していない場合は、S3 バケット所有者の AWS アカウント ID にバケット所有者のアカウント ID を入力します。 S3 この情報は、HealthOmics がバケットの所有権を検証できるようにするために必要です。

c. ローカルソースから定義フォルダを選択する：

i. zip 形式のワークフロー定義フォルダのローカルドライブの場所を入力します。

3. メインワークフロー定義ファイルパス (オプション): zip 形式のワークフロー定義フォルダまたはリポジトリから ファイルへのmainファイルパスを入力します。ワークフロー定義フォルダにファイルが 1 つしかない場合、またはメインファイルの名前が「main」の場合、このパラメータは必要ありません。

6. README ファイル (オプション) パネルで、次の情報を指定します。

1. README ファイルのソースを選択します。

a. リポジトリサービスからのインポートの場合、README ファイルパスにリポジトリ内の README ファイルへのパスを入力します。

b. S3 からファイルを選択するには、S3 の README ファイルに S3README ファイルの Amazon S3 URI を入力します。

c. ローカルソースからファイルを選択する: ローカルソースから README ファイルで、ローカルソースから README ファイルをアップロードします。

2. README ファイルパスで、ソースの README ファイルへのパスを入力します。

7. デフォルトの実行ストレージ設定パネルで、このワークフローを使用する実行のデフォルトの実行ストレージタイプと容量を指定します。

1. 実行ストレージタイプ: 一時実行ストレージのデフォルトとして静的ストレージと動的ストレージのどちらを使用するかを選択します。デフォルトは静的ストレージです。

2. Run storage capacity (オプション): 静的実行ストレージタイプでは、このワークフローに必要なデフォルトの実行ストレージ量を入力できます。このパラメータのデフォルト値は 1200 GiB です。実行を開始するときに、これらのデフォルト値を上書きできます。

8. タグ (オプション): 最大 50 個のタグをこのワークフローに関連付けることができます。

9. [次へ] を選択します。

10. ワークフローパラメータの追加 (オプション) ページで、パラメータソースを選択します。

1. ワークフロー定義ファイルからの解析の場合、HealthOmics はワークフロー定義ファイルからワークフローパラメータを自動的に解析します。
2. Git リポジトリからパラメータテンプレートを提供するには、リポジトリからパラメータテンプレートファイルへのパスを使用します。
3. ローカルソースから JSON ファイルを選択する で、パラメータを指定するローカルソースからJSONファイルをアップロードします。
4. ワークフローパラメータを手動で入力するには、パラメータ名と説明を手動で入力します。

11. パラメータプレビューパネルでは、このワークフローバージョンのパラメータを確認または変更できます。JSON ファイルを復元すると、ローカルで行った変更はすべて失われます。
12. [次へ] を選択します。
13. ワークフロー設定を確認し、ワークフローの作成を選択します。

CLI を使用したワークフローの作成

ワークフローとパラメータを定義したら、次に示すように CLI を使用してワークフローを作成できます。

```
aws omics create-workflow  \
  --name "my_workflow"   \
  --definition-zip fileb://my-definition.zip \
  --parameter-template file://my-parameter-template.json
```

Amazon S3 フォルダにあるワークフロー定義ファイルの場合は、 の代わりに `definition-uri`パラメータを使用して場所を入力します`definition-zip`。詳細については、AWS HealthOmics API リファレンスのCreateWorkflow」を参照してください。

`create-workflow` リクエストは次のように応答します。

```
{
  "arn": "arn:aws:omics:us-west-2:....",
  "id": "1234567",
  "status": "CREATING",
  "tags": {
      "resourceArn": "arn:aws:omics:us-west-2:...."
  },
  "uuid": "64c9a39e-8302-cc45-0262-2ea7116d854f"
}
```

ワークフローの作成時に使用するオプションパラメータ

ワークフローを作成するときに、任意のオプションパラメータを指定できます。詳細については、AWS HealthOmics API リファレンスのCreateWorkflow」を参照してください。

複数のワークフロー定義ファイルを含める場合は、 mainパラメータを使用して、ワークフローのメイン定義ファイルとなるファイルを指定します。

ワークフロー定義ファイルを Amazon S3 フォルダにアップロードした場合は、次の例に示すように、 definition-uriパラメータを使用して場所を指定します。アカウントが Amazon S3 バケットを所有していない場合は、所有者の AWS アカウント ID を指定します。

```
aws omics create-workflow  \
  --name Test  \
  --main multi_workflow/workflow2.wdl  \
  --definition-uri s3://omics-bucket/workflow-definition/  \
  --owner-id  123456789012  \
  --parameter-template file://params_sample_description.json
```

デフォルトの実行ストレージタイプ (DYNAMIC または STATIC) と実行ストレージ容量 (静的ストレージに必要) を指定できます。実行ストレージタイプの詳細については、「」を参照してくださいHealthOmics ワークフローでストレージタイプを実行する。

```
aws omics create-workflow  \
  --name my_workflow   \
  --definition-zip fileb://my-definition.zip \
  --parameter-template file://my-parameter-template.json   \
  --storage-type 'STATIC'  \
  --storage-capacity 1200  \
```

アクセラレーターパラメータを使用して、高速コンピューティングインスタンスで実行されるワークフローを作成します。次の例は、 --acceleratorsパラメータの使用方法を示しています。

```
aws omics create-workflow --name workflow name \
   --definition-uri s3://amzn-s3-demo-bucket1/GPUWorkflow.zip \
   --accelerators GPU
```

SDK を使用したワークフローの作成

ワークフローは、いずれかの SDKs を使用して作成できます。次の例は、Python SDK を使用してワークフローを作成する方法を示しています。

```
import boto3

omics = boto3.client('omics')

with open('definition.zip', 'rb') as f:
   definition = f.read()

response = omics.create_workflow(
   name='my_workflow',
   definitionZip=definition,
   parameterTemplate={ ... }
)
```

プライベートワークフローを更新する

HealthOmics コンソール、 AWS CLI コマンド、またはいずれかの AWS SDKs を使用してワークフローを更新できます。

> **Note**
>
> ワークフロー名に個人を特定できる情報 (PII) を含めないでください。これらの名前は CloudWatch ログに表示されます。

トピック

- コンソールを使用したワークフローの更新
- CLI を使用したワークフローの更新
- SDK を使用したワークフローの更新

コンソールを使用したワークフローの更新

ワークフローを更新するステップ

1. HealthOmics コンソールを開きます。
2. 左上のナビゲーションペイン (≡) を選択し、プライベートワークフローを選択します。
3. プライベートワークフローページで、更新するワークフローを選択します。
4. ワークフローページで、次の操作を行います。

- ワークフローにバージョンがある場合は、必ずデフォルトバージョンを選択してください。
- アクションリストから選択した編集を選択します。

5. ワークフローの編集ページで、次の値のいずれかを変更できます。

- ワークフロー名。
- ワークフローの説明。
- ワークフローのデフォルトの Run ストレージタイプ。
- デフォルトの実行ストレージ容量 (実行ストレージタイプが静的ストレージの場合) 。デフォルトの実行ストレージ設定の詳細については、「」を参照してくださいコンソールを使用したワークフローの作成。

6. 変更を保存する を選択して変更を適用します。

CLI を使用したワークフローの更新

次の例に示すように、ワークフロー名と説明を更新できます。デフォルトの実行ストレージタイプ (STATIC または DYNAMIC) を変更し、ストレージ容量 (静的ストレージタイプの場合) を実行することもできます。実行ストレージタイプの詳細については、「」を参照してくださいHealthOmics ワークフローでストレージタイプを実行する。

```
aws omics update-workflow
  --id 1234567
  --name my_workflow
  --description "updated workflow"
  --storage-type 'STATIC'
  --storage-capacity 1200
```

update-workflow リクエストに対するレスポンスを受信しません。

SDK を使用したワークフローの更新

ワークフローは、いずれかの SDKs を使用して更新できます。

次の例は、Python SDK を使用してワークフローを更新する方法を示しています。

```
import boto3

omics = boto3.client('omics')
```

```
response = omics.update_workflow(
   name='my_workflow',
   description='updated workflow'
)
```

プライベートワークフローを削除する

ワークフローが不要になった場合は、HealthOmics コンソール、 AWS CLI コマンド、またはいずれかの AWS SDKs を使用してワークフローを削除できます。次の条件を満たすワークフローを削除できます。

- ステータスは ACTIVE または FAILED です。
- アクティブな共有はありません。
- すべてのワークフローバージョンを削除しました。

ワークフローを削除しても、ワークフローを使用している進行中の実行には影響しません。

トピック

- コンソールを使用したワークフローの削除
- CLI を使用したワークフローの削除
- SDK を使用したワークフローの削除

コンソールを使用したワークフローの削除

ワークフローを削除するには

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、プライベートワークフローを選択します。
3. プライベートワークフローページで、削除するワークフローを選択します。
4. ワークフローページで、アクションリストから選択した削除を選択します。
5. ワークフローの削除モーダルで、「確認」と入力して削除を確認します。
6. [削除] を選択します。

CLI を使用したワークフローの削除

次の例は、 AWS CLI コマンドを使用してワークフローを削除する方法を示しています。この例を実行するには、 を、削除するワークフローの *workflow id* ID に置き換えます。

```
aws omics delete-workflow
  --id workflow id
```

HealthOmics はdelete-workflowリクエストにレスポンスを送信しません。

SDK を使用したワークフローの削除

ワークフローは、いずれかの SDKsを使用して削除できます。

次の例は、Python SDK を使用してワークフローを削除する方法を示しています。

```
import boto3

omics = boto3.client('omics')

response = omics.delete_workflow(
   id='1234567'
)
```

ワークフローのステータスを確認する

ワークフローを作成したら、次に示すように、get-workflow を使用してワークフローのステータスを確認し、その他の詳細を表示できます。

```
aws omics get-workflow --id 1234567
```

レスポンスには、次に示すように、ステータスを含むワークフローの詳細が含まれます。

```
{
    "arn": "arn:aws:omics:us-west-2:....",
    "creationTime": "2022-07-06T00:27:05.542459"
    "id": "1234567",
    "engine": "WDL",
    "status": "ACTIVE",
    "type": "PRIVATE",
    "main": "workflow-crambam.wdl",
```

```
    "name": "workflow_name",
    "storageType": "STATIC",
    "storageCapacity": "1200",
    "uuid": "64c9a39e-8302-cc45-0262-2ea7116d854f"
  }
```

ステータスが に移行した後、このワークフローを使用して実行を開始できますACTIVE。

ワークフロー定義からゲノムファイルを参照する

HealthOmics リファレンスストアオブジェクトは、次のような URI で参照できます。指定されたreference ID場合は、独自の account ID、reference store ID、および を使用します。

```
omics://account ID.storage.us-west-2.amazonaws.com/reference store id/reference/id
```

ワークフローによっては、参照ゲノムの ファイルSOURCEと INDEX ファイルの両方が必要になります。前の URI はデフォルトの短縮形式であり、デフォルトで SOURCE ファイルになります。いずれかのファイルを指定するには、次のように長い URI フォームを使用できます。

```
omics://account ID.storage.us-west-2.amazonaws.com/reference store id/reference/id/
source
omics://account ID.storage.us-west-2.amazonaws.com/reference store id/reference/id/
index
```

シーケンス読み取りセットを使用すると、図のように同様のパターンになります。

```
aws omics create-workflow \
     --name workflow name \
     --main sample workflow.wdl \
     --definition-uri omics://account ID.storage.us-
west-2.amazonaws.com/sequence_store_id/readSet/id \
     --parameter-template file://parameters_sample_description.json
```

FASTQ に基づくリードセットなど、一部のリードセットにはペアのリードを含めることができます。次の例では、SOURCE1 と SOURCE2 と呼ばれます。BAM や CRAM などの形式には SOURCE1 ファイルのみが含まれます。一部のリードセットには、 baiや ファイルなどの INDEX crai ファイルが含まれます。前述の URI はデフォルトの短縮形式であり、デフォルトで SOURCE1 ファイルになります。正確なファイルまたはインデックスを指定するには、次のように長い URI フォームを使用できます。

```
omics://123456789012.storage.us-west-2.amazonaws.com/<sequence_store_id>/readSet/<id>/
source1
omics://123456789012.storage.us-west-2.amazonaws.com/<sequence_store_id>/readSet/<id>/
source2
omics://123456789012.storage.us-west-2.amazonaws.com/<sequence_store_id>/readSet/<id>/
index
```

以下は、2 つの Omics Storage URIs。

```
{
   "input_fasta": "omics://123456789012.storage.us-west-2.amazonaws.com/
<reference_store_id>/reference/<id>",
   "input_cram": "omics://123456789012.storage.us-west-2.amazonaws.com/
<sequence_store_id>/readSet/<id>"
}
```

開始/実行リクエスト`--inputs file://<input_file.json>`に を追加して AWS CLI 、 の入力 JSON ファイルを参照します。

README ファイルの使用

ワークフローの手順、図、重要な情報を含む README.md ファイルをアップロードします。各ワークフローバージョンは 1 つの README ファイルをサポートしており、いつでも更新できます。

README の要件は次のとおりです。

- README ファイルはマークダウン (.md) 形式である必要があります
- 最大ファイルサイズ: 500 KiB

トピック

既存の README を使用する

Git リポジトリからエクスポートREADMEs には、通常リポジトリの外部では機能しない相対リンクが含まれています。HealthOmics Git 統合は、これらをコンソールで適切にレンダリングするための絶対リンクに自動的に変換するため、手動で URL を更新する必要がなくなります。

Amazon S3 またはローカルドライブからインポートREADMEs の場合、イメージとリンクはパブリック URLs を使用するか、適切なレンダリングのために相対パスを更新する必要があります。

> **Note**
>
> HealthOmics コンソールに表示するには、イメージをパブリックにホストする必要があります。GitHub Enterprise Server または GitLab Self-Managed リポジトリに保存されているイメージはレンダリングできません。

レンダリング条件

HealthOmics コンソールは、絶対パスを使用してパブリックにアクセス可能なイメージとリンクを補間します。プライベートリポジトリから URLsレンダリングするには、ユーザーがリポジトリにアクセスできる必要があります。カスタムドメインを使用する GitHub Enterprise Serverまたは GitLab Self-Managedリポジトリの場合、HealthOmics は相対リンクを解決したり、これらのプライベートリポジトリに保存されているイメージをレンダリングしたりすることはできません。

次の表は、 AWS コンソールの README ビューでサポートされているマークダウン要素を示しています。

要素	AWS コンソール
アラート	はい、ただしアイコンなし
バッジ	あり
基本的なテキストフォーマット	あり
コードブロック	はい。ただし、構文の強調表示とコピーボタンの機能はありません。
折りたたみ可能なセクション	あり
見出し	あり
イメージ形式	あり
イメージ (クリック可能)	あり

要素	AWS コンソール
改行	あり
Mermaid の図	グラフを開く、グラフの位置を移動する、コードをコピーできるのは のみです
Quotes	あり
サブスクリプトとスーパースクリプト	あり
テーブル	はい。ただし、テキストの整列はサポートされていません
テキストの整列	あり

HealthOmics でのワークフローのバージョニング

ワークフローを変更する必要がある場合は、新しいワークフローまたは新しいワークフローバージョンを作成できます。実行ロジックに影響を与えない許可された設定変更を除き、バージョンは変更できません。

ワークフローバージョンには以下の利点があります。

- バージョンは、関連するワークフローの論理グループを形成します。各ワークフローバージョンにユーザー定義の名前を追加して、より簡単に管理できます (特にバージョン数が多いワークフローの場合) 。
- ワークフローの複数のバージョンを同時に実行できます。
- ワークフローのすべてのバージョンは同じワークフロー ID とベース ARN を共有しているため、ワークフローを変更した後のパイプライン管理を簡素化できます。
- ワークフローバージョンは、ワークフローと同じレベルのデータ出典を提供します。バージョンはイミュータブルであり、HealthOmics はワークフローバージョンごとに一意の ARN を作成します。バージョン ARN には、次の例に示すように、ワークフロー ID とバージョン名が含まれます。

  ```
  arn:aws:omics:us-west-2:123456789012:workflow/1234567/version/
  myUniqueVersionName
  ```

- 共有ワークフローを所有している場合は、サブスクライバー (以前のバージョンを引き続き使用できるユーザー) を中断することなくワークフローを更新できます。サブスクライバーは、すべてのワークフローバージョンにアクセスできます。新しいバージョンを作成する場合、ワークフローを再共有する必要はありません。
- ワークフロー実行を開始するときに、ワークフローバージョンを指定できます。
  - ユーザーは、本番稼働用に安定したバージョンを維持し、テスト実行用に最新バージョンを試すことができます。
  - 新しいバージョンで問題が発生した場合、ユーザーはワークフローの以前のバージョンに戻すことができます。
  - 共有ワークフローのサブスクライバーは、使用するバージョンを選択できます。

トピック

デフォルトのワークフローバージョン

ワークフローの 1 つ以上のバージョンを作成すると、HealthOmics は元のワークフローをデフォルトバージョンとして扱います。実行を開始するときは、オプションで実行のワークフローバージョンを指定できます。実行の開始時にバージョンを指定しない場合、HealthOmics はデフォルトバージョンを使用します。

コンソールでは、HealthOmics はデフォルトバージョンラベルを持つ元のワークフローを示します。コンソールは、1 つ以上のワークフローバージョンを作成した後にのみこのラベルを使用します。元のワークフローは常にデフォルトバージョンのままです。デフォルトとして他のバージョンを割り当てることはできません。

ワークフローに関連付けられている他のバージョンがある場合、ワークフローのデフォルトバージョンを削除することはできません。詳細については、「プライベートワークフローを削除する」を参照してください。

ワークフローバージョンを作成する

ワークフローの新しいバージョンを作成するときは、新しいバージョンの設定値を指定する必要があります。ワークフローから設定値を継承しません。

バージョンを作成するときは、このワークフローに固有のバージョン名を指定します。HealthOmics がバージョンを作成した後で名前を変更することはできません。

バージョン名は文字または数字で始まり、大文字と小文字、数字、ハイフン、ピリオド、アンダースコアを含めることができます。最大長は 64 文字です。例えば、version1、version2、version3 などの単純な命名スキームを使用できます。ワークフローバージョンを 2.7.0、2.7.1、2.7.2 などの独自の内部バージョニング規則と一致させることもできます。

必要に応じて、バージョンの説明フィールドを使用して、このバージョンに関するメモを追加します。例: Fix for syntax error in workflow definition。

> **Note**
>
> バージョン名に個人を特定できる情報 (PII) を含めないでください。バージョン名はワークフローバージョン ARN に表示されます。

HealthOmics はワークフローバージョンに一意の ARN を割り当てます。ARN は、ワークフロー ID とバージョン名の組み合わせに基づいて一意です。

> **Warning**
>
> ワークフローバージョンを削除すると、HealthOmics ではバージョン名を別のワークフローバージョンに再利用できます。ベストプラクティスは、バージョン名を再利用しないことです。名前を再使用する場合、ワークフローと各バージョンには、出典に使用できる一意の UUID があります。

トピック

コンソールを使用してワークフローバージョンを作成する

ワークフローを作成する手順

1. HealthOmics コンソールを開きます。
2. 左上のナビゲーションペイン (≡) を選択し、プライベートワークフローを選択します。
3. プライベートワークフローページで、新しいバージョンのワークフローを選択します。
4. ワークフローの詳細ページで、新しいバージョンの作成を選択します。
5. バージョンの作成ページで、次の情報を入力します。
    1. バージョン名: ワークフロー全体で一意のワークフローバージョンの名前を入力します。
    2. バージョンの説明 (オプション): 説明フィールドを使用して、このバージョンに関するメモを追加できます。
6. ワークフロー定義パネルで、次の情報を指定します。
    1. ワークフロー言語 (オプション): ワークフローバージョンの仕様言語を選択します。それ以外の場合、HealthOmics はワークフロー定義から言語を決定します。
    2. ワークフロー定義ソースでは、Git ベースのリポジトリ、Amazon S3 の場所、またはローカルドライブから定義フォルダをインポートすることを選択します。
        a. リポジトリサービスからのインポートの場合 :

> **Note**
>
> HealthOmics は、、GitHub、GitLab、、 のパブリックリポジトリとプライベートリポジトリをサポートしていますBitbucketGitHub self-managedGitLab self-managed。

        i. 接続を選択して、 AWS リソースを外部リポジトリに接続します。接続を作成するには、「」を参照してください外部コードリポジトリに接続する。

> **Note**
>
> TLV リージョンのお客様は、ワークフローを作成するにはIAD、 (us-east-1) リージョンで接続を作成する必要があります。

ii. 完全なリポジトリ ID で、リポジトリ ID を user-name/repo-name として入力します。このリポジトリ内のファイルにアクセスできることを確認します。

iii. ソースリファレンス (オプション) に、リポジトリソースリファレンス (ブランチ、タグ、またはコミット ID) を入力します。ソース参照が指定されていない場合、HealthOmics はデフォルトのブランチを使用します。

iv. ファイルパターンを除外 で、特定のフォルダ、ファイル、または拡張子を除外するファイルパターンを入力します。これにより、リポジトリファイルをインポートする際のデータサイズを管理できます。最大 50 パターンがあり、パターンは glob パターン構文に従う必要があります。例:

A. `tests/`

B. `*.jpeg`

C. `large_data.zip`

b. S3 から定義フォルダを選択する：

i. zip 形式のワークフロー定義フォルダを含む Amazon S3 の場所を入力します。Amazon S3 バケットは、ワークフローと同じリージョンにある必要があります。

ii. アカウントが Amazon S3 バケットを所有していない場合は、S3 バケット所有者の AWS アカウント ID にバケット所有者のアカウント ID を入力します。 S3 この情報は、HealthOmics がバケットの所有権を検証できるようにするために必要です。

c. ローカルソースから定義フォルダを選択する：

i. zip 形式のワークフロー定義フォルダのローカルドライブの場所を入力します。

3. メインワークフロー定義ファイルパス (オプション): 圧縮されたワークフロー定義フォルダまたはリポジトリから ファイルへの`main`ファイルパスを入力します。ワークフロー定義フォルダにファイルが 1 つしかない場合、またはメインファイルの名前が「main」の場合、このパラメータは必要ありません。

7. デフォルトの実行ストレージ設定パネルで、このワークフローを使用する実行のデフォルトの実行ストレージタイプと容量を指定します。

1. 実行ストレージタイプ: 一時実行ストレージのデフォルトとして静的ストレージと動的ストレージのどちらを使用するかを選択します。デフォルトは静的ストレージです。

2. Run storage capacity (オプション): 静的実行ストレージタイプでは、このワークフローに必要なデフォルトの実行ストレージ量を入力できます。このパラメータのデフォルト値は 1200 GiB です。実行を開始するときに、これらのデフォルト値を上書きできます。

8. タグ (オプション): 最大 50 個のタグをこのワークフローバージョンに関連付けることができます。

9. [次へ] を選択します。
10. ワークフローパラメータの追加 (オプション) ページで、パラメータソースを選択します。
    1. ワークフロー定義ファイルからの解析の場合、HealthOmics はワークフロー定義ファイルからワークフローパラメータを自動的に解析します。
    2. Git リポジトリからパラメータテンプレートを提供するには、リポジトリからパラメータテンプレートファイルへのパスを使用します。
    3. ローカルソースから JSON ファイルを選択する で、パラメータを指定するローカルソースからJSONファイルをアップロードします。
    4. ワークフローパラメータを手動で入力するには、パラメータ名と説明を手動で入力します。
11. パラメータプレビューパネルでは、このワークフローバージョンのパラメータを確認または変更できます。JSON ファイルを復元すると、ローカルで行った変更はすべて失われます。
12. [次へ] を選択します。
13. バージョン設定を確認し、バージョンの作成を選択します。

バージョンが作成されると、コンソールはワークフローの詳細ページに戻り、ワークフローとバージョンテーブルに新しいバージョンを表示します。

CLI を使用してワークフローバージョンを作成する

`CreateWorkflowVersion` API オペレーションを使用してワークフローバージョンを作成できます。オプションのパラメータの場合、HealthOmics は次のデフォルトを使用します。

パラメータ	デフォルト
エンジン	ワークフロー定義から決定
ストレージタイプ	STATIC
ストレージ容量 (静的ストレージ用)	1200 GiB
メイン	ワークフロー定義フォルダの内容に基づいて決定されます。詳細については、「HealthOmics ワークフロー定義の要件」を参照してください。
アクセラレータ	なし

パラメータ	デフォルト
[タグ]	なし

次の CLI の例では、デフォルトの実行ストレージとして静的ストレージを使用してワークフローバージョンを作成します。

```
aws omics create-workflow-version \
--workflow-id 1234567  \
--version-name "my_version" \
--engine WDL \
--definition-zip fileb://workflow-crambam.zip \
--description "my version description" \
--main file://workflow-params.json \
--parameter-template file://workflow-params.json \
--storage-type='STATIC'   \
--storage-capacity 1200   \
--tags example123=string  \
--accelerators GPU
```

Amazon S3 フォルダにあるワークフロー定義ファイルの場合は、 の代わりに `definition-uri`パラメータを使用して場所を入力します`definition-zip`。詳細については、AWS HealthOmics API リファレンスのCreateWorkflowVersion」を参照してください。

`create-workflow-version` リクエストに対して次のレスポンスを受け取ります。

```
{
  "workflowId": "1234567",
  "versionName": "my_version",
  "arn": "arn:aws:omics:us-west-2:123456789012:workflow/1234567/version/3",
  "status": "ACTIVE",
  "tags": {
    "environment": "production",
    "owner": "team-alpha"
  },
  "uuid": "0ac9a563-355c-fc7a-1b47-a115167af8a2"
}
```

SDK を使用してワークフローバージョンを作成する

ワークフローは、いずれかの SDKs を使用して作成できます。

次の例は、Python SDK を使用してワークフローバージョンを作成する方法を示しています。

```
import boto3

omics = boto3.client('omics')

with open('definition.zip', 'rb') as f:
   definition = f.read()

response = omics.create_workflow_version(
   workflowId='1234567',
   versionName='my_version',
   requestId='my_request_1'
   definitionZip=definition,
   parameterTemplate={ ... }
)
```

ワークフローバージョンのステータスを確認する

ワークフローバージョンを作成したら、次のように get-workflow-version を使用して、ワークフローのステータスを確認し、その他の詳細を表示できます。

```
aws omics get-workflow-version
--workflow-id 9876543
--version-name "my_version"
```

レスポンスには、次に示すように、ステータスを含むワークフローの詳細が表示されます。

```
{
  "workflowId": "1234567",
  "versionName": "3.0.0",
  "arn": "arn:aws:omics:us-west-2:123456789012:workflow/1234567/version/3.0.0",
  "status": "ACTIVE",
  "description": ...
  "uuid": "0ac9a563-355c-fc7a-1b47-a115167af8a2"
}
```

このワークフローバージョンで実行を開始する前に、ステータスが に移行する必要がありますACTIVE。

ワークフローバージョンを更新する

プライベートワークフローバージョンの説明とデフォルトの実行ストレージ設定を更新できます。ワークフローバージョンのその他の情報を変更するには、新しいバージョンを作成します。

トピック

コンソールを使用してワークフローバージョンを更新する

ワークフローを更新するには

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、プライベートワークフローを選択します。
3. プライベートワークフローページで、ワークフローを選択します。
4. ワークフローページで、更新するワークフローバージョンを選択し、アクションリストから選択した編集を選択します。
   - デフォルトバージョンを選択すると、コンソールでワークフローの編集ページが開きます。詳細については、「プライベートワークフローを更新する」を参照してください。
   - ユーザー定義のバージョンを選択すると、コンソールでバージョンの編集ページが開きます。
5. バージョンの編集ページで、次の情報を入力します。
   - バージョンの説明 (オプション) - このバージョンの説明。
6. デフォルト実行ストレージ設定パネルで、このワークフローバージョンを使用する実行に次のデフォルト値を指定します。実行を開始するときにデフォルト値を上書きできます。
   - Run storage type で、Static または Dynamic を選択します。
   - 静的実行ストレージの場合は、このワークフローバージョンを使用する実行のデフォルトの実行ストレージ容量を選択します。このパラメータのデフォルト値は 1200 GiB です。
7. [Save changes] (変更の保存) をクリックします。

コンソールはワークフローの詳細ページに戻り、更新されたワークフローバージョンを含むページバナーを表示します。

CLI を使用してワークフローバージョンを更新する

次の CLI コマンドを使用して、ワークフローバージョンのパラメータを更新できます。ワークフロー ID とバージョン名の組み合わせにより、バージョンが一意に識別されます。

```
aws omics update-workflow-version
--workflow-id 1234567
--version-name "my_version"
--storage-type 'STATIC'
--storage-capacity 2400
--description "version description"
```

update-workflow-version リクエストには応答しません。

SDK を使用してワークフローバージョンを更新する

ワークフローバージョンは、いずれかの SDKsを使用して更新できます。次の python SDK の例は、ワークフローバージョンのストレージタイプと説明を更新する方法を示しています。

```
import boto3

omics = boto3.client('omics')

response = omics.update_workflow_version(
   workflowID=1234567,
   versionName='3.0.0',
   storageType='DYNAMIC',
   description='new version description'
)
```

ワークフローバージョンを削除する

ユーザー定義のワークフローバージョンは、コンソール、CLI、またはいずれかの SDKs を使用して削除できます。ワークフローバージョンを削除しても、ワークフローバージョンを使用している進行中の実行には影響しません。

を削除することはできませんデフォルトのワークフローバージョン。すべてのユーザー定義バージョンを削除してから、ワークフローを削除します。

トピック

- コンソールを使用してワークフローバージョンを削除する

- CLI を使用してワークフローバージョンを削除する
- SDK を使用してワークフローバージョンを削除する

コンソールを使用してワークフローバージョンを削除する

ワークフローバージョンを削除するには

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、プライベートワークフローを選択します。
3. プライベートワークフローページで、ワークフローを選択します。
4. ワークフローページで、削除するワークフローバージョンを選択し、アクションリストから選択した削除を選択します。
5. ワークフローバージョンの削除モーダルで、「確認」と入力して削除を確認します。
6. [削除] を選択します。

コンソールには、削除されたワークフローバージョンを含むページバナーが表示されます。

CLI を使用してワークフローバージョンを削除する

次の CLI コマンドを使用して、ユーザー定義のワークフローバージョンを削除できます。ワークフロー ID とバージョン名の組み合わせにより、バージョンが一意に識別されます。

```
aws omics delete-workflow-version
--workflow-id 9876543
--version-name "my_version"
```

`delete-workflow-version` リクエストには応答しません。

SDK を使用してワークフローバージョンを削除する

ワークフローは、いずれかの SDKsを使用して削除できます。

次の例は、Python SDK を使用してワークフローを削除する方法を示しています。

```
import boto3

omics = boto3.client('omics')
```

```
response = omics.delete_workflow_version(
   workflowID=1234567,
   versionName='3.0.0'
)
```

HealthOmics の実行を開始する

ワークフローを作成したら、ワークフローを使用して実行を開始できます。

実行を開始すると、HealthOmics はワークフローエンジンが実行中に使用する一時実行ストレージを割り当てます。データの分離とセキュリティを確保するために、HealthOmics は各実行の開始時にストレージをプロビジョニングし、実行の終了時にそれをプロビジョニング解除します。

HealthOmics は、ワークフローの実行とタスクに関連するいくつかのクォータを提供します。デフォルト値は、予期しないコストオーバーランを回避するため、意図的に控えめです。このクォータの引き上げをリクエストできます。詳細については、「HealthOmics サービスクォータ」を参照してください。

実行を開始すると、HealthOmics は実行 ID と実行 uuid を実行に割り当てます。アカウント内の実行には一意の実行 IDs。ただし、HealthOmics は削除された実行 IDsを再利用するため、実行と削除された実行は同じ実行 ID を持つことができます。また、共有ワークフローがアカウントでの実行と同じ実行 ID を持つことはまれですが、可能です。

run uuid は、アカウント間で実行を識別したり、同じ実行 ID を持つアカウント内の 2 つの実行を区別したりするために使用できるグローバル一意識別子 (ガイド) です。

> **Note**
>
> データ出典の目的で、 を使用して実行run uuidを一意に識別することをお勧めします。run uuid は、内部ラボ情報管理システム (LIMs) またはサンプル追跡システムにリンクするための最適な識別子でもあります。

トピック

- HealthOmics ワークフローでストレージタイプを実行する
- HealthOmics 実行の保持モードを実行する
- HealthOmics 実行入力
- HealthOmics での実行の開始

- HealthOmics ワークフローでライフサイクルを実行する
- HealthOmics 実行出力
- 実行失敗の理由
- HealthOmics 実行のタスクライフサイクル
- プライベート HealthOmics ワークフローの最適化を実行する

HealthOmics ワークフローでストレージタイプを実行する

実行を開始すると、HealthOmics はワークフローエンジンが実行中に使用する一時実行ストレージを割り当てます。HealthOmics は、一時実行ストレージをファイルシステムとして提供します。

特定のワークフローまたはワークフロー実行では、動的または静的実行ストレージを選択できます。デフォルトでは、HealthOmics は静的実行ストレージを提供します。

> **Note**
>
> ストレージの使用を実行すると、アカウントに料金が発生します。静的および動的実行ストレージの料金情報については、HealthOmics の料金」を参照してください。

以下のセクションでは、使用する実行ストレージタイプを決定する際に考慮すべき情報を提供します。

動的実行ストレージ

起動時間の短縮が必要な実行、ストレージのニーズが事前にわからない実行、反復的な開発テストサイクルなど、ほとんどの実行には動的実行ストレージを使用することをお勧めします。

実行に必要なストレージやスループットを見積もる必要はありません。HealthOmics は、実行中のファイルシステムの使用率に基づいて、ストレージサイズを動的にスケールアップまたはスケールダウンします。HealthOmics は、ワークフローのニーズに基づいてスループットを動的にスケーリングします。ファイルシステムエラーのストレージ不足が原因で実行が失敗することはありません。

動的実行ストレージは、静的実行ストレージよりもプロビジョニング/プロビジョニング解除の時間を短縮します。セットアップの高速化は、ほとんどのワークフローにとって利点であり、開発/テストサイクル中の利点でもあります。

実行が完了すると (成功パスまたは失敗パス)、getRun API オペレーションは storageCapacity フィールドで実行で使用される最大ストレージを返します。この情報は、ロググループにある実行マ

ニフェストomicsログでも確認できます。2 時間以内に完了する動的ストレージ実行の場合、最大ストレージ値は使用できない場合があります。

動的実行ストレージの場合、実行は NFS プロトコルを使用するファイルシステムをプロビジョニングします。NFS は、CREATE、DELETE、および RENAME ファイルオペレーションをべき等ではないものとして扱います。これにより、コードが適切に処理する必要があるオペレーションの競合状態が発生することがあります。たとえば、存在しないファイルを削除しようとすると、コードが失敗することはありません。動的実行ストレージを採用する前に、冪等性のないファイルオペレーションに強いようにワークフローコードを調整することをお勧めします。「べき等でない操作を安全に処理するためのコード例」を参照してください。

べき等でない操作を安全に処理するためのコード例

次の Python の例は、ファイルが存在しない場合に失敗せずにファイルを削除する方法を示しています。

```
import os
import errno

def remove_file(file_path):
    try:
        os.remove(file_path)
    except OSError as e:
        # If the error is "No such file or directory", ignore it (or log it)
        if e.errno != errno.ENOENT:
            # Otherwise, raise the error
            raise

# Example usage
remove_file("myfile")
```

次の例では、Bash シェルを使用します。ファイルが存在しない場合でも安全に削除するには、以下を使用します。

```
rm -f my_file
```

ファイルを安全に移動 (名前変更) するには、現在のディレクトリにファイルold_nameが存在する場合にのみ move コマンドを実行します。

```
[ -f old_name ] && mv old_name new_name
```

ディレクトリを作成するには、次のコマンドを使用します。

```
mkdir -p mydir/subdir/
```

静的実行ストレージ

静的実行ストレージの場合、実行は Lustre プロトコルを使用するファイルシステムをプロビジョニングします。このプロトコルは、デフォルトで冪等性のないファイルオペレーションに対して回復力があります。べき等でないファイルオペレーションを処理するようにワークフローコードを調整する必要はありません。

HealthOmics は、一定量の実行ストレージを割り当てます。この値は、実行を開始するときに指定します。値を指定しない場合、デフォルトの実行ストレージは 1200 GiB です。StartRun API リクエストでストレージサイズの値を指定すると、システムは値を最も近い 1200 GiB の倍数に切り上げます。そのストレージサイズが利用できない場合は、2400 GiB の最も近い倍数に切り上げられます。

静的実行ストレージの場合、HealthOmics は次のスループット値をプロビジョニングします。

- ベースラインスループットは、プロビジョニングされたストレージ容量の 1 TiB あたり 200 MB/秒です。
- プロビジョニングされたストレージ容量の TiB あたり最大 1300 MB/秒のバーストスループット。

指定されたストレージサイズが小さすぎると、実行は失敗し、ファイルシステムエラーのストレージ不足が発生します。静的実行ストレージは、既知のストレージ要件を持つ予測可能なワークフローに適しています。

静的実行ストレージは、タスクの同時実行性が高い大規模でバースト的なワークロード (大量の RNASeq サンプルが並行して処理されるなど) に適しています。動的実行ストレージよりも、GiB あたりのファイルシステムのスループットが高く、GiB あたりのコストが低くなります。

必要な静的実行ストレージの計算

ベースファイルシステムのインストールでは静的ファイルシステム容量の 7% を使用するため、静的実行ストレージ (動的実行ストレージと比較) を使用する場合、ワークフローには追加の容量が必要です。

動的実行ストレージワークフローを実行して実行で使用される最大ストレージを測定する場合は、次の計算を使用して必要な静的ストレージの最小量を決定します。

```
  static storage required =
```

```
        maximum storage in GiB used by the dynamic run storage
          + (total static file system size in GiB * 0.07)
```

例:

```
    Maximum storage measured from a dynamic run storage workflow run:  500GiB
      File system size: 1200GiB
      7% of the file system size:  84GiB
      500 + 84 = 584GiB of static run storage required for this run.
```

したがって、この実行には 1200GiB (静的実行ストレージの最小容量) で十分です。

HealthOmics 実行の保持モードを実行する

実行が完了すると、HealthOmics は実行メタデータを CloudWatch にアーカイブします。デフォルトでは、CloudWatch 保持ポリシーを変更しない限り、CloudWatch は実行データを無期限に保持します。実行出力は、削除するまで Amazon S3 にも保存されます。

調整可能な の 1 つは、リージョンmaximum number of runs (active and inactive)内の HealthOmics サービスクォータです。HealthOmics は、コンソールおよび API オペレーション (ListRuns および GetRun) で使用できるように、この実行数まで実行メタデータを保持します。実行を開始するときに、実行の保持動作を示す実行保持モードパラメータを設定できます。パラメータは、REMOVE と RETAIN の値をサポートします。

保持モードを REMOVE に設定した新しい実行の場合、HealthOmics が最大実行数を保存した後に実行を追加しようとすると、REMOVE モードを設定した最も古い実行のメタデータが自動的に削除されます。この削除は、CloudWatch または Amazon S3 に保存されているデータには影響しません。

RETAIN は、実行保持モードのデフォルト値です。このモードでの実行の場合、システムは実行メタデータを削除しません。HealthOmics が実行の最大数に達し、すべて RETAIN に設定されている場合、一部の実行を削除するまで追加の実行を作成することはできません。

最大実行数を超えるバッチを同時に実行する予定がある場合は、実行保持モードを REMOVE に設定してください。それ以外の場合、HealthOmics が最大値の後に次の実行を開始しようとすると、バッチは失敗します。

REMOVE 保持モードを使用する際のその他の考慮事項：

- 最初に保持モードとして REMOVE の使用を開始するときは、RETAIN モードを使用する 1 つ以上の実行を削除してスロットを解放することを検討してください。追加の REMOVE 実行を開始すると、自動削除が引き継がれ、新しい実行に十分なスロットが使用可能になります。

- アーカイブされた実行 (または一連の実行) を再実行するには、HealthOmics 再実行 CLI ツールを使用します。このツールの使用方法の詳細と例については、HealthOmics ツール GitHub リポジトリの「Omics の再実行」を参照してください。 HealthOmics GitHub
- 実行ごとに一意の名前を設定することをお勧めします。HealthOmics が実行を削除した後、コンソールまたは API を使用して実行名または実行 ID を見つけることはできません。ただし、CloudWatch を使用して実行名を検索できるため、一意の名前を使用して最適な検索結果を取得できます。
- CloudWatch start-query コマンドを使用して、アーカイブされた実行に関する情報を取得できます。実行名が一意でない場合、クエリは複数のマニフェストを返すことがあります。開始時刻パラメータと終了時刻パラメータは、検索の時間範囲を定義します。

```
aws logs start-query  \
    --log-group-name "/aws/omics/WorkflowLog" \
    --query-string 'filter @logStream like "manifest" and @message like "myRunName"'
 \
    --end-time <END-EPOCH-TIME> --start-time <START-EPOCH-TIME>
```

start-query コマンドはクエリ ID を返します。クエリ ID を get-query-results コマンドに渡すと、クエリ結果が返されます。

```
aws logs get-query-results --query-id QueryId
```

HealthOmics 実行入力

ワークフロー定義でワークフローまたはワークフロータスクの入力ファイルが指定されている場合、HealthOmics はファイルをワークフロー実行専用のスクラッチボリュームにステージングします。これらの入力ファイルは読み取り専用であるため、タスクがワークフロー内の他のタスクへの潜在的な入力を変更できなくなります。ディレクトリのインポートの場合、ディレクトリも読み取り専用です。

多くのゲノミクスアプリケーションは、インデックスファイルがシーケンスファイル (bamファイルのコンパニオンbaiファイルなど) と同じ場所にあることを前提としています。インデックスファイルを含めるには、ワークフロー定義でそれらをタスク入力として指定します。

トピック

- 実行パラメータサイズの管理
- Amazon S3 入力パラメータ形式

- Amazon S3 入力アーカイブの状態

実行パラメータサイズの管理

実行を開始するときは、実行パラメータ JSON オブジェクトまたはファイルで実行入力を指定します。ワークフローには、最大 50 KB の実行パラメータを指定できます。次の手法を使用して、このサイズ制約内にとどまることができます。

- ディレクトリのインポートを使用する

  多数の入力ファイルを指定するには、ファイルの場所ごとにパラメータを指定するのではなく、すべてのファイルを含む Amazon S3 の場所として 1 つのパラメータを指定します。詳細については、次のトピック (Amazon S3 入力パラメータ形式) を参照してください。
- サンプルシートを使用する

  サンプルシートは、fastq.gz アドレスの 1 つの列 (またはペア読み取りの 2 つの列) と、サンプル名などのメタデータの追加の列を含む CSV または TSV ファイルです。サンプルシートは、各入力ファイルのパラメータではなく、実行入力パラメータとして指定します。

  ワークフローは、サンプルシートがワークフロー内のデータ構造にどのようにマッピングされるかを定義します。WDL と CWL でサンプルシートのコードを記述することはできますが、NextFlow ではより一般的です。例については、nf-core GitHub サイトのサンプルシートを参照してください。

Amazon S3 入力パラメータ形式

Amazon S3 の場所を受け入れる入力パラメータの場合、 パラメータは 1 つのファイルの場所またはファイルのディレクトリ全体を指定できます。ディレクトリの使用には、次の利点があります。

- 利便性 – ディレクトリ名を パラメータとして指定します。各ファイル名は一覧表示しません。
- コンパクト性 – 入力パラメータの最大ファイルサイズは 50 KB です。入力ファイル名の長いリストを指定すると、この最大値を超える可能性があります。

Amazon S3 はフラットオブジェクトストレージシステムであるため、ディレクトリをサポートしていません。各ファイルに同じオブジェクトキープレフィックスを付けることで、ファイルを「ディレクトリ」にグループ化します。Amazon S3 オブジェクトキープレフィックスの詳細については、「プレフィックスを使用したオブジェクトの整理」を参照してください。

HealthOmics は、入力パラメータ値を次のように解釈します。

- Amazon S3 の場所がスラッシュで終わらない、または glob パターンを使用しない場合、HealthOmics はパラメータ値が 1 つの Amazon S3 オブジェクトのキーであると想定します。

  たとえば、file1.fastq を入力する`s3://myfiles/runs/inputs/a/file1.fastq`ように を指定します。
- Amazon S3 の場所がスラッシュで終わる場合、HealthOmics はパラメータ値を Amazon S3 プレフィックスとして解釈します。すべての Amazon S3 オブジェクトにそのプレフィックスをロードします。

  たとえば、キーがこのプレフィックスで始まるすべてのオブジェクトをロード`s3://myfiles/runs/inputs/a/`するように を指定できます。
- Nextflow の場合、HealthOmics は入力パラメータで Amazon S3 URIs glob パターンをサポートします。

  たとえば、キーがこのプレフィックスで始まるすべての .gz ファイルを入力する`“s3://myfiles/runs/inputs/a/*.gz”`ように を指定できます。

Amazon S3 入力でのダブルスラッシュの言語固有の処理

HealthOmics は、Amazon S3 URIs でダブルスラッシュを処理するときに各ワークフローエンジンのネイティブエンジン動作を保持するため、HealthOmics に移行するときにワークフローを変更する必要はありません。以下のセクションでは、各エンジンがさまざまなシナリオを処理する方法について説明します。

WDL

入力パラメータに URI の中央または末尾にダブルスラッシュが含まれている場合、WDL エンジンはダブルスラッシュを保持します。

入力パラメータ	予想される場所	
s3://myfiles/runs/inputs//file1.fastq	s3://myfiles/runs/inputs//file1.fastq	
s3://myfiles/runs/inputs//	s3://myfiles/runs/inputs//	

ネクストフロー

入力パラメータに URI の中間にダブルスラッシュが含まれている場合、Nextflow エンジンはダブルスラッシュを保持します。URI の末尾に二重スラッシュがある場合、Nextflow エンジンはそれを 1 つのスラッシュに解決します。

入力パラメータ	予想される場所
s3://myfiles/runs/inputs//file1.fastq	s3://myfiles/runs/inputs//file1.fastq
s3://myfiles//runs/inputs//*.gz	s3://myfiles//runs/inputs//*.gz
s3://myfiles//runs/inputs//	s3://myfiles//runs/inputs/

CWL

入力パラメータに URI の中央または末尾にダブルスラッシュが含まれている場合、CWL エンジンはダブルスラッシュを保持します。

入力パラメータ	予想される場所
s3://myfiles//runs/inputs//file1.fastq	s3://myfiles//runs/inputs//file1.fastq
s3://myfiles//runs/inputs//	s3://myfiles//runs/inputs//

Amazon S3 入力アーカイブの状態

HealthOmics は、Amazon S3 S3 オブジェクトを取得できます。次のアーカイブされたストレージ状態にあるオブジェクトの場合、HealthOmics restore で使用できるようにするオブジェクト。

- Amazon S3 Glacier の Flexible Retrieval または Deep Archive ストレージクラス。
- インテリジェント階層化のアーカイブされたアクセス階層またはディープアーカイブアクセス階層。

オブジェクトの復元の詳細については、Amazon S3ユーザーガイド」の「アーカイブされたオブジェクトの復元」を参照してください。

HealthOmics での実行の開始

実行を開始すると、実行ストレージタイプとストレージ量 (静的ストレージの場合) を設定できます。詳細については、「HealthOmics ワークフローでストレージタイプを実行する」を参照してください。

実行優先度も設定します。優先度が実行に与える影響は、実行が実行グループに関連付けられているかどうかによって異なります。詳細については、「実行優先度」を参照してください。

1 つ以上のワークフローバージョンを作成している場合は、実行の開始時にバージョンを指定できます。バージョンを指定しない場合、HealthOmics はデフォルトのワークフローバージョンを開始します。

出力ファイルの Amazon S3 の場所を指定します。大量のワークフローを同時に実行する場合は、バケットのスロットリングを避けるために、ワークフローごとに個別の Amazon S3 出力 URIs を使用します。詳細については、「Amazon S3 ユーザーガイド」の「プレフィックスを使用してオブジェクトを整理する」および「Amazon S3 パフォーマンスの最適化」ホワイトペーパーの「ストレージ接続を水平方向にスケールする」を参照してください。 Amazon S3 Amazon S3

> **Note**
>
> 実行を開始するときに IAM サービスロールを指定します。必要に応じて、 コンソールでサービスロールを作成できます。詳細については、「のサービスロール AWS HealthOmics」を参照してください。

トピック

- HealthOmics 実行パラメータ
- コンソールを使用した実行の開始
- API を使用して実行を開始する
- ワークフロー実行に関する情報を取得する
- ワークフロー実行の再実行

HealthOmics 実行パラメータ

実行を開始するときは、実行パラメータ JSON ファイルで実行入力を指定するか、パラメータ値をインラインで入力できます。実行パラメータ JSON ファイルのサイズを管理する方法については、「」を参照してください実行パラメータサイズの管理。

HealthOmics は、パラメータ値に対して次の JSON タイプをサポートしています。

JSON タイプ	キーと値の例	メモ
boolean	"b":true	値は引用符ではなく、すべて小文字です。
integer	「i」: 7	値は引用符で囲まれていません。
数値	「f」:42.3	値は引用符で囲まれていません。
文字列	「s"characters」	値は引用符で囲まれています。テキスト値と URIs。URI ターゲットは、予想される入力タイプである必要があります。
配列	「a」:[1,2,3]	値は引用符で囲まれていません。配列メンバーには、それぞれ入力パラメータで定義された型が必要です。
オブジェクト	"o":{"left"a"、"right":1}	WDL では、オブジェクトは WDL ペア、マップ、または構造体にマッピングされます。

コンソールを使用した実行の開始

ワークフロー実行を開始するには

1. HealthOmics コンソールを開きます。

2. 左側のナビゲーションペインで、実行を選択します。
3. Runs ページで、Start run を選択します。
4. 実行の詳細パネルで、次の情報を入力します。
   - ワークフローソース - 所有ワークフローまたは共有ワークフローを選択します。
   - ワークフロー ID - この実行に関連付けられたワークフロー ID。
   - ワークフローバージョン (オプション) - この実行に使用するワークフローバージョンを選択します。バージョンを選択しない場合、実行はワークフローのデフォルトバージョンを使用します。
   - 実行名 - この実行の固有の名前。
   - 実行優先度 (オプション) - この実行の優先度。数値が大きいほど優先度が高くなり、最も優先度の高いタスクが最初に実行されます。
   - Run storage type - ワークフローに指定されたデフォルトの実行ストレージタイプを上書きするには、ここでストレージタイプを指定します。静的ストレージは、実行に一定量のストレージを割り当てます。動的ストレージは、実行中の各タスクで必要に応じてスケールアップおよびスケールダウンします。
   - ストレージ容量の実行 - 静的実行ストレージの場合は、実行に必要なストレージの量を指定します。このエントリは、ワークフローに指定されたデフォルトの実行ストレージ量を上書きします。
   - Select S3 output destination - 実行出力が保存される S3 の場所。
   - 出力バケット所有者のアカウント ID (オプション) - アカウントが出力バケットを所有していない場合は、バケット所有者の AWS アカウント ID を入力します。この情報は、HealthOmics がバケットの所有権を検証できるようにするために必要です。
   - メタデータ保持モードの実行 - アカウントが最大実行数に達したときに、すべての実行のメタデータを保持するか、最も古い実行メタデータを削除するかを選択します。詳細については、「HealthOmics 実行の保持モードを実行する」を参照してください。
5. サービスロールでは、既存のサービスロールを使用するか、新しいサービスロールを作成できます。
6. (オプション) タグの場合、実行に最大 50 個のタグを割り当てることができます。
7. [次へ] を選択します。
8. パラメータ値の追加ページで、実行パラメータを指定します。パラメータを指定する JSON ファイルをアップロードするか、値を手動で入力できます。
9. [次へ] を選択します。

10. グループの実行パネルでは、オプションでこの実行の実行グループを指定できます。詳細については、「HealthOmics 実行グループの作成」を参照してください。
11. キャッシュの実行パネルでは、オプションでこの実行の実行キャッシュを指定できます。詳細については、「コンソールを使用した実行キャッシュを使用した実行の設定」を参照してください。
12. [Review and start run] を選択します。
13. 実行設定を確認したら、実行の開始を選択します。

API を使用して実行を開始する

作成した IAM ロールと Amazon S3 バケットで Start-run API オペレーションを使用します。この例では、保持モードを に設定しますREMOVE。保持モードの詳細については、「」を参照してくださいHealthOmics 実行の保持モードを実行する。

```
aws omics start-run
     --workflow-id workflow id \
     --role-arn arn:aws:iam::1234567892012:role/service-role/
OmicsWorkflow-20221004T164236 \
     --name workflow name \
     --retention-mode REMOVE
```

応答として、次の出力を取得します。uuid は実行に固有であり、 とともに出力データが書き込まれる場所を追跡するためにoutputUri使用できます。

```
{
    "arn": "arn:aws:omics:us-west-2:....:run/1234567",
    "id": "123456789",
    "uuid":"96c57683-74bf-9d6d-ae7e-f09b097db14a",
    "outputUri":"s3://bucket/folder/8405154/96c57683-74bf-9d6d-ae7e-f09b097db14a"
    "status": "PENDING"
}
```

ワークフローのパラメータテンプレートで必須パラメータが宣言されている場合は、ワークフロー実行の開始時に入力のローカル JSON ファイルを指定できます。JSON ファイルには、各入力パラメータの正確な名前と パラメータの値が含まれています。

start-run リクエスト--parameters file://<input_file.json>に を追加して AWS CLI 、の入力 JSON ファイルを参照します。実行パラメータの詳細については、「」を参照してくださいHealthOmics 実行入力。

実行のワークフローバージョンを指定できます。

```
aws omics start-run
     --workflow-id workflow id \
      ...
     --workflow-version-name '1.2.1'
```

ワークフローで指定されているデフォルトの実行ストレージタイプを上書きできます。

```
aws omics start-run
     --workflow-id workflow id \
      ...
     --storage-type STATIC
     --storage-capacity 2400
```

次に示すように、GPU ワークフロー ID で Start-run API を使用することもできます。

```
aws omics start-run
     --workflow-id workflow id \
     --role-arn arn:aws:iam::1234567892012:role/service-role/
OmicsWorkflow-20221004T164236 \
     --name GPUTestRunModel \
     --output-uri s3://amzn-s3-demo-bucket1
```

ワークフロー実行に関する情報を取得する

次に示すように、get-run API でレスポンスの ID を使用して、実行のステータスを確認できます。

```
aws omics get-run --id run id
```

この API オペレーションからのレスポンスは、ワークフロー実行のステータスを示します。可能なステータスは、PENDING、STARTING、RUNNING、および ですCOMPLETED。実行が の場合COMPLETED、出力 Amazon S3 バケットoutfile.txt内の という名前の出力ファイルは、実行 ID の後に という名前のフォルダにあります。

get-run API オペレーションは、ワークフローが Ready2Runか か、ワークフローエンジンPRIVATE、アクセラレーターの詳細など、他の詳細も返します。次の例は、プライベートワークフローの実行に対する get-run のレスポンスを示しています。「WDL with a GPU accelerator and no tags assigned to the run」で説明されています。

```
{
```

```
    "arn": "arn:aws:omics:us-west-2:123456789012:run/7830534",
    "id": "7830534",
    "uuid":"96c57683-74bf-9d6d-ae7e-f09b097db14a",
    "outputUri":"s3://bucket/folder/8405154/96c57683-74bf-9d6d-ae7e-f09b097db14a"
    "status": "COMPLETED",
    "workflowId": "4074992",
    "workflowType": "PRIVATE",
    "workflowVersionName": "3.0.0",
    "roleArn": "arn:aws:iam::123456789012:role/service-role/
OmicsWorkflow-20221004T164236",
    "name": "RunGroupMaxGpuTest",
    "runGroupId": "9938959",
    "digest":
 "sha256:a23a6fc54040d36784206234c02147302ab8658bed89860a86976048f6cad5ac",
    "accelerators": "GPU",
    "outputUri": "s3://amzn-s3-demo-bucket1",
    "startedBy": "arn:aws:sts::123456789012:assumed-role/Admin/<role_name>",
    "creationTime": "2023-04-07T16:44:22.262471+00:00",
    "startTime": "2023-04-07T16:56:12.504000+00:00",
    "stopTime": "2023-04-07T17:22:29.908813+00:00",
    "tags": {}
}
```

次に示すように、list-runs API オペレーションを使用して、すべての実行のステータスを確認できます。

```
aws omics list-runs
```

特定の実行で完了したすべてのタスクを表示するには、list-run-tasks API を使用します。

```
aws omics list-run-tasks --id task ID
```

特定のタスクの詳細を取得するには、get-run-task API を使用します。

```
aws omics get-run-task --id <run_id> --task-id task ID
```

実行が完了すると、メタデータはストリーム の CloudWatch に送信されます**manifest/run/<run ID>/<run UUID>**。

マニフェストの例を次に示します。

```
{
```

```
    "arn": "arn:aws:omics:us-east-1:123456789012:run/1695324",
    "creationTime": "2022-08-24T19:53:55.284Z",
    "resourceDigests": {
      "s3://omics-data/broad-references/hg38/v0/Homo_sapiens_assembly38.dict":
 "etag:3884c62eb0e53fa92459ed9bff133ae6",
      "s3://omics-data/broad-references/hg38/v0/Homo_sapiens_assembly38.fasta":
 "etag:e307d81c605fb91b7720a08f00276842-388",
      "s3://omics-data/broad-references/hg38/v0/Homo_sapiens_assembly38.fasta.fai":
 "etag:f76371b113734a56cde236bc0372de0a",
      "s3://omics-data/intervals/hg38-mjs-whole-chr.500M.intervals":
 "etag:27fdd1341246896721ec49a46a575334",
      "s3://omics-data/workflow-input-lists/dragen-gvcf-list.txt":
 "etag:e22f5aeed0b350a66696d8ffae453227"
    },
    "digest":
 "sha256:a5baaff84dd54085eb03f78766b0a367e93439486bc3f67de42bb38b93304964",
    "engine": "WDL",
    "main": "gatk4-basic-joint-genotyping-v2.wdl",
    "name": "1044-gvcfs",
    "outputUri": "s3://omics-data/workflow-output",
    "parameters": {
      "callset_name": "cohort",
      "input_gvcf_uris": "s3://omics-data/workflow-input-lists/dragen-gvcf-list.txt",
      "interval_list": "s3://omics-data/intervals/hg38-mjs-whole-chr.500M.intervals",
      "ref_dict": "s3://omics-data/broad-references/hg38/v0/
Homo_sapiens_assembly38.dict",
      "ref_fasta": "s3://omics-data/broad-references/hg38/v0/
Homo_sapiens_assembly38.fasta",
      "ref_fasta_index": "s3://omics-data/broad-references/hg38/v0/
Homo_sapiens_assembly38.fasta.fai"
    },
    "roleArn": "arn:aws:iam::123456789012:role/OmicsServiceRole",
    "startedBy": "arn:aws:sts::123456789012:assumed-role/admin/ahenroid-Isengard",
    "startTime": "2022-08-24T20:08:22.582Z",
    "status": "COMPLETED",
    "stopTime": "2022-08-24T20:08:22.582Z",
    "storageCapacity": 9600,
    "uuid": "a3b0ca7e-9597-4ecc-94a4-6ed45481aeab",
    "workflow": "arn:aws:omics:us-east-1:123456789012:workflow/1558364",
    "workflowType": "PRIVATE"
  },
  {
    "arn": "arn:aws:omics:us-east-1:123456789012:task/1245938",
    "cpus": 16,
```

```
    "creationTime": "2022-08-24T20:06:32.971290",
    "image": "123456789012.dkr.ecr.us-west-2.amazonaws.com/gatk",
    "imageDigest":
 "sha256:8051adab0ff725e7e9c2af5997680346f3c3799b2df3785dd51d4abdd3da747b",
    "memory": 32,
    "name": "geno-123",
    "run": "arn:aws:omics:us-east-1:123456789012:run/1695324",
    "startTime": "2022-08-24T20:08:22.278Z",
    "status": "SUCCESS",
    "stopTime": "2022-08-24T20:08:22.278Z",
    "uuid": "44c1a30a-4eee-426d-88ea-1af403858f76"
  },
  ...
```

CloudWatch ログにない場合は、実行メタデータは削除されません。実行 ID を使用して、 CLI ツールを使用してワークフロー実行を再実行することもできます。詳細を確認し、HealthOmics Tool GitHub リポジトリからツールをダウンロードします。

ワークフロー実行の再実行

次の例は、 rerunツールを使用して実行を再実行する方法を示しています。CloudWatch ログから取得できる実行 ID が必要です。

```
omics-rerun 9876543 --name workflow name --retention-mode REMOVE
```

CloudWatch に実行が存在する場合、次のようなレスポンスが表示されます。

```
Original request:
{
  "workflowId": "9679729",
  "roleArn": "arn:aws:iam::123456789012:role/DemoRole",
  "name": "sample_rerun",
  "parameters": {
    "image": "123456789012.dkr.ecr.us-west-2.amazonaws.com/default:latest",
    "file1": "omics://123456789012.storage.us-west-2.amazonaws.com/8647780323/
readSet/6389608538"
  },
  "outputUri": "s3://workflow-output-bcf2fcb1"
}
StartRun request:
{
  "workflowId": "9679729",
```

```
  "roleArn": "arn:aws:iam::123456789012:role/DemoRole",
  "name": "new test",
  "parameters": {
    "image": "123456789012.dkr.ecr.us-west-2.amazonaws.com/default:latest",
    "file1": "omics://123456789012.storage.us-west-2.amazonaws.com/8647780323/
readSet/6389608538"
  },
  "outputUri": "s3://workflow-output-bcf2fcb1"
}
StartRun response:
{
  "arn": "arn:aws:omics:us-west-2:123456789012:run/9171779",
  "id": "9171779",
  "status": "PENDING",
  "tags": {}
}
```

ワークフローが存在しない場合は、エラーメッセージが表示されます。

HealthOmics ワークフローでライフサイクルを実行する

実行のステータスをモニタリングすることで、実行の進行状況を追跡できます。HealthOmics は、実行がライフサイクルを進むにつれて実行ステータスを更新します。

実行ステータスは、次のいずれかの方法を使用して取得できます。

- HealthOmics コンソールには、各実行のステータスがRunsページに表示されます。
- GetRun API オペレーションは、現在の実行ステータスを返します。
- EventBridge イベントを使用して実行ステータスをモニタリングできます。詳細については、「での EventBridge の使用 AWS HealthOmics」を参照してください。

トピック

- 実行ステータス値
- タスクの再試行
- 実行ステータスの料金への影響

実行ステータス値

実行を開始すると、HealthOmics は実行ステータスを に設定しますPending。実行がライフサイクルを進むにつれて、HealthOmics はステータス値を更新して現在の進行状況を反映します。

> **Note**
>
> Running 以外の実行ステータスでは料金は発生しません。詳細については、次のセクションを参照ください。

HealthOmics は、次の実行ステータス値をサポートしています。

保留中

実行はキューにあり、開始を待っています。通常、実行は開始するまで短期間保留中のままになります。

- 多数のジョブを同時に送信すると、実行は保留状態になることがあります。
- 実行は、アカウントが同時実行の最大数に達した後も保留中のままになります。
- 実行がリソースの最大値のいずれかに達した実行グループの一部である場合、実行は保留中のままになります。
- 特定のキューに入れられた実行が他の実行よりも先に開始されるように、実行の優先順位を調整できます。実行優先度の詳細については、「」を参照してください実行優先度。

スタート

HealthOmics は実行を作成し、実行に必要なリソース (一時実行ストレージやエンジンノードなど) をプロビジョニングします。

- HealthOmics は、実行の開始時に一時実行ストレージをプロビジョニングし、実行が停止しているときに実行ストレージのプロビジョニングを解除します。

実行中

インポートプロセス、各タスクの処理、およびエクスポートプロセス中、実行は実行中ステータスのままになります。

- HealthOmics は、入力ファイルを一時実行ストレージファイルシステムにインポートします。入力ファイルは読み取り専用であり、タスクがワークフロー内の他のタスクへの入力を変更できないようにします。

- ファイルのエクスポート中、HealthOmics は実行ストレージファイルシステムから S3 の場所に出力ファイルをエクスポートします。
- HealthOmics は、実行ステータスが実行中の間、実行ログとタスクログを CloudWatch にリアルタイムで配信します。詳細については、「CloudWatch のログ」を参照してください。

停止中

エクスポートプロセスが完了すると、実行は停止ステータスに移行します。

- HealthOmics は、すべてのリソース (実行ストレージファイルシステムとエンジンノードを含む) のプロビジョニングを解除します。

完了

HealthOmics がリソースのプロビジョニング解除を完了すると、実行は完了に移行します。

- HealthOmics はすべての実行タスクを完了し、エラーなしで出力データをエクスポートしました。
- 実行出力は、指定された Amazon S3 URI 出力場所で使用できます。WDL および CWL の場合、HealthOmics は に関する情報を提供する実行出力概要ファイルを生成しますHealthOmics 実行出力。
- 最終実行マニフェストログとエンジンログ (該当する場合) は、CloudWatch で入手できます。
- タスクの再試行をサポートする実行の場合、完了ステータスの実行には、失敗した 1 つ以上のタスクを含めることができます。失敗したタスクごとにタスクの再試行が成功する限り、HealthOmics は実行を完了に移行します。HealthOmics は各再試行に新しいタスク ID を割り当てるため、実行には失敗した試行と完了した試行のタスク IDs が含まれます。

失敗

HealthOmics で 1 つ以上のエラーが発生し、すべての実行タスクを完了できませんでした。

- HealthOmics がリソースのプロビジョニングを解除している間、失敗した実行は停止ステータスに移行します。

Cancelled (キャンセル)

ユーザーが実行をキャンセルするリクエストを開始しました。

- HealthOmics は実行中のタスクを停止し、すべてのリソースのプロビジョニングを解除します。
- ユーザーが実行をキャンセルしても、HealthOmics は実行出力データをエクスポートしません。キャンセルされた実行の中間ファイルにアクセスすることはできません。

- アカウントでは、キャンセル前に実行中のステータス中に実行が消費したタスクとリソースに対して料金が発生します。
- 保留中または開始中のステータスで実行をキャンセルした場合、料金は発生しません。

タスクの再試行

実行中にタスクが失敗した場合、HealthOmics は次の状況でタスクを再度再試行します。

- WDL ワークフローの場合、HealthOmics は、サービスエラー (5XX HTTP ステータスコード) が原因でタスクが失敗したときにタスクの再試行をサポートします。

  デフォルトでは、HealthOmics は失敗したタスクを最大 2 回再試行します。WDL 定義ファイルを設定することで、タスクの再試行をオプトアウトできます。設定の例については、「HealthOmics ワークフロー定義のタスクリソース」を参照してください。
- Nextflow ワークフローでは、ワークフロー定義でタスクの再試行条件を設定できます。
- 実行内のすべてのタスクが最終的に完了した場合、再試行が必要であっても、HealthOmics は実行を完了に移行します。
- HealthOmics は各再試行に新しいタスク ID を割り当てるため、実行には失敗した試行と完了した試行のタスク IDs が含まれます。

実行ステータスの料金への影響

実行ステータスが実行中の場合、アカウントで料金が発生することがあります。他の実行ステータスでは料金は発生しません。たとえば、実行が開始または停止中の場合、リソースには料金はかかりません。

Running ステータスの実行には、次の請求への影響があります。

- 実行ステータスが実行中である間は、アカウントで実行ストレージファイルシステムの使用量に対して料金が発生します。実行ストレージタイプの詳細については、「」を参照してくださいHealthOmics ワークフローでストレージタイプを実行する。
- アカウントでは、ワークフロー定義の各タスクに指定したコンピューティングリソースとメモリリソース、およびタスク期間に基づいて、実行中のタスクに対して料金が発生します。詳細については、「HealthOmics タスクのコンピューティングとメモリの要件」を参照してください。
- 各タスクの最小請求しきい値は 1 分です。タスクを 1 分未満実行した場合、最低 1 分間の使用に対して料金が発生します。可能であれば、小さなタスクをグループ化してコストを最適化します。

タスクをグループ化すると、複数のシーケンシャルタスクのスピンアップを回避できるため、実行時間も短縮されます。

HealthOmics の料金の詳細については、HealthOmics の料金」を参照してください。

HealthOmics 実行出力

WDL または CWL の実行が完了すると、出力には、実行によって生成されたすべての出力を一覧表示する出力概要ファイル (JSON 形式) が含まれます。出力概要ファイルは、次の目的で使用できます。

- 実行によって生成された出力ファイルをプログラムで決定します。
- 実行が予想されるすべての出力を生成したことを確認します。

トピック

- WDL の出力概要を実行する
- CWL の出力概要を実行する

WDL の出力概要を実行する

WDL の実行が完了すると、HealthOmics は という名前の出力概要ファイルを作成しますoutput.json。

ワークフローの出力ごとに、 ファイルに対応するキーと値のペアがあります。キーには、ワークフロー名と出力名が の形式で含まれますWorkflowName.output_name。ファイル出力の場合、値はファイルが保存されている S3 の出力場所を指す S3 URI です。Array[File] 出力の場合、値は S3 URIs。

次の例は、 という名前のワークフローの output.json ファイルを示していますBWAMappingWorkflow。

```
{
  "BWAMappingWorkflow.bam_indexes": [
    "s3://omics-outputs/8886192/out/bam_indexes/0/
pbmc8k_S1_L007_R1_001.sorted.bam.bai",
    "s3://omics-outputs/8886192/out/bam_indexes/1/pbmc8k_S1_L008_R1_001.sorted.bam.bai"
  ],
```

```
  "BWAMappingWorkflow.mapping_stats": "s3://omics-outputs/8886192/out/mapping_stats/
genome_mapping_final_stats.txt",
  "BWAMappingWorkflow.merged_bam": "s3://omics-outputs/8886192/out/merged_bam/
genome_mapping.merged.bam",
  "BWAMappingWorkflow.merged_bam_index": "s3://omics-outputs/8886192/out/
merged_bam_index/genome_mapping.merged.bam.bai",
  "BWAMappingWorkflow.reference_index_tar": "s3://omics-outputs/8886192/out/
reference_index_tar/reference_index.tar",
  "BWAMappingWorkflow.sorted_bams": [
    "s3://omics-outputs/8886192/out/sorted_bams/0/pbmc8k_S1_L007_R1_001.sorted.bam",
    "s3://omics-outputs/8886192/out/sorted_bams/1/pbmc8k_S1_L008_R1_001.sorted.bam"
  ],
  "BWAMappingWorkflow.unmapped_bams": [
    "s3://omics-outputs/8886192/out/unmapped_bams/0/
pbmc8k_S1_L007_R1_001.unmapped.bam",
    "s3://omics-outputs/8886192/out/unmapped_bams/1/pbmc8k_S1_L008_R1_001.unmapped.bam"
  ]
}
```

ワークフローがファイル以外のタイプ (文字列、Int、Float、Bool など) の出力を生成する場合、フィールド値は JSON プリミティブです。例:

```
{
  "MyWorkflow.my_int_ouput": 1,
  "MyWorkflow.my_bool_output": false,
    ...
}
```

CWL の出力概要を実行する

CWL 実行が完了すると、HealthOmics は次のoutputs.json場所に という名前の出力概要ファイルを作成します。

```
{my-S3outputpath}/{runId}/{run-uuid}/logs/outputs.json
```

出力概要ファイルには出力のリストが含まれています。各出力はキーと値のペアで、キーは出力の名前です。値は、次のプロパティを含むオブジェクトです。

- location – 出力ファイルへの完全修飾パス
- basename – パスのファイル名部分
- クラス – 出力のタイプ。通常は File です。

- size – バイト単位のファイルのサイズ

次の例では、output.json ファイルに 2 つの出力ファイルのリストがあります。

```
{
  "example_output": {
    "location": "{my-S3outputpath}/{runId}/{run-uuid}/out/output.txt",
    "basename": "output.txt",
    "class": "File",
    "size": 13
  },
  "another_output": {
    "location": "{my-S3outputpath}/{runId}/{run-uuid}/out/metrics.json",
    "basename": "metrics.json",
    "class": "File",
    "size": 256
  }
}
```

実行失敗の理由

実行が失敗した場合、GetRun API オペレーションを使用して失敗の理由を取得します。

失敗の理由を確認して、実行が失敗した理由のトラブルシューティングに役立ててください。次の表に、各失敗の理由とエラーの説明を示します。

失敗の理由	エラーの説明
ASSUME_ROLE_FAILED	HealthOmics には、ロールを引き受けるアクセス許可がありません。ロールの信頼関係で HealthOmics プリンシパルを指定します。
CANNOT_START_CONTAINER_ERROR	イメージ: *#####*を使用してワークフロータスク: *name*、id: *ID* コンテナを開始できません。イメージが有効であることを確認し、もう一度試してください。
CANNOT_START_CONTAINER_SIZE_ERROR	イメージ: *#####*を使用してワークフロータスク: *name*、id: *ID* コンテナを開始できません。イメージサイズが 25 GB 未満であることを確認し、もう一度試してください。

失敗の理由	エラーの説明
ECR_PERMISSION_ERROR	HealthOmics には、イメージ URI にアクセスするアクセス許可がありません。 Amazon ECR プライベートリポジトリが存在し、HealthOmics サービスプリンシパルへのアクセス権が付与されていることを確認します。
EXPORT_FAILED	エクスポートに失敗しました。出力バケットが存在し、実行ロールにバケットへの書き込みアクセス許可があることを確認します。
FILE_SYSTEM_OUT_OF_SPACE	ファイルシステムに十分なスペースがありません。ファイルシステムのサイズを増やし、もう一度実行します。
IMAGE_VERIFICATION_FAILURE	イメージ*#####*を確認できません。この問題を解決するには、イメージをプルして ECR リポジトリにもう一度プッシュしてみてください。
インポート失敗	インポートに失敗しました。入力ファイルが存在し、実行ロールが入力にアクセスできることを確認します。
INACTIVE_OMICS_STORAGE_RESOURCE	HealthOmics ストレージ URI が ACTIVE 状態ではありません。読み取りセットをアクティブ化して、もう一度試してください。読み取りセットのアクティブ化の詳細については、「」を参照してくださいHealthOmics での読み取りセットのアクティブ化。
INPUT_URI_NOT_FOUND	指定された URI は存在しません: *uri*。URI パスが存在することを確認し、ロールがオブジェクトにアクセスできることを確認します。
INSTANCE_RESERVATION_FAILED	ワークフロー実行を完了するのに十分なインスタンス容量がありません。ワークフローの実行を再試行してください。
INVALID_ECR_IMAGE_URI	Amazon ECR イメージ URI 構造が無効です。有効な URI を指定して、もう一度試してください。

失敗の理由	エラーの説明
INVALID_TASK_RESOURCE_VALUE	リクエストされた GPU、CPU、またはメモリが、使用可能なコンピューティング容量に対して高すぎるか、タスク *ID* の最小値 1 未満です。
INVALID_URI_INPUT	URI 構造が有効な *URI* ではありません。URI 構造を確認して、もう一度試してください。
MODIFIED_INPUT_RESOURCE	指定された URI *URI* は、実行の開始後に変更されました。実行を再試行します。
OUT_OF_MEMORY_ERROR	ワークフロータスク *ID* のメモリが不足しました。ワークフロー定義のメモリ値を増やし、実行を再試行してください。
RUN_TASK_FAILED	タスクが失敗したため、実行に失敗しました。タスクの失敗をデバッグするには、GetRunTask API オペレーションと Amazon CloudWatch Logs ストリームを使用します。
RUN_TIMED_OUT	*##*後にタイムアウトを実行します。
SERVICE_ERROR	サービスに一時的なエラーが発生しました。ワークフローの実行を再試行してください。
UNSUPPORTED_INPUT_SIZE	合計入力サイズが大きすぎます。入力サイズを減らして、もう一度試してください。
WORKFLOW_RUN_FAILED	ワークフローの実行に失敗しました。CloudWatch Logs エンジンログストリーム *ID* を確認して、障害をデバッグします。
WORKFLOW_VER_VALIDATION_FAILED	HealthOmics は、リクエストされた Nextflow バージョン: *### ###* -- をサポートしていません。サポートされている最新バージョンは *#####*です。Nextflow バージョンをサポートされているバージョンに変更し、もう一度試してください。
サポートされていない_GPU_INSTANCE_TYPE	リクエストされたインスタンスタイプは、 *#####*ではサポートされていません。このリージョンでサポートされている GPU インスタンスタイプで実行を再試行します。使用可能なインスタンスタイプは *GPU #########* です。

応答しない実行のガイダンス

新しいワークフローを開発する場合、コードに問題があり、タスクが適切にプロセスを終了できないと、実行または特定のタスクが「停止」または「ハング」する可能性があります。これは、タスクを長期間実行するのが通常であるため、トラブルシューティングとキャッチが難しい場合があります。応答しない実行を防止して識別するには、以下のセクションで推奨されるベストプラクティスに従ってください。

応答しない実行を防ぐためのベストプラクティス

- タスクコードで開いているすべてのファイルを閉じていることを確認します。開くファイルが多すぎると、ワークフローエンジン内でスレッドの問題が発生することがあります。
- ワークフロータスクによって作成されたバックグラウンドプロセスは、タスクが終了したときに終了する必要があります。ただし、バックグラウンドプロセスが正常に終了しない場合は、タスクコードでそのプロセスを明示的にシャットダウンする必要があります。
- プロセスが終了せずにループしないようにします。これにより、実行が応答しなくなる可能性があり、解決するにはワークフロー定義コードの変更が必要です。
- タスクに適切なメモリと CPU 割り当てを提供します。CloudWatch ログを分析するか、ワークフローが正常に完了した実行アナライザーの実行で を使用して、最適なコンピューティング割り当てがあることを確認します。Run Analyzer `headroom`パラメータを使用してヘッドルームを追加し、プロセスを完了するための十分なリソースを確保します。バックグラウンドオペレーティングシステムのプロセスを考慮して、割り当てられたメモリと CPU に少なくとも 5% のヘッドルームを含めます。
  - さらに、インスタンスのスループットを向上させる必要がある場合は、インスタンスの帯域幅サイズを増やします。vCPUs (サイズ 4xl 以下) の Amazon EC2 インスタンスでは、スループットがバーストする可能性があります。Amazon EC2 インスタンスのスループットの詳細については、Amazon EC2 の使用可能なインスタンス帯域幅」を参照してください。
- 実行に正しいファイルシステムサイズを使用していることを確認します。静的実行ストレージを使用している応答しない実行の場合は、静的実行ストレージの割り当てを増やして、ファイルシステムの IO スループットとストレージ容量を増やすことを検討してください。実行マニフェストを分析して最大ファイルシステムストレージを確認し、Run Analyzer を使用してファイルシステムの割り当てを増やす必要があるかどうかを確認します。

応答しない実行をキャッチするためのベストプラクティス

- 新しいワークフローを開発するときは、最大実行時間制限が設定されている実行グループを使用して、ランナウェイコードをキャッチします。例えば、実行が完了するまでに 1 時間かかる場合

は、2 時間または 3 時間 (またはユースケースに応じて異なる期間) 後にタイムアウトする実行グループに配置して、ランアウェイジョブをキャッチします。また、処理時間の差異を考慮してバッファを適用します。

- 最大ランタイム制限が異なる一連の実行グループを設定します。たとえば、数時間後に実行を終了する実行グループと、数日後に実行を終了する実行グループを、予想されるワークフロー期間に基づいて割り当てることができます。
- HealthOmics の最大実行時間サービス制限は 604,800 秒、つまり 7 日で、クォータツールのリクエストで調整できます。このクォータのサービス制限の引き上げをリクエストするのは、1 週間に近づく実行がある場合のみです。実行時間が短い実行と長い実行が混在していて、実行グループを使用していない場合は、実行時間が長い実行を、最大実行時間サービス制限の高い別のアカウントに配置することを検討してください。
- CloudWatch ログで、応答しないと思われるタスクがないか調べます。通常、タスクが通常のログステートメントを出力し、長期間出力していない場合、タスクは停止またはフリーズする可能性があります。

応答しない実行が発生した場合の対処方法

- 追加コストが発生しないように実行をキャンセルします。
- タスクログを調べて、プロセスが正しく終了しなかったかどうかを確認します。
- エンジンログを検査して、異常なエンジン動作を特定します。
- 応答しない実行のタスクログとエンジンログを、正常に完了した同じ実行のログと比較します。これにより、応答しない動作の原因となった可能性のある違いを特定できます。
- 根本原因を特定できない場合は、サポートケースを作成し、以下を含めます。
  - スタックした実行の ARN と、正常に完了した同一の実行の ARN。
  - エンジンログ (実行がキャンセルまたは失敗すると使用可能)
  - 応答しないタスクのタスクログ。トラブルシューティングのためにワークフロー内のすべてのタスクにタスクログは必要ありません。

HealthOmics 実行のタスクライフサイクル

タスクは、実行内の 1 つのプロセスです。HealthOmics は、ワークフロー内の各タスクを、タスクに必要なリソースに最適なオミクスコンピューティングインスタンスタイプにマッピングします。ワークフロー定義で必要なリソースを指定します。詳細については、「」を参照してくださいHealthOmics タスクのコンピューティングとメモリの要件。

HealthOmics は、タスクが使用する一時実行ストレージを提供します。HealthOmics は、タスク入力ファイルを読み取り専用ファイルとして一時実行ストレージにコピーします。HealthOmics はシンボリックリンクを提供し、タスクが作業ディレクトリから入力ファイルにアクセスできるようにします。タスクは、ワークフロー定義ファイルで宣言したファイルにのみアクセスできます。

タスクステータス値

タスクのステータスをモニタリングすることで、タスクの進行状況を追跡できます。実行を開始すると、HealthOmics は実行の各タスクPendingのタスクステータスを に設定します。タスクが開始されてライフサイクルが進むと、HealthOmics はステータス値を更新して現在の進行状況を反映します。

タスクのステータスは、次のいずれかの方法を使用して取得できます。

- HealthOmics コンソールには、実行の各タスクのステータスがRun detailsページに表示されます。
- GetRunTask API オペレーションはタスクのステータスを返します。
- EventBridge イベントを使用してタスクステータスをモニタリングできます。詳細については、「での EventBridge の使用 AWS HealthOmics」を参照してください。

GetRunTask API オペレーションを使用して、タスクの現在のステータスを取得できます。HealthOmics コンソールには、実行の各タスクのステータスがRun detailsページに表示されます。

HealthOmics は、次のタスクステータス値をサポートしています。

保留中

タスクはキューにあり、開始を待っています。タスクは、開始するまで短期間保留中のままです。

- アカウントが同時タスクの最大数に達した後も、タスクは保留中のままになります。
- 実行がリソースの最大値のいずれかに達した実行グループの一部である場合、タスクは保留中のままになります。
- 特定のキューに入れられた実行とそのタスクが他のキューに入れられた実行の前に開始されるように、実行の優先順位を調整できます。実行優先度の詳細については、「」を参照してください。 実行優先度

スタート

HealthOmics はタスクを作成し、ワークフロータスクノードなど、タスクに必要なリソースをプロビジョニングしています。

実行中

HealthOmics がタスクを処理している間、タスクのステータスは実行中です。

停止中

タスクの処理と出力データのエクスポートが完了すると、タスクは Stopping に移行します。

- HealthOmics はワークフロータスクノードのプロビジョニングを解除します。

完了

HealthOmics はタスクの処理を完了し、出力データを実行ストレージファイルシステムに転送しました。

失敗

HealthOmics でタスクの処理中にエラーが発生しましたが、完了していません。

- タスクは停止ステータス (HealthOmics はリソースのプロビジョニングを解除) に移行し、失敗ステータスに移行します。
- エラーがサービスエラー (5XX HTTP ステータスコード) であり、ワークフローがこのタスクの再試行をサポートしている場合、HealthOmics はタスクの再処理を試みます。HealthOmics は、再試行に新しいタスク ID を割り当てます。

Cancelled (キャンセル)

HealthOmics は、ユーザーが開始した実行のキャンセルリクエスト後にタスクを停止します。

- タスクは停止ステータス (HealthOmics はリソースのプロビジョニングを解除) に移行し、次にキャンセルステータスに移行します。

ワークフロータスクのトラブルシューティング

以下は、タスクのトラブルシューティングに関するベストプラクティスと考慮事項です。

- タスクログSTDERRは、 STDOUTに依存し、タスクによって生成されます。タスクで使用されるアプリケーションがこれらのいずれも生成しない場合、タスクログは作成されません。デバッグを支援するには、 `verbose` モードでアプリケーションを使用します。
- タスクで実行されているコマンドと補間された値を表示するには、`set -x`Bash コマンドを使用します。これにより、タスクが正しい入力を使用しているかどうかを判断し、エラーによってタスクが意図したとおりに実行されない原因を特定できます。

- echo コマンドを使用して、変数の値を STDOUTまたは に出力しますSTDERR。これにより、期待どおりに設定されていることを確認できます。
- などのコマンドを使用してls -l <name_of_input_file>、入力が存在し、予想されるサイズであることを確認します。そうでない場合、バグによる空の出力を生成する以前のタスクの問題が明らかになる可能性があります。
- タスクスクリプトdf -Ph . | awk 'NR==2 {print $4}'の コマンドを使用して、タスクで現在使用可能な領域を決定し、追加のストレージ割り当てでワークフローを実行する必要がある状況を特定するのに役立ちます。

上記のコマンドのいずれかをタスクスクリプトに含めることは、タスクコンテナにこれらのコマンドも含まれ、コンテナ環境pathの にあることを前提としています。

プライベート HealthOmics ワークフローの最適化を実行する

合計コスト、合計実行時間、またはその両方の組み合わせで実行を最適化できます。HealthOmics は、実行の最適化の決定に役立つデータとツールを提供します。実行の最適化は Ready2Run ワークフローには適用されません。これは、サービスがこれらのワークフローのリソースプロビジョニングを管理する方法を制御できないためです。

最初のステップでは、実行中のタスクの現在のタスクリソースの使用状況とコストを理解し、実行コストとパフォーマンスを最適化する方法を適用します。

トピック

アナライザーの実行

HealthOmics には、Run Analyzer という名前のオープンソースツールが用意されています。このツールは、実行のタスクレベルのリソース使用状況情報を抽出し、コストと実行パフォーマンスの最適化の機会を提案します。

> **Note**
>
> Run Analyzer は、ツールの実行時の AWS 定価に基づいてタスクコストと潜在的なコスト削減を見積もります。最適化の推奨事項を評価し、ユースケースに適した推奨事項を実装します。採用した最適化をテストして、実行で機能することを確認します。

Run Analyzer は次のタスクを実行します。

- メモリとコンピューティングのボトルネックを評価します。
- メモリまたは CPU 用に過剰にプロビジョニングされているタスクを特定し、コストを削減できる新しいインスタンスサイズを推奨します。
- 個々のタスクのコスト見積もりを計算し、レコメンデーションを適用すると潜在的なコスト削減を計算します。
- タスクのタイムラインビューが表示され、タスクの依存関係と処理シーケンスを確認できます。タイムラインは、長時間実行されるタスクを特定するのにも役立ちます。
- 実行ストレージのファイルシステムサイズに関する推奨事項を提供します。
- タスクのプロビジョニング時間を表示して、大きなコンテナ負荷によってプロビジョニング時間が遅くなる可能性のある領域を特定できるようにします。
- ツールには、最適化レコメンデーションの積極性を制御するために使用できる入力パラメータ (ヘッドルーム) が含まれています。

以下のセクションでは、Run Analyzer を使用して実行を最適化するための具体的な提案を示します。

実行コストを決定する

以下の方法とガイドラインを使用して、実行コストを決定できます。

- 請求期間の合計実行コストを表示するには、次の手順に従います。
  1. Billing and Cost Management コンソールを開き、Bills を選択します。
  2. サービス別の料金で、Omics を展開します。
  3. リージョンを展開し、オミクスインスタンスタイプ、実行ストレージタイプ、および Ready2Run ワークフロー別に項目化されたすべての実行のコストを表示します。
- 各実行の情報を含むコストレポートを生成するには、次の手順に従います。

1. Billing and Cost Management コンソールを開き、データエクスポートを選択します。
2. Create を選択して、新しいデータエクスポートを作成します。
3. データエクスポートのエクスポート名を入力します。他のフィールドをデフォルト値のままにして、CUR (コストと使用状況) レポートを作成します。
4. 時間の詳細度には、時間単位または日単位を選択します。
5. データエクスポートストレージ設定で、以下の設定ステップを実行します。

   a. データエクスポート用に Amazon S3 バケットを設定します。

   b. ファイルバージョニングでは、既存のエクスポートファイルを上書きするか、毎回新しいファイルを作成するかを選択します。

   システムは 24 時間以内に最初のレポートを生成し、それ以降のレポートは 1 日に 1 回生成します。
6. データエクスポートの作成方法の詳細については、「データエクスポートAWS ユーザーガイド」の「データエクスポートの作成」を参照してください。

- 実行にタグを付けて、チームやプロジェクトなどのカテゴリ別にコストをモニタリングおよび最適化できます。タグを使用する場合は、以下の手順に従ってタグカテゴリ別に実行コストを表示します。

  1. Billing and Cost Management コンソールを開き、Cost Explorer を選択します。
  2. レポートパラメータ > グループ化基準で、ディメンションとしてタグ を選択し、目的のタグ名を選択します。
- タスクのリソース使用状況を確認するには、CloudWatch で実行マニフェストログを表示します。詳細については、「CloudWatch Logs による HealthOmics のモニタリング」を参照してください。
- ツールアナライザーの実行を使用して、実行のタスクリソース使用状況情報を抽出します。

実行時間の使用状況を確認する

実行時間の使用状況を調査するには、次の方法を使用できます。

- コンソールの Runs ページから、実行の合計実行時間を表示できます。
- 実行の詳細ページから、次の項目を表示できます。
  - 実行の合計実行時間を表示します。

  - 実行内の各タスクの実行時間を表示します。
  - リンクのいずれかを選択して Amazon S3 のログを表示するか、CloudWatch で実行ログまたはマニフェストログを実行します。
- タスクの実行 リストから、タスクのログを表示 リンクを選択して、CloudWatch でタスクログを表示します。
- `listRuns` API オペレーションへのレスポンスには実行開始時刻と停止時刻が含まれているため、合計実行時間を計算できます。
- このアナライザーの実行ツールは、タイムラインビューにタスク期間を表示します。このツールは、タスク処理シーケンスを視覚的に表現し、予想される順序と一致させることができます。

実行を最適化する方法

HealthOmics は、データステージング (データのインポートやデータエクスポートなど) を実行するリソースを自動的にプロビジョニング、管理、最適化します。また、HealthOmics はワークフローのワークフローエンジンを起動して実行します。ただし、さまざまな実行設定を設定することで、実行開始時間、タスク開始時間、全体的なタスク実行時間に影響を与えることができます。ワークフロー定義と設計に対する全体的なアプローチは、タスクの実行時間にも影響します。次のリストは、実行とタスクのパフォーマンスに影響を与える可能性のある要因を示しています。

ストレージタイプを実行する

実行ストレージタイプは、実行パフォーマンスと実行プロビジョニング時間に影響します。動的実行ストレージは、実行ストレージのニーズに応じて動的にスケールするため、プロビジョニングが速くなり、メモリが不足することはありません。動的実行ストレージは、開発中のワークフローにも適しており、問題のトラブルシューティングのためにワークフローを開始および停止することがよくあります。

静的実行ストレージでは、ファイルシステムのプロビジョニング時間が長くなりますが、一部の実行はより速く完了できます。通常、実行のタスク同時実行数が高い場合や、9.6 TiB を超えるファイルシステム容量が必要な場合です。静的実行ストレージは、I/O 要件が高い長時間実行されるワークフローに適しています。

特定の実行の各実行ストレージタイプのコストとパフォーマンスを評価するのに役立つように、A/B テストを試して、どの実行ストレージタイプがパフォーマンスを向上させるかを確認できます。また、開発サイクルに動的実行ストレージを使用することを検討し、本番環境の大規模な実行には静的実行ストレージを使用することを検討してください。

実行ストレージタイプの詳細については、「」を参照してください。HealthOmics ワークフローでストレージタイプを実行する

オーバープロビジョニング実行の静的ストレージ

ワークフロータスクの計算が I/O によって制約されている場合は、静的実行ストレージの過剰プロビジョニングを検討してください。ストレージコストはサイズとともに増加しますが、ファイルシステムの最大スループットも増加します。高価なコンピューティングタスクで I/O ボトルネックが発生している場合は、ファイルシステムサイズを大きくしてタスクの実行時間を短縮することで、全体的なコストを削減できます。

コンテナイメージサイズの縮小

各タスクが開始されると、HealthOmics はタスクに指定したコンテナをロードします。コンテナが大きいほど、ロードに時間がかかります。コンテナをできるだけ小さくして、新しいタスクの起動効率を向上させます。コンテナに大規模なデータセットを追加する場合は、データセットを S3 に保存し、ワークフローで S3 からデータをインポートすることを検討してください。HealthOmics がサポートするコンテナの最大サイズについては、「」を参照してくださいHealthOmics ワークフローの固定サイズクォータ。

タスクサイズ

小さなシーケンシャルタスクを 1 つのタスクに結合して、タスクのプロビジョニング時間を節約できます。また、HealthOmics には 1 分間の最小タスク期間料金がかかるため、タスクを組み合わせるとコストを削減できます。結合タスク内では、Unix パイプを使用してファイルのシリアル化と逆シリアル化の I/O コストを回避できる場合があります。

ファイル圧縮

ワークフロー中間ファイルを過度に圧縮しないでください。ほとんどのゲノミクス形式は、「gzip」または「block gzip」圧縮を使用します。タスク入力ファイルを解凍し、タスク出力ファイルを再圧縮すると、タスク全体の CPU 使用率の大部分を消費する可能性があります。一部のゲノミクスアプリケーションでは、出力をシリアル化するときに圧縮レベルを設定できます。圧縮レベルを減らすことで、CPU 時間を短縮できますが、ファイルが大きいほどディスクへの書き込みに費やす時間が長くなります。タスクとアプリケーションに応じて、実行時間が最短になる中間ファイルに最適な圧縮レベルを見つけることができます。まず、出力ファイルが最も大きいタスクをターゲットにすることをお勧めします。圧縮レベル 2 は、いくつかのシナリオに適しています。ユースケースではこのレベルから始めて、他の圧縮レベルを試して結果を比較できます。

スレッド数

タスク定義でスレッドを指定する場合は、スレッドの数をリクエストvCPUs の数と同じ値に設定します。

コンピューティングとメモリを指定する

タスクでメモリまたはコンピューティングリソースを指定しない場合、HealthOmics は最小のインスタンスタイプ (`omics.c.large`) をデフォルトの として割り当てます。HealthOmics でより大きなインスタンスタイプを割り当てる場合は、メモリとコンピューティングの要件を明示的に宣言します。

HealthOmics は、リクエストした vCPUsメモリ、GPU リソースの数を割り当てます。たとえば、15vCPUs と 33GiB をリクエストすると、HealthOmics はタスクに omics.m.4xl インスタンス (16 vCPUs、64GB) を割り当てますが、タスクで使用できる vCPUsと 33GiB は 15 個のみです。したがって、オミクスインスタンスに一致する vCPUs とメモリリソースをリクエストすることをお勧めします。

複数のサンプルを 1 回の実行にバッチ処理する

ファイルシステムのプロビジョニングには実行開始時に時間がかかるため、複数のサンプルを同じ実行にバッチ処理することで、プロビジョニング時間を短縮できます。このアプローチを決定する前に、次の要因を考慮してください。

- 1 つのサンプルが正しくないとワークフローが失敗する可能性があるため、サンプルをバッチ処理すると失敗したワークフローの数が増える可能性があります。ワークフローがほとんどの場合成功すると確信できない場合は、サンプルごとに 1 回実行することをお勧めします。
- HealthOmics は、ワークフロー全体に 1 つの実行ストレージファイルシステムを割り当てます。サンプルのバッチの場合は、すべてのサンプルを処理するのに十分な量の実行ストレージを指定してください。
- ワークフローあたりの実行ストレージの最大量があるため、 はバッチに追加できるサンプルの数を制限する可能性があります。
- 最小実行ストレージサイズは 1.2 TiB であるため、ワークフローが各サンプルで最小よりもはるかに少ないストレージを使用すると、バッチ処理によってコストを削減できます。
- 実行ストレージは複数の同時接続を処理できるため、同じ実行ストレージを使用する複数のタスクがあっても I/O ボトルネックは発生しません。
- 各実行には独自のタグセットがあります。ワークフローに予算編成や追跡の情報でタグ付けする場合は、個別の実行を使用することをお勧めします。

- IAM ロールは実行全体に適用されます。各ユーザーは、サンプルのバッチのすべてのデータにアクセスできます。ワークフローを分離することで、よりきめ細かなアクセス許可を使用できるようになります。
- HealthOmics は、同時ワークフローの最大数とワークフロー内の同時タスクの最大数のアカウントレベルのクォータを設定します。これらのクォータの引き上げをリクエストする方法については、「」を参照してくださいHealthOmics サービスクォータ。

コンテナイメージのパラメータを使用する

ワークフローに URIs、コンテナイメージをパラメータ化します。これらは実行パラメータであり、HealthOmics は、実行を開始する前に実行がコンテナにアクセスできることを検証します。そうしないと、完了したタスクに対して料金が発生したときに、実行中にタスクが失敗します。また、これらはパラメータ化された入力であるため、HealthOmics は実行マニフェストでチェックサムを生成し、実行の出所を改善します。

linter を使用する

新しいワークフローを実行する前に、linter を使用して一般的なワークフローエラーを見つけます。詳細については、「HealthOmics のワークフローlinters」を参照してください。

EventBridge を使用して問題にフラグを付ける

EventBridge カスタマイズされたアラートを使用して、ビジネスロジックに固有の異常を検出します。

シーケンスストアを使用する

ストレージコストを節約するために、ソースデータにシーケンスストアを使用することを検討してください。詳細については、HealthOmics ブログ記事の「Store omics data cost-effectively at any scale with HealthOmics」を参照してください。

実行間のファイルサイズの差異の影響

多くの場合、ユーザーは少数のテストデータを使用して実行を設計およびテストし、その後、本番環境の実行でファイルサイズに大きなばらつきがあるさまざまなデータに遭遇します。実行を最適化するときは、必ずこの差異を考慮してください。

次のリストは、ファイルサイズに大きな差異がある場合の最適化に関する推奨事項を示しています。

テストデータ内のさまざまなファイルサイズ

開発中は、代表的な分散量を持つテストデータを使用してください。

Run Analyzer を使用する

Run Analyzer ツールは、さまざまなサンプルで使用して、データサイズの差異を考慮します。

実行アナライザーを使用して、本番稼働用データサンプルの実行間の差異を把握できます。Run Analyzer の `--batch` モードを使用して、実行のバッチの統計を生成し、データセットの外れ値を処理するために必要な最大コンピューティングリソースを分析します。

たとえば、実行アナライザーにデータのフルフローセルをバッチモードで提供して、フルフローセルのピーク vCPU とメモリ使用率を把握できます。

入力データセットのサイズ分散を減らす

サンプルサイズに大きなばらつきがある場合は、HealthOmics のアップストリームでサンプルを分岐させ、バッチごとに異なるファイルシステムサイズを選択して、実行ストレージコストを節約できます。

WDL では、 `size`関数を使用して、大きなサンプルと小さなサンプルの個々のタスクのリソース割り当てを分割します。この戦略を最もコストの高いタスクに適用して、最も大きな影響を与えます。

Nextflow では、ファイルサイズまたはファイル名に基づいてリソース割り当てを階層化するための条件付きリソースを使用します。詳細については、Nextflow GitHub サイトの「条件付きプロセスリソース」を参照してください。

最適化が早すぎない

大幅なパフォーマンス調整作業に投資する前に、ワークフローコードとロジックを確定します。コードを変更すると、必要なリソースに大きな影響を与える可能性があります。開発プロセスで実行の最適化が早すぎると、最適化が過剰になるか、後でワークフロー定義が変更された場合に再度最適化が必要になる場合があります。

Run Analyzer ツールを定期的に再実行する

ワークフロー定義を経時的に変更した場合、またはサンプル分散が変更された場合は、Run Analyzer ツールを定期的に実行して、追加の最適化を行うことができます。

リソースの同時実行を最適化する方法

HealthOmics には、実行を大規模に処理する際のコストの制御と管理に役立つ以下の機能があります。

- 実行グループを使用して、コストとリソース使用量を制御します。タスクあたりの同時実行数、vCPUs、GPUs、合計実行時間に対して、実行グループの最大値を設定できます。別々のチームまたはグループが同じアカウントを使用している場合は、チームごとに個別の実行グループを作成できます。チームごと、および実行グループの最大値を設定することで、リソースの使用状況とコストを制御できます。詳細については、「HealthOmics 実行グループの作成」を参照してください。
- 開発中に、最大値が低い個別の実行グループを設定して、ランナウェイタスクをキャッチできます。
- Service Quotas は、過剰なリソースリクエストからアカウントを保護するのに役立ちます。クォータ値の増加をリクエストする方法など、Service Quotas の詳細については、「」を参照してください。 HealthOmics サービスクォータ

HealthOmics で実行グループと実行グループを削除する

実行グループまたは実行グループが不要になった場合は、、API AWS CLI、またはコンソールを使用して削除できます。

実行の削除に加えて、実行をキャンセルすることもできます。実行をキャンセルするには、そのステータスが PENDING、、STARTINGRUNNING、または である必要がありますSTOPPING。

> **Note**
>
> 実行をキャンセルしても、HealthOmics は実行出力を保存しません。

次の AWS CLI コマンドは、実行をキャンセルする方法を示しています。この例を実行するには、 をキャンセルする実行の ID *run id*に置き換えます。成功した場合、レスポンスはありません。

```
aws omics cancel-run --id run id
```

次の AWS CLI コマンドは実行を削除します。実行は、完了またはキャンセルされた場合にのみ削除できます。この例を実行するには、 を、削除する実行の *run id* ID に置き換えます。実行が正常に削除された場合、レスポンスはありません。

```
aws omics delete-run --id run id
```

実行グループを削除することもできます。実行グループは、ステータスが PENDING、、STARTING、RUNNINGまたは である実行グループに関連付けられた実行がない場合にのみ削除できますSTOPPING。

次の例は、 を使用して実行グループ AWS CLI を削除する方法を示しています。レスポンスは送信されません。この例を実行するには、 を、削除する実行グループの *run group id* ID に置き換えます。

```
aws omics delete-run-group --id run group id
```

HealthOmics 実行グループの作成

オプションで実行グループを作成して、グループに追加する実行のコンピューティングリソースを制限できます。実行グループは以下に役立ちます。

- サービス制限を超えないように実行をキューに入れます。
- 最大実行期間を設定して、ランアウェイタスクをキャッチします。
- 最も重要な実行が最初に完了するように、各実行の優先度を管理します。

最大同時 vCPU、GPU、または実行を設定すると、最大数に達すると実行タスクがキューに入れられます。最大実行期間を設定すると、最大実行期間を超えると実行は失敗します。

実行優先度設定を使用して、実行グループ内の優先度を確立します。

サービスの制限は、実行グループの制限よりも優先されます。たとえば、実行グループの最大値をリージョンのサービス最大値よりも高い値に設定すると、HealthOmics はサービス最大値を適用します。

トピック

実行優先度

実行優先度を使用して、実行グループで実行の優先度を確立できます。

複数の実行の優先度が同じ場合、最初に開始された実行の優先度が高くなります。

実行グループにない実行の優先度を設定することもできます。優先度は、実行グループに含まれていない他のすべての実行の優先度と比較されます。

実行の開始時に実行優先度を設定します。詳細については、「HealthOmics での実行の開始」を参照してください。

コンソールを使用した実行グループの作成

実行グループを作成するには

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、グループの実行を選択します。
3. グループの実行ページで、実行グループの作成を選択します。
4. 実行グループの詳細の作成ページで、次の情報を入力します。
   - 実行グループ名 - この実行グループの一意の名前。
   - 同時実行の最大 vCPU - 実行グループ内のすべてのアクティブな実行で同時に実行できる vCPUs の最大数。
   - 最大 GPUs - 実行グループ内のすべてのアクティブな実行で同時に実行できる GPUs の最大数。
   - 実行あたりの最大実行時間 (分) - 各実行の最大時間 (分単位) 。実行が最大実行時間を超えると、実行は自動的に失敗します。
   - 最大同時実行数 - 同時に実行できる実行の最大数。
5. (オプション) 実行グループに最大 50 個のタグを追加できます。
6. 実行グループの作成 を選択します。

CLI を使用した実行グループの作成

実行グループを作成するには、create-run-group API オペレーションを使用して、 という名前の実行グループを作成しますTestRunGroup。次の例では、最大 20 CPUs、10 GPUs、5 回の実行、最大実行時間を 600 分に設定します。

```
aws omics create-run-group --name TestRunGroup \
--max-cpus 20 \
```

```
--max-gpus 10 \
--max-duration 600 \
--max-runs 5
```

この API オペレーションからのレスポンスには、新しく作成された の ID が含まれますRunGroup。

```
{
    "arn": "arn:aws:omics:us-west-2:12345678901:runGroup/2839621",
    "id": "2839621",
    "tags": {}
}
```

実行グループに関する追加情報を取得するには、次の例に示すように、get-run-group API オペレーションでこの ID を使用します。

```
aws omics get-run-group --id run group id
```

レスポンスには、実行グループの制限設定と割り当てられたタグが含まれます。

```
{
    "arn": "arn:aws:omics:us-west-2:776893852117:runGroup/2839621",
    "id": "2839621",
    "name": "TestRunGroup",
    "maxCpus": 20,
    "maxRuns": 5,
    "maxDuration": 600,
    "creationTime": "2024-06-12T15:35:39.191730+00:00",
    "tags": {},
    "maxGpus": 10
}
```

list-run-group API オペレーションを使用して、作成されたすべての実行グループを表示することもできます。

```
aws omics list-run-groups
```

HealthOmics 実行のコールキャッシュ

AWS HealthOmics は、プライベートワークフローの再開とも呼ばれるコールキャッシュをサポートしています。コールキャッシュは、実行の完了後に完了したワークフロータスクの出力を保存しま

す。後続の実行では、タスク出力を再度計算するのではなく、キャッシュからのタスク出力を使用できます。コールキャッシュにより、コンピューティングリソースの使用量が減少し、実行時間が短縮され、コンピューティングコストが削減されます。

実行が完了したら、キャッシュされたタスク出力ファイルにアクセスできます。高度なタスクのデバッグとトラブルシューティングを実行するには、ワークフロー定義でこれらのファイルをタスク出力として指定することで、中間タスクファイルをキャッシュできます。

呼び出しキャッシュを使用して、失敗した実行から完了したタスク結果を保存できます。次の実行は、完了したタスクを再計算するのではなく、最後に正常に完了したタスクから開始されます。

HealthOmics がタスクに一致するキャッシュエントリを見つけられない場合、実行は失敗しません。HealthOmics は、タスクとその依存タスクを再計算します。

コールキャッシュの問題のトラブルシューティングについては、「」を参照してくださいコールキャッシュの問題のトラブルシューティング。

トピック

コールキャッシュの仕組み

コールキャッシュを使用するには、実行キャッシュを作成し、キャッシュされたデータに関連付けられた Amazon S3 の場所を持つように設定します。実行を開始するときは、実行キャッシュを指定します。実行キャッシュは 1 つのワークフロー専用ではありません。複数のワークフローからの実行では、同じキャッシュを使用できます。

実行のエクスポートフェーズ中、システムは完了したタスク出力を Amazon S3 の場所にエクスポートします。中間タスクファイルをエクスポートするには、ワークフロー定義でこれらのファイルをタスク出力として宣言します。コールキャッシュは内部的にメタデータを保存し、キャッシュエントリごとに一意のハッシュを作成します。

実行中のタスクごとに、ワークフローエンジンは、このタスクに一致するキャッシュエントリがあるかどうかを検出します。一致するキャッシュエントリがない場合、HealthOmics はタスクを計算します。一致するキャッシュエントリがある場合、エンジンはキャッシュされた結果を取得します。

キャッシュエントリを一致させるために、HealthOmics はネイティブワークフローエンジンに含まれるハッシュメカニズムを使用します。HealthOmics は、これらの既存のハッシュ実装を拡張して、S3 eTags や ECR コンテナダイジェストなどの HealthOmics 変数を考慮します。

HealthOmics は、次のワークフロー言語バージョンのコールキャッシュをサポートしています。

- WDL バージョン 1.0、1.1、および 開発バージョン
- Nextflow バージョン 23.10 および 24.10
- すべての CWL バージョン

> **Note**
>
> HealthOmics は、Ready2Run ワークフローのコールキャッシュをサポートしていません。

トピック

責任共有モデル

ユーザーと は、タスクと実行がコールキャッシュに適した候補 AWS かどうかを判断する責任を共有します。コールキャッシュは、すべてのタスクがべき等である場合に最良の結果を達成します (同じ入力を使用してタスクを繰り返し実行すると、同じ結果が得られます) 。

ただし、タスクに非決定的な要素 (乱数生成やシステム時間など) が含まれている場合、同じ入力を使用してタスクを繰り返し実行すると、出力が異なる場合があります。これは、次の方法でコールキャッシュの有効性に影響を与える可能性があります。

- HealthOmics が、タスク実行が現在の実行に対して生成する出力と同じではないキャッシュエントリ (前回の実行で作成) を使用する場合、実行はキャッシュなしで同じ実行とは異なる結果を生成する可能性があります。
- HealthOmics は、非決定的なタスク出力のため、一致する必要があるタスクに一致するキャッシュエントリを見つけられない場合があります。有効なキャッシュエントリが見つからない場合、実行はタスクを不必要に再計算するため、コールキャッシュを使用するコスト削減のメリットが減ります。

以下は、コールキャッシュの結果に影響する非決定的な結果を引き起こす可能性のある既知のタスク動作です。

- 乱数ジェネレーターの使用。
- システム時刻によって異なります。
- 同時実行の使用 (レース条件により出力の変動が発生する可能性があります) 。
- タスク入力パラメータで指定されている範囲を超えるローカルファイルまたはリモートファイルを取得します。

非決定的な動作を引き起こす可能性のあるその他のシナリオについては、Nextflow ドキュメントサイトの「非決定的なプロセス入力」を参照してください。

タスクが非決定的な出力を生成すると思われる場合は、Nextflow のキャッシュオプトアウトなどのワークフローエンジン機能を使用して、非決定的な特定のタスクがキャッシュされないようにすることを検討してください。

無効なコールキャッシュや予想とは異なる出力がリスクをもたらす可能性のある環境でコールキャッシュを有効にする前に、特定のワークフローとタスクの要件を徹底的に確認することをお勧めします。例えば、コールキャッシュが臨床ユースケースに適しているかどうかを判断する際には、コールキャッシュの潜在的な制限を慎重に検討する必要があります。

タスクのキャッシュ要件

HealthOmics は、次の要件を満たすタスクのタスク出力をキャッシュします。

- タスクはコンテナを定義する必要があります。HealthOmics は、コンテナのないタスクの出力をキャッシュしません。
- タスクは 1 つ以上の出力を生成する必要があります。ワークフロー定義でタスク出力を指定します。

- ワークフロー定義で動的値を使用しないでください。たとえば、実行ごとに増加する値を持つタスクにパラメータを渡すと、HealthOmics はタスク出力をキャッシュしません。

> **Note**
>
> 実行内の複数のタスクが同じコンテナイメージを使用する場合、HealthOmics はこれらのタスクすべてに同じイメージバージョンを提供します。HealthOmics がイメージをプルすると、実行期間中のコンテナイメージの更新は無視されます。このアプローチは、予測可能で一貫したエクスペリエンスを提供し、実行中にデプロイされるコンテナイメージの更新によって発生する可能性のある問題を防ぎます。

キャッシュパフォーマンスを実行する

実行のコールキャッシュを有効にすると、実行パフォーマンスに次のような影響が生じることがあります。

- 最初の実行中、HealthOmics は実行中のタスクのキャッシュデータを保存します。コールキャッシュはエクスポートデータの量を増やすため、この実行ではエクスポート時間が長くなることがあります。
- 以降の実行では、キャッシュから実行を再開するときに、処理ステップの数が短縮され、実行時間が短縮される可能性があります。
- また、中間ファイルを出力として宣言することを選択した場合、このデータが詳細になる可能性があるため、エクスポート時間がさらに長くなる可能性があります。

キャッシュデータの保持と無効化イベント

実行キャッシュの主な目的は、実行中のタスクの計算を最適化することです。タスクに一致する有効なキャッシュエントリがある場合、HealthOmics はタスクを再計算する代わりにキャッシュエントリを使用します。それ以外の場合、HealthOmics はデフォルトのサービス動作に戻ります。これは、タスクとその依存タスクを再計算することです。このアプローチを使用しても、キャッシュミスによって実行が失敗することはありません。

実行キャッシュサイズを管理することをお勧めします。時間の経過とともに、ワークフローエンジンまたは HealthOmics サービスの更新、または実行タスクまたは実行タスクで行った変更が原因で、キャッシュエントリが無効になる場合があります。以下のセクションでは、追加の詳細について説明します。

トピック

マニフェストバージョンの更新とデータの鮮度

HealthOmics サービスは定期的に新機能やワークフローエンジンの更新を導入し、一部またはすべての実行キャッシュエントリを無効にする場合があります。この場合、実行で 1 回限りのキャッシュミスが発生する可能性があります。

HealthOmics は、キャッシュエントリごとに JSON マニフェストファイルを作成します。2025 年 2 月 12 日以降に開始された実行の場合、マニフェストファイルにはバージョンパラメータが含まれます。サービスの更新によってキャッシュエントリが無効になる場合、HealthOmics は削除対象のレガシーキャッシュエントリを識別できるようにバージョン番号を増分します。

次の例は、 バージョンが 2 に設定されているマニフェストファイルを示しています。

```
{
     "arn": "arn:aws:omics:us-west-2:12345678901:runCache/0123456/
cacheEntry/1234567-195f-3921-a1fa-ffffcef0a6a4",
     "s3uri": "s3://example/1234567-d0d1-e230-
d599-10f1539f4a32/1348677/4795326/7e8c69b1-145f-3991-a1fa-ffffcef0a6a4",
     "taskArn": "arn:aws:omics:us-west-2:12345678901:task/4567891",
     "workDir": "/mnt/workflow/1234567-d0d1-e230-d599-10f1539f4a32/workdir/call-
TxtFileCopyTask/5w6tn5feyga7noasjuecdeoqpkltrfo3/wxz2fuddlo6hc4uh5s2lreaayczduxdm",
     "files": [
         {
             "name": "output_txt_file",
             "path": "out/output_txt_file/outfile.txt",
             "etag": "ajdhyg9736b9654673b9fbb486753bc8"
         }
     ],
     "nextflowContext": {},
     "otherOutputs": {},
     "version": 2,
  }
```

有効でなくなったキャッシュエントリを含む実行の場合は、キャッシュを再構築して新しい有効なエントリを作成します。実行ごとに次の手順を実行します。

1. キャッシュ保持を CACHE ALWAYS に設定して実行を 1 回開始します。この実行により、新しいキャッシュエントリが作成されます。
2. 後続の実行では、キャッシュ保持を以前の設定 (CACHE ALWAYS または CACHE ON FAILURE) に設定します。

有効でなくなったキャッシュエントリをクリーンアップするには、キャッシュ Amazon S3 バケットからこれらのキャッシュエントリを削除できます。HealthOmics はこれらのキャッシュエントリを再利用しません。有効でないエントリを保持することを選択した場合、実行には影響しません。

> **Note**
>
> コールキャッシュは、キャッシュに指定された Amazon S3 の場所にタスク出力データを保存し、 に料金が発生します AWS アカウント。

キャッシュ動作の実行

実行キャッシュ動作を設定して、失敗した実行 (失敗時のキャッシュ) またはすべての実行 (常にキャッシュ) のタスク出力を保存できます。実行キャッシュを作成するときは、このキャッシュを使用するすべての実行のデフォルトのキャッシュ動作を設定します。実行を開始するときに、デフォルトの動作を上書きできます。

Cache on failure は、複数のタスクが正常に完了した後に失敗するワークフローをデバッグする場合に便利です。ハッシュによって考慮されるすべての一意の変数が前回の実行と同じである場合、後続の実行は最後に正常に完了したタスクから再開されます。

Cache always は、正常に完了したワークフローでタスクを更新する場合に便利です。以下のステップに従うことをお勧めします。

1. 新しい実行を作成します。キャッシュ動作を常にキャッシュに設定し、実行を開始します。
2. 実行が完了したら、ワークフローのタスクを更新し、動作セットキャッシュで常に新しい実行を開始します。この実行は、更新されたタスクと、更新されたタスクに依存する後続のタスクを処理します。他のすべてのタスクでは、キャッシュされた結果が使用されます。
3. 必要に応じて、更新されたタスクの開発が完了するまで、ステップ 2 を繰り返します。
4. 必要に応じて、今後の実行で更新されたタスクを使用します。これらの実行に新規または異なる入力を使用する予定がある場合は、後続の実行を失敗時にキャッシュに切り替えることを忘れないでください。

> **Note**
>
> Cache always モードは、同じテストデータセットを使用している間は推奨されますが、実行のバッチには推奨されません。このモードを大量のバッチ実行に設定すると、システムは大量のデータを Amazon S3 にエクスポートできるため、エクスポート時間とストレージコストが増加します。

コントロール実行キャッシュサイズ

HealthOmics は、実行キャッシュデータを削除または自動アーカイブしたり、キャッシュデータを管理するための Amazon S3 クリーンアップルールを適用したりしません。Amazon S3 ストレージコストを節約し、実行キャッシュサイズを管理できるように、定期的なキャッシュクリーンアップを実行することをお勧めします。ファイルを直接削除することも、実行キャッシュバケットにデータ保持/レプリケーションポリシーを設定することもできます。

たとえば、Amazon S3 ライフサイクルポリシーを設定して 90 日後にオブジェクトを期限切れにしたり、各開発プロジェクトの終了時にキャッシュデータを手動でクリーンアップしたりできます。

以下の情報は、キャッシュデータサイズを管理するのに役立ちます。

- Amazon S3 を確認することで、キャッシュ内のデータ量を表示できます。HealthOmics はキャッシュサイズをモニタリングまたはレポートしません。
- 有効なキャッシュエントリを削除しても、後続の実行は失敗しません。HealthOmics は、タスクとその依存タスクを再計算します。
- HealthOmics がタスクに一致するエントリを見つけられないようにキャッシュ名またはディレクトリ構造を変更すると、HealthOmics はタスクを再計算します。

キャッシュエントリがまだ有効かどうかを確認する必要がある場合は、キャッシュマニフェストのバージョン番号を確認してください。詳細については、「マニフェストバージョンの更新とデータの鮮度」を参照してください。

実行キャッシュの作成

実行キャッシュを作成するときは、キャッシュデータの Amazon S3 の場所を指定します。このデータはすぐにアクセス可能である必要があります。コールキャッシュは、Glacier にアーカイブされたオブジェクト (GFR や GDA ストレージクラスなど) を取得しません。

キャッシュデータの Amazon S3 バケットが別の によって所有されている場合は AWS アカウント、実行キャッシュを作成するときにそのアカウント ID を指定します。

コンソールを使用した実行キャッシュの作成

コンソールから、以下の手順に従って実行キャッシュを作成します。

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、キャッシュの実行を選択します。
3. キャッシュの実行ページで、実行キャッシュの作成を選択します。
4. 実行キャッシュの作成ページのキャッシュ詳細の実行パネルで、次のフィールドを設定します。
   a. 実行キャッシュの名前を入力します。
   b. (オプション) 説明を入力します。
   c. キャッシュされた出力の S3 の場所を入力します。ワークフローと同じリージョンのバケットを選択します。
   d. (オプション) バケット所有者 AWS アカウント の を入力して、バケットの所有権を確認します。値を入力しない場合、デフォルト値はアカウント ID です。
   e. キャッシュ動作で、デフォルトの動作 (失敗した実行の出力をキャッシュするか、すべての実行の出力をキャッシュするか) を設定します。実行を開始するときに、オプションでデフォルトの動作を上書きできます。
5. (オプション) 1 つ以上のタグを実行キャッシュに関連付けます。
6. 実行キャッシュの作成 を選択します。コンソールには、新しい実行キャッシュがキャッシュの実行テーブルに表示されます。

CLI を使用した実行キャッシュの作成

createcreate-run-cache CLI コマンドを使用して、実行キャッシュを作成します。デフォルトのキャッシュ動作は ですCACHE_ON_FAILURE。

```
aws omics create-run-cache \
      --name "workflow 123 run cache" \
      --description "my run cache" \
      --cache-s3-location "s3://amzn-s3-demo-bucket" \
      --cache-behavior "CACHE_ALWAYS"                \
      --cache-bucket-owner-id  "111122223333"
```

作成が成功すると、次のフィールドを含むレスポンスを受け取ります。

```
{
  "arn": "string",
  "id": "string",
  "status": "ACTIVE"
  "tags": {}
  }
```

実行キャッシュの更新

キャッシュ名、説明、タグ、またはキャッシュ動作は変更できますが、キャッシュの S3 の場所は変更できません。

コンソールを使用した実行キャッシュの更新

コンソールから、以下の手順に従って実行キャッシュを更新します。

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、キャッシュの実行を選択します。
3. Run caches テーブルから、更新する実行キャッシュを選択し、Edit を選択します。
4. キャッシュ詳細の実行パネルでは、実行キャッシュ名、説明、キャッシュ動作フィールドを更新できます。
5. (オプション) 1 つ以上の新しいタグを実行キャッシュに関連付けるか、既存のタグを削除します。
6. 実行キャッシュの保存 を選択します。

CLI を使用した実行キャッシュの更新

updateupdate-run-cache CLI コマンドを使用して、実行キャッシュを更新します。

```
aws omics update-run-cache \
      --name "workflow 123 run cache" \
      --id "workflow id" \
      --description "my run cache" \
```

```
      --cache-behavior "CACHE_ALWAYS"
```

更新が成功すると、データフィールドのないレスポンスを受け取ります。

実行キャッシュの削除

アクティブな実行が使用していない場合は、実行キャッシュを削除できます。実行キャッシュを使用している場合は、実行が完了するまで待機するか、実行をキャンセルできます。

実行キャッシュを削除すると、リソースとそのメタデータは削除されますが、Amazon S3 内のデータは削除されません。キャッシュを削除した後は、キャッシュを再アタッチしたり、後続の実行に使用したりすることはできません。

キャッシュされたデータは、検査のために Amazon S3 に残ります。標準の S3 Deleteオペレーションを使用して、古いキャッシュデータを削除できます。または、Amazon S3 ライフサイクルポリシーを作成して、使用しなくなったキャッシュデータを期限切れにします。

コンソールを使用した実行キャッシュの削除

コンソールから、以下の手順に従って実行キャッシュを削除します。

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、キャッシュの実行を選択します。
3. Run caches テーブルから、削除する実行キャッシュを選択します。
4. Run caches テーブルメニューから、Delete を選択します。
5. モーダルダイアログから、後で参照できるように Amazon S3 キャッシュデータリンクを保存し、実行キャッシュを削除することを確認します。

   Amazon S3 リンクを使用してキャッシュされたデータを検査することはできますが、データを別の実行キャッシュに再リンクすることはできません。検査が完了したら、キャッシュデータを削除します。

CLI を使用した実行キャッシュの削除

deletedelete-run-cache CLI コマンドを使用して、実行キャッシュを削除します。

```
aws omics delete-run-cache \
      --id "my cache id"
```

削除が成功すると、データフィールドのないレスポンスを受け取ります。

実行キャッシュの内容

HealthOmics は、S3 バケット内の次の構造で実行キャッシュを整理します。

```
s3://{cache.S3location}/{cache.uuid}/runID/taskID/{cacheentry.uuid}/
```

cache.uuid は、キャッシュのグローバルに一意の ID です。cacheentry.uuid は、キャッシュされたタスクのグローバルに一意の uuid です。HealthOmics は uuid をキャッシュとタスクに割り当てます。

すべてのワークフローエンジンについて、キャッシュには次のファイルが含まれます。

- {cacheentryuuid}.json ファイル – HealthOmics はこのマニフェストファイルを作成します。このマニフェストファイルには、キャッシュ内のすべての項目のリストやキャッシュバージョンなど、キャッシュに関する情報が含まれています。
- タスク出力ファイル – 各タスク出力は、タスクで定義された 1 つ以上のファイルで構成されます。

Nextflow を使用するワークフローの場合、Nextflow エンジンはこれらの追加ファイルをキャッシュに作成します。

- command.out ファイル – このファイルには、タスク実行の stdout コンテンツが含まれています。
- .exitcode ファイル – このファイルには、タスク終了コード (整数) が含まれています。

> **Note**
>
> 高度なトラブルシューティングのために実行キャッシュ内の中間タスクファイルにアクセスする場合は、ワークフロー定義でこれらのファイルをタスク出力として宣言します。

エンジン固有のキャッシュ機能

HealthOmics は、ワークフローエンジン間でコールキャッシュの一貫した実装を提供しようとします。各ワークフローエンジンが特定のケースをどのように処理するかによって、いくつかの違いがあります。

- ネクストフロー

  - 異なる Nextflow バージョン間でのキャッシュは保証されません。たとえば、v23.10.0 でタスクを実行し、その後 v24.10.8 で同じタスクを実行すると、HealthOmics は 2 回目の実行をキャッシュミスと見なす場合があります。
  - キャッシュfalseディレクティブを使用して、個々のタスクのキャッシュをオフにできます。このディレクティブの詳細については、Nextflow 仕様の「プロセス」を参照してください。
  - HealthOmics は Nextflow lenient モードを使用しますが、ディープキャッシュモードはサポートしていません。
  - タスクの入力への S3 パスで glob パターンを使用する場合、キャッシュは個々の S3 オブジェクトを評価します。新しいオブジェクトを追加すると、HealthOmics は新しいオブジェクトを使用するタスクのみを再計算します。
  - HealthOmics はタスクの再試行をキャッシュしません。この動作は、Nextflow のデフォルトの動作と一致します。
- WDL
  - HealthOmics は、WDL ワークフローの開発バージョンを使用する場合、入力用の新しい「ディレクトリ」タイプをサポートしています。コールキャッシュの場合、ディレクトリ内のオブジェクトが変更されると、HealthOmics はディレクトリを入力するすべてのタスクを再計算します。
  - HealthOmics はタスクレベルのキャッシュをサポートしていますが、ワークフローレベルのキャッシュはサポートしていません。
- CWL
  - タスクからの定数出力は、マニフェストから明示的には表示されません。HealthOmics は、定数出力を中間ファイルとしてキャッシュします。

実行キャッシュの使用

デフォルトでは、 実行は実行キャッシュを使用しません。実行にキャッシュを使用するには、実行を開始するときに実行キャッシュと実行キャッシュの動作を指定します。

実行が完了したら、 コンソール、CloudWatch Logs、または API オペレーションを使用して、キャッシュヒットを追跡したり、キャッシュの問題をトラブルシューティングしたりできます。詳細については、「通話キャッシュ情報の追跡」および「コールキャッシュの問題のトラブルシューティング」を参照してください。

実行の 1 つ以上のタスクが非決定的な出力を生成する場合は、実行にコールキャッシュを使用しないか、これらの特定のタスクをキャッシュから除外することを強くお勧めします。詳細については、「責任共有モデル」を参照してください。

> **Note**
>
> 実行を開始するときに IAM サービスロールを指定します。コールキャッシュを使用するには、サービスロールに実行キャッシュ Amazon S3 の場所にアクセスするためのアクセス許可が必要です。詳細については、「のサービスロール AWS HealthOmics」を参照してください。

トピック

コンソールを使用した実行キャッシュを使用した実行の設定

コンソールから、実行の開始時に実行キャッシュを設定します。

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、実行を選択します。
3. 実行ページで、開始する実行を選択します。
4. 「実行の開始」を選択し、「実行の開始」のステップ 1 と 2 を完了しますコンソールを使用した実行の開始。
5. 実行の開始のステップ 3 で、既存の実行キャッシュの選択を選択します。
6. キャッシュ ID の実行ドロップダウンリストからキャッシュを選択します。
7. デフォルトの実行キャッシュ動作を上書きするには、実行のキャッシュ動作を選択します。詳細については、「キャッシュ動作の実行」を参照してください。
8. 「実行を開始する」のステップ 4 に進みます。

CLI を使用した実行キャッシュを使用した実行の設定

実行キャッシュを使用する実行を開始するには、cache-id パラメータを start-run CLI コマンドに追加します。必要に応じて、 `cache-behavior`パラメータを使用して、実行キャッシュ用に設定した

デフォルトの動作を上書きします。次の例は、 コマンドのキャッシュフィールドのみを示しています。

```
aws omics start-run \
        ...
      --cache-id "xxxxxx"    \
      --cache-behavior  CACHE_ALWAYS
```

オペレーションが成功すると、データフィールドのないレスポンスを受け取ります。

実行キャッシュのエラーケース

以下のシナリオでは、キャッシュ動作が常にキャッシュに設定されている実行であっても、HealthOmics がタスク出力をキャッシュしない場合があります。

- 最初のタスクが正常に完了する前に実行でエラーが発生した場合、エクスポートするキャッシュ出力はありません。
- エクスポートプロセスが失敗した場合、HealthOmics はタスク出力を Amazon S3 キャッシュの場所に保存しません。
- filesystem out of space エラーが原因で実行が失敗した場合、呼び出しキャッシュはタスク出力を保存しません。
- 実行をキャンセルした場合、呼び出しキャッシュはタスク出力を保存しません。
- 実行に実行タイムアウトが発生した場合、失敗時にキャッシュを使用するように実行を設定しても、呼び出しキャッシュはタスク出力を保存しません。

通話キャッシュ情報の追跡

コンソール、 CLI、または CloudWatch Logs を使用して、コールキャッシュイベント (キャッシュヒットの実行など) を追跡できます。

トピック

- コンソールを使用してキャッシュヒットを追跡する
- CLI を使用して通話キャッシュを追跡する
- CloudWatch Logs を使用して通話キャッシュを追跡する

コンソールを使用してキャッシュヒットを追跡する

実行の詳細ページに、実行タスクテーブルに各タスクのキャッシュヒット情報が表示されます。このテーブルには、関連付けられたキャッシュエントリへのリンクも含まれています。次の手順を使用して、実行のキャッシュヒット情報を表示します。

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、実行を選択します。
3. 実行ページで、検査する実行を選択します。
4. 実行の詳細ページで、タスクの実行タブを選択してタスクテーブルを表示します。
5. タスクにキャッシュヒットがある場合、キャッシュヒット列には Amazon S3 の実行キャッシュエントリの場所へのリンクが含まれます。
6. リンクを選択して、実行キャッシュエントリを検査します。

CLI を使用して通話キャッシュを追跡する

get-run CLI コマンドを使用して、実行が呼び出しキャッシュを使用したかどうかを確認します。

```
aws omics get-run --id 1234567
```

レスポンスで、 `cacheId`フィールドが設定されている場合、実行はそのキャッシュを使用します。

list-run-tasks CLI コマンドを使用して、実行中の各キャッシュされたタスクのキャッシュデータの場所を取得します。

```
aws omics list-run-tasks --id 1234567
```

レスポンスで、タスクの cacheHit フィールドが true の場合、cacheS3Uri フィールドはそのタスクのキャッシュデータの場所を提供します。

get-run-task CLI コマンドを使用して、特定のタスクのキャッシュデータの場所を取得することもできます。

```
aws omics get-run-task --id 1234567 --task-id <task_id>
```

CloudWatch Logs を使用して通話キャッシュを追跡する

HealthOmics は /aws/omics/WorkflowLog CloudWatch ロググループにキャッシュアクティビティログを作成します。実行キャッシュごとにログストリームがあります: runCache/<cache_id>/<cache_uuid>。

コールキャッシュを使用する実行の場合、HealthOmics はこれらのイベントの CloudWatch Logs エントリを生成します。

- キャッシュエントリの作成 (CACHE_ENTRY_CREATED)
- キャッシュエントリの一致 (CACHE_HIT)
- キャッシュエントリの一致に失敗する (CACHE_MISS)

これらのログの詳細については、「」を参照してくださいCloudWatch のログ 。

/aws/omics/WorkflowLog ロググループで次の CloudWatch Insights クエリを使用して、このキャッシュの実行ごとのキャッシュヒット数を返します。

```
filter @logStream like 'runCache/<CACHE_ID>/'
 fields @timestamp, @message
 filter logMessage like 'CACHE_HIT'
 parse "run: *," as run
 stats count(*) as cacheHits by run
```

次のクエリを使用して、各実行によって作成されたキャッシュエントリの数を返します。

```
filter @logStream like 'runCache/<CACHE_ID>/'
 fields @timestamp, @message
 filter logMessage like 'CACHE_ENTRY_CREATED'
 parse "run: *," as run
 stats count(*) as cacheEntries by run
```

HealthOmics ワークフローの共有

プライベートワークフローの所有者は、同じリージョン AWS アカウント の とワークフローを共有できます。ワークフローを複数のワークフローと共有するには AWS アカウント、同じワークフローの複数の共有を作成します。

所有者は、共有を削除することで、共有ワークフローへのアクセスを取り消すことができます。

> **Note**
>
> HealthOmics は、ワークフローがサブスクライバーのアカウントで実行されている間、共有ワークフローが Amazon ECR リポジトリに自動的にアクセスできるようにします。共有ワークフローに追加のリポジトリアクセスを付与する必要はありません。

ワークフローを共有すると、サブスクライバーは任意のワークフローバージョンを使用できます。共有ワークフローのバージョンレベルのアクセスコントロールが必要な場合は、ワークフローバージョンを使用するのではなく、個別のワークフローを作成することをお勧めします。

トピック

共有ワークフローへのサブスクライブ

共有ワークフローにサブスクライブするには、以下の全体的な手順に従ってワークフローを受け入れて使用します。

1. コンソールまたは API を使用して共有を受け入れます。現在のリージョンを共有リクエストと同じリージョンに設定します。
   - コンソールで共有リクエストを検索するには、すべてのリソース共有ページに移動し、共有タブを選択します。
2. コンソールまたは API を使用して、共有ワークフローの実行を作成します。
   - コンソールでワークフローの詳細ページを検索するには、「自分と共有」に移動し (ステップ 1 を参照) 、共有ワークフローのリソースリンクを選択します。
3. ワークフロー用に独自の入力データを指定します。
4. 共有ワークフローは で実行されます AWS アカウント。

共有ワークフローのサブスクライバーとして、システムは次のワークフローアクションを実行することをブロックします。

- 共有ワークフローのエクスポート
- 共有ワークフローの再実行
  - 共有ワークフローの新しい実行を作成します。
- ワークフローを再共有します。
- ワークフローへのタグの割り当て。
- ワークフローを削除します。
  - ワークフローが不要になった場合は、ワークフロー共有を削除します。

リソース共有の詳細についてはでのクロスアカウントリソース共有 AWS HealthOmics、「」を参照してください。

ワークフロー共有のステータスのモニタリング

HealthOmics は、ワークフロー共有のステータス変更ごとに EventBridge にイベントを送信します。特定のステータス変更に関する通知を受信する場合は、EventBridge ルールを設定して、ワークフロー共有ステータス変更イベントをモニタリングします。例:

- ワークフロー共有リクエストを受信するたびに、およびユーザーがワークフロー共有を取り消すたびに通知が必要です。
- ワークフロー共有リクエストを開始した後、ユーザーがリクエストを承諾または拒否したときに通知を受け取る必要があります。

イベントの使用の詳細については、「」を参照してくださいでの EventBridge の使用 AWS HealthOmics。

コンソールを使用したプライベートワークフローの共有

コンソールから、プライベートワークフローをワークフローと同じリージョン AWS アカウント のと共有できます。

プライベートワークフローを共有するには

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、プライベートワークフローを選択します。

3. プライベートワークフローページのワークフローテーブルから、共有するワークフローを選択し、共有を選択します。
4. 共有ワークフローページの共有詳細パネルで、共有のわかりやすい名前を入力し、サブスクライバー AWS アカウント の を入力します。
5. リソースの共有を選択します。コンソールでは、すべてのリソース共有ページにリソース共有が表示されます。

共有の初期状態は保留中です。サブスクライバーが共有を受け入れると、状態はアクティブに変わります。

CLI を使用したプライベートワークフローの共有

ワークフロー共有を作成するには、create-share API オペレーションを使用します。プリンシパルサブスクライバーは AWS アカウント 、ワークフローにアクセスするユーザーの です。

```
aws omics create-share \
  --resource-arn "arn:aws:omics:us-west-2:555555555555:workflow/123456" \
  --principal-subscriber "123456789012" \
  --name "my_Share-123"
```

作成が成功すると、共有 ID とステータスを含むレスポンスを受け取ります。

```
{
"shareId": "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a",
"name": "my_Share-123",
"status": "PENDING"
}
```

共有は、サブスクライバーが `accept-share` API オペレーションを使用して承諾するまで保留状態のままです。

その他の API の使用例でのクロスアカウントリソース共有 AWS HealthOmicsについては、「」を参照してください。

コンソールを使用した共有ワークフローの受け入れ

コンソールを使用して、提供されたワークフロー共有を受け入れることができます。コンソールをワークフローと同じリージョンに設定してください。

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、すべてのリソース共有を選択し、共有タブを選択します。
3. 「自分と共有されているリソース」テーブル から、ワークフロー共有を選択し、「承諾」を選択します。

ワークフローを承諾したら、共有ワークフローのリソースリンクを選択して詳細を表示します。

コンソールを使用した共有ワークフローの実行

ワークフロー共有を受け入れると、ワークフローで実行を開始できます。

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、すべてのリソース共有を選択し、共有タブを選択します。
3. 自分と共有されているリソース テーブルから、共有ワークフローのリソースリンクを選択します。
4. ワークフローの詳細ページで、実行の作成を選択します。

   コンソールで実行の作成ページが開き、ワークフロータイプ (共有) とワークフロー ID が事前に入力されています。
5. 実行フォームの作成で残りのフィールドを設定します。詳細については、「コンソールを使用した実行の開始」を参照してください。

API を使用して共有ワークフローを実行する

get-workflow を使用して、共有ワークフローの ARN を取得します。

```
aws omics get-workflow --id 1234567 \
--workflow-owner-id 55555555555
```

ワークフローを実行するときは、ワークフロー所有者の AWS アカウント ID と共有ワークフローの ARN を指定します。

```
aws omics start-run --id 1234567 --workflow-owner-id 55555555555 \
--role-arn arn:aws:iam::1234567892012:role/service-role/OmicsWorkflow-20221004T164236 \
--name ArchiveTest --retention-mode REMOVE
```

HealthOmics で Ready2Run ワークフロー

Ready2Run ワークフローは、サードパーティーのパブリッシャーによって公開される事前設定されたワークフローです。Sentieon Inc などの一部のパブリッシャーは、サブスクリプションベースのワークフローを提供しています。他の Ready2Run ワークフローはサブスクリプションを必要とせず、NF-Core ワークフローなどの一部のワークフローはオープンソースです。

Ready2Run ワークフローは、次のシナリオに適しています。

- 基盤となるインフラストラクチャをセットアップすることなく、パイプライン出力の分析と結果の生成に集中したいと考えています。
- 確立されたワークフローを使用して結果をレプリケートしたい。
- ソフトウェア開発者は、アプリケーションを HealthOmics SDK と直接統合する必要があります。

HealthOmics は Ready2Run ワークフローのバージョニングをサポートしています。バージョンを提供する Ready2Run ワークフローでは、実行の開始時にバージョン名を指定できます。

すべての Ready2Run ワークフローは、トラブルシューティングに使用できる CloudWatch ログを含むログを提供します。

> **Note**
>
> Sentieon Ready2Run ワークフローはサブスクリプションベースです。Sentieon Ready2Run ワークフローをアカウントで初めて実行すると、Sentieon は の 2 週間の評価ライセンスを自動的に作成します AWS アカウント。ライセンスは、すべての Sentieon Ready2Run ワークフローで有効です。評価期間が終了したら、永続的ライセンスをリクエストするか、評価ライセンスの拡張をリクエストできます。詳細については、「Subscribing to Sentieon Ready2Run workflows」を参照してください。

トピック

HealthOmics で使用可能な Ready2Run ワークフロー

次の表に、HealthOmics で使用できる Ready2Run ワークフローを示します。

HealthOmics コンソールにログインして、入力パラメータやワークフロー図など、これらのワークフローに関する詳細情報を表示できます。Ready2Run ワークフローの料金情報については、HealthOmics の料金」を参照してください。

> **Note**
>
> 各 Ready2Run ワークフローには最大入力ファイルサイズがあります。これらの最大ファイルサイズは調整できません。

[Workflow name] (ワークフロー名)	パブリッシャー	サブスクリプションが必要ですか？	最大入力ファイルサイズ (GiB)	推定実行時間 (HH:MM)
AlphaFold for 601-1200	Google DeepMind	なし	1	11:15
最大 600 個のリテンション用の AlphaFold	Google DeepMind	なし	1	7:30
Bases2Fastq for 2x150	要素バイオサイエンス	なし	1,000	1:45
2x300 用の Bases2Fastq	要素バイオサイエンス	なし	1,000	1:30
Bases2Fastq for 2x75	要素バイオサイエンス	なし	500	0:45
最大 800 個のリテンションに対応する ESMFold	メタリサーチ	なし	1	0:15

[Workflow name] (ワークフロー名)	パブリッシャー	サブスクリプションが必要ですか？	最大入力ファイルサイズ (GiB)	推定実行時間 (HH:MM)
GATK-BP fq2bam	ブロードインスティテュート	なし	64	10:10
30x ゲノムの GATK-BP Germline bam2vcf	ブロードインスティテュート	なし	39	2:45
30x ゲノム用の GATK-BP Germline fq2vcf	ブロードインスティテュート	なし	64	12:30
GATK-BP Somatic WES bam2vcf	ブロードインスティテュート	なし	86	1:30
NVIDIA Parabricks BAM2FQ2BAM WGS 最大 30X	NVIDIA 株式会社	なし	80	1:39
NVIDIA Parabricks BAM2FQ2BAM WGS 最大 50X	NVIDIA 株式会社	なし	120	2:45
NVIDIA Parabricks BAM2FQ2BAM WGS 最大 5X	NVIDIA 株式会社	なし	20	0:18

[Workflow name] (ワークフロー名)	パブリッシャー	サブスクリプションが必要ですか？	最大入力ファイルサイズ (GiB)	推定実行時間 (HH:MM)
NVIDIA Parabricks FQ2BAM WGS、最大 30X	NVIDIA 株式会社	なし	71	1:00
NVIDIA Parabricks FQ2BAM WGS、最大 50X	NVIDIA 株式会社	なし	137	1:45
NVIDIA Parabricks FQ2BAM WGS 最大 5X	NVIDIA 株式会社	なし	13	0:15
NVIDIA Parabricks Germline DeepVariant WGS 最大 30X	NVIDIA 株式会社	なし	71	2:00
NVIDIA Parabricks Germline DeepVariant WGS、最大 50X	NVIDIA 株式会社	なし	137	3:30
NVIDIA Parabricks Germline DeepVariant WGS、最大 5X	NVIDIA 株式会社	なし	12	0:30

[Workflow name] (ワークフロー名)	パブリッシャー	サブスクリプションが必要ですか？	最大入力ファイルサイズ (GiB)	推定実行時間 (HH:MM)
NVIDIA Parabricks Germline HaplotypeCaller WGS、最大 30X	NVIDIA 株式会社	なし	71	1:15
NVIDIA Parabricks Germline HaplotypeCaller WGS、最大 50X	NVIDIA 株式会社	なし	137	2:00
NVIDIA Parabricks Germline HaplotypeCaller WGS、最大 5X	NVIDIA 株式会社	なし	13	0:15
NVIDIA Parabricks Somatic Mutect2 WGS、最大 50X	NVIDIA 株式会社	なし	196	0:45
KallistoBUStools を使用した scRNAseq	NF-Core	なし	119	1:30
Salmon Alevin-fry を使用した scRNAseq	NF-Core	なし	119	2:30
scRNAseq と STARsolo	NF-Core	なし	119	2:30

[Workflow name] (ワークフロー名)	パブリッシャー	サブスクリプションが必要ですか？	最大入力ファイルサイズ (GiB)	推定実行時間 (HH:MM)
最大 300 倍の Sentieon Germline BAM WES	Sentieon, Inc.	あり	9	1:00
最大 32 倍の Sentieon Germline BAM WGS	Sentieon, Inc.	あり	18	1:30
最大 100 倍の Sentieon Germline FASTQ WES	Sentieon, Inc.	あり	5	0:45
最大 300 倍の Sentieon Germline FASTQ WES	Sentieon, Inc.	あり	26	2:00
最大 32 倍の Sentieon Germline FASTQ WGS	Sentieon, Inc.	あり	51	3:30
ONT 用 Sentieon LongRead	Sentieon, Inc.	あり	25	1:30
PacBio HiFi 用 Sentieon LongRead	Sentieon, Inc.	あり	58	4:00
Sentieon Somatic WES	Sentieon, Inc.	あり	50	2:30

[Workflow name] (ワークフロー名)	パブリッシャー	サブスクリプションが必要ですか？	最大入力ファイルサイズ (GiB)	推定実行時間 (HH:MM)
Sentieon Somatic WGS	Sentieon, Inc.	あり	113	4:30
最大 40 倍の Ultima Genomics DeepVariant	Ultima ゲノミクス	なし	91	1:55

Ready2Run ワークフローを使用する場合、ワークフローは事前設定されており、編集できません。プライベートワークフローとは対照的に、Ready2Run ワークフローは以下をサポートしていません。

- 最大入力ファイルサイズを増やす
- コンピューティングリソースまたは実行ストレージの変更
- ワークフロー定義またはコンテナの変更
- 実行グループへの実行の追加
- ワークフローの共有

パブリッシャーが GitHub で Ready2Run ワークフローを共有している場合は、Ready2Run ワークフローに基づいて独自のプライベートワークフローを作成できます。次の表は、各パブリッシャーの GitHub ワークフローへのリンクを示しています。

パブリッシャー	GitHub のワークフロー
Google DeepMind、Meta Research	タンパク質折りたたみワークフロー
要素バイオサイエンス	詳細については、Element Biosciences にお問い合わせください。
ブロードインスティテュート	GATK ワークフロー
NVIDIA 株式会社	Parabricks ワークフロー

パブリッシャー	GitHub のワークフロー
nf-core	NF-Core ワークフロー
センティオン	Sentieon ワークフロー
Ultima ゲノミクス	Ultima Genomics ワークフロー

Sentieon Ready2Run ワークフローへのサブスクライブ

Sentieon Ready2Run ワークフローはサブスクリプションベースです。Sentieon Ready2Run ワークフローをアカウントで初めて実行すると、Sentieon は の 2 週間の評価ライセンスを自動的に作成します AWS アカウント。ライセンスは、すべての Sentieon Ready2Run ワークフローで有効です。評価期間が終了したら、永続的ライセンスをリクエストするか、評価ライセンスの拡張をリクエストできます。

Sentieon Ready2Run ワークフローをサブスクライブするには、次の手順に従います。

- 以下の手順に従って、 AWS 正規ユーザー ID を見つけます。
- ソフトウェアライセンスをリクエストするには、Sentieon サポートグループ (support@sentieon.com) に E メールを送信します。 AWS E メールに正規ユーザー ID を入力します。

コンソールを使用した HealthOmics Ready2Run ワークフローの開始

コンソールでの Ready2Run ワークフローの使用は、プライベートワークフローの使用と似ています。主な違いの 1 つは、ワークフローパブリッシャーが独自のデータを作成せずにワークフローを試すことができるようにサンプルデータを提供することです。

コンソールで Ready2Run ワークフローを使用するには

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、Ready2Run ワークフローを選択します。
3. Ready2Run ワークフローページで、使用するワークフローを選択します。コンソールで、そのワークフローの詳細ページが開きます。

4. 詳細タブには、名前、実行あたりの定価、説明、ワークフロー言語タイプ、実行ストレージ容量、ステータス、作成日、説明を含むパラメータなどの情報が一覧表示されます。詳細タブには、ワークフローにサブスクリプションが必要かどうかも表示されます。
5. ワークフローを使用するには、実行の作成を選択します。
6. 実行の詳細の指定 ページで、実行名を入力します。必要に応じて、ワークフローバージョンを指定できます。実行に実行優先度を追加することもできます。
7. 実行出力の Amazon S3 の場所を入力または選択します。
8. メタデータ保持の実行モードでは、実行メタデータを保持するか削除するかを選択します。
9. サービスロールパネルで、既存のサービスロールを使用するか、新しいサービスロールを作成するかを選択します。
10. (オプション) 実行の識別と管理に役立つタグを追加します。
11. [次へ] を選択します。
12. パラメータの追加 ページで、実行パラメータ値を追加するオプションのいずれかを選択します。

    - Amazon S3 の場所からパラメータファイル (JSON 形式) を選択します。
    - ローカルドライブからパラメータファイル (JSON 形式) を選択します。
    - パラメータ値を手動で入力します。
    - ワークフローパブリッシャーから提供された Ready2Run サンプルデータを使用してワークフローを実行します。
13. JSON ファイルをアップロードすると、コンソールはファイルを解析し、インライン検証を実行します。その後、必要に応じてパラメータの値を手動で更新できます。
14. [次へ] を選択します。
15. 入力を確認し、実行の開始を選択します。

API を使用して HealthOmics Ready2Run ワークフローを開始する

ほとんどの API オペレーションは、Ready2Run ワークフローとプライベートワークフローと同様に動作します。

使用可能な Ready2Run ワークフローのリストを返すには、 `type`パラメータを READY2RUN に設定して list-workflows を使用します。

```
aws omics list-workflows --type READY2RUN
```

list-workflows レスポンスから実行するワークフローを特定したら、 --idパラメータで get-workflow を使用して詳細を取得できます。

```
aws omics get-workflow --type READY2RUN --id workflow id
```

Ready2Run ワークフローを実行するには、次の例に示すようにREADY2RUN、ワークフロータイプパラメータを に設定して Start-run API オペレーションを使用できます。

```
aws-omics start-run \
  --workflow-type READY2RUN \
  --workflow-id workflow id \
  --output-uri &example-s3-bucket; \
  --role-arn arn:aws:iam::1234567892012:role/service-role/OmicsWorkflow-20221004T164236
 \
  --parameters file:///path/to/parameters.json
```

ワークフローバージョンを指定するには、この例に示すように Workflow-version パラメータを使用します。

```
aws-omics start-run \
  --workflow-type READY2RUN \
  ...
  --version-name '3.0.0'
```

実行をモニタリングするには、次に示すように get-run API オペレーションを使用できます。

```
aws-omics get-run \
  --id run id
```

HealthOmics ストレージ

HealthOmics ストレージを使用して、ゲノミクスデータを効率的かつ低コストで保存、取得、整理、共有できます。HealthOmics ストレージは、異なるデータオブジェクト間の関係を理解し、同じソースデータから発生した読み取りセットを定義できます。これにより、データの出所が提供されます。

ACTIVE 状態に保存されているデータは、すぐに取得できます。30 日以上アクセスされていないデータは ARCHIVE状態で保存されます。アーカイブされたデータにアクセスするには、API オペレーションまたはコンソールを使用してデータを再アクティブ化します。

HealthOmics シーケンスストアは、ファイルのコンテンツの整合性を維持するように設計されています。ただし、アクティブな階層化とアーカイブ階層化中の圧縮のため、インポートされたデータファイルとエクスポートされたファイルのビット単位の同等性は保持されません。

取り込み中、HealthOmics はエンティティタグまたは HealthOmics ETag を生成して、データファイルのコンテンツの整合性を検証できるようにします。シーケンス部分は、読み取りセットのソースレベルで ETag として識別され、キャプチャされます。ETag 計算では、実際のファイルやゲノムデータは変更されません。リードセットが作成された後、ETag はリードセットソースのライフサイクルを通じて変更されません。つまり、同じファイルを再インポートすると、同じ ETag 値が計算されます。

トピック

HealthOmics ETags とデータ出所

HealthOmics ETag (エンティティタグ) は、シーケンスストアに取り込まれたコンテンツのハッシュです。これにより、取り込まれたデータファイルのコンテンツの整合性を維持しながら、データの取得と処理が簡素化されます。ETag は、オブジェクトのメタデータではなくセマンティックコンテンツへの変更を反映します。指定されたリードセットタイプとアルゴリズムによって、ETag の計算方法が決まります。ETag 計算では、実際のファイルやゲノムデータは変更されません。読み取りセットのファイルタイプスキーマがそれを許可すると、シーケンスストアはデータ出所にリンクされたフィールドを更新します。

ファイルにはビット単位のアイデンティティとセマンティックアイデンティティがあります。ビット単位のアイデンティティは、ファイルのビットが同じであることを意味します。セマンティックアイデンティティは、ファイルのコンテンツが同じであることを意味します。セマンティックアイデンティティは、ファイルのコンテンツの整合性をキャプチャする際に、メタデータの変更や圧縮の変更に強いです。

HealthOmics シーケンスストアのリードセットは、オブジェクトのライフサイクル全体で圧縮/解凍サイクルとデータ出所の追跡が行われます。この処理中、取り込まれたファイルのビット単位のアイデンティティは変更される可能性があり、ファイルがアクティブ化されるたびに変更されることが予想されますが、ファイルのセマンティックアイデンティティは維持されます。セマンティックアイデンティティは HealthOmics エンティティタグ、またはシーケンスストアの取り込み中に計算され、リードセットメタデータとして利用できる ETag としてキャプチャされます。

読み取りセットのファイルタイプスキーマで許可されている場合、シーケンスストアの更新フィールドはデータの出所にリンクされます。uBAM、BAM、および CRAM ファイルの場合、新しい `@CO`または `Comment` タグが ヘッダーに追加されます。コメントには、シーケンスストア ID と取り込みタイムスタンプが含まれます。

Amazon S3 ETags

Amazon S3 URI を使用してファイルにアクセスする場合、Amazon S3 API オペレーションは Amazon S3 ETag 値とチェックサム値も返すことがあります。Amazon S3 ETag とチェックサムの値は、ファイルのビット単位のアイデンティティを表すため、HealthOmics ETags とは異なります。説明メタデータとオブジェクトの詳細については、Amazon S3 Object API ドキュメントを参照してください。Amazon S3 ETag 値は、読み取りセットのアクティベーションサイクルごとに変更される可能性があり、それを使用してファイルの読み取りを検証できます。ただし、ファイルのライフサイクル中にファイル ID の検証に使用する Amazon S3 ETag 値は整合性が保たれないため、

キャッシュしないでください。対照的に、HealthOmics ETag は、リードセットのライフサイクルを通じて一貫性を維持します。

HealthOmics が ETags を計算する方法

ETag は、取り込まれたファイルコンテンツのハッシュから生成されます。ETag アルゴリズムファミリーはデフォルトで MD5up に設定されていますが、シーケンスストアの作成時に異なる方法で設定できます。ETag が計算されると、アルゴリズムと計算されたハッシュが読み取りセットに追加されます。ファイルタイプでサポートされている MD5 アルゴリズムは次のとおりです。

- FASTQ_MD5up – 非圧縮で完全な FASTQ リードセットソースの MD5 ハッシュを計算します。
- BAM_MD5up – SAM で表される非圧縮 BAM または uBAM リードセットソースのアラインメントセクションの MD5 ハッシュを、利用可能な場合はリンクされたリファレンスに基づいて計算します。
- CRAM_MD5up – リンクされたリファレンスに基づいて、SAM で表される非圧縮 CRAM リードセットソースのアライメントセクションの MD5 ハッシュを計算します。

> **Note**
>
> MD5 ハッシュは衝突に対して脆弱であることが知られています。このため、2 つの異なるファイルは、既知の衝突を悪用するために製造された場合、同じ ETag を持つ可能性があります。

SHA256 ファミリーでは、次のアルゴリズムがサポートされています。アルゴリズムは次のように計算されます。

- FASTQ_SHA256up – 非圧縮で完全な FASTQ リードセットソースの SHA-256 ハッシュを計算します。
- BAM_SHA256up – SAM で表される非圧縮 BAM または uBAM リードセットソースのアライメントセクションの SHA-256 ハッシュが使用可能な場合は、リンクされたリファレンスに基づいて計算します。
- CRAM_SHA256up – リンクされたリファレンスに基づいて、SAM で表される非圧縮 CRAM リードセットソースのアライメントセクションの SHA-256 ハッシュを計算します。

SHA512 ファミリーでは、次のアルゴリズムがサポートされています。アルゴリズムは次のように計算されます。

- FASTQ_SHA512up – 非圧縮で完全な FASTQ リードセットソースの SHA-512 ハッシュを計算します。
- BAM_SHA512up – SAM で表される非圧縮 BAM または uBAM リードセットソースのアライメントセクションの SHA-512 ハッシュを、利用可能な場合はリンクされたリファレンスに基づいて計算します。
- CRAM_SHA512up – リンクされたリファレンスに基づいて、SAM で表される非圧縮 CRAM リードセットソースのアライメントセクションの SHA-512 ハッシュを計算します。

HealthOmics リファレンスストアの作成

HealthOmics のリファレンスストアは、リファレンスゲノムを保存するためのデータストアです。各 AWS アカウント およびリージョンに 1 つのリファレンスストアを持つことができます。コンソールまたは CLI を使用してリファレンスストアを作成できます。

トピック

コンソールを使用したリファレンスストアの作成

参照ストアを作成するには

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、HealthOmics の開始方法を選択します。
3. Genomics データストレージオプションから参照ゲノムを選択します。
4. 以前にインポートした参照ゲノムを選択するか、新しい参照ゲノムをインポートできます。参照ゲノムをインポートしていない場合は、右上の「参照ゲノムのインポート」を選択します。
5. リファレンスゲノムのインポートジョブの作成ページで、クイック作成または手動作成オプションを選択してリファレンスストアを作成し、次の情報を指定します。

   - 参照ゲノム名 - このストアの一意の名前。
   - 説明 (オプション) - このリファレンスストアの説明。

- IAM ロール - 参照ゲノムにアクセスできるロールを選択します。
- Amazon S3 からのリファレンス - Amazon S3 バケット内のリファレンスシーケンスファイルを選択します。
- タグ (オプション) - このリファレンスストアには最大 50 個のタグを指定します。

CLI を使用したリファレンスストアの作成

次の例は、 を使用してリファレンスストアを作成する方法を示しています AWS CLI。 AWS リージョンごとに 1 つのリファレンスストアを持つことができます。

リファレンスストアは、拡張子 `.fasta`、、`.fa`、`.fas`、、、`.fsa`、`.faa.fna`、、、`.ffn`、`.frn.mpfa`、 `.seq`の FASTA ファイルのストレージをサポートしています`.txt`。これらの拡張機能`bgzip`のバージョンもサポートされています。

次の例では、 をリファレンスストアに選択した名前*`reference store name`*に置き換えます。

```
aws omics create-reference-store --name "reference store name"
```

リファレンスストア ID と名前、ARN、およびリファレンスストアが作成された時刻のタイムスタンプを含む JSON レスポンスを受け取ります。

```
{
    "id": "3242349265",
    "arn": "arn:aws:omics:us-west-2:555555555555:referenceStore/3242349265",
    "name": "MyReferenceStore",
    "creationTime": "2022-07-01T20:58:42.878Z"
}
```

リファレンスストア ID は追加の AWS CLI コマンドで使用できます。次の例に示すように、list-reference-stores コマンドを使用して、アカウントにリンクされたリファレンスストア IDs のリストを取得できます。

```
aws omics list-reference-stores
```

これに応じて、新しく作成したリファレンスストアの名前を受け取ります。

```
{
    "referenceStores": [
```

```
        {
              "id": "3242349265",
              "arn": "arn:aws:omics:us-west-2:555555555555:referenceStore/3242349265",
              "name": "MyReferenceStore",
             "creationTime": "2022-07-01T20:58:42.878Z"
         }
     ]
}
```

リファレンスストアを作成したら、インポートジョブを作成してゲノムリファレンスファイルをロードできます。そのためには、 を使用するか、IAM ロールを作成してデータにアクセスする必要があります。以下は、ポリシーの例です。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetBucketLocation"

            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1",
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
        }
    ]
}
```

次の例のような信頼ポリシーも必要です。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                   "omics.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

参照ゲノムをインポートできるようになりました。この例では、Genome Reference Consortium Human Build 38 (hg38) を使用しています。これはオープンアクセスであり、 のオープンデータのレジストリ AWSから入手できます。このデータをホストするバケットは、米国東部 (オハイオ) に基づいています。他の AWS リージョンでバケットを使用するには、リージョンでホストされている Amazon S3 バケットにデータをコピーします。次の AWS CLI コマンドを使用して、ゲノムを Amazon S3 バケットにコピーします。

```
aws s3 cp s3://broad-references/hg38/v0/Homo_sapiens_assembly38.fasta s3://amzn-s3-
demo-bucket
```

その後、インポートジョブを開始できます。*reference store ID*、*role ARN*、 を独自の入力*source file path*に置き換えます。

```
aws omics start-reference-import-job --reference-store-id reference store ID --role-
arn role ARN --sources source file path
```

データがインポートされると、JSON で次のレスポンスを受け取ります。

```
{
        "id": "7252016478",
        "referenceStoreId": "3242349265",
        "roleArn": "arn:aws:iam::111122223333:role/OmicsReferenceImport",
        "status": "CREATED",
        "creationTime": "2022-07-01T21:15:13.727Z"
}
```

次のコマンドを使用して、ジョブのステータスをモニタリングできます。次の例では、 *reference store ID*と *job ID*を、詳細を確認するリファレンスストア ID とジョブ ID に置き換えます。

```
aws omics get-reference-import-job --reference-store-id reference store ID --id job ID
```

レスポンスでは、そのリファレンスストアの詳細とそのステータスを含むレスポンスを受け取ります。

```
{
    "id": "7252016478",
    "referenceStoreId": "3242349265",
    "roleArn": "arn:aws:iam::555555555555:role/OmicsReferenceImport",
    "status": "RUNNING",
    "creationTime": "2022-07-01T21:15:13.727Z",
    "sources": [
        {
            "sourceFile": "s3://amzn-s3-demo-bucket/Homo_sapiens_assembly38.fasta",
            "status": "IN_PROGRESS",
            "name": "MyReference"
        }
    ]
}
```

インポートされたリファレンスを見つけるには、リファレンスを一覧表示し、リファレンス名に基づいてフィルタリングします。をリファレンスストア ID *reference store ID*に置き換え、オプションのフィルターを追加してリストを絞り込みます。

```
aws omics list-references --reference-store-id reference store ID --filter
 name=MyReference
```

応答として、次の情報を受け取ります。

```
{
    "references": [
        {
            "id": "1234567890",
            "arn": "arn:aws:omics:us-west-2:555555555555:referenceStore/1234567890/
reference/1234567890",
            "referenceStoreId": "12345678",
            "md5": "7ff134953dcca8c8997453bbb80b6b5e",
            "status": "ACTIVE",
            "name": "MyReference",
            "creationTime": "2022-07-02T00:15:19.787Z",
```

```
            "updateTime": "2022-07-02T00:15:19.787Z"
        }
    ]
}
```

リファレンスメタデータの詳細については、get-reference-metadata API オペレーションを使用します。次の例では、 をリファレンスストア ID *reference store ID*に置き換え、 を詳細を確認するリファレンス ID *reference ID*に置き換えます。

```
aws omics get-reference-metadata --reference-store-id reference store ID --id reference
 ID
```

応答として以下の情報を受け取ります。

```
{
    "id": "1234567890",
    "arn": "arn:aws:omics:us-west-2:555555555555:referenceStore/referencestoreID/
reference/referenceID",
    "referenceStoreId": "1234567890",
    "md5": "7ff134953dcca8c8997453bbb80b6b5e",
    "status": "ACTIVE",
    "name": "MyReference",
    "creationTime": "2022-07-02T00:15:19.787Z",
    "updateTime": "2022-07-02T00:15:19.787Z",
    "files": {
        "source": {
            "totalParts": 31,
            "partSize": 104857600,
            "contentLength": 3249912778
        },
        "index": {
            "totalParts": 1,
            "partSize": 104857600,
            "contentLength": 160928
        }
    }
}
```

get-reference を使用して、リファレンスファイルの一部をダウンロードすることもできます。次の例では、 をリファレンスストア ID *reference store ID*に、 をダウンロード元のリファレンス ID *reference ID* に置き換えます。

```
aws omics get-reference --reference-store-id reference store ID --id reference ID --
part-number 1 outfile.fa
```

HealthOmics シーケンスストアの作成

HealthOmics シーケンスストアは、 FASTQ (gzip のみ) および の非整列形式のゲノムファイルのストレージをサポートしていますuBAM。また、 BAMおよび のアラインされた形式もサポートしていますCRAM。

インポートされたファイルは読み取りセットとして保存されます。読み取りセットにタグを追加し、IAM ポリシーを使用して読み取りセットへのアクセスを制御できます。整列されたリードセットには、ゲノムシーケンスを整列させるために参照ゲノムが必要ですが、整列されていないリードセットではオプションです。

リードセットを保存するには、まずシーケンスストアを作成します。シーケンスストアを作成するときは、オプションの Amazon S3 バケットをフォールバックの場所と S3 アクセスログが保存される場所として指定できます。フォールバックの場所は、直接アップロード中に読み取りセットの作成に失敗したファイルを保存するために使用されます。フォールバックロケーションは、2023 年 5 月 15 日以降に作成されたシーケンスストアで利用できます。シーケンスストアを作成するときに、フォールバックの場所を指定します。

最大 5 つのリードセットタグキーを指定できます。これらのキーのいずれかに一致するタグキーを使用して読み取りセットを作成または更新すると、読み取りセットタグは対応する Amazon S3 オブジェクトに伝達されます。HealthOmics によって作成されたシステムタグは、デフォルトで伝播されます。

トピック

- コンソールを使用したシーケンスストアの作成
- CLI を使用したシーケンスストアの作成
- シーケンスストアの更新
- シーケンスストアの読み取りセットタグの更新
- ゲノムファイルのインポート

コンソールを使用したシーケンスストアの作成

シーケンスストアを作成するには

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、シーケンスストアを選択します。
3. シーケンスストアの作成ページで、次の情報を入力します。
   - シーケンスストア名 - このストアの一意の名前。
   - 説明 (オプション) - このシーケンスストアの説明。
4. S3 のフォールバックの場所には、Amazon S3 の場所を指定します。HealthOmics はフォールバックの場所を使用して、直接アップロード中に読み取りセットの作成に失敗したファイルを保存します。HealthOmics サービスに Amazon S3 フォールバックロケーションへの書き込みアクセスを許可する必要があります。ポリシーの例についてはフォールバックの場所を設定するを参照してください。

   フォールバックロケーションは、2023 年 5 月 16 日より前に作成されたシーケンスストアでは使用できません。
5. (オプション) S3 伝達のリードセットタグキーの場合、最大 5 つのリードセットキーを入力して、リードセットから基盤となる S3 オブジェクトに伝達できます。読み取りセットから S3 オブジェクトにタグを伝播することで、タグやエンドユーザーに基づいて S3 アクセス許可を付与し、Amazon S3 getObjectTagging API オペレーションを通じて伝播されたタグを表示できます。

   a. テキストボックスに 1 つのキー値を入力します。コンソールは、次のキーを追加する新しいテキストボックスを作成します。
   b. (オプション) 削除 を選択して、すべてのキーを削除します。
6. データ暗号化で、データ暗号化を が所有および管理するか、カスタマー管理の CMK AWS を使用するかを選択します。
7. (オプション) S3 データアクセスで、Amazon S3 経由でシーケンスストアにアクセスするための新しいロールとポリシーを作成するかどうかを選択します。
8. (オプション) S3 アクセスログ記録では、Amazon S3 がアクセスログレコードを収集`Enabled`するかどうかを選択します。

   S3 のアクセスログの場所には、ログを保存する Amazon S3 の場所を指定します。このフィールドは、S3 アクセスログ記録を有効にした場合にのみ表示されます。

9. タグ (オプション) - このシーケンスストアには最大 50 個のタグを指定します。これらのタグは、読み取りセットのインポート/タグ更新中に設定される読み取りセットタグとは別のものです。

ストアを作成すると、 の準備が整いますゲノムファイルのインポート。

CLI を使用したシーケンスストアの作成

次の例では、 をシーケンスストアに選択した名前*sequence store name*に置き換えます。

```
aws omics create-sequence-store --name sequence store name --fallback-location "s3://
amzn-s3-demo-bucket"
```

JSON で次のレスポンスを受け取ります。これには、新しく作成したシーケンスストアの ID 番号が含まれます。

```
{
    "id": "3936421177",
    "arn": "arn:aws:omics:us-west-2:111122223333:sequenceStore/3936421177",
    "name": "sequence_store_example_name",
    "creationTime": "2022-07-13T20:09:26.038Z"
    "fallbackLocation" : "s3://amzn-s3-demo-bucket"
}
```

以下に示すように、list-sequence-stores コマンドを使用して、アカウントに関連付けられているすべてのシーケンスストアを表示することもできます。

```
aws omics list-sequence-stores
```

次のレスポンスが表示されます。

```
{
    "sequenceStores": [
        {
            "arn": "arn:aws:omics:us-west-2:111122223333:sequenceStore/3936421177",
            "id": "3936421177",
            "name": "MySequenceStore",
            "creationTime": "2022-07-13T20:09:26.038Z",
            "updatedTime": "2024-09-13T04:11:31.242Z",
```

```
            "fallbackLocation" : "s3://amzn-s3-demo-bucket",
            "status": "Active"
        }
    ]
}
```

get-sequence-store を使用すると、次の例に示すように、ID を使用してシーケンスストアの詳細を確認できます。

```
aws omics get-sequence-store --id sequence store ID
```

次のレスポンスが表示されます。

```
{
  "arn": "arn:aws:omics:us-west-2:123456789012:sequenceStore/sequencestoreID",
  "creationTime": "2024-01-12T04:45:29.857Z",
  "updatedTime": "2024-09-13T04:11:31.242Z",
  "description": null,
  "fallbackLocation": null,
  "id": "2015356892",
  "name": "MySequenceStore",
  "s3Access": {
      "s3AccessPointArn": "arn:aws:s3:us-
west-2:123456789012:accesspoint/592761533288-2015356892",
      "s3Uri": "s3://592761533288-2015356892-ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
s3alias/592761533288/sequenceStore/2015356892/",
      "accessLogLocation": "s3://IAD-seq-store-log/2015356892/"
  },
  "sseConfig": {
      "keyArn": "arn:aws:kms:us-west-2:123456789012:key/eb2b30f5-635d-4b6d-b0f9-
d3889fe0e648",
      "type": "KMS"
  },
  "status": "Active",
  "statusMessage": null,
  "setTagsToSync": ["withdrawn","protocol"],
}
```

作成後、複数のストアパラメータを更新することもできます。これは、 コンソールまたは API updateSequenceStoreオペレーションを使用して実行できます。

シーケンスストアの更新

シーケンスストアを更新するには、次の手順に従います。

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、シーケンスストアを選択します。
3. 更新するシーケンスストアを選択します。
4. 詳細パネルで、編集を選択します。
5. 詳細の編集ページで、次のフィールドを更新できます。

   - シーケンスストア名 - このストアの一意の名前。
   - 説明 - このシーケンスストアの説明。
   - S3 のフォールバックの場所、Amazon S3 の場所を指定します。HealthOmics はフォールバックの場所を使用して、直接アップロード中に読み取りセットの作成に失敗したファイルを保存します。
   - S3 伝達のリードセットタグキーは、Amazon S3 に伝達するために最大 5 つのリードセットキーを入力できます。
   - (オプション) S3 アクセスログ記録では、Amazon S3 がアクセスログレコードを収集`Enabled`するかどうかを選択します。

     S3 のアクセスログの場所には、ログを保存する Amazon S3 の場所を指定します。このフィールドは、S3 アクセスログ記録を有効にした場合にのみ表示されます。
   - タグ (オプション) - このシーケンスストアには最大 50 個のタグを指定します。

シーケンスストアの読み取りセットタグの更新

シーケンスストアの読み取りセットタグやその他のフィールドを更新するには、次の手順に従います。

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、シーケンスストアを選択します。
3. 更新するシーケンスストアを選択します。
4. [詳細] タブを選択します。
5. [編集] を選択します。
6. 必要に応じて、新しいリードセットタグを追加するか、既存のタグを削除します。

7. 必要に応じて、名前、説明、フォールバックの場所、または S3 データアクセスを更新します。
8. [Save changes] (変更の保存) をクリックします。

ゲノムファイルのインポート

ゲノムファイルをシーケンスストアにインポートするには、次の手順に従います。

ゲノミクスファイルをインポートするには

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、シーケンスストアを選択します。
3. シーケンスストアページで、ファイルをインポートするシーケンスストアを選択します。
4. 個々のシーケンスストアページで、ゲノムファイルのインポートを選択します。
5. インポートの詳細の指定ページで、次の情報を入力します。
   - IAM ロール - Amazon S3 のゲノムファイルにアクセスできる IAM ロール。
   - 参照ゲノム - このゲノムデータの参照ゲノム。
6. インポートマニフェストを指定ページで、次の情報マニフェストファイルを指定します。マニフェストファイルは、ゲノミクスデータの重要な情報を記述する JSON または YAML ファイルです。マニフェストファイルの詳細については、「」を参照してくださいHealthOmics シーケンスストアへの読み取りセットのインポート。
7. インポートジョブの作成をクリックします。

HealthOmics リファレンスストアとシーケンスストアの削除

リファレンスストアとシーケンスストアの両方を削除できます。シーケンスストアは、リードセットが含まれていない場合にのみ削除でき、リファレンスストアはリファレンスが含まれていない場合にのみ削除できます。シーケンスまたはリファレンスストアを削除すると、そのストアに関連付けられたタグも削除されます。

次の例は、 を使用してリファレンスストアを削除する方法を示しています AWS CLI。アクションが成功すると、レスポンスは受信されません。次の例では、 をリファレンスストア ID *reference store ID* に置き換えます。

```
aws omics delete-reference-store --id reference store ID
```

次の例は、シーケンスストアを削除する方法を示しています。アクションが成功した場合、レスポンスは受信されません。次の例では、 をシーケンスストア ID *sequence store ID* に置き換えます。

```
aws omics delete-sequence-store --id sequence store ID
```

次の例に示すように、リファレンスストアのリファレンスを削除することもできます。リファレンスは、読み取りセット、バリアントストア、または注釈ストアで使用されていない場合にのみ削除できます。次の例では、 をリファレンスストア ID *reference store ID*に置き換え、 を削除するリファレンスの ID *reference ID*に置き換えます。

```
aws omics delete-reference  --id reference ID --reference-store-id reference store ID
```

HealthOmics シーケンスストアへの読み取りセットのインポート

シーケンスストアを作成したら、インポートジョブを作成して、読み取りセットをデータストアにアップロードします。Amazon S3 バケットからファイルをアップロードすることも、同期 API オペレーションを使用して直接アップロードすることもできます。Amazon S3 バケットは、シーケンスストアと同じリージョンにある必要があります。

アライメントされたリードセットとアライメントされていないリードセットの任意の組み合わせをシーケンスストアにアップロードできますが、インポート内のリードセットのいずれかがアライメントされている場合は、参照ゲノムを含める必要があります。

リファレンスストアの作成に使用した IAM アクセスポリシーを再利用できます。

以下のトピックでは、シーケンスストアに読み取りセットをインポートし、インポートされたデータに関する情報を取得するために実行する主要なステップについて説明します。

トピック

- リードセットデータファイルをダウンロードする

Amazon S3 にファイルをアップロードする

次の例は、Amazon S3 バケットにファイルを移動する方法を示しています。

```
aws s3 cp s3://1000genomes/phase1/data/HG00100/alignment/
HG00100.chrom20.ILLUMINA.bwa.GBR.low_coverage.20101123.bam s3://your-bucket
aws s3 cp s3://1000genomes/phase3/data/HG00146/sequence_read/SRR233106_1.filt.fastq.gz
 s3://your-bucket
aws s3 cp s3://1000genomes/phase3/data/HG00146/sequence_read/SRR233106_2.filt.fastq.gz
 s3://your-bucket
aws s3 cp s3://1000genomes/data/HG00096/alignment/
HG00096.alt_bwamem_GRCh38DH.20150718.GBR.low_coverage.cram s3://your-bucket
aws s3 cp s3://gatk-test-data/wgs_ubam/NA12878_20k/NA12878_A.bam s3://your-bucket
```

この例CRAMで使用されるサンプル BAMと には、異なるゲノム参照 Hg19と が必要ですHg38。詳細について、またはこれらのリファレンスにアクセスするには、「 のオープンデータレジストリ」の「The Broad Genome References」を参照してください AWS。

マニフェストファイルの作成

また、インポートジョブをモデル化するには、JSON でマニフェストファイルを作成する必要があります `import.json` (次の例を参照) 。コンソールでシーケンスストアを作成する場合、 `sequenceStoreId`または を指定する必要がないため`roleARN`、マニフェストファイルは`sources`入力で始まります。

API manifest

次の例では、 API を使用して 3 つの読み取りセットをインポートします。1 つは FASTQ、1 つは BAM、1 つは ですCRAM。

```
{
  "sequenceStoreId": "3936421177",
  "roleArn": "arn:aws:iam::555555555555:role/OmicsImport",
  "sources":
  [
      {
          "sourceFiles":
          {
```

```
            "source1": "s3://amzn-s3-demo-bucket/
HG00100.chrom20.ILLUMINA.bwa.GBR.low_coverage.20101123.bam"
        },
        "sourceFileType": "BAM",
        "subjectId": "mySubject",
        "sampleId": "mySample",
        "referenceArn": "arn:aws:omics:us-
west-2:555555555555:referenceStore/0123456789/reference/0000000001",
        "name": "HG00100",
        "description": "BAM for HG00100",
        "generatedFrom": "1000 Genomes"
    },
    {
        "sourceFiles":
        {
            "source1": "s3://amzn-s3-demo-bucket/SRR233106_1.filt.fastq.gz",
            "source2": "s3://amzn-s3-demo-bucket/SRR233106_2.filt.fastq.gz"
        },
        "sourceFileType": "FASTQ",
        "subjectId": "mySubject",
        "sampleId": "mySample",
        // NOTE: there is no reference arn required here
        "name": "HG00146",
        "description": "FASTQ for HG00146",
        "generatedFrom": "1000 Genomes"
    },
    {
        "sourceFiles":
        {
            "source1": "s3://amzn-s3-demo-bucket/
HG00096.alt_bwamem_GRCh38DH.20150718.GBR.low_coverage.cram"
        },
        "sourceFileType": "CRAM",
        "subjectId": "mySubject",
        "sampleId": "mySample",
        "referenceArn": "arn:aws:omics:us-
west-2:555555555555:referenceStore/0123456789/reference/0000000001",
        "name": "HG00096",
        "description": "CRAM for HG00096",
        "generatedFrom": "1000 Genomes"
    },
    {
        "sourceFiles":
        {
```

```
            "source1": "s3://amzn-s3-demo-bucket/NA12878_A.bam"
        },
        "sourceFileType": "UBAM",
        "subjectId": "mySubject",
        "sampleId": "mySample",
        // NOTE: there is no reference arn required here
        "name": "NA12878_A",
        "description": "uBAM for NA12878",
        "generatedFrom": "GATK Test Data"
    }
  ]
}
```

Console manifest

このサンプルコードは、 コンソールを使用して単一の読み取りセットをインポートするために使用されます。

```
[
  {
      "sourceFiles":
      {
          "source1": "s3://amzn-s3-demo-bucket/
HG00100.chrom20.ILLUMINA.bwa.GBR.low_coverage.20101123.bam"
      },
      "sourceFileType": "BAM",
      "subjectId": "mySubject",
      "sampleId": "mySample",
      "name": "HG00100",
      "description": "BAM for HG00100",
      "generatedFrom": "1000 Genomes"
  },
  {
      "sourceFiles":
      {
          "source1": "s3://amzn-s3-demo-bucket/SRR233106_1.filt.fastq.gz",
          "source2": "s3://amzn-s3-demo-bucket/SRR233106_2.filt.fastq.gz"
      },
      "sourceFileType": "FASTQ",
      "subjectId": "mySubject",
      "sampleId": "mySample",
      "name": "HG00146",
      "description": "FASTQ for HG00146",
```

```
      "generatedFrom": "1000 Genomes"
  },
  {
      "sourceFiles":
      {
          "source1": "s3://your-bucket/
HG00096.alt_bwamem_GRCh38DH.20150718.GBR.low_coverage.cram"
      },
      "sourceFileType": "CRAM",
      "subjectId": "mySubject",
      "sampleId": "mySample",
      "name": "HG00096",
      "description": "CRAM for HG00096",
      "generatedFrom": "1000 Genomes"
  },
  {
      "sourceFiles":
      {
          "source1": "s3://amzn-s3-demo-bucket/NA12878_A.bam"
      },
      "sourceFileType": "UBAM",
      "subjectId": "mySubject",
      "sampleId": "mySample",
      "name": "NA12878_A",
      "description": "uBAM for NA12878",
      "generatedFrom": "GATK Test Data"
  }
]
```

または、マニフェストファイルを YAML 形式でアップロードすることもできます。

インポートジョブの開始

インポートジョブを開始するには、次の AWS CLI コマンドを使用します。

```
aws omics start-read-set-import-job --cli-input-json file://import.json
```

ジョブが正常に作成されたことを示す次のレスポンスが表示されます。

```
{
  "id": "3660451514",
  "sequenceStoreId": "3936421177",
```

```
  "roleArn": "arn:aws:iam::111122223333:role/OmicsImport",
  "status": "CREATED",
  "creationTime": "2022-07-13T22:14:59.309Z"
}
```

インポートジョブをモニタリングする

インポートジョブが開始されたら、次のコマンドを使用してその進行状況をモニタリングできます。次の例では、 をシーケンスストア ID *sequence store id*に置き換え、 をインポート ID *job import ID* に置き換えます。

```
aws omics get-read-set-import-job --sequence-store-id sequence store id --id job import
 ID
```

以下は、指定されたシーケンスストア ID に関連付けられているすべてのインポートジョブのステータスを示しています。

```
{
  "id": "1234567890",
  "sequenceStoreId": "1234567890",
  "roleArn": "arn:aws:iam::111122223333:role/OmicsImport",
  "status": "RUNNING",
  "statusMessage": "The job is currently in progress.",
  "creationTime": "2022-07-13T22:14:59.309Z",
  "sources": [
      {
          "sourceFiles":
          {
              "source1": "s3://amzn-s3-demo-bucket/
HG00100.chrom20.ILLUMINA.bwa.GBR.low_coverage.20101123.bam"
          },
          "sourceFileType": "BAM",
          "status": "IN_PROGRESS",
          "statusMessage": "The job is currently in progress."
          "subjectId": "mySubject",
          "sampleId": "mySample",
          "referenceArn": "arn:aws:omics:us-
west-2:111122223333:referenceStore/3242349265/reference/8625408453",
          "name": "HG00100",
          "description": "BAM for HG00100",
          "generatedFrom": "1000 Genomes",
          "readSetID": "1234567890"
```

```
      },
      {
          "sourceFiles":
          {
              "source1": "s3://amzn-s3-demo-bucket/SRR233106_1.filt.fastq.gz",
              "source2": "s3://amzn-s3-demo-bucket/SRR233106_2.filt.fastq.gz"
          },
          "sourceFileType": "FASTQ",
          "status": "IN_PROGRESS",
          "statusMessage": "The job is currently in progress."
          "subjectId": "mySubject",
          "sampleId": "mySample",
          "name": "HG00146",
          "description": "FASTQ for HG00146",
          "generatedFrom": "1000 Genomes",
          "readSetID": "1234567890"
      },
      {
          "sourceFiles":
          {
              "source1": "s3://amzn-s3-demo-bucket/
HG00096.alt_bwamem_GRCh38DH.20150718.GBR.low_coverage.cram"
          },
          "sourceFileType": "CRAM",
          "status": "IN_PROGRESS",
          "statusMessage": "The job is currently in progress."
          "subjectId": "mySubject",
          "sampleId": "mySample",
          "referenceArn": "arn:aws:omics:us-
west-2:111122223333:referenceStore/3242349265/reference/1234568870",
          "name": "HG00096",
          "description": "CRAM for HG00096",
          "generatedFrom": "1000 Genomes",
          "readSetID": "1234567890"
      },
      {
          "sourceFiles":
          {
              "source1": "s3://amzn-s3-demo-bucket/NA12878_A.bam"
          },
          "sourceFileType": "UBAM",
          "status": "IN_PROGRESS",
          "statusMessage": "The job is currently in progress."
          "subjectId": "mySubject",
```

```
            "sampleId": "mySample",
            "name": "NA12878_A",
            "description": "uBAM for NA12878",
            "generatedFrom": "GATK Test Data",
            "readSetID": "1234567890"
        }
    ]
}
```

インポートされたシーケンスファイルを検索する

ジョブが完了したら、list-read-sets API オペレーションを使用して、インポートされたシーケンスファイルを検索できます。次の例では、 をシーケンスストア ID *sequence store id* に置き換えます。

```
aws omics list-read-sets --sequence-store-id sequence store id
```

次のレスポンスが表示されます。

```
{
    "readSets": [
        {
            "id": "0000000001",
            "arn": "arn:aws:omics:us-west-2:111122223333:sequenceStore/01234567890/
readSet/0000000001",
            "sequenceStoreId": "1234567890",
            "subjectId": "mySubject",
            "sampleId": "mySample",
            "status": "ACTIVE",
            "name": "HG00100",
            "description": "BAM for HG00100",
            "referenceArn": "arn:aws:omics:us-
west-2:111122223333:referenceStore/01234567890/reference/0000000001",
            "fileType": "BAM",
            "sequenceInformation": {
                "totalReadCount": 9194,
                "totalBaseCount": 928594,
                "generatedFrom": "1000 Genomes",
                "alignment": "ALIGNED"
            },
            "creationTime": "2022-07-13T23:25:20Z"
            "creationType": "IMPORT",
```

```
            "etag": {
                "algorithm": "BAM_MD5up",
                "source1": "d1d65429212d61d115bb19f510d4bd02"
            }
        },
        {
            "id": "0000000002",
            "arn": "arn:aws:omics:us-west-2:111122223333:sequenceStore/0123456789/
readSet/0000000002",
            "sequenceStoreId": "0123456789",
            "subjectId": "mySubject",
            "sampleId": "mySample",
            "status": "ACTIVE",
            "name": "HG00146",
            "description": "FASTQ for HG00146",
            "fileType": "FASTQ",
            "sequenceInformation": {
                "totalReadCount": 8000000,
                "totalBaseCount": 1184000000,
                "generatedFrom": "1000 Genomes",
                "alignment": "UNALIGNED"
            },
            "creationTime": "2022-07-13T23:26:43Z"
            "creationType": "IMPORT",
            "etag": {
                "algorithm": "FASTQ_MD5up",
                "source1": "ca78f685c26e7cc2bf3e28e3ec4d49cd"
            }
        },
        {
            "id": "0000000003",
            "arn": "arn:aws:omics:us-west-2:111122223333:sequenceStore/0123456789/
readSet/0000000003",
            "sequenceStoreId": "0123456789",
            "subjectId": "mySubject",
            "sampleId": "mySample",
            "status": "ACTIVE",
            "name": "HG00096",
            "description": "CRAM for HG00096",
            "referenceArn": "arn:aws:omics:us-
west-2:111122223333:referenceStore/0123456789/reference/0000000001",
            "fileType": "CRAM",
            "sequenceInformation": {
                "totalReadCount": 85466534,
```

```
                "totalBaseCount": 24000004881,
                "generatedFrom": "1000 Genomes",
                "alignment": "ALIGNED"
            },
            "creationTime": "2022-07-13T23:30:41Z"
            "creationType": "IMPORT",
            "etag": {
                "algorithm": "CRAM_MD5up",
                "source1": "66817940f3025a760e6da4652f3e927e"
            }
        },
        {
            "id": "0000000004",
            "arn": "arn:aws:omics:us-west-2:111122223333:sequenceStore/0123456789/
readSet/0000000004",
            "sequenceStoreId": "0123456789",
            "subjectId": "mySubject",
            "sampleId": "mySample",
            "status": "ACTIVE",
            "name": "NA12878_A",
            "description": "uBAM for NA12878",
            "fileType": "UBAM",
            "sequenceInformation": {
                "totalReadCount": 20000,
                "totalBaseCount": 5000000,
                "generatedFrom": "GATK Test Data",
                "alignment": "ALIGNED"
            },
            "creationTime": "2022-07-13T23:30:41Z"
            "creationType": "IMPORT",
            "etag": {
                "algorithm": "BAM_MD5up",
                "source1": "640eb686263e9f63bcda12c35b84f5c7"
            }
        }
    ]
}
```


読み取りセットの詳細を取得する

読み取りセットの詳細については、GetReadSetMetadata API オペレーションを使用します。次の例では、 をシーケンスストア ID *sequence store id*に置き換え、 を読み取りセット ID *read set id* に置き換えます。

```
aws omics get-read-set-metadata --sequence-store-id sequence store id --id read set id
```

次のレスポンスが表示されます。


```
{
"arn": "arn:aws:omics:us-west-2:123456789012:sequenceStore/2015356892/
readSet/9515444019",
"creationTime": "2024-01-12T04:50:33.548Z",
"creationType": "IMPORT",
"creationJobId": "33222111",
"description": null,
"etag": {
  "algorithm": "FASTQ_MD5up",
  "source1": "00d0885ba3eeb211c8c84520d3fa26ec",
  "source2": "00d0885ba3eeb211c8c84520d3fa26ec"
},
"fileType": "FASTQ",
"files": {
  "index": null,
  "source1": {
    "contentLength": 10818,
    "partSize": 104857600,
    "s3Access": {
      "s3Uri": "s3://accountID-sequence store ID-ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
s3alias/592761533288/sequenceStore/2015356892/readSet/9515444019/
import_source1.fastq.gz"
},
    "totalParts": 1
  },
  "source2": {
    "contentLength": 10818,
    "partSize": 104857600,
    "s3Access": {
      "s3Uri": "s3://accountID-sequence store ID-ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
s3alias/592761533288/sequenceStore/2015356892/readSet/9515444019/
import_source1.fastq.gz"
    },
    "totalParts": 1
  }
},
"id": "9515444019",
"name": "paired-fastq-import",
```

```
"sampleId": "sampleId-paired-fastq-import",
"sequenceInformation": {
  "alignment": "UNALIGNED",
  "generatedFrom": null,
  "totalBaseCount": 30000,
  "totalReadCount": 200
},
"sequenceStoreId": "2015356892",
"status": "ACTIVE",
"statusMessage": null,
"subjectId": "subjectId-paired-fastq-import"
}
```


リードセットデータファイルをダウンロードする

Amazon S3 API オペレーションを使用して、アクティブな読み取りセットのオブジェクトにアクセスできます。 GetObjectオブジェクトの URI は GetReadSetMetadata API レスポンスで返されます。詳細については、「Amazon S3 URIs を使用した HealthOmics リードセットへのアクセス」を参照してください。

または、HealthOmics GetReadSet API オペレーションを使用します。個々のパートをダウンロードすることで、 GetReadSetを使用して並行してダウンロードできます。これらのパートは Amazon S3 のパートと似ています。以下は、読み取りセットからパート 1 をダウンロードする方法の例です。次の例では、 をシーケンスストア ID *sequence store id*に置き換え、 を読み取りセット ID *read set id* に置き換えます。

```
aws omics get-read-set --sequence-store-id sequence store id --id read set id  --part-
number 1 outfile.bam
```

HealthOmics Transfer Manager を使用して、HealthOmics リファレンスまたはリードセットのファイルをダウンロードすることもできます。HealthOmics Transfer Manager はこちらからダウンロードできますhttps://pypi.org/project/amazon-omics-tools/。Transfer Manager の使用と設定の詳細については、この GitHub リポジトリを参照してください。

HealthOmics シーケンスストアへの直接アップロード

HealthOmics Transfer Manager を使用して、シーケンスストアにファイルを追加することをお勧めします。Transfer Manager の使用の詳細については、この GitHub リポジトリを参照してください。直接アップロード API オペレーションを使用して、読み取りセットをシーケンスストアに直接アップロードすることもできます。

直接アップロードの読み取りセットは、 PROCESSING_UPLOAD 状態で最初に存在します。つまり、ファイルパーツは現在アップロード中であり、読み取りセットメタデータにアクセスできます。パートがアップロードされ、チェックサムが検証されると、読み込みセットは になり、インポートされた読み込みセットと同じようにACTIVE動作します。

直接アップロードが失敗した場合、読み取りセットのステータスは と表示されますUPLOAD_FAILED。アップロードに失敗したファイルのフォールバックの場所として Amazon S3 バケットを設定できます。フォールバックロケーションは、2023 年 5 月 15 日以降に作成されたシーケンスストアで利用できます。

トピック

- を使用してシーケンスストアに直接アップロードする AWS CLI
- フォールバックの場所を設定する

を使用してシーケンスストアに直接アップロードする AWS CLI

開始するには、マルチパートアップロードを開始します。これを行うには AWS CLI、次の例に示すように、 を使用します。

AWS CLI コマンドを使用して直接アップロードするには

1. 次の例に示すように、データを分離してパートを作成します。

```
split -b 100MiB SRR233106_1.filt.fastq.gz source1_part_
```

2. ソースファイルがパートになったら、次の例に示すように、マルチパートリードセットアップロードを作成します。*sequence store ID* とその他のパラメータをシーケンスストア ID とその他の値に置き換えます。

```
aws omics create-multipart-read-set-upload \
--sequence-store-id sequence store ID \
--name upload name \
--source-file-type FASTQ \
--subject-id subject ID \
--sample-id sample ID \
--description "FASTQ for HG00146" "description of upload" \
--generated-from "1000 Genomes""source of imported files"
```

レスポンスで uploadIDおよびその他のメタデータを取得します。アップロードプロセスの次のステップuploadIDには、 を使用します。

```
{
"sequenceStoreId": "1504776472",
"uploadId": "7640892890",
"sourceFileType": "FASTQ",
"subjectId": "mySubject",
"sampleId": "mySample",
"generatedFrom": "1000 Genomes",
"name": "HG00146",
"description": "FASTQ for HG00146",
"creationTime": "2023-11-20T23:40:47.437522+00:00"
}
```

3. 読み取りセットをアップロードに追加します。ファイルが十分に小さい場合は、このステップを 1 回だけ実行する必要があります。大きなファイルの場合は、ファイルの各部分に対してこのステップを実行します。以前に使用したパート番号を使用して新しいパートをアップロードすると、以前にアップロードしたパートが上書きされます。

   次の例では、*sequence store ID*、*upload ID*、およびその他のパラメータを値に置き換えます。

```
aws omics upload-read-set-part \
--sequence-store-id sequence store ID \
--upload-id upload ID \
--part-source SOURCE1 \
--part-number part number \
--payload  source1/source1_part_aa.fastq.gz
```

   レスポンスは、アップロードされたファイルが意図したファイルと一致することを確認するために使用できる ID です。

```
{
"checksum": "984979b9928ae8d8622286c4a9cd8e99d964a22d59ed0f5722e1733eb280e635"
}
```

4. 必要に応じて、ファイルの一部のアップロードを続行します。読み取りセットがアップロードされたことを確認するには、以下に示すように listlist-read-set-upload-parts API オペレーションを

使用します。次の例では、*sequence store ID* 、、 *upload IDpart source*を独自の入力に置き換えます。

```
aws omics list-read-set-upload-parts \
 --sequence-store-id sequence store ID \
 --upload-id upload ID \
 --part-source SOURCE1
```

レスポンスは、読み取りセットの数、サイズ、および最後に更新された時刻のタイムスタンプを返します。

```
{
"parts": [
    {
        "partNumber": 1,
        "partSize": 104857600,
        "partSource": "SOURCE1",
        "checksum": "MVMQk+vB9C3Ge8ADHkbKq752n3BCUzyl41qEkqlOD5M=",
        "creationTime": "2023-11-20T23:58:03.500823+00:00",
        "lastUpdatedTime": "2023-11-20T23:58:03.500831+00:00"
    },
    {
        "partNumber": 2,
        "partSize": 104857600,
        "partSource": "SOURCE1",
        "checksum": "keZzVzJNChAqgOdZMvOmjBwrOPM0enPj1UAfs0nvRto=",
        "creationTime": "2023-11-21T00:02:03.813013+00:00",
        "lastUpdatedTime": "2023-11-21T00:02:03.813025+00:00"
    },
    {
        "partNumber": 3,
        "partSize": 100339539,
        "partSource": "SOURCE1",
        "checksum": "TBkNfMsaeDpXzEf3ldlbi0ipFDPaohKHyZ+LF1J4CHk=",
        "creationTime": "2023-11-21T00:09:11.705198+00:00",
        "lastUpdatedTime": "2023-11-21T00:09:11.705208+00:00"
    }
]
}
```

5. アクティブなマルチパートリードセットのアップロードをすべて表示するには、以下に示すように list-multipart-read-set-uploads を使用します。を独自のシーケンスストアの ID *sequence store ID*に置き換えます。

```
aws omics list-multipart-read-set-uploads --sequence-store-id
             sequence store ID
```

この API は、進行中のマルチパートリードセットアップロードのみを返します。取り込まれた読み取りセットが になった後ACTIVE、またはアップロードが失敗した場合、アップロードは list-multipart-read-set-uploads API へのレスポンスでは返されません。アクティブな読み取りセットを表示するには、list-read-sets API を使用します。list-multipart-read-set-uploads のレスポンスの例を以下に示します。

```
{
"uploads": [
    {
        "sequenceStoreId": "1234567890",
        "uploadId": "8749584421",
        "sourceFileType": "FASTQ",
        "subjectId": "mySubject",
        "sampleId": "mySample",
        "generatedFrom": "1000 Genomes",
        "name": "HG00146",
        "description": "FASTQ for HG00146",
        "creationTime": "2023-11-29T19:22:51.349298+00:00"
    },
    {
        "sequenceStoreId": "1234567890",
        "uploadId": "5290538638",
        "sourceFileType": "BAM",
        "subjectId": "mySubject",
        "sampleId": "mySample",
        "generatedFrom": "1000 Genomes",
        "referenceArn": "arn:aws:omics:us-
west-2:123456789012:referenceStore/8168613728/reference/2190697383",
        "name": "HG00146",
        "description": "BAM for HG00146",
        "creationTime": "2023-11-29T19:23:33.116516+00:00"
    },
    {
```

```
        "sequenceStoreId": "1234567890",
        "uploadId": "4174220862",
        "sourceFileType": "BAM",
        "subjectId": "mySubject",
        "sampleId": "mySample",
        "generatedFrom": "1000 Genomes",
        "referenceArn": "arn:aws:omics:us-
west-2:123456789012:referenceStore/8168613728/reference/2190697383",
        "name": "HG00147",
        "description": "BAM for HG00147",
        "creationTime": "2023-11-29T19:23:47.007866+00:00"
    }
]
}
```

6. ファイルのすべての部分をアップロードしたら、次の例に示すように、complete-multipart-read-set-upload を使用してアップロードプロセスを完了します。パートの *sequence store ID*、*upload ID*、および パラメータを独自の値に置き換えます。

```
aws omics complete-multipart-read-set-upload \
--sequence-store-id sequence store ID \
--upload-id upload ID \
--parts '[{"checksum":"gaCBQMe+rpCFZxLpoP6gydBoXaKKDA/
Vobh5zBDb4W4=","partNumber":1,"partSource":"SOURCE1"}]'
```

complete-multipart-read-set-upload のレスポンスは、インポートされたリードセットIDs です。

```
{
"readSetId": "0000000001"
}
```

7. アップロードを停止するには、アップロード ID で abort-multipart-read-set-upload を使用してアップロードプロセスを完了します。*sequence store ID* と を独自のパラメータ値*upload ID*に置き換えます。

```
aws omics abort-multipart-read-set-upload \
--sequence-store-id sequence store ID \
--upload-id upload ID
```

8. アップロードが完了したら、次に示すように get-read-set を使用して読み取りセットからデータを取得します。アップロードがまだ処理中の場合、get-read-set は制限されたメタデータを返

し、生成されたインデックスファイルは使用できなくなります。*sequence store ID* と他のパラメータを独自の入力に置き換えます。

```
aws omics get-read-set
 --sequence-store-id sequence store ID \
 --id read set ID \
 --file SOURCE1 \
 --part-number 1 myfile.fastq.gz
```

9. アップロードのステータスを含むメタデータを確認するには、get-read-set-metadata API オペレーションを使用します。

```
aws omics get-read-set-metadata --sequence-store-id sequence store ID --id read set
 ID
```

レスポンスには、ファイルタイプ、リファレンス ARN、ファイル数、シーケンスの長さなどのメタデータの詳細が含まれます。また、 ステータスも含まれます。可能なステータスは、PROCESSING_UPLOAD、ACTIVE、および ですUPLOAD_FAILED。

```
{
"id": "0000000001",
"arn": "arn:aws:omics:us-west-2:555555555555:sequenceStore/0123456789/
readSet/0000000001",
"sequenceStoreId": "0123456789",
"subjectId": "mySubject",
"sampleId": "mySample",
"status": "PROCESSING_UPLOAD",
"name": "HG00146",
"description": "FASTQ for HG00146",
"fileType": "FASTQ",
"creationTime": "2022-07-13T23:25:20Z",
"files": {
    "source1": {
        "totalParts": 5,
        "partSize": 123456789012,
        "contentLength": 6836725,

    },
    "source2": {
        "totalParts": 5,
        "partSize": 123456789056,
        "contentLength": 6836726
```

```
    }
},
'creationType": "UPLOAD"
}
```

フォールバックの場所を設定する

シーケンスストアを作成または更新するときに、アップロードに失敗したファイルのフォールバックの場所として Amazon S3 バケットを設定できます。これらの読み取りセットのファイル部分は、フォールバックの場所に転送されます。フォールバックロケーションは、2023 年 5 月 15 日以降に作成されたシーケンスストアで利用できます。

次の例に示すように、Amazon S3 バケットポリシーを作成して、Amazon S3 フォールバックロケーションへの書き込みアクセスを HealthOmics に付与します。 Amazon S3

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "omics.amazonaws.com"
    },
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
}
```

フォールバックログまたはアクセスログの Amazon S3 バケットがカスタマーマネージドキーを使用している場合は、キーポリシーに次のアクセス許可を追加します。

```
 {
    "Sid": "Allow use of key",
    "Effect": "Allow",
    "Principal": {
        "Service": "omics.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

HealthOmics リードセットを Amazon S3 バケットにエクスポートする

読み取りセットをバッチエクスポートジョブとして Amazon S3 バケットにエクスポートできます。そのためには、まず、次の IAM ポリシーの例のように、バケットへの書き込みアクセス権を持つ IAM ポリシーを作成します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1",
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    }
  ]
}
```

JSON

```
{
"Version": "2012-10-17",
"Statement": [
  {
      "Effect": "Allow",
      "Principal": {
          "Service": [
              "omics.amazonaws.com"
          ]
      },
      "Action": "sts:AssumeRole"
```

```
  }
]
}
```

IAM ポリシーが設定されたら、読み取りセットのエクスポートジョブを開始します。次の例は、start-read-set-export-job API オペレーションを使用してこれを行う方法を示しています。次の例では、、、、 *sequence store ID* *destination* などのすべてのパラメータを入力*role ARN*に置き換えます。

```
aws omics start-read-set-export-job
--sequence-store-id sequence store id \
--destination valid s3 uri \
--role-arn role ARN \
--sources readSetId=read set id_1 readSetId=read set id_2
```

オリジンシーケンスストアと送信先 Amazon S3 バケットに関する情報を含む次のレスポンスを受け取ります。

```
{
"id": <job-id>,
"sequenceStoreId": <sequence-store-id>,
"destination": <destination-s3-uri>,
"status": "SUBMITTED",
"creationTime": "2022-10-22T01:33:38.079000+00:00"
}
```

ジョブが開始されたら、次に示すように get-read-set-export-job API オペレーションを使用してそのステータスを判断できます。*sequence store ID* と をそれぞれシーケンスストア ID とジョブ ID *job ID*に置き換えます。

```
aws omics get-read-set-export-job --id job-id --sequence-store-id sequence store ID
```

次に示すように、list-read-set-export-jobs API オペレーションを使用して、シーケンスストアに対して初期化されたすべてのエクスポートジョブを表示できます。をシーケンスストア ID *sequence store ID*に置き換えます。

```
aws omics list-read-set-export-jobs --sequence-store-id sequence store ID.
```

```
{
```

```
"exportJobs": [
  {
      "id": <job-id>,
      "sequenceStoreId": <sequence-store-id>,
      "destination": <destination-s3-uri>,
      "status": "COMPLETED",
      "creationTime": "2022-10-22T01:33:38.079000+00:00",
      "completionTime": "2022-10-22T01:34:28.941000+00:00"
  }
]
}
```

読み取りセットのエクスポートに加えて、Amazon S3 アクセス URIs を使用して共有することもできます。詳細については[Amazon S3 URIs を使用した HealthOmics リードセットへのアクセス]を参照してください。

Amazon S3 URIs を使用した HealthOmics リードセットへのアクセス

Amazon S3 URI パスを使用して、アクティブなシーケンスストアの読み取りセットにアクセスできます。

Amazon S3 URI パスを使用すると、Amazon S3 オペレーションを使用して、読み取りセットを一覧表示、共有、ダウンロードできます。S3 APIsを加速します。 S3 さらに、S3 APIs へのアクセスを他のアカウントと共有し、データへのクロスリージョン読み取りアクセスを提供できます。

HealthOmics は、アーカイブされた読み取りセットへの Amazon S3 URI アクセスをサポートしていません。読み取りセットをアクティブ化すると、毎回同じ URI パスに復元されます。

HealthOmics ストアにデータをロードすると、Amazon S3 URI は Amazon S3 アクセスポイントに基づいているため、次のような Amazon S3 URIs を読み取る業界標準ツールと直接統合できます。

- Integrative Genomics Viewer (IGV) や UCSC Genome Browser などのビジュアル分析アプリケーション。
- CWL、WDL、Nextflow などの Amazon S3 拡張機能を使用した一般的なワークフロー。
- アクセスポイントの Amazon S3 URIs の認証と読み取り、または署名付き Amazon S3 URIs読み取りが可能な任意のツール。
- Mountpoint や CloudFront などの Amazon S3 ユーティリティ。

Amazon S3 Mountpoint を使用すると、Amazon S3 バケットをローカルファイルシステムとして使用できます。Mountpoint の詳細については、「Mountpoint for Amazon S3」を参照してください。

Amazon CloudFront は、高いパフォーマンス、セキュリティ、開発者の利便性を実現するために構築されたコンテンツ配信ネットワーク (CDN) サービスです。Amazon CloudFront の使用の詳細については、Amazon CloudFront ドキュメントを参照してください。シーケンスストアで CloudFront を設定するには、 AWS HealthOmics チームにお問い合わせください。

データ所有者のルートアカウントは、シーケンスストアプレフィックスのアクション S3:GetObject、S3:GetObjectTagging、S3:List Bucket に対して有効になっています。アカウントのユーザーがデータにアクセスするには、IAM ポリシーを作成し、ユーザーまたはロールにアタッチします。ポリシーの例についてはAmazon S3 URIs を使用したデータアクセスのアクセス許可を参照してください。

アクティブな読み取りセットで次の Amazon S3 API オペレーションを使用して、データを一覧表示および取得できます。アーカイブされた読み取りセットは、アクティブ化された後に Amazon S3 URIs を介してアクセスできます。

- GetObject — Amazon S3 からオブジェクトを取得します。
- HeadObject — HEAD オペレーションは、オブジェクト自体を返さずにオブジェクトからメタデータを取得します。このオペレーションは、オブジェクトのメタデータのみが必要な場合に便利です。
- ListObjects および ListObject v2 — バケット内のオブジェクトの一部またはすべて (最大 1,000) を返します。
- CopyObject — Amazon S3 に既に保存されているオブジェクトのコピーを作成します。HealthOmics は Amazon S3 アクセスポイントへのコピーをサポートしていますが、アクセスポイントへの書き込みはサポートしていません。

HealthOmics シーケンスストアは、ETags を通じてファイルのセマンティックアイデンティティを維持します。ファイルのライフサイクル全体を通して、ビット単位のアイデンティティに基づく Amazon S3 ETag は変更される可能性がありますが、HealthOmics ETag は変わりません。詳細についてはHealthOmics ETags とデータ出所を参照してください。

トピック

- HealthOmics ストレージの Amazon S3 URI 構造
- ホスト型またはローカル IGV を使用した読み取りセットへのアクセス
- HealthOmics での Samtools または HTSlib の使用

- Mountpoint HealthOmics の使用
- HealthOmics での CloudFront の使用

HealthOmics ストレージの Amazon S3 URI 構造

Amazon S3 URIsには、 omics:subjectId および omics:sampleIdリソースタグがあります。これらのタグを使用して、 などのパターンで IAM ポリシーを使用してアクセスを共有できます"s3:ExistingObjectTag/omics:subjectId": "pattern desired"。

ファイル構造は次のとおりです。

.../*account_id*/sequenceStore/*seq_store_id*/readSet/*read_set_id*/*files*.

Amazon S3 からシーケンスストアにインポートされたファイルの場合、シーケンスストアは元のソース名を維持しようとします。名前が競合すると、システムは読み込みセット情報を追加して、ファイル名が一意であることを確認します。例えば、fastq 読み取りセットの場合、両方のファイル名が同じであれば、名前を一意にするために、 sourceXは .fastq.gz または .fq.gz の前に挿入されます。直接アップロードの場合、ファイル名は次のパターンに従います。

- FASTQ の場合 — *read_set_name_sourcex*.fastq.gz
- uBAM/BAM/CRAM の場合 — *read_set_name.file ###*。拡張子は .bamまたは です.cram。例は NA193948.bam です。

BAM または CRAM の読み取りセットの場合、インデックスファイルは取り込みプロセス中に自動的に生成されます。生成されたインデックスファイルには、ファイル名の末尾に適切なインデックス拡張子が適用されます。パターン <*####################*>.<*############*> *######*インデックス拡張子は .baiまたは です.crai。

ホスト型またはローカル IGV を使用した読み取りセットへのアクセス

IGV は、BAM ファイルと CRAM ファイルを分析するために使用されるゲノムブラウザです。一度にゲノムの一部しか表示されないため、 ファイルとインデックスの両方が必要です。IGV はローカルでダウンロードして使用でき、AWS がホストする IGV を作成するガイドがあります。パブリックウェブバージョンには CORS が必要なため、サポートされていません。

ローカル IGV は、ファイルにアクセスするためにローカル AWS 設定に依存します。その設定で使用されるロールに、アクセスするリードセットの s3 URI に対する kms:Decrypt および s3:GetObject

アクセス許可を有効にするポリシーがアタッチされていることを確認します。その後、IGV で「File > load from URL」を使用し、ソースとインデックスの URI に貼り付けることができます。または、署名付き URLsを生成して同じ方法で使用することもできます。これにより、AWS 設定がバイパスされます。CORS は Amazon S3 URI アクセスではサポートされていないため、CORS に依存するリクエストはサポートされていないことに注意してください。

AWS ホスト型 IGV の例では、AWS Cognito を使用して、環境内で正しい設定とアクセス許可を作成します。アクセスする読み取りセットの Amazon S3 URI に対する kms:Decrypt および s3:GetObject アクセス許可を有効にするポリシーが作成されていることを確認して、このポリシーを Cognito ユーザープールに割り当てられたロールに追加します。その後、IGV で「ファイル > URL からのロード」を使用して、ソースとインデックスの URI に を入力できます。または、署名付き URLsを生成して同じ方法で使用することもできます。これにより、AWS 設定がバイパスされます。

シーケンスストアは「Amazon」タブには表示されないことに注意してください。 は、 AWS プロファイルが設定されているリージョンで、ユーザーが所有するバケットのみを表示するためです。

HealthOmics での Samtools または HTSlib の使用

HTSlib は、Samtools、rSamtools、PySam などのいくつかのツールで共有されるコアライブラリです。HTSlib バージョン 1.20 以降を使用して、Amazon S3 アクセスポイントをシームレスにサポートします。HTSlib ライブラリの古いバージョンでは、次の回避策を使用できます。

- を使用して HTS Amazon S3 ホストの環境変数を設定します。 `export HTS_S3_HOST="s3.region.amazonaws.com"`
- 使用するファイルの署名済み URL を生成します。BAM または CRAM を使用している場合は、ファイルとインデックスの両方の署名付き URL が生成されていることを確認します。その後、両方のファイルをライブラリで使用できます。
- Mountpoint を使用して、HTSlibライブラリを使用しているのと同じ環境にシーケンスストアまたはリードセットプレフィックスをマウントします。ここから、ローカルファイルパスを使用してファイルにアクセスできます。

Mountpoint HealthOmics の使用

Mountpoint for Amazon S3 は、Amazon S3 バケットをローカルファイルシステムとしてマウントするためのシンプルで高スループットのファイルクライアントです。Mountpoint for Amazon S3 を使用すると、アプリケーションはオープンや読み取りなどのファイルオペレーションを通じて Amazon

S3 に保存されているオブジェクトにアクセスできます。Mountpoint for Amazon S3 は、これらのオペレーションを Amazon S3 オブジェクト API コールに自動的に変換するため、アプリケーションはファイルインターフェイスを介して Amazon S3 のエラスティックストレージとスループットにアクセスできます。

Mountpoint のインストール手順を使用して、Mountpoint をインストールできます。Mountpoint は、インストールにローカルな AWS プロファイルを使用し、Amazon S3 プレフィックスレベルで動作します。使用するプロファイルに、アクセスするリードセット (複数可) またはシーケンスストアの Amazon S3 URI プレフィックスに対する s3:GetObject、s3:ListBucket、および kms:Decrypt アクセス許可を有効にするポリシーがあることを確認します。その後、次のパスを使用してバケットをマウントできます。

```
mount-s3 access point arn local path to mount --prefix prefix to sequence store or read
 set --region region
```

HealthOmics での CloudFront の使用

Amazon CloudFront は、高いパフォーマンス、セキュリティ、開発者の利便性を実現するために構築されたコンテンツ配信ネットワーク (CDN) サービスです。CloudFront を使用するお客様は、サービスチームと協力して CloudFront ディストリビューションを有効にする必要があります。アカウントチームと協力して HealthOmics サービスチームを関与させます。

HealthOmics での読み取りセットのアクティブ化

次の例 AWS CLIに示すように、start-read-set-activation-job API オペレーションまたは を使用してアーカイブされた読み取りセットをアクティブ化できます。*sequence store ID* と をシーケンスストア ID とリードセット ID *read set id*に置き換えます。 IDs

```
aws omics start-read-set-activation-job
     --sequence-store-id sequence store ID \
     --sources readSetId=read set ID readSetId=read set id_1 read set id_2
```

次に示すように、アクティベーションジョブ情報を含むレスポンスを受け取ります。

```
{
    "id": "12345678",
    "sequenceStoreId": "1234567890",
```

```
    "status": "SUBMITTED",
    "creationTime": "2022-10-22T00:50:54.670000+00:00"
}
```

アクティベーションジョブが開始されたら、get-read-set-activation-job API オペレーションを使用して進行状況をモニタリングできます。以下は、 を使用してアクティベーションジョブのステータス AWS CLI を確認する方法の例です。*job ID* と をそれぞれシーケンスストア ID とジョブ IDs *sequence store ID*に置き換えます。

```
aws omics get-read-set-activation-job --id job ID --sequence-store-id sequence store ID
```

レスポンスは、次に示すように、アクティベーションジョブを要約します。

```
{
    "id": 123567890,
    "sequenceStoreId": 123467890,
    "status": "SUBMITTED",
    "statusUpdateReason": "The job is submitted and will start soon.",
    "creationTime": "2022-10-22T00:50:54.670000+00:00",
    "sources": [
        {
            "readSetId": <reads set id_1>,
            "status": "NOT_STARTED",
            "statusUpdateReason": "The source is queued for the job."
        },
        {
            "readSetId": <read set id_2>,
            "status": "NOT_STARTED",
            "statusUpdateReason": "The source is queued for the job."
        }
    ]
}
```

getget-read-set-metadataオペレーションを使用して、アクティベーションジョブのステータスを確認できます。可能なステータスは、ACTIVE、ACTIVATING、および ですARCHIVED。次の例では、をシーケンスストア ID *sequence store ID*に置き換え、 を読み取りセット ID *read set ID* に置き換えます。

```
aws omics get-read-set-metadata --sequence-store-id sequence store ID --id read set ID
```

次のレスポンスは、読み取りセットがアクティブであることを示しています。


```
{
    "id": "12345678",
    "arn": "arn:aws:omics:us-west-2:555555555555:sequenceStore/1234567890/
readSet/12345678",
    "sequenceStoreId": "0123456789",
    "subjectId": "mySubject",
    "sampleId": "mySample",
    "status": "ACTIVE",
    "name": "HG00100",
    "description": "HG00100 aligned to HG38 BAM",
    "fileType": "BAM",
    "creationTime": "2022-07-13T23:25:20Z",
    "sequenceInformation": {
        "totalReadCount": 1513467,
        "totalBaseCount": 163454436,
        "generatedFrom": "Pulled from SRA",
        "alignment": "ALIGNED"
    },
    "referenceArn": "arn:aws:omics:us-west-2:555555555555:referenceStore/0123456789/
reference/0000000001",
    "files": {
        "source1": {
            "totalParts": 2,
            "partSize":  10485760,
            "contentLength": 17112283,
            "s3Access": {
        "s3Uri": "s3://accountID-sequence store ID-ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
s3alias/592761533288/sequenceStore/2015356892/readSet/9515444019/
import_source1.fastq.gz"
},
         },
        "index": {
            "totalParts": 1,
            "partSize": 53216,
            "contentLength": 10485760
            "s3Access": {
        "s3Uri": "s3://accountID-sequence store ID-ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
s3alias/592761533288/sequenceStore/2015356892/readSet/9515444019/
import_source1.fastq.gz"
},
        }
    },
```

```
    "creationType": "IMPORT",
    "etag": {
        "algorithm": "BAM_MD5up",
        "source1": "d1d65429212d61d115bb19f510d4bd02"
    }
}
```

次の例に示すように、list-read-set-activation-jobs を使用してすべてのリードセットアクティベーションジョブを表示できます。次の例では、 をシーケンスストア ID *sequence store ID* に置き換えます。

```
aws omics list-read-set-activation-jobs --sequence-store-id sequence store ID
```

次のレスポンスが表示されます。

```
{
    "activationJobs": [
        {
            "id": 1234657890,
            "sequenceStoreId": "1234567890",
            "status": "COMPLETED",
            "creationTime": "2022-10-22T01:33:38.079000+00:00",
            "completionTime": "2022-10-22T01:34:28.941000+00:00"
        }
    ]
}
```

HealthOmics 分析

HealthOmics 分析は、ゲノムバリアントと注釈の保存と分析をサポートします。Analytics には、バリアントストアと注釈ストアの 2 種類のストレージリソースが用意されています。これらのリソースを使用して、ゲノムバリアントデータと注釈データを保存、変換、クエリします。データストアにデータをインポートした後、Athena を使用してデータに対して高度な分析を実行できます。

HealthOmics コンソールまたは API を使用して、ストアの作成と管理、データのインポート、共同作業者との分析ストアデータの共有を行うことができます。

バリアントストアは VCF 形式のデータをサポートし、注釈ストアは TSV/CSV および GFF3 形式をサポートします。ゲノム座標は、ゼロベースの半分閉じた半開き間隔として表されます。データが HealthOmics 分析データストアにある場合、VPC ファイルへのアクセスは によって管理されます AWS Lake Formation。その後、Amazon Athena を使用して VCF ファイルをクエリできます。クエリは Athena クエリエンジンバージョン 3 を使用する必要があります。Athena クエリエンジンのバージョンの詳細については、Amazon Athena ドキュメントを参照してください。

トピック

HealthOmics バリアントストアの作成

以下のトピックでは、 コンソールと API を使用して HealthOmics バリアントストアを作成する方法について説明します。

トピック

コンソールを使用したバリアントストアの作成

HealthOmics コンソールを使用してバリアントストアを作成できます。

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、バリアントストアを選択します。
3. バリアントストアの作成ページで、次の情報を入力します。
   - バリアントストア名 - このストアの一意の名前。
   - 説明 (オプション) - このバリアントストアの説明。
   - 参照ゲノム - このバリアントストアの参照ゲノム。
   - データ暗号化 - データ暗号化を AWS 自分で所有および管理するかどうかを選択します。
   - タグ (オプション) - このバリアントストアには最大 50 個のタグを指定します。
4. バリアントストアの作成 を選択します。

API を使用したバリアントストアの作成

HealthOmics `CreateVariantStore` API オペレーションを使用してバリアントストアを作成します。このオペレーションは、 を使用して実行することもできます AWS CLI。

バリアントストアを作成するには、ストアの名前とリファレンスストアの ARN を指定します。バリアントストアは、ステータスが READY に変わったときにデータを取り込む準備ができています。

次の例では AWS CLI 、 を使用してバリアントストアを作成します。

```
aws omics create-variant-store --name myvariantstore \
    --reference referenceArn="arn:aws:omics:us-
west-2:555555555555:referenceStore/123456789/reference/5987565360"
```

バリアントストアの作成を確認するには、次のレスポンスを受け取ります。

```
{
    "creationTime": "2022-11-03T18:19:52.296368+00:00",
    "id": "45aeb91d5678",
    "name": "myvariantstore",
    "reference": {
        "referenceArn": "arn:aws:omics:us-west-2:555555555555:referenceStore/123456789/
reference/5987565360"
```

```
    },
    "status": "CREATING"
}
```

バリアントストアの詳細については、get-variant-store API を使用します。

```
aws omics get-variant-store --name myvariantstore
```

次のレスポンスが表示されます。

```
{
    "id": "45aeb91d5678",
    "reference": {
        "referenceArn": "arn:aws:omics:us-west-2:555555555555:referenceStore/123456789/
reference/5987565360"
    },
    "status": "ACTIVE",
    "storeArn": "arn:aws:omics:us-west-2:555555555555:variantStore/myvariantstore",
    "name": "myvariantstore",
    "creationTime": "2022-11-03T18:19:52.296368+00:00",
    "updateTime": "2022-11-03T18:30:56.272792+00:00",
    "tags": {},
    "storeSizeBytes": 0
}
```

アカウントに関連付けられているすべてのバリアントストアを表示するには、list-variant-stores API を使用します。

```
aws omics list-variant-stores
```

次のレスポンスの例に示すように、すべてのバリアントストアとその IDs、ステータス、およびその他の詳細を一覧表示するレスポンスを受け取ります。

```
{
    "variantStores": [
        {
            "id": "45aeb91d5678",
            "reference": {
                "referenceArn": "arn:aws:omics:us-
west-2:55555555555:referenceStore/5506874698"
            },
```

```
            "status": "ACTIVE",
            "storeArn": "arn:aws:omics:us-west-2:55555555555:variantStore/
new_variant_store",
            "name": "variantstore",
            "creationTime": "2022-11-03T18:19:52.296368+00:00",
            "updateTime": "2022-11-03T18:30:56.272792+00:00",
            "statusMessage": "",
            "storeSizeBytes": 141526
        }
    ]
}
```

また、ステータスやその他の基準に基づいて list-variant-stores API のレスポンスをフィルタリングすることもできます。

2023 年 5 月 15 日以降に作成された分析ストアにインポートされた VCF ファイルには、バリアント効果予測子 (VEP) 注釈のスキーマが定義されています。これにより、インポートされた VCF データのクエリと解析が容易になります。この変更は、 annotation fieldsパラメータが API または CLI コールに含まれている場合を除き、2023 年 5 月 15 日より前に作成されたストアには影響しません。これらのストアでは、 annotation fieldsパラメータを使用するとリクエストが失敗します。

HealthOmics バリアントストアのインポートジョブの作成

次の例は、 を使用してバリアントストアのインポートジョブ AWS CLI を作成する方法を示しています。

```
aws omics start-variant-import-job \
       --destination-name myvariantstore \
       --runLeftNormalization false \
       --role-arn  arn:aws:iam::55555555555:role/roleName \
       --items source=s3://my-omics-bucket/sample.vcf.gz source=s3://my-omics-bucket/
sample2.vcf.gz
```

```
{
    "destinationName": "store_a",
    "roleArn": "....",
    "runLeftNormalization": false,
    "items": [
        {"source": "s3://my-omics-bucket/sample.vcf.gz"},
        {"source": "s3://my-omics-bucket/sample2.vcf.gz"}
```

```
    ]
}
```

2023 年 5 月 15 日以降に作成されたストアの場合、次の例は --annotation-fieldsパラメータを追加する方法を示しています。注釈フィールドはインポートで定義されます。

```
aws omics start-variant-import-job \
   --destination-name annotationparsingvariantstore \
   --role-arn arn:aws:iam::123456789012:role/<role_name> \
   --items source=s3://pathToS3/sample.vcf
   --annotation-fields '{"VEP": "CSQ"}'
```

```
{
    "jobId": "981e2286-e954-4391-8a97-09aefc343861"
}
```

get-variant-import-job を使用してステータスを確認します。

```
aws omics get-variant-import-job --job-id 08279950-a9e3-4cc3-9a3c-a574f9c9e229
```

インポートジョブのステータスを示す JSON レスポンスを受け取ります。VCF の VEP 注釈は、ID/値のペアとして INFO 列に保存されている情報について解析されます。Ensembl Variant Effect Predictor 注釈のデフォルト ID INFO 列は CSQ ですが、 --annotation-fieldsパラメータを使用して、INFO 列で使用されるカスタム値を指定できます。現在、解析は VEP 注釈でサポートされています。

2023 年 5 月 15 日より前に作成されたストア、または VEP 注釈を含まない VCF ファイルの場合、レスポンスには注釈フィールドは含まれません。

```
{
    "creationTime": "2023-04-11T17:52:37.241958+00:00",
    "destinationName": "annotationparsingvariantstore",
    "id": "7a1c67e3-b7f9-434d-817b-9c571fd63bea",
    "items": [

    {
       "jobStatus": "COMPLETED",
       "source": "s3://amzn-s3-demo-bucket/NA12878.2k.garvan.vcf"
    }
 ],
```

```
    "roleArn": "arn:aws:iam::555555555555:role/<role_name>",

    "runLeftNormalization": false,
    "status": "COMPLETED",
    "updateTime": "2023-04-11T17:58:22.676043+00:00",
}
```

VCF ファイルの一部である VEP 注釈は、次の構造を持つ事前定義されたスキーマとして保存されます。Extras フィールドを使用して、デフォルトのスキーマに含まれていない追加の VEP フィールドを保存できます。

```
annotations struct<
   vep: array<struct<
      allele:string,
      consequence: array<string>,
      impact:string,
      symbol:string,
      gene:string,
      `feature_type`: string,
      feature: string,
      biotype: string,
      exon: struct<rank:string, total:string>,
      intron: struct<rank:string, total:string>,
      hgvsc: string,
      hgvsp: string,
      `cdna_position`: string,
      `cds_position`: string,
      `protein_position`: string,
      `amino_acids`: struct<reference:string, variant: string>,
      codons: struct<reference:string, variant: string>,
      `existing_variation`: array<string>,
      distance: string,
      strand: string,
      flags: array<string>,
      symbol_source: string,
      hgnc_id: string,
      `extras`: map<string, string>
    >>
>
```

解析はベストエフォートアプローチで実行されます。VEP エントリが VEP 標準仕様に従っていない場合、解析されず、配列の行は空になります。

新しいバリアントストアの場合、get-variant-import-job のレスポンスには、図のように注釈フィールドが含まれます。

```
aws omics get-variant-import-job --job-id 08279950-a9e3-4cc3-9a3c-a574f9c9e229
```

インポートジョブのステータスを示す JSON レスポンスを受け取ります。

```
{
    "creationTime": "2023-04-11T17:52:37.241958+00:00",
    "destinationName": "annotationparsingvariantstore",
    "id": "7a1c67e3-b7f9-434d-817b-9c571fd63bea",
    "items": [

    {
    "jobStatus": "COMPLETED",
    "source": "s3://amzn-s3-demo-bucket/NA12878.2k.garvan.vcf"
    }
 ],
    "roleArn": "arn:aws:iam::123456789012:role/<role_name>",
    "runLeftNormalization": false,
    "status": "COMPLETED",
    "updateTime": "2023-04-11T17:58:22.676043+00:00",
    "annotationFields" : {"VEP": "CSQ"}
  }
}
```

list-variant-import-jobs を使用して、すべてのインポートジョブとそのステータスを表示できます。

```
aws omics list-variant-import-jobs --ids 7a1c67e3-b7f9-434d-817b-9c571fd63bea
```

レスポンスには、次のように情報が含まれています。

```
{
    "variantImportJobs": [
    {
        "creationTime": "2023-04-11T17:52:37.241958+00:00",
        "destinationName": "annotationparsingvariantstore",
        "id": "7a1c67e3-b7f9-434d-817b-9c571fd63bea",
        "roleArn": "arn:aws:iam::55555555555:role/roleName",
        "runLeftNormalization": false,
```

```
        "status": "COMPLETED",
        "updateTime": "2023-04-11T17:58:22.676043+00:00",
        "annotationFields" : {"VEP": "CSQ"}
        }
    ]
  }
}
```

必要に応じて、次のコマンドを使用してインポートジョブをキャンセルできます。

```
aws omics cancel-variant-import-job
     --job-id edd7b8ce-xmpl-47e2-bc99-258cac95a508
```

HealthOmics 注釈ストアの作成

注釈ストアは、TSV、VPC、GFF ファイルなどの注釈データベースを表すデータストアです。同じ参照ゲノムが指定されている場合、注釈ストアはインポート中にバリアントストアと同じ座標系にマッピングされます。以下のトピックでは、HealthOmics コンソールと AWS CLI を使用して注釈ストアを作成および管理する方法を示します。

トピック

- コンソールを使用した注釈ストアの作成
- API を使用した注釈ストアの作成

コンソールを使用した注釈ストアの作成

HealthOmics コンソールで注釈ストアを作成するには、次の手順に従います。

注釈ストアを作成するには

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、注釈ストアを選択します。
3. 注釈ストアページで、注釈ストアの作成を選択します。
4. 注釈ストアの作成ページで、次の情報を入力します。
    - 注釈ストア名 - このストアの一意の名前。
    - 説明 (オプション) - このリファレンスゲノムの説明。

- データ形式とスキーマの詳細 - データファイル形式を選択し、このストアのスキーマ定義をアップロードします。
- リファレンスゲノム - この注釈のリファレンスゲノム。
- データ暗号化 - データ暗号化を AWS 自分で所有および管理するかどうかを選択します。
- タグ (オプション) - この注釈ストアには最大 50 個のタグを指定します。

5. 注釈ストアの作成 を選択します。

API を使用した注釈ストアの作成

次の例は、 を使用して注釈ストアを作成する方法を示しています AWS CLI。すべての AWS CLI および API オペレーションで、データの形式を指定する必要があります。

```
aws omics create-annotation-store --name my_annotation_store \
           --store-format GFF \
           --reference referenceArn="arn:aws:omics:us-
west-2:555555555555:referenceStore/6505293348/reference/5987565360"
           --version-name new_version
```

注釈ストアの作成を確認するために、次のレスポンスを受け取ります。

```
{
           "creationTime": "2022-08-24T20:34:19.229500Z",
           "id": "3b93cdef69d2",
           "name": "my_annotation_store",
           "reference": {
               "referenceArn": "arn:aws:omics:us-
west-2:555555555555:referenceStore/6505293348/reference/5987565360"
           },
           "status": "CREATING"
           "versionName": "my_version"
       }
```

注釈ストアの詳細については、get-annotation-store API を使用します。

```
aws omics get-annotation-store --name my_annotation_store
```

次のレスポンスが表示されます。

```
{
```

```
          "id": "eeb019ac79c2",
          "reference": {
              "referenceArn": "arn:aws:omics:us-
west-2:555555555555:referenceStore/5638433913/reference/5871590330“
          },
          "status": "ACTIVE",
          "storeArn": "arn:aws:omics:us-west-2:555555555555:annotationStore/gffstore",
          "name": "my_annotation_store",
          "creationTime": "2022-11-05T00:05:19.136131+00:00",
          "updateTime": "2022-11-05T00:10:36.944839+00:00",
          "tags": {},
          "storeFormat": "GFF",
          "statusMessage": "",
          "storeSizeBytes": 0,
          "numVersions": 1
      }
```

アカウントに関連付けられているすべての注釈ストアを表示するには、list-annotation-stores API オペレーションを使用します。

```
aws omics list-annotation-stores
```

次のレスポンスの例に示すように、すべての注釈ストアとその IDs、ステータス、およびその他の詳細を一覧表示するレスポンスを受け取ります。

```
{
           "annotationStores": [
               {
                  "id": "4d8f3eada259",
                   "reference":
                       "referenceArn": "arn:aws:omics:us-
west-2:555555555555:referenceStore/5638433913/reference/5871590330"
                   },
                   "status": "CREATING",
                   "name": "gffstore",
                   "creationTime": "2022-09-27T17:30:52.182990+00:00",
                   "updateTime": "2022-09-27T17:30:53.025362+00:00"
               }
           ]
       }
```

ステータスやその他の条件に基づいてレスポンスをフィルタリングすることもできます。

HealthOmics 注釈ストアのインポートジョブの作成

トピック

API を使用した注釈インポートジョブの作成

次の例は、 を使用して注釈インポートジョブ AWS CLI を開始する方法を示しています。

```
aws omics start-annotation-import-job \
           --destination-name myannostore \
           --version-name myannostore \
           --role-arn arn:aws:iam::123456789012:role/roleName \
           --items source=s3://my-omics-bucket/sample.vcf.gz
           --annotation-fields '{"VEP": "CSQ"}'
```

2023 年 5 月 15 日より前に作成された注釈ストアは、注釈フィールドが含まれている場合、エラーメッセージを返します。注釈ストアのインポートジョブに関連する API オペレーションの出力は返されません。

その後、get-annotation-import-job API オペレーションと job ID パラメータを使用して、注釈のインポートジョブの詳細を確認できます。

```
aws omics get-annotation-import-job --job-id 9e4198fb-fa85-446c-9301-9b823a1a8ba8
```

注釈フィールドを含む次のレスポンスを受け取ります。

```
{
          "creationTime": "2023-04-11T19:09:25.049767+00:00",
          "destinationName": "parsingannotationstore",
          "versionName": "parsingannotationstore",
          "id": "9e4198fb-fa85-446c-9301-9b823a1a8ba8",
          "items": [
              {
                  "jobStatus": "COMPLETED",
```

```
                "source": "s3://my-omics-bucket/sample.vep.vcf"
            }
        ],
        "roleArn": "arn:aws:iam::55555555555:role/roleName",
        "runLeftNormalization": false,
        "status": "COMPLETED",
        "updateTime": "2023-04-11T19:13:09.110130+00:00",
        "annotationFields" : {"VEP": "CSQ"}
    }
```

すべての注釈ストアのインポートジョブを表示するには、list-annotation-import-jobs を使用します。

```
aws omics list-annotation-import-jobs --ids 9e4198fb-fa85-446c-9301-9b823a1a8ba8
```

レスポンスには、注釈ストアのインポートジョブの詳細とステータスが含まれます。

```
{
        "annotationImportJobs": [
        {
            "creationTime": "2023-04-11T19:09:25.049767+00:00",
            "destinationName": "parsingannotationstore",
            "versionName": "parsingannotationstore",
            "id": "9e4198fb-fa85-446c-9301-9b823a1a8ba8",
            "roleArn": "arn:aws:iam::55555555555:role/roleName",
            "runLeftNormalization": false,
            "status": "COMPLETED",
            "updateTime": "2023-04-11T19:13:09.110130+00:00",
            "annotationFields" : {"VEP": "CSQ"}
        }
        ]
    }
```

TSV 形式と VCF 形式の追加パラメータ

TSV 形式と VCF 形式の場合、入力を解析する方法を API に通知する追加のパラメータがあります。

> **Important**
>
> クエリエンジンでエクスポートされた CSV 注釈データは、データセットのインポートから直接情報を返します。インポートされたデータに数式またはコマンドが含まれている場合、

ファイルは CSV インジェクションの対象となる可能性があります。したがって、クエリエンジンでエクスポートされたファイルは、セキュリティ警告を求められる可能性があります。悪意のあるアクティビティを回避するには、エクスポートファイルを読み取るときにリンクとマクロをオフにします。

TSV パーサーは、次の表に示す、ゲノム座標の左正規化や標準化などの基本的なバイオインフォマティクス操作も実行します。

形式タイプ	説明
ジェネリック	汎用テキストファイル。ゲノム情報がありません。
CHR_POS	開始位置 - 1、終了位置を追加します。これはと同じですPOS。
CHR_POS_REF_ALT	contig、1 ベース位置、ref および alt アレル情報が含まれます。
CHR_START_END_REF_ALT_ONE_BASE	contig、start、end、ref、alt の各アレル情報が含まれます。座標は 1 ベースです。
CHR_START_END_ZERO_BASE	競合位置、開始位置、終了位置が含まれます。座標は 0 ベースです。
CHR_START_END_ONE_BASE	競合位置、開始位置、終了位置が含まれます。座標は 1 ベースです。
CHR_START_END_REF_ALT_ZERO_BASE	contig、start、end、ref、alt の各アレル情報が含まれます。座標は 0 ベースです。

TSV インポート注釈ストアリクエストは次の例のようになります。

```
aws omics start-annotation-import-job \
--destination-name tsv_anno_example \
--role-arn arn:aws:iam::555555555555:role/demoRole \
--items source=s3://demodata/genomic_data.bed.gz \
```

```
--format-options '{ "tsvOptions": {
        "readOptions": {
            "header": false,
            "sep": "\t"
        }
    }
}'
```

TSV 形式の注釈ストアの作成

次の例では、ヘッダー、行、コメントを含むタブ制限ファイルを使用して注釈ストアを作成します。座標は でCHR_START_END_ONE_BASED、OMIM のヒューマンジーンマップの概要からの HG19 遺伝子マップが含まれています。 https://www.omim.org/downloads

```
aws omics create-annotation-store --name mimgenemap \
  --store-format TSV \
  --reference=referenceArn=arn:aws:omics:us-
west-2:555555555555:referenceStore/6505293348/reference/2310864158 \
  --store-options=tsvStoreOptions='{
    annotationType=CHR_START_END_ONE_BASE,
    formatToHeader={CHR=chromosome, START=genomic_position_start,
 END=genomic_position_end},
    schema=[
      {chromosome=STRING},
      {genomic_position_start=LONG},
      {genomic_position_end=LONG},
      {cyto_location=STRING},
      {computed_cyto_location=STRING},
      {mim_number=STRING},
      {gene_symbols=STRING},
      {gene_name=STRING},
      {approved_gene_name=STRING},
      {entrez_gene_id=STRING},
      {ensembl_gene_id=STRING},
      {comments=STRING},
      {phenotypes=STRING},
      {mouse_gene_symbol=STRING}]}'
```

ヘッダーの有無にかかわらず、ファイルをインポートできます。CLI リクエストでこれを指定するには、次のインポートジョブの例に示すようにheader=false、 を使用します。

```
aws omics start-annotation-import-job \
```

```
   --role-arn arn:aws:iam::555555555555:role/demoRole \
   --items=source=s3://amzn-s3-demo-bucket/annotation-examples/hg38_genemap2.txt \
   --destination-name output-bucket \
   --format-options=tsvOptions='{readOptions={sep="\t",header=false,comment="#"}}'
```

次の の例では、ベッドファイルの注釈ストアを作成します。Bed ファイルは、タブで区切られたシンプルなファイルです。この例では、列は、Chromine、Start、End、およびリージョン名です。座標はゼロベースであり、データにはヘッダーがありません。

```
aws omics create-annotation-store \
   --name cexbed --store-format TSV \
   --reference=referenceArn=arn:aws:omics:us-
west-2:555555555555:referenceStore/6505293348/reference/2310864158 \
   --store-options=tsvStoreOptions='{
   annotationType=CHR_START_END_ZERO_BASE,
   formatToHeader={CHR=chromosome, START=start, END=end},
   schema=[{chromosome=STRING}, {start=LONG}, {end=LONG}, {name=STRING}]}'
```

その後、次の CLI コマンドを使用して、ベッドファイルを注釈ストアにインポートできます。

```
aws omics start-annotation-import-job \
   --role-arn arn:aws:iam::555555555555:role/demoRole \
   --items=source=s3://amzn-s3-demo-bucket/TruSeq_Exome_TargetedRegions_v1.2.bed \
   --destination-name cexbed \
   --format-options=tsvOptions='{readOptions={sep="\t",header=false,comment="#"}}'
```

次の例では、VPC ファイルの最初の数列と注釈情報を含む列を含むタブ区切りファイルの注釈ストアを作成します。これには、ゲノム位置と、ゲノム、開始、参照、および代替アレルに関する情報が含まれ、ヘッダーが含まれます。

```
aws omics create-annotation-store --name gnomadchrx --store-format TSV \
--reference=referenceArn=arn:aws:omics:us-
west-2:555555555555:referenceStore/6505293348/reference/2310864158 \
--store-options=tsvStoreOptions='{
    annotationType=CHR_POS_REF_ALT,
    formatToHeader={CHR=chromosome, POS=start, REF=ref, ALT=alt},
    schema=[
        {chromosome=STRING},
        {start=LONG},
        {ref=STRING},
```

```
        {alt=STRING},
        {filters=STRING},
        {ac_hom=STRING},
        {ac_het=STRING},
        {af_hom=STRING},
        {af_het=STRING},
        {an=STRING},
        {max_observed_heteroplasmy=STRING}]}'
```

次に、次の CLI コマンドを使用して、ファイルを注釈ストアにインポートします。

```
aws omics start-annotation-import-job \
  --role-arn arn:aws:iam::555555555555:role/demoRole \
   --items=source=s3://amzn-s3-demo-bucket/
gnomad.genomes.v3.1.sites.chrM.reduced_annotations.tsv \
   --destination-name gnomadchrx \
   --format-options=tsvOptions='{readOptions={sep="\t",header=true,comment="#"}}'
```

次の例は、顧客が mim2gene ファイルの注釈ストアを作成する方法を示しています。mim2gene ファイルは、OMIM の遺伝子と別の遺伝子識別子の間のリンクを提供します。これはタブ区切りで、コメントが含まれています。

```
aws omics create-annotation-store \
  --name mim2gene \
  --store-format TSV \
  --reference=referenceArn=arn:aws:omics:us-
west-2:555555555555:referenceStore/6505293348/reference/2310864158 \
  --store-options=tsvStoreOptions='
    {annotationType=GENERIC,
    formatToHeader={},
    schema=[
        {mim_gene_id=STRING},
        {mim_type=STRING},
        {entrez_id=STRING},
        {hgnc=STRING},
        {ensembl=STRING}]}'
```

その後、次のようにデータをストアにインポートできます。

```
aws omics start-annotation-import-job \
   --role-arn arn:aws:iam::555555555555:role/demoRole \
```

```
    --items=source=s3://xquek-dev-aws/annotation-examples/mim2gene.txt \
    --destination-name mim2gene \
    --format-options=tsvOptions='{readOptions={sep="\t",header=false,comment="#"}}'
```

VCF 形式のインポートジョブの開始

VCF ファイルには、次に示すようにignoreFilterField、これらのパラメータを無視または含める 2 つの追加入力 ignoreQualFieldと があります。

```
aws omics start-annotation-import-job --destination-name annotation_example\
  --role-arn arn:aws:iam::555555555555:role/demoRole \
  --items source=s3://demodata/example.garvan.vcf \
  --format-options '{ "vcfOptions": {
    "ignoreQualField": false,
    "ignoreFilterField": false
    }
   }'
```

図に示すように、注釈ストアのインポートをキャンセルすることもできます。キャンセルが成功した場合、この AWS CLI 呼び出しに対するレスポンスは受信されません。ただし、インポートジョブ ID が見つからないか、インポートジョブが完了すると、エラーメッセージが表示されます。

```
aws omics cancel-annotation-import-job --job-id edd7b8ce-xmpl-47e2-bc99-258cac95a508
```

> **Note**
>
> get-annotation-import-job、get-variant-import-job、list-annotation-import-jobs、list-variant-import-jobs のメタデータインポートジョブ履歴は、2 年後に自動削除されます。インポートされたバリアントと注釈データは自動削除されず、データストアに残ります。

HealthOmics 注釈ストアの新しいバージョンの作成

注釈ストアの新しいバージョンを作成して、注釈データベースのさまざまなバージョンを収集できます。これにより、注釈データが整理され、定期的に更新されます。

既存の注釈ストアの新しいバージョンを作成するには、次の例に示すように create-annotation-store-version API を使用します。

```
aws omics create-annotation-store-version \
     --name my_annotation_store \
     --version-name my_version
```

注釈ストアのバージョン ID で次のレスポンスが表示され、注釈の新しいバージョンが作成されていることを確認します。

```
{
     "creationTime": "2023-07-21T17:15:49.251040+00:00",
     "id": "3b93cdef69d2",
     "name": "my_annotation_store",
     "reference": {
         "referenceArn": "arn:aws:omics:us-
west-2:555555555555:referenceStore/6505293348/reference/5987565360"
     },
     "status": "CREATING",
     "versionName": "my_version"
}
```

注釈ストアバージョンの説明を更新するには、update-annotation-store-version を使用して注釈ストアバージョンに更新を追加できます。

```
aws omics update-annotation-store-version \
    --name my_annotation_store \
    --version-name my_version \
    --description "New Description"
```

注釈ストアのバージョンが更新されたことを確認する次のレスポンスが表示されます。

```
{
     "storeId": "4934045d1c6d",
     "id": "2a3f4a44aa7b",
     "description":"New Description",
     "status": "ACTIVE",
     "name": "my_annotation_store",
     "versionName": "my_version",
     "creation Time": "2023-07-21T17:20:59.380043+00:00",
     "updateTime": "2023-07-21T17:26:17.892034+00:00"
}
```

注釈ストアバージョンの詳細を表示するには、get-annotation-store-version を使用します。

```
aws omics get-annotation-store-version --name my_annotation_store --version-name
 my_version
```

バージョン名、ステータス、その他の詳細が記載されたレスポンスが送信されます。

```
{
     "storeId": "4934045d1c6d",
     "id": "2a3f4a44aa7b",
     "status": "ACTIVE",
     "versionArn": "arn:aws:omics:us-west-2:555555555555:annotationStore/
my_annotation_store/version/my_version",
     "name": "my_annotation_store",
     "versionName": "my_version",
     "creationTime": "2023-07-21T17:15:49.251040+00:00",
     "updateTime": "2023-07-21T17:15:56.434223+00:00",
     "statusMessage": "",
     "versionSizeBytes": 0
    }
```

注釈ストアのすべてのバージョンを表示するには、次の例に示すように listlist-annotation-store-versions を使用できます。

```
aws omics list-annotation-store-versions --name my_annotation_store
```

以下の情報を含むレスポンスを受け取ります。

```
{
  "annotationStoreVersions": [
    {
     "storeId": "4934045d1c6d",
     "id": "2a3f4a44aa7b",
     "status": "CREATING",
     "versionArn": "arn:aws:omics:us-west-2:555555555555:annotationStore/
my_annotation_store/version/my_version_2",
     "name": "my_annotation_store",
     "versionName": "my_version_2",
     "creation Time": "2023-07-21T17:20:59.380043+00:00",
     "versionSizeBytes": 0
    },
    {
     "storeId": "4934045d1c6d",
```

```
     "id": "4934045d1c6d",
     "status": "ACTIVE",
     "versionArn": "arn:aws:omics:us-west-2:555555555555:annotationStore/
my_annotation_store/version/my_version_1",
     "name": "my_annotation_store",
     "versionName": "my_version_1",
     "creationTime": "2023-07-21T17:15:49.251040+00:00",
     "updateTime": "2023-07-21T17:15:56.434223+00:00",
     "statusMessage": "",
     "versionSizeBytes": 0
    }
}
```

注釈ストアのバージョンが不要になった場合は、次の例に示すように、delete-annotation-store-versions を使用して注釈ストアのバージョンを削除できます。

```
aws omics delete-annotation-store-versions --name my_annotation_store --versions
 my_version
```

ストアバージョンがエラーなしで削除された場合は、次のレスポンスが表示されます。

```
{
  "errors": []
}
```

エラーが発生した場合は、次のようにエラーの詳細を含むレスポンスを受け取ります。

```
{
  "errors": [
    {
      "versionName": "my_version",
      "message": "Version with versionName: my_version was not found."
    }
  ]
}
```

アクティブなインポートジョブを持つ注釈ストアバージョンを削除しようとすると、次に示すようにエラーを含むレスポンスが表示されます。

```
{
  "errors": [
```

```
    {
      "versionName": "my_version",
      "message": "version has an inflight import running"
    }
  ]
}
```

この場合、次の例に示すように、注釈ストアのバージョンを強制的に削除できます。

```
aws omics delete-annotation-store-versions --name my_annotation_store --versions
 my_version --force
```

HealthOmics 分析ストアの削除

バリアントまたは注釈ストアを削除すると、システムはそのストアおよび関連するタグにインポートされたすべてのデータも削除します。

次の例は、 を使用してバリアントストアを削除する方法を示しています AWS CLI。アクションが成功すると、バリアントストアのステータスは に移行しますDELETING。

```
aws omics delete-variant-store --id <variant-store-id>
```

次の例は、注釈ストアを削除する方法を示しています。アクションが成功すると、注釈ストアのステータスは に移行しますDELETING。複数のバージョンが存在する場合、注釈ストアを削除することはできません。

```
aws omics delete-annotation-store --id <annotation-store-id>
```

HealthOmics 分析データのクエリ

バリアントストアでクエリを実行するには、 AWS Lake Formation と Amazon Athena または Amazon EMR を使用します。クエリを実行する前に、Lake Formation と Amazon Athena のセットアップ手順 (以下のセクションで説明) を完了してください。

Amazon EMR の詳細については、「チュートリアル: Amazon EMR の開始方法」を参照してください。

2024 年 9 月 26 日以降に作成されたバリアントストアの場合、HealthOmics はストアをサンプル ID でパーティション分割します。このパーティショニングは、HealthOmics がサンプル ID を使用して

バリアント情報の保存を最適化することを意味します。サンプル情報をフィルターとして使用するクエリは、クエリがスキャンするデータが少ないため、より迅速に結果を返します。

HealthOmics はパーティションファイル名としてサンプル IDsを使用します。データを取り込む前に、サンプル ID に PHI データが含まれているかどうかを確認します。その場合は、データを取り込む前にサンプル ID を変更します。サンプル IDs に含めるコンテンツと含めないコンテンツの詳細については、 AWS HIPAA コンプライアンスウェブページのガイダンスを参照してください。

トピック

- HealthOmics を使用するように Lake Formation を設定する
- クエリ用の Athena の設定
- HealthOmics バリアントストアでのクエリの実行

HealthOmics を使用するように Lake Formation を設定する

Lake Formation を使用して HealthOmics データストアを管理する前に、次の Lake Formation 設定手順を実行します。

トピック

- Lake Formation 管理者の作成または検証
- Lake Formation コンソールを使用したリソースリンクの作成
- AWS RAM リソース共有のアクセス許可の設定

Lake Formation 管理者の作成または検証

Lake Formation でデータレイクを作成する前に、1 人以上の管理者を定義します。

管理者は、リソースリンクを作成するアクセス許可を持つユーザーとロールです。アカウントごと、リージョンごとにデータレイク管理者を設定します。

Lake Formation コンソールで管理者ユーザーを作成する

1. AWS Lake Formation コンソールを開く: Lake Formation コンソールを開く
2. コンソールに「Lake Formation へようこそ」パネルが表示されたら、「開始方法」を選択します。

   Lake Formation は、データレイク管理者テーブルにユーザーを追加します。

3. それ以外の場合は、左側のメニューから管理ロールとタスクを選択します。
4. 必要に応じて管理者を追加します。

Lake Formation コンソールを使用したリソースリンクの作成

ユーザーがクエリできる共有リソースを作成するには、デフォルトのアクセスコントロールを無効にする必要があります。デフォルトのアクセスコントロールの無効化の詳細については、Lake Formation ドキュメントの「データレイクのデフォルトのセキュリティ設定を変更する」を参照してください。リソースリンクを個別に作成することも、グループとして作成して、Amazon Athena または他の AWS サービス (Amazon EMR など) のデータにアクセスすることもできます。

AWS Lake Formation コンソールでリソースリンクを作成し、HealthOmics Analytics ユーザーと共有する

1. AWS Lake Formation コンソールを開く: Lake Formation コンソール
2. プライマリナビゲーションバーで、データベースを選択します。
3. Databases テーブルで、目的のデータベースを選択します。
4. 作成メニューから、リソースリンクを選択します。
5. リソースリンク名を入力します。Athena からデータベースにアクセスする場合は、小文字 (最大 256 文字) のみを使用して名前を入力します。
6. [作成] を選択します。
7. 新しいリソースリンクがデータベースに一覧表示されるようになりました。

Lake Formation コンソールを使用して共有リソースへのアクセスを許可する

Lake Formation データベース管理者は、次の手順を使用して共有リソースへのアクセスを許可できます。

1. AWS Lake Formation コンソールを開きます: https://console.aws.amazon.com/lakeformation/
2. プライマリナビゲーションバーで、データベースを選択します。
3. データベースページで、以前に作成したリソースリンクを選択します。
4. Actions メニューから、ターゲットに対する付与を選択します。
5. プリンシパルの「データアクセス許可の付与」ページで、IAM ユーザーまたはロールを選択します。

6. IAM ユーザーまたはロールのドロップダウンメニューから、アクセスを許可するユーザーを見つけます。
7. 次に、LF タグまたはカタログリソースカードで、名前付きデータカタログリソースオプションを選択します。
8. テーブルオプションドロップダウンメニューから、すべてのテーブルまたは以前に作成したテーブルを選択します。
9. テーブルのアクセス許可カードで、テーブルのアクセス許可で「Describe and Select」を選択します。
10. 次に、Grant を選択します。

Lake Formation のアクセス許可を表示するには、プライマリナビゲーションペインからデータレイクのアクセス許可を選択します。この表は、使用可能なデータベースとリソースリンクを示しています。

AWS RAM リソース共有のアクセス許可の設定

AWS Lake Formation コンソールで、プライマリナビゲーションバーでデータレイクのアクセス許可を選択してアクセス許可を表示します。データアクセス許可ページで、リソースタイプ、データベース、および RAM リソース共有の共有リソース**ARN**に関連する テーブルを表示できます。 AWS Resource Access Manager （ AWS RAM) リソース共有を受け入れる必要がある場合、 はコンソールで AWS Lake Formation 通知します。

HealthOmics は、ストアの作成中に AWS RAM リソース共有を暗黙的に受け入れることができます。 AWS RAM リソース共有を受け入れるには、 または `CreateAnnotationStore` API オペレーションを呼び出す IAM ユーザー`CreateVariantStore`またはロールで、次のアクションを許可する必要があります。

- `ram:GetResourceShareInvitations` - このアクションにより、HealthOmics は招待を検索できます。
- `ram:AcceptResourceShareInvitation` - このアクションによりHealthOmics は FAS トークンを使用して招待を受け入れることができます。

これらのアクセス許可がないと、ストアの作成中に認可エラーが表示されます。

これらのアクションを含むポリシーの例を次に示します。このポリシーを、 AWS RAM リソース共有を受け入れる IAM ユーザーまたはロールに追加します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "omics:*",
        "ram:AcceptResourceShareInvitation",
        "ram:GetResourceShareInvitations"
      ],
      "Resource": "*"
    }
  ]
}
```

クエリ用の Athena の設定

Athena を使用してバリアントと注釈をクエリできます。クエリを実行する前に、次のセットアップタスクを実行します。

トピック

- Athena コンソールを使用してクエリ結果の場所を設定する
- Athena エンジン v3 でワークグループを設定する

Athena コンソールを使用してクエリ結果の場所を設定する

クエリ結果の場所を設定するには、次の手順に従います。

1. Athena コンソールを開く: Athena コンソール
2. プライマリナビゲーションバーで、クエリエディタを選択します。
3. クエリエディタで、設定タブを選択し、管理を選択します。
4. クエリ結果を保存する場所の S3 プレフィックスを入力します。

Athena エンジン v3 でワークグループを設定する

ワークグループを設定するには、次の手順に従います。

1. Athena コンソールを開く: Athena コンソール
2. プライマリナビゲーションバーで、ワークグループを選択し、ワークグループを作成します。
3. ワークグループの名前を入力します。
4. エンジンのタイプとして Athena SQL を選択します。
5. アップグレードクエリエンジンで、手動を選択します。
6. クエリバージョンエンジンで、Athena バージョン 3 を選択します。
7. [Create workgroup] (ワークグループの作成) を選択します。

HealthOmics バリアントストアでのクエリの実行

Amazon Athena を使用してバリアントストアでクエリを実行できます。バリアントストアと注釈ストアのゲノム座標は、ゼロベース、半クローズの半オープン間隔として表されることに注意してください。

Athena コンソールを使用してシンプルなクエリを実行する

次の例は、単純なクエリを実行する方法を示しています。

1. Athena クエリエディタを開く: Athena クエリエディタ
2. ワークグループ で、セットアップ中に作成したワークグループを選択します。
3. データソースが AwsDataCatalog であることを確認します。
4. データベースで、Lake Formation のセットアップ中に作成したデータベースリソースリンクを選択します。
5. クエリ 1 タブのクエリエディタに次のクエリをコピーします。

```
SELECT * from omicsvariants limit 10
```

6. クエリを実行するには、[実行] を選択します。コンソールは、結果テーブルにomicsvariantsテーブルの最初の 10 行を入力します。

Athena コンソールを使用して複雑なクエリを実行する

次の例は、複雑なクエリを実行する方法を示しています。このクエリを実行するには、注釈ストアClinVarに をインポートします。

複雑なクエリを実行する

1. Athena クエリエディタを開く: Athena クエリエディタ
2. ワークグループ で、セットアップ中に作成したワークグループを選択します。
3. データソースが AwsDataCatalog であることを確認します。
4. データベース で、Lake Formation のセットアップ中に作成したデータベースリソースリンクを選択します。
5. 右上+の を選択して、クエリ 2 という名前の新しいクエリタブを作成します。
6. クエリ 2 タブのクエリエディタに次のクエリをコピーします。

```
SELECT variants.sampleid,
  variants.contigname,
  variants.start,
  variants."end",
  variants.referenceallele,
  variants.alternatealleles,
  variants.attributes AS variant_attributes,
  clinvar.attributes AS clinvar_attributes
FROM omicsvariants as variants
INNER JOIN omicsannotations as clinvar ON
  variants.contigname=CONCAT('chr',clinvar.contigname)
  AND variants.start=clinvar.start
  AND variants."end"=clinvar."end"
  AND variants.referenceallele=clinvar.referenceallele
  AND variants.alternatealleles=clinvar.alternatealleles
WHERE clinvar.attributes['CLNSIG']='Likely_pathogenic'
```

7. Run を選択してクエリの実行を開始します。

HealthOmics 分析ストアの共有

バリアントストアまたは注釈ストアの所有者として、そのストアを他の AWS アカウントと共有できます。所有者は、共有を削除することで、共有リソースへのアクセスを取り消すことができます。

共有ストアのサブスクライバーは、まず共有を受け入れます。その後、共有ストアを使用するワークフローを定義できます。データは、 AWS Glue と Lake Formation の両方でテーブルとして表示されます。

ストアへのアクセスが不要になった場合は、共有を削除します。

リソース共有の詳細については[でのクロスアカウントリソース共有 AWS HealthOmics](#)、「」を参照してください。

ストア共有の作成

ストア共有を作成するには、create-share API オペレーションを使用します。プリンシパルサブスクライバーは、共有をサブスクライブする AWS アカウント ユーザーの です。次の の例では、バリアントストアの共有を作成します。ストアを複数のアカウントと共有するには、同じストアの複数の共有を作成します。

```
aws omics create-share  \
        --resource-arn "arn:aws:omics:us-west-2:555555555555:variantStore/
omics_dev_var_store" \
        --principal-subscriber "123456789012" \
        --name "my_Share-123"
```

作成が成功すると、共有 ID とステータスを含むレスポンスを受け取ります。

```
{
       "shareId": "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a",
       "name": "my_Share-123",
       "status": "PENDING"
  }
```

共有は、サブスクライバーが accept-share API オペレーションを使用して承諾するまで保留状態のままです。

でのクロスアカウントリソース共有 AWS HealthOmics

クロスアカウント共有を使用すると、コピーを作成したり、IAM リソースポリシーを変更したりすることなく、共同作業者とリソースを共有できます。次のリソースは、クロスアカウント共有をサポートしています。

- HealthOmics バリアントストア
- HealthOmics 注釈ストア
- プライベートワークフロー

リソースの共有には、次のステップが含まれます。

1. リソース所有者は共有を作成し、リソースの ARN と目的のサブスクライバー AWS アカウントの を指定します。リソース共有は、サブスクライバーが共有を受け入れるまで保留状態のままです。
2. サブスクライバーはリソース共有を受け入れて、リソースにアクセスします。リソース共有はアクティブ化状態に移行します。
3. HealthOmics サービスは、サブスクライバーアカウントにリソースへのアクセスを提供します。
4. リソース所有者は共有を削除するか、サブスクライバーは共有へのアクセスを取り消すことができます。サブスクライバーは、共有または関連付けられたリソースを削除することはできません。

トピック

共有の作成

create-share API オペレーションを使用して共有を作成できます。プリンシパルサブスクライバーは、共有リソースをサブスクライブする AWS アカウント ユーザーの です。次の の例では、バリアントストアの共有を作成します。

```
aws omics create-share \
    --resource-arn "arn:aws:omics:us-west-2:555555555555:variantStore/
omics_dev_var_store" \
    --principal-subscriber "123456789012"  \
    --name "my_Share-123"
```

作成が成功すると、共有 ID とステータスを含むレスポンスを受け取ります。

```
{
"shareId": "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a",
     "name": "my_Share-123",
     "status": "PENDING"
}
```

共有は、サブスクライバーが accept-share API オペレーションを使用して承諾するまで保留状態のままです。

```
 aws omics accept-share \
    --share-id "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a"
```

サブスクライバーが共有を受け入れると、共有はアクティブ状態に移行します。

```
{
"status": "ACTIVATING"
}
```

共有に関する情報を取得する

get-share API オペレーションを使用して、共有に関する情報を取得します。

```
aws omics get-share --share-id "495c21bedc889d07d0ab69d710a6841e-
dd75ab7a1a9c384fa848b5bd8e5a7e0a"
```

API レスポンスには、共有に関するメタデータ情報が含まれます。

```
{
  "share":
    {
      "shareId": "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a",
      "name": "my_Share-123",
      "resourceArn": "arn:aws:omics:us-west-2:555555555555:variantStore/
omics_dev_var_store",
      "principalSubscriber": "123456789012",
      "ownerId": "555555555555",
      "status": "PENDING"
    }
}
```

所有している共有を表示する

list-shares API を使用して、所有する各共有に関する情報を取得します。

```
aws omics list-shares  --resource-owner SELF
```

API レスポンスには、所有する各共有のメタデータが含まれます。

他のアカウントから受け入れられた共有を表示する

list-shares API を使用して、他のアカウントから承諾したすべての共有を表示します。

```
aws omics list-shares  --resource-owner OTHER
```

API レスポンスには、承諾した各共有のメタデータが含まれます。

共有を削除する

削除共有 API を使用して、不要になった共有を削除します。

```
aws omics delete-share \
    --share-id "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a"
```

HealthOmics でのリソースのタグ付け

トピック

重要な注意点

HealthOmics は、AWS 責任共有モデルポリシーに基づいて顧客データを保護します。つまり、すべての顧客データは移行中と保管中の両方で暗号化されます。ただし、データストアやジョブベースのオペレーションなどのリソースのお客様が入力したすべての名前が暗号化されるわけではありません。個人を特定できる情報や保護対象医療情報は含めないでください。詳細については、「AWS HealthOmics のセキュリティ」を参照してください。

HealthOmics リソースのタグ付け

タグを使用して AWS リソースにメタデータを割り当てることができます。各タグは、ユーザー定義のキーと値で構成されるラベルです。タグは、リソースの管理、識別、整理、検索、フィルタリングに役立ちます。

このトピックでは、一貫性のある効果的なタグ付け戦略を実装する目的で広く使用されているタグ付けカテゴリと戦略について説明します。以下のセクションでは、AWS リソース、タグ付け、請求の詳細、および に関する基本的な知識を前提としています AWS Identity and Access Management。

各 タグは 2 つの部分で構成されます:

- タグキー (CostCenter、Environment、Project など) タグキーでは、大文字と小文字が区別されます。
- タグ値 (111122223333 や Production など) 。タグキーと同様に、タグ値では大文字と小文字が区別されます。

タグを使用し、リソースを目的、所有者、環境などの基準別に分類できます。詳細については、「AWS タグ付け戦略」を参照してください。

リソースのタグは、リソースのサービスコンソール、サービス API、または から追加、変更、または削除できます AWS CLI。

タグ付けを有効にするには、TagResources が承認されていることを確認します。次の例のような IAM ポリシーをアタッチすることで、TagResources を承認できます。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "omics:Create*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "omics:Start*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "omics:Tag*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "omics:Untag*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "omics:List*",
            "Resource": "*"
        }
    ]
}
```

ベストプラクティス

AWS リソースのタグ付け戦略を作成するときは、ベストプラクティスに従います。

- 個人を特定できる情報 (PII)、保護対象医療情報 (PHI)、またはその他の機密情報をタグに保存しないでください。
- タグには、標準化された、大文字と小文字の区別がある形式を使用し、すべてのリソースタイプに一貫して適用します。
- リソースアクセスコントロールの管理、コスト追跡、オートメーション、整理など、複数の目的に対応したタグガイドラインを考慮します。
- 自動化されたツールを使用して、リソースタグを管理できます。AWS Resource Groups と Resource Groups Tagging API を使用すると、タグをプログラムで制御できるため、タグとリソースを自動的に管理、検索、フィルタリングできます。
- タグ付けは、より多くのタグを使用する場合により効果的です。
- タグは、ユーザーのニーズの変化に応じて編集または変更できます。ただし、アクセスコントロールタグを更新するには、リソースへのアクセスを制御するためにこれらのタグを参照するポリシーも更新する必要があります。

タグ付け要件

タグには、次の要件があります。

- キーに aws: というプレフィックスを付けることはできません。
- キーはタグセットごとに一意であることが必要です。
- キーに使用できる文字数は 1 ~ 128 文字です。
- 値に使用できる文字数は 0 ~ 256 文字です。
- 値はタグセットごとに一意である必要はありません。
- キーと値に使用できる文字は、Unicode 文字、数字、空白、および _ . : / = + - @ の記号です。
- キーと値は大文字と小文字が区別されます。

シーケンスストアのリードセットタグ

シーケンスストアの場合、読み取りセットで作成されたタグは、読み取りセットリソースレベルにあります。リードセットには、S3 APIs を使用してアクセス、検索、制限できるオブジェクトも含ま

れています。デフォルトでは、サンプル ID (omics:sampleId) とサブジェクト ID (omics:subjectId) がオブジェクトに追加されます。

さらに、読み取りセットとその下のオブジェクト間で最大 5 つのタグを同期できます。同期するタグの設定は、 propogatedSetLevelTagsパラメータを使用してストアの作成または更新中に設定されたストアレベルの設定です。

ストアに既にデータがある場合、キーの更新には時間がかかる場合があります。この更新中、HealthOmics はストアのステータスを に変更しますUpdating。完了すると、HealthOmics はストアのステータスを に設定しますActive。タグが伝播している間は、タグに依存するアクセス許可が適用されない場合があります。アクセス許可は、タグの伝播が完了した後に適用されます。

タグが読み取りセットで設定または更新されると、システムはストア設定に基づいて、その読み取りセットのオブジェクトを更新するかどうかを決定します。

HealthOmics リソースへのタグの追加

リソースにタグを追加すると、AWS リソースを識別して整理し、リソースへのアクセスを管理するのに役立ちます。まず、リソースに 1 つ以上のタグ (キーと値のペア) を追加します。リソースごとに最大 50 個のタグを使用できます。キーフィールドと値フィールドで使用できる文字にも制限があります。

タグを追加したら、IAM ポリシーを作成して、これらのタグに基づいてリソースへのアクセス AWS を管理できます。HealthOmics コンソールまたは を使用して AWS CLI 、リソースにタグを追加できます。リポジトリにタグを追加すると、追加したリポジトリに影響が生じる場合があります。データストアにタグを追加する前に、タグを使用してデータストアなどのリソースへのアクセスを制御する可能性のある IAM ポリシーを確認してください。

サービスタグは、件名とシーケンスストアのサンプル ID の両方に対して自動生成されます。

を使用して HealthOmics リソースにタグ AWS CLI を追加するには、次の手順に従います。たとえば、シーケンスストアの作成中にタグを追加するには、 で次のコマンドを使用します AWS CLI。シーケンスストアの名前は MySequenceStore で、キーを持つ 2 つの追加タグは key1 と key2 で、値はそれぞれ value1 と value2 です。
:

```
aws omics create-sequence-store --name "MySequenceStore" --tags key1=value1,key2=value2
```

出力はタグを一覧表示しません。次のレスポンスが返されます。

```
{
    "id": "6860403586",
    "referenceStoreId": "4889894479",
    "roleArn": "arn:aws:iam::555555555555:role/ImportTest",
    "status": "CREATED",
    "creationTime": "2022-07-21T01:19:07.194Z"
}
```

既存のリソースにタグを追加するには、次のコマンド例を実行します。

```
aws omics tag-resource --resource-arn arn:aws:omics:us-
west-2:555555555555:sequenceStore/2275234794 --tags key1=value1,key2=value2
```

正常に実行されると、このコマンドは空のレスポンスを返します。

リソースのタグを一覧表示します

を使用して HealthOmics リソースの AWS タグのリスト AWS CLI を表示するには、次の手順に従います。タグが追加されていない場合、返されるリストは空になります。

ターミナルまたはコマンドラインで、次の例に示すように list-tags-for-resource コマンドを実行します。

```
aws omics list-tags-for-resource --resource-arn arn:aws:omics:us-
west-2:555555555555:sequenceStore/2275234794
```

タグのリストが JSON 形式で返されます。

```
 {
    "tags": {
        "key1": "value1",
        "key2": "value2"
    }
}
```

データストアからのタグの削除

リソースに関連付けられた 1 つ以上のタグを削除できます。タグを削除しても、そのタグに関連付けられた他の AWS リソースからタグを削除することにはなりません。

ターミナルまたはコマンドラインで untag-resource コマンドを実行し、タグを削除するリソースの Amazon リソースネーム (ARN) と、削除するタグのタグキーを指定します。

```
aws omics untag-resource --resource-arn arn:aws:omics:us-
west-2:555555555555:sequenceStore/2275234794 --tag-keys key1,key2
```

成功した場合、このコマンドはレスポンスを返しません。リソースに関連付けられているタグを確認するには、list-tags-for-resource コマンドを実行します。

HealthOmics の IAM アクセス許可

AWS Identity and Access Management (IAM) を使用して、HealthOmics API およびストアやワークフローなどのリソースへのアクセスを管理できます。HealthOmics を使用するアカウントのユーザーおよびアプリケーションの場合、IAM ユーザー、グループ、またはロールに適用できるアクセス許可ポリシーでアクセス許可を管理します。

アカウント内のユーザーとアプリケーションのアクセス許可を管理するには、HealthOmics が提供するポリシーを使用するか、独自のポリシーを作成します。HealthOmics コンソールは、複数のサービスを使用して、関数の設定とトリガーに関する情報を取得します。提供されたポリシーをそのまま使用することも、より制限の厳しいポリシーの開始点として使用することもできます。

HealthOmics は IAM サービスロールを使用して、ユーザーに代わって他のサービスにアクセスします。例えば、Amazon S3 からデータを読み取るワークフローを実行するときに、サービスロールを作成または選択します。一部の機能では、他の サービスのリソースに対するアクセス許可も設定する必要があります。HealthOmics の使用を開始する前に、これらの要件を確認してください。

詳細については、IAM ユーザーガイドの 「IAM とは」を参照してください。

トピック

- HealthOmics のアイデンティティベースの IAM ポリシー
- のサービスロール AWS HealthOmics
- リソースのアクセス許可
- Amazon S3 URIs を使用したデータアクセスのアクセス許可

HealthOmics のアイデンティティベースの IAM ポリシー

アカウントのユーザーに HealthOmics へのアクセスを許可するには、 AWS Identity and Access Management (IAM) でアイデンティティベースのポリシーを使用します。ID ベースのポリシーは、IAM ユーザー、またはユーザーに関連付けられている IAM グループとロールに直接適用できます。また、別のアカウントのユーザーに、アカウントのロールを引き受けて HealthOmics リソースにアクセスするアクセス許可を付与することもできます。

ワークフローバージョンでアクションを実行するアクセス許可をユーザーに付与するには、ワークフローと特定のワークフローバージョンをリソースリストに追加する必要があります。

次の IAM ポリシーは、ユーザーがすべての HealthOmics API アクションにアクセスし、サービスロールを HealthOmics に渡すことを許可します。

Example ユーザーポリシー

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "omics:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "omics.amazonaws.com"
        }
      }
    }
  ]
}
```

HealthOmics を使用する場合は、他の AWS のサービスともやり取りします。これらのサービスにアクセスするには、各サービスが提供する管理ポリシーを使用します。リソースのサブセットへのアクセスを制限するには、管理ポリシーを開始点として使用して、より制限の厳しい独自のポリシーを作成できます。

- AmazonS3FullAccess – ジョブで使用される Amazon S3 バケットとオブジェクトへのアクセス。

- AmazonEC2ContainerRegistryFullAccess – ワークフローコンテナイメージの Amazon ECR レジストリとリポジトリへのアクセス。
- AWSLakeFormationDataAdmin – 分析ストアによって作成された Lake Formation データベースとテーブルへのアクセス。
- ResourceGroupsandTagEditorFullAccess – HealthOmics リソースに HealthOmics タグ付け API オペレーションをタグ付けします。

上記のポリシーでは、ユーザーが IAM ロールを作成することはできません。これらのアクセス許可を持つユーザーがジョブを実行するには、管理者は HealthOmics にデータソースへのアクセス許可を付与するサービスロールを作成する必要があります。詳細については、「のサービスロール AWS HealthOmics」を参照してください。

実行のカスタム IAM アクセス許可を定義する

StartRun リクエストによって参照されるワークフロー、実行、または実行グループを認可リクエストに含めることができます。これを行うには、ワークフロー、実行、または実行グループの必要な組み合わせを IAM ポリシーに一覧表示します。たとえば、ワークフローの使用を特定の実行または実行グループに制限できます。ワークフローを実行グループでのみ使用するように指定することもできます。

以下は、単一の実行グループを持つ単一のワークフローを許可する IAM ポリシーの例です。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "omics:StartRun"
          ],
          "Resource": [
              "arn:aws:omics:us-west-2:123456789012:workflow/1234567",
              "arn:aws:omics:us-west-2:123456789012:runGroup/2345678"
          ]
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "omics:StartRun"
            ],
            "Resource": [
                "arn:aws:omics:us-west-2:123456789012:run/*",
                "arn:aws:omics:us-west-2:123456789012:runGroup/2345678"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "omics:GetRun",
                "omics:ListRunTasks",
                "omics:GetRunTask",
                "omics:CancelRun",
                "omics:DeleteRun"
            ],
            "Resource": [
                "arn:aws:omics:us-west-2:123456789012:run/*"
            ]
        }
    ]
}
```

のサービスロール AWS HealthOmics

サービスロールは、 アカウントのリソースにアクセスするためのアクセス許可を AWS サービスに付与する AWS Identity and Access Management (IAM) ロールです。インポートジョブを開始するとき、または実行を開始する AWS HealthOmics ときに、 にサービスロールを指定します。

HealthOmics コンソールで必要なロールを作成できます。HealthOmics API を使用してリソースを管理する場合は、IAM コンソールを使用してサービスロールを作成します。詳細については、[「 にアクセス許可を委任するロールを作成する AWS のサービス」](#)を参照してください。

サービスロールには、次の信頼ポリシーが必要です。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "omics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

信頼ポリシーは、HealthOmics サービスがロールを引き受けることを許可します。

トピック

- IAM サービスポリシーの例
- AWS CloudFormation テンプレートの例

IAM サービスポリシーの例

これらの例では、リソース名とアカウント IDsは、 を実際の値に置き換えるためのプレースホルダーです。

次の例は、実行の開始に使用できるサービスロールのポリシーを示しています。このポリシーは、Amazon S3 出力場所、ワークフローロググループ、および実行用の Amazon ECR コンテナにアクセスするためのアクセス許可を付与します。

> **Note**
>
> 実行にコールキャッシュを使用している場合は、s3 アクセス許可のリソースとして実行キャッシュの Amazon S3 の場所を追加します。

Example 実行を開始するためのサービスロールポリシー

JSON

```
{
"Version": "2012-10-17",
"Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "s3:GetObject",
              "s3:PutObject"
          ],
          "Resource": [
              "arn:aws:s3:::amzn-s3-demo-bucket1/*"
          ]
      },
      {
          "Effect": "Allow",
          "Action": [
              "s3:ListBucket"
          ],
          "Resource": [
              "arn:aws:s3:::amzn-s3-demo-bucket1"
          ]
      },
      {
          "Effect": "Allow",
          "Action": [
              "logs:DescribeLogStreams",
              "logs:CreateLogStream",
              "logs:PutLogEvents"
          ],
          "Resource": [
              "arn:aws:logs:us-east-1:123456789012:log-group:/aws/omics/
WorkflowLog:log-stream:*"
          ]
      },
      {
          "Effect": "Allow",
          "Action": [
              "logs:CreateLogGroup"
          ],
```

```
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:/aws/omics/
WorkflowLog:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchCheckLayerAvailability"
            ],
            "Resource": [
                "arn:aws:ecr:us-east-1:123456789012:repository/*"
            ]
        }
    ]
}
```

次の例は、ストアのインポートジョブに使用できるサービスロールのポリシーを示しています。このポリシーは、Amazon S3 の入力場所 にアクセスするためのアクセス許可を付与します。

Example リファレンスストアジョブのサービスロール

JSON

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

        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }

    ]
}
```

AWS CloudFormation テンプレートの例

次のサンプル AWS CloudFormation テンプレートは、 というプレフィックスが付いた名前の Amazon S3 バケットにアクセスしomics-、ワークフローログをアップロードするアクセス許可を HealthOmics に付与するサービスロールを作成します。

Example リファレンスストア、Amazon S3、CloudWatch Logs のアクセス許可

```
Parameters:
  bucketName:
    Description: Bucket name
    Type: String

Resources:
  serviceRole:
    Type: AWS::IAM::Role
    Properties:
      Policies:
        - PolicyName: read-reference
          PolicyDocument:
            Version: 2012-10-17
            Statement:
            - Effect: Allow
              Action:
                - omics:*
              Resource: !Sub arn:${AWS::Partition}:omics:${AWS::Region}:
${AWS::AccountId}:referenceStore/*
        - PolicyName: read-s3
          PolicyDocument:
            Version: 2012-10-17
            Statement:
            - Effect: Allow
```

```
              Action:
                - s3:ListBucket
              Resource: !Sub arn:${AWS::Partition}:s3:::${bucketName}
            - Effect: Allow
              Action:
                - s3:GetObject
                - s3:PutObject
              Resource: !Sub arn:${AWS::Partition}:s3:::${bucketName}/*
        - PolicyName: upload-logs
          PolicyDocument:
            Version: 2012-10-17
            Statement:
            - Effect: Allow
              Action:
                - logs:DescribeLogStreams
                - logs:CreateLogStream
                - logs:PutLogEvents
              Resource: !Sub arn:${AWS::Partition}:logs:${AWS::Region}:
${AWS::AccountId}:loggroup:/aws/omics/WorkflowLog:log-stream:*
            - Effect: Allow
              Action:
                - logs:CreateLogGroup
              Resource: !Sub arn:${AWS::Partition}:logs:${AWS::Region}:
${AWS::AccountId}:loggroup:/aws/omics/WorkflowLog:*
      AssumeRolePolicyDocument: |
        {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "omics.amazonaws.com"
                ]
              }
            }
          ]
        }
```

リソースのアクセス許可

AWS HealthOmics は、ジョブの実行時またはストアの作成時に、ユーザーに代わって他の サービスのリソースを作成してアクセスします。場合によっては、リソースにアクセスしたり、HealthOmics がリソースにアクセスしたりできるように、他の サービスでアクセス許可を設定する必要があります。

セクション

Amazon ECR のアクセス許可

HealthOmics サービスがプライベート Amazon ECR リポジトリからコンテナでワークフローを実行する前に、コンテナのリソースポリシーを作成します。このポリシーは、HealthOmics サービスがコンテナを使用するためのアクセス許可を付与します。このリソースポリシーは、ワークフローによって参照される各プライベートリポジトリに追加します。

> **Note**
>
> プライベートリポジトリとワークフローは同じリージョンにある必要があります。

以下のセクションでは、必要なポリシー設定について説明します。

トピック

Amazon ECR リポジトリのリソースポリシーを作成する

リソースポリシーを作成して、HealthOmics サービスがリポジトリ内のコンテナを使用してワークフローを実行できるようにします。このポリシーは、HealthOmics サービスプリンシパルが必要な Amazon ECR アクションにアクセスするためのアクセス許可を付与します。

ポリシーを作成するには、次の手順に従います。

1. Amazon ECR コンソールでプライベートリポジトリページを開き、アクセス権を付与するリポジトリを選択します。
2. サイドバーのナビゲーションから、アクセス許可を選択します。
3. JSON の編集 を選択します。
4. [Add Statement (ステートメントの追加)] を選択します。
5. 次のポリシーステートメントを追加し、ポリシーの保存を選択します。

   JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "omics workflow access",
            "Effect": "Allow",
            "Principal": {
                "Service": "omics.amazonaws.com"
            },
            "Action": [
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage",
                "ecr:BatchCheckLayerAvailability"
            ],
            "Resource": "*"
        }
    ]
}
```

クロスアカウントコンテナを使用したワークフローの実行

ワークフローとコンテナを所有する AWS アカウントが異なる場合は、次のクロスアカウントアクセス許可を設定する必要があります。

1. リポジトリの Amazon ECR ポリシーを更新して、ワークフローを所有するアカウントにアクセス許可を明示的に付与します。

2. ワークフローを所有するアカウントのサービスロールを更新して、コンテナイメージへのアクセスを許可します。

次の例は、ワークフローを所有するアカウントへのアクセスを許可する Amazon ECR リソースポリシーを示しています。

この例では、以下のようになっています：

- ワークフローアカウント ID: 111122223333
- コンテナリポジトリアカウント ID: 444455556666
- コンテナ名: samtools

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "omics.amazonaws.com"
            },
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ]
        },
        {
            "Sid": "allow access to the service role of the account that owns the
 workflow",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/DemoCustomer"
            },
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ]
```

```
        }
    ]
}
```

セットアップを完了するには、ワークフローを所有するアカウントのサービスロールに次のポリシーステートメントを追加します。このポリシーは、サービスロールが「samtools」コンテナイメージにアクセスするためのアクセス許可を付与します。アカウント番号、コンテナ名、リージョンを独自の値に置き換えてください。

```
{
    "Sid": "CrossAccountEcrRepoPolicy",
    "Effect": "Allow",
    "Action": ["ecr:BatchCheckLayerAvailability", "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer"],
    "Resource": "arn:aws:ecr:us-west-2:444455556666:repository/samtools"
}
```

共有ワークフローの Amazon ECR リポジトリポリシー

> **Note**
>
> HealthOmics は、ワークフローがサブスクライバーのアカウントで実行されている間、共有ワークフローがワークフロー所有者のアカウントの Amazon ECR リポジトリに自動的にアクセスできるようにします。共有ワークフローに追加のリポジトリアクセスを付与する必要はありません。詳細については、HealthOmics ワークフローの共有」を参照してください。

デフォルトでは、サブスクライバーは基盤となるコンテナを使用する Amazon ECR リポジトリにアクセスできません。必要に応じて、リポジトリのリソースポリシーに条件キーを追加することで、Amazon ECR リポジトリへのアクセスをカスタマイズできます。以下のセクションでは、ポリシーの例を示します。

特定のワークフローへのアクセスを制限する

条件ステートメントで個々のワークフローを一覧表示できるため、これらのワークフローのみがリポジトリ内のコンテナを使用できます。SourceArn 条件キーは、共有ワークフローの ARN を指定します。次の の例では、指定されたワークフローにこのリポジトリを使用するアクセス許可を付与します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
     {
       "Sid": "OmicsAccessPrincipal",
       "Effect": "Allow",
       "Principal": {
         "Service": "omics.amazonaws.com"
       },
       "Action": [
         "ecr:GetDownloadUrlForLayer",
         "ecr:BatchGetImage",
         "ecr:BatchCheckLayerAvailability"
       ],
       "Resource": "*",
       "Condition": {
          "StringEquals": {
             "aws:SourceArn": "arn:aws:omics:us-
east-1:111122223333:workflow/1234567"
          }
       }
     }
  ]
}
```

特定のアカウントへのアクセスを制限する

条件ステートメントにサブスクライバーアカウントを一覧表示して、これらのアカウントのみがリポジトリ内のコンテナを使用するアクセス許可を持つようにできます。SourceAccount 条件キーは、サブスクライバー AWS アカウント の を指定します。次の の例では、指定されたアカウントにこのリポジトリを使用するアクセス許可を付与します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Sid": "OmicsAccessPrincipal",
      "Effect": "Allow",
      "Principal": {
        "Service": "omics.amazonaws.com"
      },
      "Action": [
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

次のポリシー例に示すように、特定のサブスクライバーに対する Amazon ECR アクセス許可を拒否することもできます。

JSON

```
{
      "Version": "2012-10-17",
      "Statement": [
          {
              "Sid": "OmicsAccessPrincipal",
              "Effect": "Allow",
              "Principal": {
                  "Service": "omics.amazonaws.com"
              },
              "Action": [
                  "ecr:GetDownloadUrlForLayer",
                  "ecr:BatchGetImage",
                  "ecr:BatchCheckLayerAvailability"
              ],
              "Resource": "*",
              "Condition": {
```

```
                "StringNotEquals": {
                  "aws:SourceAccount": "111122223333"
                }
            }
        }
    ]
}
```

Lake Formation 許可

HealthOmics で分析機能を使用する前に、Lake Formation でデフォルトのデータベース設定を行います。

Lake Formation でリソースのアクセス許可を設定するには

1. Lake Formation コンソールでデータカタログ設定ページを開きます。
2. 新しく作成されたデータベースとテーブルのデフォルトのアクセス許可で、データベースとテーブルの IAM アクセスコントロール要件のチェックを解除します。
3. [保存] を選択します。

HealthOmics Analytics は、サービスポリシーに次の例のような正しい RAM アクセス許可がある場合、データを自動的に受け入れます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "omics:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ram:AcceptResourceShareInvitation",
```

```
        "ram:GetResourceShareInvitations"
      ],
      "Resource": "*"
    }
  ]
}
```

Amazon S3 URIs を使用したデータアクセスのアクセス許可

HealthOmics API オペレーションまたは Amazon S3 API オペレーションを使用して、シーケンスストアデータにアクセスできます。

HealthOmics API アクセスの場合、HealthOmics のアクセス許可は IAM ポリシーを通じて管理されます。ただし、S3 Access には、ストアの S3 アクセスポリシーでの明示的な許可と IAM ポリシーの 2 つのレベルの設定が必要です。HealthOmics での IAM ポリシーの使用の詳細については、HealthOmics のサービスロール」を参照してください。

Amazon S3 API を使用してオブジェクトを読み取る機能を共有するには、次の 3 つの方法があります。 APIs

1. ポリシーベースの共有 – この共有では、S3 アクセスポリシーで IAM プリンシパルを有効にし、IAM ポリシーを記述して IAM プリンシパルにアタッチする必要があります。詳細については、次のトピックを参照してください。
2. 署名付き URLs – シーケンスストア内のファイルの共有可能な署名付き URL を生成することもできます。Amazon S3 を使用した署名付き URLs の作成の詳細については、Amazon S3 S3 ドキュメントの「署名付き URLs」を参照してください。シーケンスストア S3 アクセスポリシーは、署名付き URL 機能を制限するためのステートメントをサポートしています。
3. 引き受けたロール – データ所有者のアカウント内に、ユーザーがそのロールを引き受けることを許可するアクセスポリシーを持つロールを作成します。

トピック

ポリシーベースの共有

直接 S3 URI を使用してシーケンスストアデータにアクセスする場合、HealthOmics は関連する S3 バケットアクセスポリシーのセキュリティ対策を強化します。

新しい S3 アクセスポリシーには、次のルールが適用されます。既存のポリシーの場合、ポリシーを次に更新するときにルールが適用されます。

- S3 アクセスポリシーは、次のポリシー要素をサポートします。
  - バージョン、ID、ステートメント、Sid、効果、プリンシパル、アクション、リソース、条件
- S3 アクセスポリシーは、次の条件キーをサポートしています。
  - s3:ExistingObjectTag/<key>、s3:prefix、s3:signatureversion、s3:TlsVersion
  - ポリシーは、次の条件演算子を持つ aws:PrincipalArn もサポートしています: ArnEquals と ArnLike

サポートされていない要素または条件を含めるようにポリシーを追加または更新しようとすると、システムはリクエストを拒否します。

トピック

- デフォルトの S3 アクセスポリシー
- アクセスポリシーのカスタマイズ
- IAM ポリシー
- タグベースのアクセス制御

デフォルトの S3 アクセスポリシー

シーケンスストアを作成すると、HealthOmics はデフォルトの S3 アクセスポリシーを作成し、データストア所有者のルートアカウントに、シーケンスストア内のすべてのアクセス可能なオブジェクトに対して、S3:GetObject、S3GetObjectTagging、S3:ListBucket のアクセス許可を付与します。デフォルトで作成されるポリシーは次のとおりです。

JSON

```
{
    "Version": "2012-10-17",
    "Statement":
```

```
    [
        {
            "Effect": "Allow",
            "Principal":
            {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action":
            [
                "s3:GetObject",
                "s3:GetObjectTagging"
            ],
            "Resource": "arn:aws:s3:us-
west-2:222222222222:accesspoint/111111111111-1234567890/object/111111111111/
sequenceStore/1234567890/*"
        },
        {
            "Effect": "Allow",
            "Principal":
            {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:us-
west-2:222222222222:accesspoint/111111111111-1234567890/111111111111/
sequenceStore/1234567890/*"
        }
    ]
}
```

アクセスポリシーのカスタマイズ

S3 アクセスポリシーが空白の場合、S3 アクセスは許可されません。既存のポリシーがあり、s3 アクセスを削除する必要がある場合は、 deleteS3AccessPolicy を使用してすべてのアクセスを削除します。

共有の制限を追加したり、他のアカウントへのアクセスを許可したりするには、PutS3AccessPolicy API を使用してポリシーを更新できます。ポリシーの更新は、シーケンスストアのプレフィックスまたは指定されたアクションを超えることはできません。

IAM ポリシー

Amazon S3 APIs を使用してユーザーまたは IAM プリンシパルにアクセスを許可するには、S3 アクセスポリシーのアクセス許可に加えて、IAM ポリシーを作成してプリンシパルにアタッチし、アクセスを許可する必要があります。Amazon S3 API アクセスを許可するポリシーは、シーケンスストアレベルまたは読み取りセットレベルで適用できます。読み取りセットレベルでは、プレフィックスまたはサンプル ID パターンまたはサブジェクト ID パターンのリソースタグフィルターを使用して、アクセス許可を制限できます。

シーケンスストアがカスタマーマネージドキー (CMK) を使用している場合、プリンシパルには復号化に KMS キーを使用する権限も必要です。詳細については、「 AWS Key Management Service デベロッパーガイド」の「クロスアカウント KMS アクセス」を参照してください。

次の例では、シーケンスストアへのアクセス権をユーザーに付与します。追加の条件またはリソースベースのフィルターを使用して、アクセスを微調整できます。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:root"
      },
      "Action":
      [
         "s3:GetObject",
         "s3:GetObjectTagging"
      ],
      "Resource": "arn:aws:s3:us-
west-2:222222222222:accesspoint/111111111111-1234567890/object/111111111111/
sequenceStore/1234567890/*",
      "Condition": {
        "StringEquals": {
          "s3:ExistingObjectTag/omics:readSetStatus": "ACTIVE"
        }
      }
    },
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:root"
      },
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:us-
west-2:222222222222:accesspoint/111111111111-1234567890",
      "Condition": {
        "StringLike": {
          "s3:prefix": "111111111111/sequenceStore/1234567890/*"
        }
      }
    }
  ]
}
```

タグベースのアクセス制御

タグベースのアクセスコントロールを使用するには、まずシーケンスストアを更新して、使用されるタグキーを伝達する必要があります。この設定は、シーケンスストアの作成時または更新時に設定されます。タグが伝播されると、タグ条件を使用して制限をさらに追加できます。制限は、S3 アクセスポリシーまたは IAM ポリシーに設定できます。以下は、設定されるタブベースの S3 アクセスポリシーの例です。

```
{
    "Sid": "tagRestrictedGets",
    "Effect": "Allow",
    "Principal":
    {
        "AWS": "arn:aws:iam::<target_restricted_account_id>:root"
    },
    "Action":
    [
        "s3:GetObject",
        "s3:GetObjectTagging"
    ],
    "Resource": "arn:aws:s3:us-west-2:222222222222:accesspoint/111111111111-1234567890/
object/111111111111/sequenceStore/1234567890/*",
    "Condition":
    {
        "StringEquals":
        {
```

```
            "s3:ExistingObjectTag/tagKey1": "tagValue1",
            "s3:ExistingObjectTag/tagKey2": "tagValue2"
        }
    }
}
```

制限の例

シナリオ: データ所有者がユーザーが「取り消された」データをダウンロードする能力を制限できる共有の作成。

このシナリオでは、データ所有者 (アカウント #111111111111) がデータストアを管理していました。このデータ所有者は、研究者 (アカウント #999999999999) を含む幅広いサードパーティーユーザーとデータを共有します。データを管理する一環として、データ所有者は参加者データの取り消しリクエストを定期的に取得します。この取り消しを管理するために、データ所有者はまずリクエストの受信時に直接ダウンロードアクセスを制限し、最終的には要件に従ってデータを削除します。

このニーズを満たすために、データ所有者はシーケンスストアを設定し、各リードセットは「ステータス」のタグを受け取ります。このタグは、取り消しリクエストが通過した場合に「取り消し」に設定されます。タグがこの値に設定されたデータの場合、このファイルでユーザーが「getObject」を実行できないようにする必要があります。この設定を行うには、データ所有者が 2 つのステップを実行する必要があります。

ステップ 1. シーケンスストアでは、ステータスタグが伝播されるように更新されていることを確認します。これを行うには、 `createSequenceStore`または を呼び出す`propogatedSetLevelTags`ときに に「ステータス」キーを追加します。`updateSequenceStore`.

ステップ 2. ストアの s3 アクセスポリシーを更新して、ステータスタグが取り消されたオブジェクトの getObject を制限します。これは、 `PutS3AccesPolicy` API を使用してストアアクセスポリシーを更新することによって行われます。次のポリシーでは、オブジェクトを一覧表示するときに削除されたファイルを表示することはできますが、アクセスすることはできません。

- ステートメント 1 (restrictedGetWithdrawal): アカウント 999999999999 は、取り消されたオブジェクトを取得できません。
- ステートメント 2 (ownerGetAll): データ所有者であるアカウント 111111111111 は、取り消されたオブジェクトを含むすべてのオブジェクトを取得できます。
- ステートメント 3 (everyoneListAll): すべての共有アカウント 111111111111 および 999999999999 は、プレフィックス全体で ListBucketオペレーションを実行できます。

JSON

```
{
    "Version": "2012-10-17",
    "Statement":
    [
        {
            "Sid": "restrictedGetWithdrawal",
            "Effect": "Allow",
            "Principal":
            {
                "AWS": "arn:aws:iam::999999999999:root"
            },
            "Action":
            [
                "s3:GetObject",
                "s3:GetObjectTagging"
            ],
            "Resource": "arn:aws:s3:us-
west-2:222222222222:accesspoint/111111111111-1234567890/object/111111111111/
sequenceStore/1234567890/*",
            "Condition":
            {
                "StringNotEquals":
                {
                    "s3:ExistingObjectTag/status": "withdrawn"
                }
            }
        },
        {
            "Sid": "ownerGetAll",
            "Effect": "Allow",
            "Principal":
            {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action":
            [
                "s3:GetObject",
                "s3:GetObjectTagging"
            ],
```

```
            "Resource": "arn:aws:s3:us-
west-2:222222222222:accesspoint/111111111111-1234567890/object/111111111111/
sequenceStore/1234567890/*",
            "Condition":
            {
                "StringEquals":
                {
                    "s3:ExistingObjectTag/omics:readSetStatus": "ACTIVE"
                }
            }
        },
        {
            "Sid": "everyoneListAll",
            "Effect": "Allow",
            "Principal":
            {
                "AWS": [
                    "arn:aws:iam::111111111111:root",
                    "arn:aws:iam::999999999999:root"
                ]
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:us-
west-2:222222222222:accesspoint/111111111111-1234567890",
            "Condition":
            {
                "StringLike":
                {
                    "s3:prefix": "111111111111/sequenceStore/1234567890/*"
                }
            }
        }
    ]
}
```

AWS HealthOmics のセキュリティ

で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からメリットを得られます。

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 – AWS は、 で AWS サービスを実行するインフラストラクチャを保護する責任を担います AWS クラウド。 は、お客様が安全に使用できるサービス AWS も提供します。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。AWS HealthOmics に適用されるコンプライアンスプログラムの詳細については、「コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム」を参照してください。
- 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、AWS HealthOmics 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の目的を達成するように AWS HealthOmics を設定する方法について説明します。また、AWS HealthOmics リソースのモニタリングや保護に役立つ他の AWS サービスの使用方法についても説明します。

トピック

- でのデータ保護 AWS HealthOmics
- HealthOmics での ID とアクセスの管理
- のコンプライアンス検証 AWS HealthOmics
- HealthOmics の耐障害性
- AWS HealthOmics およびインターフェイス VPC エンドポイント (AWS PrivateLink)

でのデータ保護 AWS HealthOmics

責任 AWS 共有モデル、AWS HealthOmics 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、 内のすべてのデフォルトのセキュリティコントロールを使用します AWS のサービス。
- Amazon Macie などの高度な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 AWS HealthOmics AWS CLIまたは他の AWS のサービス を使用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。

保管中の暗号化

トピック

- AWS 所有のキー
- カスタマーマネージドキー
- カスタマーマネージドキーの作成
- カスタマーマネージドキーを使用するために必要な IAM アクセス許可
- 詳細はこちら

保管中の機密データを保護するために、 はサービス所有の AWS Key Management Service (AWS KMS) キーを使用してデフォルトで暗号化 AWS HealthOmics を提供します。カスタマーマネージドキーもサポートされています。カスタマーマネージドキーの詳細については、「Amazon Key Management Service」を参照してください。

すべての HealthOmics データストア (ストレージと分析) は、カスタマーマネージドキーの使用をサポートしています。データストアの作成後に暗号化設定を変更することはできません。データストアが を使用している場合 AWS 所有のキー、AWS_OWNED_KMS_KEY と表示され、保管時の暗号化に使用される特定のキーは表示されません。

HealthOmics ワークフローの場合、カスタマーマネージドキーは一時ファイルシステムではサポートされていません。ただし、すべてのデータは保管時に XTS-AES-256 ブロック暗号暗号化アルゴリズムを使用して自動的に暗号化され、ファイルシステムが暗号化されます。ワークフロー実行を開始するために使用される IAM ユーザーとロールは、ワークフロー入出力バケットに使用される AWS KMS キーにもアクセスできる必要があります。ワークフローでは許可は使用されず、 AWS KMS 暗号化は Amazon S3 バケットの入力と出力に制限されます。ワークフロー APIs の両方に使用される IAM ロールには、使用される AWS KMS キーと、入出力 Amazon S3 バケットへのアクセス権も必要です。IAM ロールとアクセス許可を使用して、アクセスまたは AWS KMS ポリシーを制御できます。詳細については、「 の認証とアクセスコントロール AWS KMS」を参照してください。

HealthOmics Analytics AWS Lake Formation で を使用すると、Lake Formation に関連付けられた復号アクセス許可も入力および出力 Amazon S3 バケットに付与されます。がアクセス許可 AWS Lake Formation を管理する方法の詳細については、 AWS Lake Formation ドキュメントを参照してください。

HealthOmics Analytics は、Amazon S3 バケット内の暗号化されたデータを読み取るアクセス許可を Lake Formation kms:Decrypt に付与します。 Amazon S3 Lake Formation を介してデータをクエリするアクセス許可がある限り、暗号化されたデータを読み取ることができます。データへのアクセスは、KMS キーポリシーではなく、Lake Formation のデータアクセスコントロールによって制御されます。詳細については、Lake Formation ドキュメントのAWS 「Integrated AWS service requests」を参照してください。

AWS 所有のキー

デフォルトでは、HealthOmics は AWS 所有のキー を使用して保管中のデータを自動的に暗号化します。このデータには、個人を特定できる情報 (PII) や保護対象医療情報 (PHI) などの機密情報が含まれる可能性があるためです。 AWS 所有のキー aren はアカウントに保存されません。これらは、複数の AWS アカウントで使用するために AWS が所有および管理する KMS キーのコレクションの一部です。

AWS のサービスでは、 AWS 所有のキー を使用してデータを保護できます。を表示、管理、アクセスしたり AWS 所有のキー、それらの使用を監査したりすることはできません。ただし、データを暗号化するキーを保護するための作業やプログラムを操作したり変更したりする必要はありません。

の使用に対して月額料金や使用料金は請求されず AWS 所有のキー、アカウントの AWS KMS クォータにはカウントされません。詳細については、「AWS マネージドキー」を参照してください。

カスタマーマネージドキー

HealthOmics は、作成、所有、管理する対称カスタマーマネージドキーを使用して、既存の AWS 所有の暗号化に 2 番目の暗号化レイヤーを追加します。この暗号化レイヤーはユーザーが完全に制御できるため、次のようなタスクを実行できます。

- キーポリシー、IAM ポリシー、許可の確立と維持
- キー暗号化マテリアルのローテーション
- キーポリシーの有効化と無効化
- タグの追加
- キーエイリアスの作成
- 削除のためのキースケジューリング

CloudTrail を使用して、HealthOmics が AWS KMS ユーザーに代わって に送信するリクエストを追跡することもできます。別途 AWS KMS 料金がかかります。詳細については、「カスタマーマネージドキー」を参照してください。

カスタマーマネージドキーの作成

AWS マネジメントコンソールまたは AWS KMS APIs を使用して、対称カスタマーマネージドキーを作成できます。

AWS Key Management Service デベロッパーガイドの「対称カスタマーマネージドキーの作成」の手順に従います。

キーポリシーは、カスタマーマネージドキーへのアクセスを制御します。すべてのカスタマーマネージドキーには、キーポリシーが 1 つだけ必要です。このポリシーには、そのキーを使用できるユーザーとその使用方法を決定するステートメントが含まれています。カスタマーマネージドキーを作成するときに、キーポリシーを指定できます。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の管理」を参照してください。

HealthOmics Analytics リソースでカスタマーマネージドキーを使用するには、呼び出し元のプリンシパルがキーポリシーで kms:CreateGrant オペレーションを必要とします。これにより、システムは FAS トークンを使用して、指定された KMS キーへのアクセスを制御するカスタマーマネージドキーへの許可を作成できます。このキーは、HealthOmics が必要とする kms:grant オペレーションへのアクセス権をユーザーに付与します。 HealthOmics 詳細については、「許可の使用」を参照してください。

HealthOmics 分析では、呼び出し元のプリンシパルに次の API オペレーションを許可する必要があります。

- kms:CreateGrant は、特定のカスタマーマネージドキーに権限を追加し、HealthOmics Analytics の権限オペレーションへのアクセスを許可します。
- kms:DescribeKey は、キーの検証に必要なカスタマーマネージドキーの詳細を提供します。これはすべてのオペレーションに必要です。
- kms:GenerateDataKey は、すべての書き込みオペレーションで保管中のリソースを暗号化するためのアクセスを提供します。また、このアクションは、呼び出し元がキーを使用するためのアクセス権を持っていることをサービスが検証するために使用できるカスタマーマネージドキーの詳細を提供します。
- kms:Decrypt は、暗号化されたリソースの読み取りまたは検索オペレーションへのアクセスを提供します。

HealthOmics ストレージリソースでカスタマーマネージドキーを使用するには、HealthOmics サービスプリンシパルと呼び出し元プリンシパルをキーポリシーで許可する必要があります。これにより、呼び出し元がキーにアクセスできることをサービスで検証し、サービスプリンシパルを使用してカスタマーマネージドキーを使用してストア管理を実行できます。HealthOmics ストレージの場合、サービスプリンシパルのキーポリシーは、次の API オペレーションを許可する必要があります。

- kms:DescribeKey は、キーの検証に必要なカスタマーマネージドキーの詳細を提供します。これはすべてのオペレーションに必要です。
- kms:GenerateDataKey は、すべての書き込みオペレーションで保管中のリソースを暗号化するためのアクセスを提供します。また、このアクションは、呼び出し元がキーを使用するためのアクセス権を持っていることをサービスが検証するために使用できるカスタマーマネージドキーの詳細を提供します。
- kms:Decrypt は、暗号化されたリソースの読み取りまたは検索オペレーションへのアクセスを提供します。

次の例は、サービスプリンシパルがカスタマーマネージドキーを使用して暗号化された HealthOmics シーケンスまたはリファレンスストアを作成して操作できるようにするポリシーステートメントを示しています。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
{
    "Effect": "Allow",
    "Principal": {
        "Service": "omics.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:Encrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
        }
    ]
}
```

次の例は、Amazon S3 バケットからデータを復号するためのデータストアのアクセス許可を作成するポリシーを示しています。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "omics:GetReference",
                "omics:GetReferenceMetadata"
            ],
            "Resource": [
                "arn:AWS:omics:{{region}}:{{accountId}}:referenceStore/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:AWS:s3:::[[s3path]]/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:AWS:kms:{{region}}:{111122223333}:key/{{key_id}}"
            ],
            "Condition": {
              "StringEquals": {
                  "kms:ViaService": [
                    "s3.{{region}}.amazonAWS.com"
                  ]
              }
           }
        }
    ]
}
```

カスタマーマネージドキーを使用するために必要な IAM アクセス許可

カスタマーマネージドキーを使用して AWS KMS 暗号化されたデータストアなどのリソースを作成する場合、IAM ユーザーまたはロールのキーポリシーと IAM ポリシーの両方に必要なアクセス許可があります。

kms:ViaService 条件キーを使用して、KMS キーの使用を HealthOmics からのリクエストのみに制限できます。

キーポリシーの詳細については、AWS Key Management Service デベロッパーガイドの「IAM ポリシーの有効化」を参照してください。

トピック

Analytics API のアクセス許可

HealthOmics 分析の場合、ストアを作成する IAM ユーザーまたはロールには、kms:CreateGrant、kms:GenerateDataKey、kms:Decrypt、および kms:DescribeKey アクセス許可と、必要な HealthOmics アクセス許可が必要です。

Storage API のアクセス許可

HealthOmics ストレージ APIs、次の API オペレーションを呼び出す IAM ユーザーまたはロールには、リストされたアクセス許可が必要です。

CreateReferenceStore、CreateSequenceStore

ストアを作成するには、IAM 発信者がアクセス`kms:DescribeKey`許可と必要な HealthOmics アクセス許可を持っている必要があります。HealthOmics サービスプリンシパルは を呼び出し`kms:GenerateDataKeyWithoutPlaintext`て、データのロードとアクセスのアクセス検証チェックを実行します。

StartReadSetImportJob、StartReferenceImportJob

データインポートジョブを開始するには、IAM 呼び出し元がインポートするストアの KMS キーに対する `kms:Decrypt`および `kms:GenerateDataKey` アクセス許可と、インポートするオ

ブジェクトを含む Amazon S3 バケットに対する kms:Decrypt アクセス許可を持っている必要があります。さらに、呼び出しに渡されるロールには、インポートするオブジェクトを含む Amazon S3 バケットに対するkms:Decryptアクセス許可が必要です。IAM 呼び出し元には、ジョブにロールを渡すアクセス許可も必要です。

CreateMultipartReadSetUpload、UploadReadSetPart、CompleteMultipartReadSetUpload

マルチパートアップロードを完了するには、IAM 発信者がマルチパートアップロードを作成、アップロード、完了kms:Decryptkms:GenerateDataKeyするための と が必要です。

StartReadSetExportJob

データエクスポートジョブを開始するには、IAM 呼び出し元に、ストアの KMS キーが からエクスポートされるkms:Decryptアクセス許可kms:GenerateDataKeyと、オブジェクトを受信する Amazon S3 バケットに対するkms:Decryptアクセス許可が必要です。さらに、呼び出しに渡されるロールには、オブジェクトを受信する Amazon S3 バケットに対するkms:Decryptアクセス許可が必要です。IAM 呼び出し元には、ジョブにロールを渡すアクセス許可も必要です。

StartReadsetActivationJob

読み取りセットのアクティベーションジョブを開始するには、IAM 呼び出し元にオブジェクトに対する kms:Decryptおよび アクセスkms:GenerateDataKey許可が必要です。

GetReference、GetReadSet

ストアからオブジェクトを読み取るには、IAM 呼び出し元にオブジェクトに対するkms:Decryptアクセス許可が必要です。

読み取りセット S3 GetObject

Amazon S3 GetObject API を使用してストアからオブジェクトを読み取るには、IAM 呼び出し元にオブジェクトに対するkms:Decryptアクセス許可が必要です。カスタマーマネージドキーと AWS 所有のキー 設定の両方にこのアクセス許可を設定します。

HealthOmics が AWS KMS で許可を使用する方法

HealthOmics Analytics には、カスタマーマネージド KMS キーを使用するための許可が必要です。グラントは HealthOmics ワークフローでは必須ではなく、使用されません。HealthOmics Storage は、サービスプリンシパルから直接カスタマーマネージドキーを使用するため、グラントを使用しないでください。カスタマーマネージドキーで暗号化された分析ストアを作成すると、HealthOmics 分析は CreateGrant リクエストを AWS KMS に送信して、ユーザーに代わって許可を作成します。AWS KMS の許可は、顧客アカウントの KMS キーへのアクセス権を HealthOmics に付与するために使用されます。

HealthOmics 分析がユーザーに代わって作成する許可を取り消したり廃止したりすることはお勧めしません。アカウントで AWS KMS キーを使用するアクセス許可を HealthOmics に付与する許可を取り消しまたは廃止すると、HealthOmics はこのデータにアクセスしたり、データストアにプッシュされた新しいリソースを暗号化したり、プル時に復号したりすることはできません。

HealthOmics の許可を取り消すか廃止すると、変更はすぐに行われます。アクセス権を取り消すには、許可を取り消すのではなく、データストアを削除することをお勧めします。データストアを削除すると、HealthOmics はユーザーに代わって許可を廃止します。

AWS HealthOmics の暗号化キーのモニタリング

CloudTrail を使用して、カスタマーマネージドキーを使用するときに が AWS KMS ユーザーに代わって AWS HealthOmics に送信するリクエストを追跡できます。CloudTrail ログのログエントリは、HealthOmics によって行われたリクエストを明確に区別するために、userAgent フィールドに HealthOmics.amazonAWS.com を表示します。 userAgent HealthOmics

次の例は、カスタマーマネージドキーによって暗号化されたデータにアクセスするために HealthOmics によって呼び出される AWS KMS オペレーションをモニタリングするための CreateGrant、GenerateDataKey、Decrypt、および DescribeKey の CloudTrail イベントです。

次に、CreateGrant を使用して HealthOmics 分析が顧客提供の KMS キーにアクセスできるようにする方法も示し、HealthOmics がその KMS キーを使用して保管中のすべての顧客データを暗号化できるようにします。

独自の権限を作成する必要はありません。HealthOmics は、CreateGrant リクエストを AWS KMS に送信することで、ユーザーに代わって許可を作成します。の許可 AWS KMS は、顧客アカウントの AWS KMS キーへのアクセスを HealthOmics に許可するために使用されます。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "xx:test",
        "arn": "arn:AWS:sts::555555555555:assumed-role/user-admin/test",
        "accountId": "xx",
        "accessKeyId": "xxx",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "xxxx",
                "arn": "arn:AWS:iam::555555555555:role/user-admin",
                "accountId": "555555555555",
```

```
                "userName": "user-admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-11-11T01:36:17Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "apigateway.amazonAWS.com"
    },
    "eventTime": "2022-11-11T02:34:41Z",
    "eventSource": "kms.amazonAWS.com",
    "eventName": "CreateGrant",
    "AWSRegion": "us-west-2",
    "sourceIPAddress": "apigateway.amazonAWS.com",
    "userAgent": "apigateway.amazonAWS.com",
    "requestParameters": {
        "granteePrincipal": "AWS Internal",
        "keyId": "arn:AWS:kms:us-west-2:555555555555:key/a6e87d77-cc3e-4a98-a354-
e4c275d775ef",
        "operations": [
            "CreateGrant",
            "RetireGrant",
            "Decrypt",
            "GenerateDataKey"
        ]
    },
    "responseElements": {
        "grantId": "4869b81e0e1db234342842af9f5531d692a76edaff03e94f4645d493f4620ed7",
        "keyId": "arn:AWS:kms:us-west-2:245126421963:key/xx-cc3e-4a98-a354-
e4c275d775ef"
    },
    "requestID": "d31d23d6-b6ce-41b3-bbca-6e0757f7c59a",
    "eventID": "3a746636-20ef-426b-861f-e77efc56e23c",
    "readOnly": false,
    "resources": [
        {
            "accountId": "245126421963",
            "type": "AWS::KMS::Key",
            "ARN": "arn:AWS:kms:us-west-2:245126421963:key/xx-cc3e-4a98-a354-
e4c275d775ef"
        }
    ],
    "eventType": "AWSApiCall",
```

```
    "managementEvent": true,
    "recipientAccountId": "245126421963",
    "eventCategory": "Management"
}
```

次の例は、GenerateDataKey を使用して、ユーザーが保存する前にデータを暗号化するために必要なアクセス許可を持っていることを確認する方法を示しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "EXAMPLEUSER",
        "arn": "arn:AWS:sts::111122223333:assumed-role/Sampleuser01",
        "accountId": "111122223333",
        "accessKeyId": "EXAMPLEKEYID",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "EXAMPLEROLE",
                "arn": "arn:AWS:iam::111122223333:role/Sampleuser01",
                "accountId": "111122223333",
                "userName": "Sampleuser01"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2021-06-30T21:17:06Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedBy": "omics.amazonAWS.com"
    },
    "eventTime": "2021-06-30T21:17:37Z",
    "eventSource": "kms.amazonAWS.com",
    "eventName": "GenerateDataKey",
    "AWSRegion": "us-east-1",
    "sourceIPAddress": "omics.amazonAWS.com",
    "userAgent": "omics.amazonAWS.com",
    "requestParameters": {
        "keySpec": "AES_256",
        "keyId": "arn:AWS:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
    },
```

```
    "responseElements": null,
    "requestID": "EXAMPLE_ID_01",
    "eventID": "EXAMPLE_ID_02",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::KMS::Key",
            "ARN": "arn:AWS:kms:us-east-1:111122223333:key/EXAMPLE_KEY_ARN"
        }
    ],
    "eventType": "AWSApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```


詳細はこちら

以下のリソースは、保管時のデータの暗号化に関する詳細情報を提供します。

AWS Key Management Service の基本概念の詳細については、 AWS KMS ドキュメントを参照してください。

AWS KMS ドキュメントのセキュリティのベストプラクティスの詳細については、「」を参照してください。

転送中の暗号化

AWS HealthOmics は TLS 1.2 以降を使用して、パブリックエンドポイントおよびバックエンドサービスを介して転送中のデータを暗号化します。

HealthOmics での ID とアクセスの管理

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御 AWS のサービス するのに役立つ です。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS HealthOmics リソースの使用を許可する (アクセス許可を付与する) かを制御します。IAM は、追加料金なしで使用できる AWS のサービス です。

トピック

対象者

AWS Identity and Access Management (IAM) の使用方法は、AWS HealthOmics で行う作業によって異なります。

サービスユーザー – AWS HealthOmics サービスを使用してジョブを実行する場合、管理者は必要な認証情報とアクセス許可を提供します。さらに多くの AWS HealthOmics 機能を使用して作業を行う場合は、追加のアクセス許可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。AWS HealthOmics の機能にアクセスできない場合は、「」を参照してくださいAWS HealthOmics ID とアクセスのトラブルシューティング。

サービス管理者 – 社内の AWS HealthOmics リソースを担当している場合は、通常、AWS HealthOmics へのフルアクセスがあります。サービスユーザーがどの AWS HealthOmics 機能やリソースにアクセスするかを決めるのは管理者の仕事です。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。AWS HealthOmics で IAM を使用する方法の詳細については、「」を参照してくださいが IAM と AWS HealthOmics 連携する方法。

IAM 管理者 – IAM 管理者は、AWS HealthOmics へのアクセスを管理するポリシーの作成方法の詳細について確認する場合があります。IAM で使用できる AWS HealthOmics アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS HealthOmics。

アイデンティティを使用した認証

認証とは、ID 認証情報 AWS を使用して にサインインする方法です。として、IAM ユーザーとして AWS アカウントのルートユーザー、または IAM ロールを引き受けることによって、認証 (にサインイン AWS) される必要があります。

ID ソースを介して提供された認証情報を使用して、フェデレーティッド ID AWS として にサインインできます。 AWS IAM Identity Center （ IAM Identity Center) ユーザー、会社のシングルサインオン認証、Google または Facebook 認証情報は、フェデレーション ID の例です。フェデレーティッド ID としてサインインする場合、IAM ロールを使用して、前もって管理者により ID フェデレーションが設定されています。フェデレーション AWS を使用して にアクセスすると、間接的にロールを引き受けることになります。

ユーザーのタイプに応じて、 AWS Management Console または AWS アクセスポータルにサインインできます。へのサインインの詳細については AWS、「 AWS サインイン ユーザーガイド」の「 へのサインイン AWS アカウント方法」を参照してください。

AWS プログラムで にアクセスする場合、 はソフトウェア開発キット (SDK) とコマンドラインインターフェイス (CLI) AWS を提供し、認証情報を使用してリクエストを暗号化して署名します。AWS ツールを使用しない場合は、自分でリクエストに署名する必要があります。リクエストに自分で署名する推奨方法の使用については、「IAM ユーザーガイド」の「API リクエストに対するAWS Signature Version 4」を参照してください。

使用する認証方法を問わず、追加セキュリティ情報の提供をリクエストされる場合もあります。たとえば、 では、アカウントのセキュリティを高めるために多要素認証 (MFA) を使用する AWS ことをお勧めします。詳細については、「AWS IAM Identity Center ユーザーガイド」の「多要素認証」および「IAM ユーザーガイド」の「IAM のAWS 多要素認証」を参照してください。

AWS アカウント ルートユーザー

を作成するときは AWS アカウント、アカウント内のすべての およびリソースへの AWS のサービス 完全なアクセス権を持つ 1 つのサインインアイデンティティから始めます。この ID は AWS アカウント ルートユーザーと呼ばれ、アカウントの作成に使用した E メールアドレスとパスワードでサインインすることでアクセスできます。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザーの認証情報は保護し、ルートユーザーでしか実行できないタスクを実行するときに使用します。ルートユーザーとしてサインインする必要があるタスクの完全なリストについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーティッドアイデンティティ

ベストプラクティスとして、管理者アクセスを必要とするユーザーを含む人間のユーザーに、ID プロバイダーとのフェデレーションを使用して一時的な認証情報 AWS のサービス を使用して にアクセスすることを要求します。

フェデレーティッド ID は、エンタープライズユーザーディレクトリ、ウェブ ID プロバイダー、AWS Directory Service、アイデンティティセンターディレクトリ、または ID ソースを介して提供された認証情報 AWS のサービス を使用して にアクセスするユーザーです。フェデレーティッド ID がにアクセスすると AWS アカウント、ロールを引き受け、ロールは一時的な認証情報を提供します。

アクセスを一元管理する場合は、 AWS IAM Identity Centerを使用することをお勧めします。IAM Identity Center でユーザーとグループを作成するか、独自の ID ソースのユーザーとグループのセットに接続して同期し、すべての AWS アカウント とアプリケーションで使用できます。IAM Identity Center の詳細については、「AWS IAM Identity Center ユーザーガイド」の「What is IAM Identity Center?」(IAM Identity Center とは) を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、単一のユーザーまたはアプリケーションに対して特定のアクセス許可 AWS アカウント を持つ 内の ID です。可能であれば、パスワードやアクセスキーなどの長期的な認証情報を保有する IAM ユーザーを作成する代わりに、一時的な認証情報を使用することをお勧めします。ただし、IAM ユーザーでの長期的な認証情報が必要な特定のユースケースがある場合は、アクセスキーをローテーションする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「長期的な認証情報を必要とするユースケースのためにアクセスキーを定期的にローテーションする」を参照してください。

IAM グループは、IAM ユーザーの集団を指定するアイデンティティです。グループとしてサインインすることはできません。グループを使用して、複数のユーザーに対して一度に権限を指定できます。多数のユーザーグループがある場合、グループを使用することで権限の管理が容易になります。例えば、IAMAdmins という名前のグループを設定して、そのグループに IAM リソースを管理する許可を与えることができます。

ユーザーは、ロールとは異なります。ユーザーは 1 人の人または 1 つのアプリケーションに一意に関連付けられますが、ロールはそれを必要とする任意の人が引き受けるようになっています。ユーザーには永続的な長期の認証情報がありますが、ロールでは一時認証情報が提供されます。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

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 ID です。これは IAM ユーザーに似ていますが、特定のユーザーには関連付けられていません。で IAM ロールを一時的に引き受けるには AWS Management Console、ユーザーから IAM ロール (コンソール) に切り替えることができます。ロールを引き受けるには、 または AWS API オペレーションを AWS CLI 呼び出すか、カスタ

ム URL を使用します。ロールを使用する方法の詳細については、「IAM ユーザーガイド」の「ロールを引き受けるための各種方法」を参照してください。

IAM ロールと一時的な認証情報は、次の状況で役立ちます:

- フェデレーションユーザーアクセス – フェデレーティッド ID に許可を割り当てるには、ロールを作成してそのロールの許可を定義します。フェデレーティッド ID が認証されると、その ID はロールに関連付けられ、ロールで定義されている許可が付与されます。フェデレーションのロールについては、「IAM ユーザーガイド」の「サードパーティー ID プロバイダー (フェデレーション) 用のロールを作成する」を参照してください。IAM Identity Center を使用する場合は、許可セットを設定します。アイデンティティが認証後にアクセスできるものを制御するため、IAM Identity Center は、権限セットを IAM のロールに関連付けます。アクセス許可セットの詳細については、「AWS IAM Identity Center User Guide」の「Permission sets」を参照してください。
- 一時的な IAM ユーザー権限 - IAM ユーザーまたはロールは、特定のタスクに対して複数の異なる権限を一時的に IAM ロールで引き受けることができます。
- クロスアカウントアクセス - IAM ロールを使用して、自分のアカウントのリソースにアクセスすることを、別のアカウントの人物 (信頼済みプリンシパル) に許可できます。クロスアカウントアクセス権を付与する主な方法は、ロールを使用することです。ただし、一部の では AWS のサービス、(プロキシとしてロールを使用する代わりに) リソースに直接ポリシーをアタッチできます。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。
- クロスサービスアクセス — 一部の は他の の機能 AWS のサービス を使用します AWS のサービス。例えば、あるサービスで呼び出しを行うと、通常そのサービスによって Amazon EC2 でアプリケーションが実行されたり、Amazon S3 にオブジェクトが保存されたりします。サービスでは、呼び出し元プリンシパルの許可、サービスロール、またはサービスリンクロールを使用してこれを行う場合があります。
  - 転送アクセスセッション (FAS) – IAM ユーザーまたはロールを使用して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。

- サービスロール - 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「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。
- サービスにリンクされたロール – 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。

- Amazon EC2 で実行されているアプリケーション – IAM ロールを使用して、EC2 インスタンスで実行され、 AWS CLI または AWS API リクエストを行うアプリケーションの一時的な認証情報を管理できます。これは、EC2 インスタンス内でのアクセスキーの保存に推奨されます。EC2 インスタンスに AWS ロールを割り当て、そのすべてのアプリケーションで使用できるようにするには、インスタンスにアタッチされたインスタンスプロファイルを作成します。インスタンスプロファイルにはロールが含まれ、EC2 インスタンスで実行されるプログラムは一時的な認証情報を取得できます。詳細については、「IAM ユーザーガイド」の「Amazon EC2 インスタンスで実行されるアプリケーションに IAM ロールを使用して許可を付与する」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

でアクセスを制御する AWS には、ポリシーを作成し、ID AWS またはリソースにアタッチします。ポリシーは AWS 、アイデンティティまたはリソースに関連付けられているときにアクセス許可を定義する のオブジェクトです。 は、プリンシパル (ユーザー、ルートユーザー、またはロールセッション) がリクエストを行うときに、これらのポリシー AWS を評価します。ポリシーでの権限により、リクエストが許可されるか拒否されるかが決まります。ほとんどのポリシーは JSON ドキュメント AWS として に保存されます。JSON ポリシードキュメントの構造と内容の詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

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。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き受けることができます。

IAM ポリシーは、オペレーションの実行方法を問わず、アクションの許可を定義します。例えば、`iam:GetRole` アクションを許可するポリシーがあるとします。そのポリシーを持つユーザーは、 AWS Management Console、、 AWS CLIまたは AWS API からロール情報を取得できます。

アイデンティティベースのポリシー

アイデンティティベースポリシーは、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。アイデンティティベース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。

アイデンティティベースのポリシーは、さらにインラインポリシーまたはマネージドポリシーに分類できます。インラインポリシーは、単一のユーザー、グループ、またはロールに直接埋め込まれています。管理ポリシーは、 内の複数のユーザー、グループ、ロールにアタッチできるスタンドアロンポリシーです AWS アカウント。管理ポリシーには、 AWS 管理ポリシーとカスタマー管理ポリシーが含まれます。マネージドポリシーまたはインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

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、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。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは、IAM の AWS マネージドポリシーを使用できません。

アクセスコントロールリスト (ACL)

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。

Amazon S3、および Amazon VPC は AWS WAF、ACLs。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「アクセスコントロールリスト (ACL) の概要」を参照してください。

その他のポリシータイプ

AWS は、一般的でない追加のポリシータイプをサポートします。これらのポリシータイプでは、より一般的なポリシータイプで付与された最大の権限を設定できます。

- アクセス許可の境界 - アクセス許可の境界は、アイデンティティベースポリシーによって IAM エンティティ (IAM ユーザーまたはロール) に付与できる権限の上限を設定する高度な機能です。エンティティにアクセス許可の境界を設定できます。結果として得られる権限は、エンティティのアイデンティティベースポリシーとそのアクセス許可の境界の共通部分になります。`Principal` フィールドでユーザーまたはロールを指定するリソースベースのポリシーでは、アクセス許可の境界は制限されません。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。アクセス許可の境界の詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可の境界」を参照してください。
- サービスコントロールポリシー (SCPs) – SCPs は、 の組織または組織単位 (OU) の最大アクセス許可を指定する JSON ポリシーです AWS Organizations。 AWS Organizations は、ビジネスが所有する複数の をグループ化して一元管理するためのサービス AWS アカウント です。組織内のすべての機能を有効にすると、サービスコントロールポリシー (SCP) を一部またはすべてのアカウントに適用できます。SCP は、各 を含むメンバーアカウントのエンティティのアクセス許可を制限します AWS アカウントのルートユーザー。Organizations と SCP の詳細については、「AWS Organizations ユーザーガイド」の「サービスコントロールポリシー (SCP)」を参照してください。
- リソースコントロールポリシー (RCP) – RCP は、所有する各リソースにアタッチされた IAM ポリシーを更新することなく、アカウント内のリソースに利用可能な最大数のアクセス許可を設定するために使用できる JSON ポリシーです。RCP は、メンバーアカウントのリソースのアクセス許可を制限し、組織に属しているかどうかにかかわらず AWS アカウントのルートユーザー、 を含む ID の有効なアクセス許可に影響を与える可能性があります。RCP をサポートする のリストを含む Organizations と RCP の詳細については、AWS Organizations RCPs「リソースコントロールポリシー (RCPs」を参照してください。 AWS のサービス
- セッションポリシー - セッションポリシーは、ロールまたはフェデレーションユーザーの一時的なセッションをプログラムで作成する際にパラメータとして渡す高度なポリシーです。結果としてセッションの権限は、ユーザーまたはロールのアイデンティティベースポリシーとセッションポリシーの共通部分になります。また、リソースベースのポリシーから権限が派生する場合もありま

す。これらのポリシーのいずれかを明示的に拒否した場合、権限は無効になります。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。が複数のポリシータイプが関与する場合にリクエストを許可するかどうか AWS を決定する方法については、「IAM ユーザーガイド」の「ポリシー評価ロジック」を参照してください。

が IAM と AWS HealthOmics 連携する方法

IAM を使用して AWS HealthOmics へのアクセスを管理する前に、AWS HealthOmics で使用できる IAM 機能を確認してください。

で使用できる IAM 機能 AWS HealthOmics

IAM の機能	HealthOmics のサポート
アイデンティティベースポリシー	はい
リソースベースのポリシー	いいえ
ポリシーアクション	はい
ポリシーリソース	あり
ポリシー条件キー	いいえ
ACL	なし
ABAC (ポリシーのタグ)	あり
一時的な認証情報	はい
プリンシパル権限	はい
サービスロール	はい
サービスリンクロール	なし

HealthOmics 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。

サービス間の混乱した代理の防止

混乱した代理問題は、アクションを実行する許可を持たないエンティティが、より特権のあるエンティティにアクションを実行するように強制できるセキュリティの問題です。では AWS、サービス間のなりすましにより、混乱した代理問題が発生する可能性があります。サービス間でのなりすましは、あるサービス (呼び出し元サービス) が、別のサービス (呼び出し対象サービス) を呼び出すときに発生する可能性があります。呼び出し元サービスが操作され、それ自身のアクセス許可を使用して、本来アクセス許可が付与されるべきではない方法で別の顧客のリソースに対して働きかけることがあります。これを防ぐため、 AWS では、アカウント内のリソースへのアクセス許可が付与されたサービスプリンシパルですべてのサービスのデータを保護するために役立つツールを提供しています。

AWS HealthOmics がリソースに別のサービスに付与するアクセス許可を制限するには、リソースポリシーで `aws:SourceArn`および `aws:SourceAccount` グローバル条件コンテキストキーを使用することをお勧めします。

HealthOmics が引き受けるロールの混乱した代理問題を防ぐには、ロールの信頼ポリシー`arn:aws:omics:region:accountNumber:*`で `aws:SourceArn` の値を に設定します。ワイルドカード (*) は、すべての HealthOmics リソースに 条件を適用します。

次の信頼関係ポリシーは、HealthOmics にリソースへのアクセスを許可し、 `aws:SourceArn` および `aws:SourceAccount` グローバル条件コンテキストキーを使用して混乱した代理問題を防ぎます。HealthOmics のロールを作成するときは、このポリシーを使用します。

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "omics.amazonaws.com"
```

```
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "accountNumber"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:omics:region:accountNumber:*"
        }
      }
    }
  ]
}
```

HealthOmics 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。

HealthOmics のアイデンティティベースのポリシーの例

AWS HealthOmics アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS HealthOmics。

HealthOmics 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: なし

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。

HealthOmics の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。

HealthOmics アクションのリストを確認するには、「サービス認可リファレンス」の「AWS HealthOmics で定義されるアクション」を参照してください。

HealthOmics のポリシーアクションは、アクションの前に次のプレフィックスを使用します。

```
omics
```

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": [
      "omics:action1",
      "omics:action2"
         ]
```

AWS HealthOmics アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS HealthOmics。

HealthOmics のポリシーリソース

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": "*"
```

HealthOmics リソースタイプとその ARNs「AWS HealthOmics で定義されるリソース」を参照してください。 各リソースの ARN を指定できるアクションについては、「AWS HealthOmics で定義されるアクション」を参照してください。

AWS HealthOmics アイデンティティベースのポリシーの例を表示するには、「」を参照してくださいのアイデンティティベースのポリシーの例 AWS HealthOmics。

HealthOmics のポリシー条件キー

ポリシー条件キーは HealthOmics ではサポートされていません。

HealthOmicsACLs)

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。

HealthOmics を使用した属性ベースのアクセスコントロール (ABAC)

ABAC (ポリシー内のタグ) のサポート: あり

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。

HealthOmics リソースのタグ付けの詳細については、「」を参照してくださいHealthOmics でのリソースのタグ付け。

次の例は、特定のタグなしでリソースへのアクセスを拒否する IAM ポリシーを記述する方法を示しています。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "omics:*"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "Null": {
                  "aws:RequestTag/MyCustomTag": "true"
                }
            }
        }
    ]
}
```

HealthOmics 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 などの詳細については、IAM ユーザーガイドの「IAM AWS のサービス 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方法 AWS Management Console を使用して にサインインする場合、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。

HealthOmics のクロスサービスプリンシパル許可

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。

HealthOmics のサービスロール

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**
>
> サービスロールのアクセス許可を変更すると、HealthOmics の機能が破損する可能性があります。HealthOmics が指示する場合にのみ、サービスロールを編集します。

HealthOmics のサービスにリンクされたロール

サービスにリンクされたロールのサポート: なし

サービスにリンクされたロールは、 にリンクされたサービスロールの一種です AWS のサービス。サービスは、ユーザーに代わってアクションを実行するロールを引き受けることができます。サービスにリンクされたロールは に表示され AWS アカウント 、サービスによって所有されます。IAM 管理者は、サービスにリンクされたロールの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。

サービスにリンクされたロールの作成または管理の詳細については、「IAM と提携するAWS のサービス」を参照してください。表の「サービスリンクロール」列に Yes と記載されたサービスを見つけます。サービスにリンクされたロールに関するドキュメントをサービスで表示するには、[はい] リンクを選択します。

のアイデンティティベースのポリシーの例 AWS HealthOmics

デフォルトでは、ユーザーとロールには AWS HealthOmics リソースを作成または変更するアクセス許可はありません。また、、 AWS Command Line Interface （ AWS CLI ） AWS Management Console、または AWS API を使用してタスクを実行することはできません。IAM 管理者は、リソースで必要なアクションを実行するための権限をユーザーに付与する IAM ポリシーを作成できます。その後、管理者はロールに IAM ポリシーを追加し、ユーザーはロールを引き継ぐことができます。

これらサンプルの JSON ポリシードキュメントを使用して、IAM アイデンティティベースのポリシーを作成する方法については、「IAM ユーザーガイド」の「IAM ポリシーを作成する (コンソール)」を参照してください。

各リソースタイプの ARN の形式など、AWS HealthOmics で定義されるアクションとリソースタイプの詳細については、「サービス認可リファレンス」の「AWS HealthOmics 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。 ARNs

トピック

- ポリシーに関するベストプラクティス
- HealthOmics コンソールの使用
- 自分の権限の表示をユーザーに許可する

ポリシーに関するベストプラクティス

ID ベースのポリシーは、アカウント内で誰かが AWS HealthOmics 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。

HealthOmics コンソールの使用

AWS HealthOmics コンソールにアクセスするには、最小限のアクセス許可のセットが必要です。これらのアクセス許可により、 の AWS HealthOmics リソースの詳細を一覧表示および表示できます AWS アカウント。最小限必要な許可よりも制限が厳しいアイデンティティベースのポリシーを作成

すると、そのポリシーを持つエンティティ (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。

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。

自分の権限の表示をユーザーに許可する

この例では、ユーザーアイデンティティにアタッチされたインラインおよびマネージドポリシーの表示を IAM ユーザーに許可するポリシーの作成方法を示します。このポリシーには、コンソールで、または AWS CLI または AWS API を使用してプログラムでこのアクションを実行するアクセス許可が含まれています。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
```

```
        }
    ]
}
```

AWS の 管理ポリシー AWS HealthOmics

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、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るように、多くの一般的なユースケースにアクセス許可を提供するように設計されています。

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。

AWS マネージドポリシー: AmazonOmicsFullAccess

`AmazonOmicsFullAccess` ポリシーを IAM ID にアタッチして、HealthOmics へのフルアクセスを許可できます。

このポリシーは、すべての HealthOmics アクションへのフルアクセス許可を付与します。注釈ストアまたはバリアントストアを作成すると、Omics は Resource Access Manager (RAM) コンソールの Resource Share Invitation を通じてそのストアへのアクセスも許可します。Lake Formation を介したリソース共有の招待の詳細については、「Lake Formation でのクロスアカウントデータ共有」を参

照してください。Omics 管理者ポリシーの場合、Amazon S3 バケットにアクセスするには次のアクセス許可も必要です。

- PutObject
- GetObject
- ListBucket
- AbortMultipartUpload
- ListMultipartUploadParts

JSON

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "omics:*"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "ram:AcceptResourceShareInvitation",
    "ram:GetResourceShareInvitations"
   ],
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "aws:CalledViaLast": "omics.amazonaws.com"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": "iam:PassRole",
   "Resource": "*",
   "Condition": {
```

```
    "StringEquals": {
     "iam:PassedToService": "omics.amazonaws.com"
    }
   }
  }
 ]
}
```

AWS マネージドポリシー: AmazonOmicsReadOnlyAccess

AWSOmicsReadOnlyAccess ポリシーを IAM ID にアタッチするには、その ID のアクセス許可を読み取り専用アクセスに制限します。

JSON

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "omics:Get*",
                "omics:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 管理ポリシーに対する HealthOmics の更新

このサービスがこれらの変更の追跡を開始してからの HealthOmics の AWS マネージドポリシーの更新に関する詳細を表示します。このページの変更に関する自動アラートについては、HealthOmics ドキュメント履歴ページの RSS フィードにサブスクライブしてください。

変更	説明	日付
AmazonOmicsFullAccess - 新しいポリシーが追加されました	HealthOmics は、すべてのアクションとリソースへのフルアクセスをユーザーに許可する新しいポリシーを追加しました。詳細については、「AmazonOmicsFullAccess」を参照してください。	2023 年 2 月 23 日
HealthOmics が変更の追跡を開始しました	HealthOmics は、 AWS 管理ポリシーの変更の追跡を開始しました。	2022 年 11 月 29 日
AmazonOmicsReadOnlyAccess - 新しいポリシーを追加	HealthOmics は、アクセスを読み取り専用に制限する新しいポリシーを追加しました。詳細については、AmazonOmicsReadOnlyAccess を参照してください。	2022 年 11 月 29 日

AWS HealthOmics ID とアクセスのトラブルシューティング

次の情報は、AWS HealthOmics と IAM の使用時に発生する可能性がある一般的な問題の診断と修正に役立ちます。

トピック

- HealthOmics でアクションを実行する権限がない
- iam:PassRole を実行する権限がありません
- 自分の 以外のユーザーに HealthOmics リソース AWS アカウント へのアクセスを許可したい

HealthOmics でアクションを実行する権限がない

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omics:*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:
 omics:GetWidget on resource: my-example-widget
```

この場合、omics:*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 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、AWS HealthOmics にロールを渡すことができるようにポリシーを更新する必要があります。

一部の AWS のサービス では、新しいサービスロールまたはサービスにリンクされたロールを作成する代わりに、そのサービスに既存のロールを渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

次の例のエラーは、 という IAM ユーザーがコンソールを使用して marymajor AWS HealthOmics でアクションを実行しようとすると発生します。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 HealthOmics 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:

- AWS HealthOmics がこれらの機能をサポートしているかどうかを確認するには、「」を参照してくださいが IAM と AWS HealthOmics 連携する方法。
- 所有 AWS アカウント している のリソースへのアクセスを提供する方法については、「IAM ユーザーガイド」の「所有 AWS アカウント している別の の IAM ユーザーへのアクセスを提供する」を参照してください。
- リソースへのアクセスをサードパーティーに提供する方法については AWS アカウント、IAM ユーザーガイドの「サードパーティー AWS アカウント が所有する へのアクセスを提供する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」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。

のコンプライアンス検証 AWS HealthOmics

サードパーティーの監査者は、複数のコンプライアンスプログラムの一環として AWS HealthOmics のセキュリティと AWS コンプライアンスを評価します。これには、HIPAA、FedRAMP などが含まれます。次の表は、HealthOmics サービスのコンプライアンス証明書を示しています。

証明書	リンク
HIPAA	HIPAA 対応サービスリファレンス
HiTrust-CSF	Health Information Trust Alliance 共通セキュリティフレームワーク
FedRAMP Moderate (East/West)	連邦リスク認可管理プログラム
ISO/CSA スター	ISO および CSA STAR 認定

証明書	リンク
C5	クラウドコンピューティングコンプライアンスコントロールカタログ
DoD CC SRG IL2	Department of Defense Cloud Computing セキュリティ要件ガイド
ENS High	Esquema Nacional de Seguridad
FINMA	スイス金融市場監督局
ISMAP	情報システムのセキュリティ管理および評価プログラム
OSPAR	外部委託サービスプロバイダーの監査レポート
PCI	Payment Card Industry データセキュリティ標準
ピン留め	銀行関連付け CCI - サードパーティー資格
PiTuKri	クラウドサービスの情報セキュリティを評価するための基準
SOC 1、2、3	システムおよび組織のコントロール

特定のコンプライアンスプログラムの対象となるすべての AWS サービスのリストについては、「コンプライアンスプログラムによる AWS 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 コンプライアンスプログラム」を参照してください。

を使用して、サードパーティーの監査レポートをダウンロードできます AWS Artifact。詳細については、「Downloading Reports in AWS Artifact」を参照してください。

HealthOmics データストアは、内部ファイルの命名とリソースのタグ付けにサンプル ID を使用します。データを取り込む前に、サンプル ID に PHI データが含まれているかどうかを確認します。その場合は、データを取り込む前にサンプル ID を変更します。詳細については、 AWS HIPAA コンプライアンスウェブページのガイダンスを参照してください。

を使用する際のお客様のコンプライアンス責任 AWS HealthOmics は、お客様のデータの機密性、貴社のコンプライアンス目的、適用される法律および規制によって決まります。 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。
- Architecting for HIPAA Security and Compliance ホワイトペーパー – このホワイトペーパーでは、企業が AWS を使用して HIPAA 準拠のアプリケーションを作成する方法について説明します。
- 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」 – AWS Configでは、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。

HealthOmics の耐障害性

AWS グローバルインフラストラクチャは、 AWS リージョン およびアベイラビリティーゾーンを中心に構築されています。 は、低レイテンシー、高スループット、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 AWS リージョン 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。

AWS リージョン および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。

グローバル AWS インフラストラクチャに加えて、AWS HealthOmics には、データの耐障害性とバックアップのニーズをサポートするのに役立つ機能がいくつか用意されています。

AWS HealthOmics およびインターフェイス VPC エンドポイント (AWS PrivateLink)

VPC と 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 AWS HealthOmics 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 HealthOmics API オペレーションにプライベートにアクセスするために使用できるテクノロジーである を利用しています。VPC 内のインスタンスは、パブリック IP アドレスがなくても HealthOmics API オペレーションと通信できます。VPC と HealthOmics 間のトラフィックは Amazon ネットワーク外に流れません。

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。

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)」を参照してください。

VPC エンドポイントポリシーは、イスラエル (テルアビブ) を除くすべてのリージョンの HealthOmics でサポートされています。デフォルトでは、HealthOmics へのフルアクセスはエンドポイントを介して許可されます。

HealthOmics VPC エンドポイントに関する考慮事項

HealthOmics のインターフェイス VPC エンドポイントを設定する前に、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プロパティと制限を確認してください。

HealthOmics は、VPC からのすべての HealthOmics Storage API アクションの呼び出しをサポートしています。

デフォルトでは、VPC エンドポイントポリシーは HealthOmics ではサポートされていませんが、HealthOmics Storage オペレーションの完全な HealthOmics HealthOmics アクセス用の VPC エンドポイントを作成できます。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。

HealthOmics 用のインターフェイス VPC エンドポイントの作成

Amazon VPC コンソールまたは AWS Command Line Interface () を使用して、HealthOmics サービスの VPC エンドポイントを作成できます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。

次のサービス名を使用して HealthOmics の VPC エンドポイントを作成します。

- com.amazonaws.*region*.storage-omics
- com.amazonaws.*region*.control-storage-omics
- com.amazonaws.*region*.analytics-omics
- com.amazonaws.*region*.workflows-omics
- com.amazonaws.*region*.tags-omics

米国東部 (バージニア北部) および米国西部 (オレゴン) リージョンは FIPS AWS PrivateLink エンドポイントをサポートしています。これらのリージョンでは、次のサービス名を使用することもできます。

- com.amazonaws.*region*.storage-omics-fips
- com.amazonaws.*region*.control-storage-omics-fips
- com.amazonaws.*region*.analytics-omics-fips
- com.amazonaws.*region*.workflows-omics-fips
- com.amazonaws.*region*.tags-omics-fips

エンドポイントのプライベート DNS を有効にする場合、 などのリージョンのデフォルト DNS 名を使用して HealthOmics に API リクエストを行うことができますomics.us-east-1.amazonaws.com。

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を介したサービスへのアクセス」を参照してください。

HealthOmics の VPC エンドポイントポリシーの作成

HealthOmics へのアクセスを制御するエンドポイントポリシーを VPC エンドポイントにアタッチできます。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

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。

例: HealthOmics アクションの VPC エンドポイントポリシー。

HealthOmics のエンドポイントポリシーの例を次に示します。エンドポイントにアタッチすると、このポリシーは、すべてのリソースのすべてのプリンシパルに対して HealthOmics アクションへのアクセスを許可します。

API

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "omics:List*"
         ],
         "Resource":"*"
      }
   ]
}
```

AWS CLI

```
aws ec2 modify-vpc-endpoint \
    --vpc-endpoint-id vpce-id \
    --region us-west-2 \
    --policy-document \
    "{\"Statement\":[{\"Principal\":\"*\",\"Effect\":\"Allow\",\"Action\":
[\"omics:List*\"],\"Resource\":\"*\"}]}"
```

Amazon S3 URIs を使用してリードセットにアクセスするための特別な考慮事項

プライベート接続を使用しているときに Amazon S3 URIs を介して読み取りセットにアクセスするには、シーケンスストアで PrivateLink インターフェイスエンドポイントを設定します。設定後、エンドポイントの形式は次のとおりです。

```
com.amazonaws.region.storage-omics
com.amazonaws.region.control-storage-omics
```

ゲートウェイエンドポイントを使用するには、Amazon S3 のゲートウェイエンドポイントガイドに従ってゲートウェイエンドポイントを設定します。HealthOmics は Amazon S3 バケットを所有しているため、バケットポリシーを作成または調整する必要はありません。ゲートウェイエンドポイントは、データにアクセスするユーザーまたはロールにアタッチされたポリシーに依存しますが、より制限の厳しいポリシーでエンドポイントを設定することもできます。これらのポリシーには、Amazon S3 アクセスポイント ARN および Amazon S3 アクションに基づくアクセスの制限を含めることができます。

AWS HealthOmics のモニタリング

モニタリングは、AWS HealthOmics およびその他の AWS ソリューションの信頼性、可用性、パフォーマンスを維持する上で重要な部分です。 には、AWS HealthOmics をモニタリングし、問題が発生したときに報告し、必要に応じて自動アクションを実行するための以下のモニタリングツール AWS が用意されています。

- Amazon CloudWatch は、 AWS リソースと で実行されるアプリケーションを AWS リアルタイムでモニタリングします。メトリクスの収集と追跡、カスタマイズしたダッシュボードの作成、および指定したメトリクスが指定したしきい値に達したときに通知またはアクションを実行するアラームの設定を行うことができます。例えば、CloudWatch で Amazon EC2 インスタンスの CPU 使用率などのメトリクスを追跡し、必要に応じて新しいインスタンスを自動的に起動できます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。
- Amazon CloudWatch Logs では、Amazon EC2 インスタンス、CloudTrail、その他ソースから得たログファイルのモニタリング、保存、およびアクセスが可能です。CloudWatch Logs は、ログファイル内の情報をモニタリングし、特定のしきい値が満たされたときに通知します。高い耐久性を備えたストレージにログデータをアーカイブすることも可能です。詳細については、『Amazon CloudWatch Logs ユーザーガイド』を参照してください。
- AWS CloudTrailは、 AWS アカウント により、またはそのアカウントに代わって行われた API コールや関連イベントを取得し、指定した Amazon S3 バケットにログファイルを配信します。AWSを呼び出したユーザーとアカウント、呼び出し元の IP アドレス、および呼び出しの発生日時を特定できます。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。
- Amazon EventBridge は、アプリケーションをさまざまなイベントソースのデータに簡単に接続できるようにするサーバーレスイベントバスサービスです。EventBridge は、独自のアプリケーション、Software-as-a-Service (SaaS) アプリケーション、および AWS のサービスからリアルタイムデータのストリームを配信し、そのデータを Lambda などのターゲットにルーティングします。これにより、サービスで発生したイベントをモニタリングし、イベント駆動型アーキテクチャを構築できます。詳細については、「Amazon EventBridge ユーザーガイド」を参照してください。

> **Note**
>
> サービスの更新については、Personal Health Dashboard を設定してモニタリングします。ダッシュボードの管理方法の詳細については、「AWS Health Dashboard の開始方法」を参照してください。

トピック

S3 アクセスログ記録

ストアが作成したアクセスログを使用して、HealthOmics シーケンスストアデータへの Amazon S3 API アクセスをモニタリングできます。CloudWatch を使用して、HealthOmics API オペレーションからの S3 アクセスをモニタリングできます。CloudWatch は、独自のアカウントから発信される Amazon S3 アクセスを可視化します。データ所有者としてサードパーティーアカウントへのアクセスを共有する場合、CloudWatch ではアクセスログ記録を使用できません。代わりに、ストアの S3 アクセスログを使用します。これにより、設定された Amazon S3 バケット のデータへのすべての S3 アクセスがログに記録されます。 Amazon S3

`CreateSequenceStore` または `UpdateSequenceStore` API オペレーションを使用して S3 アクセスログを設定します。また、HealthOmics サービスプリンシパル (`omics.amazonaws.com`) に、設定された S3 プレフィックスに対する`s3:PutObject`アクセス許可があることを確認してください。

> **Note**
>
> ログは、レプリケート先バケットのデフォルトの暗号化設定を使用します。バケットがカスタマーマネージドキーを使用する場合、サービスプリンシパルは書き込みにキーを使用するためのアクセス権を持っている必要があります。

アクセスログ記録をオフにするには、 を使用し`UpdateSequenceStore`、アクセスログ設定を空白に設定します。

CloudWatch メトリクスによる HealthOmics のモニタリング

CloudWatch を使用して HealthOmics CloudWatch は raw データを収集し、読み取り可能なほぼリアルタイムのメトリクスに加工します。これらの統計は 15 か月間保持されるため、履歴情報にアクセ

スし、ウェブアプリケーションまたはサービスの動作をより的確に把握できます。また、特定のしきい値を監視するアラームを設定し、これらのしきい値に達したときに通知を送信したりアクションを実行したりできます。詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

AWS HealthOmics サービスは、 AWS/Omics名前空間で次のメトリクスを報告します。

API コールカウントメトリクスは、次の AWS HealthOmics APIs。API オペレーションディメンションのみがレポートされます。

- リファレンスストア APIs
  CreateReferenceStore、DeleteReferenceStore、StartReferenceImportJob
- Sequence store and read set APIs —
  CreateSequenceStore、DeleteSequenceStore、StartReadSetImportJob、StartReadSetActivationJob、St
- バリアントストア APIs —
  CreateVariantStore、DeleteVariantStore、StartVariantImportJob、CancelVariantImportJob
- 注釈ストア APIs —
  CreateAnnotationStore、DeleteAnotationStore、StartAnnotationImportJob、CancelAnnotationImportJob
- ワークフロー、実行、実行グループ APIs —
  CreateWorkflow、DeleteWorkflow、StartRun、CancelRun、DeleteRun、CreateRunGroup、DeleteRunGr

AWS HealthOmics メトリクスの表示

の CloudWatch メトリクス AWS HealthOmics は CloudWatch コンソールで表示できます。

メトリクスを表示する方法 (CloudWatch コンソール)

1. AWS マネジメントコンソールにサインインし、CloudFront コンソール を開きます。
2. メトリクスを選択し、すべてのメトリクスを選択し、AWS/使用状況を選択します。
3. の Filter ServiceAWS HealthOmics。
4. ディメンションを選択してメトリクスの名前を選んだら、グラフに追加 を選択します。
5. 日付範囲の値を選択します。選択した日付範囲のメトリクスカウントがグラフに表示されます。

CloudWatch を使用したアラームの作成

CloudWatch アラームは指定期間中に単一のメトリクスを監視し、1 つ以上のアクションを実行して Amazon Simple Notification Service (Amazon SNS) トピックまたは Auto Scaling ポリシーに通知を送信します。アクションは、複数の指定期間にわたって特定のしきい値を基準としたメトリクスの値に応じて実行されます。アラームの状態が変わったときにも、CloudWatch は Amazon SNS メッセージを送信できます。

CloudWatch アラームがアクションを呼び出すのは、状態が変わってから指定期間が経過するまで、その新しい状態が続いた場合に限ります。

メトリクスを表示する方法 (CloudWatch コンソール)

1. AWS マネジメントコンソールにサインインし、CloudFront コンソール を開きます。
2. [Alarms]、[Create Alarm] の順に選択します。
3. AWS/Usage を選択し、サービスディメンションを使用して AWS HealthOmics メトリクスを選択します。
4. [Time Range] (時間の範囲) で、モニタリングする期間を選択し、[Next] (次へ) を選択します。
5. [Name] (名前) と [Description] (説明) を入力します。
6. 常に >= を選択し、最大値を入力します。
7. アラーム状態に達したときに CloudWatch から E メールを送信する場合は、「アクション」セクションの「このアラームが発生するたびに、「状態は ALARM」を選択します。通知の送信先として、メーリングリストを選択するか、新しいリストを選択して新しいメーリングリストを作成します。
8. [Alarm Preview] (アラームの確認) セクションでアラームをプレビューします。アラームに問題がなければ、[Create Alarm] (アラームの作成) を選択します。

CloudWatch Logs による HealthOmics のモニタリング

HealthOmics は、実行の理解とトラブルシューティングに役立つさまざまな口グを生成します。口グは CloudWatch と Amazon S3 の 2 つの場所で使用できます。

デフォルトでは、実行の口グ記録はオンになっています。必要に応じて、 startrunリクエストLogLevel = OFFで を設定することで、実行の口グ記録をオフにできます。

> **Note**
>
> サービスの更新については、Personal Health Dashboard を設定してモニタリングします。ダッシュボードの管理方法の詳細については、「AWS Health Dashboard の開始方法」を参照してください。

トピック

HealthOmics ワークフローのログタイプ

HealthOmics は、ワークフローに次のタイプのログを提供します。

- エンジンログ – 基盤となるワークフローエンジン (Nextflow、WDL、CWL) は、実行のエンジンログを生成します。これらのログは、ワークフロー定義の問題のトラブルシューティングに役立ちます。
- マニフェストログの実行 – これらのログは、タスクのステータス、開始時刻、停止時刻、失敗の理由 (タスクが失敗した場合) など、各実行タスクに関する高レベルの情報を提供します。

  マニフェストログを実行すると、リソース最適化の機会を理解するのに役立つリソース使用率統計もレポートされます。これらの統計には以下が含まれます。

  - cpusAverage
  - cpusMaximum
  - cpusReserved
  - gpusReserved
  - memoryAverageGiB
  - memoryMaximumGiB
  - memoryReservedGiB
  - runningSeconds

- 実行ログ – 実行ログは、全体的な実行ステータスと、個々のタスクが開始、実行、停止、完了した時刻を提供します。実行ログでは、ファイルのインポートとエクスポートの手順も確認できます。
- タスクログ – タスクログは、実行中の個々のタスクに関する詳細なログ情報を提供します。タスクログの出力は、タスク定義とコード内のログステートメントを使用する場所によって異なります。タスクログが必要なレベルのインサイトを提供しない場合は、タスク定義にログステートメントを追加して、よりインサイトの多いタスクログを生成することを検討してください。
- キャッシュログの実行 – キャッシュログの実行は、実行キャッシュの全体的なステータスとタスク出力のキャッシュを提供します。キャッシュログを実行すると、キャッシュを使用する実行ごとにキャッシュヒットとキャッシュミスを可視化できます。
- Outputs.json – WDL および CWL ワークフローの場合、HealthOmics は実行完了後に という名前のエンジン生成ファイルを Amazon S3 バケット`outputs.json`に配信します。このファイルには、実行のすべての出力のリストとマップが含まれます。

CloudWatch のログ

HealthOmics CloudWatch ワークフローログは、ロググループ にあります`/aws/omics/WorkflowLog`。また、get-run API オペレーションの出力は、エンジンログと実行ログの CloudWatch ログストリーム ARNs を提供します。

デフォルトでは、 は CloudWatch Logs を無期限に AWS 保持します。ロググループの保持ポリシーを調整して、保持期間を 10 年から 1 日の間で設定できます。

次の表は、HealthOmics の CloudWatch Logs の概要を示しています。 HealthOmics

ログ名	CloudWatch Logs で利用可能	がログを利用できる場合	ログストリーム形式
エンジンログ	はい、失敗した実行の場合	実行完了後	run/*runID*/engine
マニフェストログを実行する	あり	実行完了後	manifest/run/*runID*/*runUUID*
ログの実行	あり	リアルタイムで	run/*runID*

ログ名	CloudWatch Logs で利用可能	がログを利用できる場合	ログストリーム形式
タスクログ	あり	リアルタイムで	run/*runID*/task/*taskID*
キャッシュログを実行する	あり	リアルタイムで	runCache/*runCacheId* /*runCacheUUID*
Outputs.json (WDL および CWL)	なし	該当なし	該当なし

Amazon S3 のログ

実行が完了すると、エンジンログは S3 バケットに配信され、削除するまで無期限に使用できます。これらのログは、ワークフローに指定した S3 出力 URI のログディレクトリにあります。

logs ディレクトリへのパスの形式は ですs3://{user_provided_path}/logs/。

次の表は、Amazon S3 バケットで利用可能な HealthOmics ログの概要を示しています。 Amazon S3

ログ名	Amazon S3 で利用可能	がログを利用できる場合	ログストリームパス
エンジンログ	あり	実行完了後	s3://*user_provided_path* /logs/engine.log
Outputs.json (WDL および CWL)	あり	実行完了後	s3://*user_provided_path* /*runID*/*runUUID*/logs/outputs.json
マニフェストログの実行、ログの実行、タスクログの実行	なし	該当なし	該当なし

CLI のインタラクティブ CloudWatch Logs

インタラクティブモードで Live Tail コマンドを使用して、CloudWatch Logs をインタラクティブに表示できます。実行の進行状況をリアルタイムで追跡し、最大 5 つのキーワードを定義してログで強調表示できます。

```
aws logs start-live-tail  \
  --mode interactive  \
  --log-group-identifiers arn:aws:logs:region:account-ID:log-group:/aws/omics/
WorkflowLog
```

詳細については、 AWS CLI 「 コマンドリファレンス」の「Start live tail」を参照してください。

コンソールから CloudWatch Logs にアクセスする

実行のログにアクセスするには、HealthOmics コンソールの実行の詳細ページからこれらのログに直接リンクできます。

1. HealthOmics コンソールを開きます。
2. 左側のナビゲーションペインで、実行を選択します。
3. Runs テーブルから実行を選択します。
4. 実行の詳細ページで、次のいずれかのアクションを選択できます。

    a. Run summary から、View run logs を選択します。コンソールが CloudWatch コンソールで実行ログを開きます。

    b. Run summary から、View logs in Amazon S3 を選択します。コンソールは、Amazon S3 コンソールでログフォルダを開きます。

    c. タスクの実行から、タスクのログの表示、実行ログの表示、または実行マニフェストログの表示を選択します。コンソールは CloudWatch コンソールでログを開きます。

CloudWatch コンソールからログに移動することもできます。

1. CloudWatch コンソール https://console.aws.amazon.com/cloudwatch/ を開きます。
2. 左側のメニューから、ロググループを選択します。
3. `/aws/omics/WorkflowLog` グループを選択します。

    ロググループのリストが長い場合は、検索テキストボックスにオミクスを入力してリストを絞り込むことができます。

4. ロググループの詳細ページが開いたら、表示するログストリームを選択します。コンソールには、このログストリームのイベントが表示されます。

を使用した AWS HealthOmics API コールのログ記録 AWS CloudTrail

AWS HealthOmics は AWS CloudTrail、HealthOmics 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 HealthOmics 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。キャプチャされた呼び出しには、HealthOmics コンソールからの呼び出しと HealthOmics API オペレーションへのコード呼び出しが含まれます。証跡を作成する場合は、HealthOmics のイベントなど、Amazon S3 バケットへの CloudTrail イベントの継続的な配信を有効にすることができます。証跡を設定しない場合でも、CloudTrail コンソールの [イベント履歴] で最新のイベントを表示できます。CloudTrail で収集された情報を使用して、HealthOmics に対するリクエスト、リクエスト元の IP アドレス、リクエスト者、リクエスト日時などの詳細を確認できます。

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 HealthOmics 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。HealthOmics 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。

HealthOmics 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 HealthOmics アクションは CloudTrail によってログに記録され、 AWS HealthOmics API リファレンスに記載されています。例えば、CreateReferenceStore、StartVariantImportJob、CreateWorkflow 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、誰がリクエストを生成したかという情報が含まれます。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 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。

HealthOmics ログファイルエントリについて

「トレイル」は、指定した Amazon S3 バケットにイベントをログファイルとして配信するように設定できます。CloudTrail のログファイルは、単一か複数のログエントリを含みます。イベントは任意ソースからの単一リクエストを表し、リクエストされたアクション、アクションの日時、リクエストパラメータなどの情報を含みます。CloudTrail ログファイルは、パブリック API 呼び出しの順序付けられたスタックトレースではないため、特定の順序では表示されません。

次の例は、CreateWorkflow アクションを示す CloudTrail ログエントリを示しています。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIU53LOGOMTOPXXNPG:username",
        "arn": "arn:aws:sts::account:assumed-role/admin/username",
        "accountId": "account-id",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
```

```
                "principalId": "AROAIU53LOGOMTOPXXNPG",
                "arn": "arn:aws:iam::account:role/admin",
                "accountId": "account",
                "userName": "admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2022-07-23T18:26:09Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2022-07-23T18:46:42Z",
    "eventSource": "omics.amazonaws.com",
    "eventName": "CreateWorkflow",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.176",
    "userAgent": "aws-cli/1.22.45 Python/3.9.13 Darwin/20.6.0 botocore/1.23.45",
    "requestParameters": {
        "name": "parameter_name",
        "definitionZip": "czM6Ly93b3JrZmxvd2RlZi1oZWxsby9kZWZpbml0aW9uLnppcA==",
        "requestId": "d788a73c-b81b-45fb-a8a6-d8bb4449ec8a"
    },
    "responseElements": {
        "id": "1002571",
        "arn": "arn:aws:omics:us-west-2:555555555555:instance/i-b188560f ",
        "status": "CREATING",
        "tags": {
            "resourceArn": "arn:aws:omics:us-west-2:083685709690:workflow/1002571"
        }
    },
    "requestID": "842d731d-f264-4b08-a2c9-2f7d45e1eaa3",
    "eventID": "76872ca2-f208-4193-807d-7dd7ea34e6b2",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "083685709690",
    "eventCategory": "Management"
}
```

での EventBridge の使用 AWS HealthOmics

HealthOmics は、リソースのステータスが変更されたときに Amazon EventBridge にイベントを送信します。リソースには、インポートジョブ、エクスポートジョブ、リソース共有、ワークフロー、タスク、実行が含まれます。リソースのタイプごとに、イベントを生成するステータス変更のリストがあります。

イベントバスは、イベントを受信して送信先に配信するルーターです。アカウントには、 AWS サービスからイベントを自動的に受信するデフォルトのイベントバスが含まれています。追加のカスタムイベントバスを作成できます。

EventBridge ルールを作成して、イベントバスがイベントを受信したときに実行するアクションを指定します。たとえば、リソースのステータス変更を通知するルールを作成できます。

イベントを使用する一般的なシナリオは次のとおりです。

- ユーザーがリソースを共有したとき、または共有を取り消したときを監視します。
- 実行が失敗するか正常に完了したかをモニタリングするには。

EventBridge の使用の詳細については、「Amazon EventBridge とは」を参照してください。

トピック

HealthOmics の EventBridge を設定する

EventBridge イベントをモニタリングする前に、EventBridge バスを作成し、目的のイベントのルールを作成します。

EventBridge バスを設定する

のデフォルトのイベントバスを使用する AWS アカウント か、カスタムイベントバスを設定できます。カスタムイベントバスを設定するには、次の手順に従います。

1. EventBridge コンソールを開きます: https://console.aws.amazon.com/events/。

2. 左側のナビゲーションで、イベントバスを選択します。
3. [イベントバスの作成 (Create event bus)] を選択します。
4. イベントバスの作成フォームに、バスの名前を入力します。
5. Create を選択してバスを作成します。

EventBridge ルールを作成します

次の手順は、シンプルなルールを作成する方法を示しています。ルールの詳細については、EventBridge のルール」を参照してください。

1. EventBridge コンソールを開きます: https://console.aws.amazon.com/events/。
2. 左のナビゲーションで、ルール を選択します。
3. [ルールを作成] を選択します。コンソールでルールの作成フォームが開きます。
4. ルールの詳細の定義で、ルールの名前を指定します。
   - Name に、バスの名前を入力します。
   - イベントバスで、このルールのバスを選択します。
   - [次へ] を選択します。
5. ビルドイベントパターンで、イベントソースで AWS イベントまたは EventBridge パートナーイベントを選択します。
6. イベントパターンまで下にスクロールします。
   a. イベントソースで、AWS サービスを選択します。
   b. AWS サービスの場合は、テキストフィルターにオミクスを入力し、サービスAWS HealthOmicsとして を選択します。
   c. イベントタイプで、目的のイベント (またはすべてのイベント) を選択します。
   d. [次へ] を選択します。
7. Select target (s) で、イベントのターゲットを選択します。たとえば、AWS サービス、選択した CloudWatch ロググループを選択し、ロググループを設定します。

   多くのターゲットタイプで、EventBridge はターゲットにイベントを送信するためのアクセス許可が必要です。コンソールでは、これらのアクセス許可が自動的に作成されます。
8. (オプション) タグを設定する で、タグをルールに関連付けます。
9. 「確認と更新」で、設定を確認し、ルールの作成を選択します。

HealthOmics の EventBridge イベント

次の表に、HealthOmics が EventBridge に送信するイベントと、そのイベントで可能なステータス値のリストを示します。

イベント名	使用可能なステータス値
注釈インポートジョブのステータスの変更	送信済み、進行中、キャンセル済み、完了済み、失敗、または失敗して完了
Annotation Store 共有ステータスの変更	保留中、アクティブ化中、アクティブ、削除中、削除済み、失敗
Annotation Store のステータスの変更	作成、作成、更新、更新、削除、削除、または作成に失敗しました
読み取りセットのアクティベーションジョブのステータスの変更	送信済み、進行中、完了、失敗、または失敗して完了
読み取りセットのエクスポートジョブのステータスの変更	送信済み、進行中、完了、失敗、または失敗して完了
読み取りセットのインポートジョブのステータスの変更	送信済み、進行中、完了、失敗、または失敗して完了
読み取りセットのステータス変更	アップロード、アップロード失敗、アクティブ、アーカイブ済み、アクティブ化、または削除の処理
リファレンスインポートジョブのステータスの変更	送信済み、進行中、完了、失敗、または失敗して完了
リファレンスステータスの変更	アクティブまたは削除済み
リファレンスストアのステータスの変更	作成、更新、アクティブ、または削除済み
実行ステータスの変更	保留中、開始中、実行中、停止中、完了済み、削除済み、失敗、またはキャンセル済み
Sequence Store のステータスの変更	作成、更新、アクティブ、または削除済み

イベント名	使用可能なステータス値
タスクステータスの変更	保留中、開始中、実行中、停止中、完了済み、削除済み、失敗、またはキャンセル済み
バリアントインポートジョブのステータスの変更	送信済み、進行中、キャンセル済み、完了済み、失敗、または失敗して完了
バリアントストアの共有ステータスの変更	保留中、アクティブ化中、アクティブ、削除中、削除済み、失敗
バリアントストアのステータスの変更	作成、作成、更新、更新、削除、削除、または作成に失敗しました
ワークフロー共有ステータスの変更	保留中、アクティブ化中、アクティブ、削除中、削除済み、失敗
ワークフローステータスの変更	作成成功、作成失敗、削除成功、または削除失敗

イベントメッセージの構造

HealthOmics は、EventBridge に状態変更イベントメッセージを送信するためのベストエフォート配信を提供します。イベントは、メタデータの詳細も含めた JSON 構造を持つオブジェクトです。メタデータを入力として使用して、イベントを再作成したり、詳細を確認したりできます。イベントには次のフィールドが含まれます。

- `version` — 現在、すべてのイベントで 0 (ゼロ) 。
- `id` — イベントごとに生成されるバージョン 4 UUID。
- `detail-type` — 送信されるイベントのタイプ。
- `account` — バケット所有者の 12 桁の AWS アカウント ID。
- `source` — イベントを生成したサービスを識別します。
- `time` — イベントが発生した時刻。
- `region` — バケット AWS リージョン の を識別します。
- `resources` — バケットの Amazon リソースネーム (ARN) を含む JSON 配列。
- `detail` — イベントに関する情報を含む JSON オブジェクト。

実行イベントには、次のフィールドが含まれます。

- uuid — 実行の汎用一意識別子。
- workflowId — この実行に関連付けられたワークフローのワークフロー識別子。
- workflowName — この実行に関連付けられたワークフローの名前。
- runId — 実行識別子。
- runName — 実行名。
- runOutputUri — 実行が出力データを書き込む URI。

イベントメッセージの例

次の例は、実行ステータスの変更に関するイベントで、追加のフィールドを示しています。

```
{
    "version":"0",
    "id":"c0e540f4-df38-b986-86c1-3e3730f971fe",
    "detail-type":"Run Status Change",
    "source":"aws.omics",
    "account":"123456789012",
    "time":"2022-10-20T22:07:35Z",
    "region":"us-west-2",
    "resources":[
        "arn:aws:omics:us-west-2:123456789012:run/2101313"
    ],
    "detail":{
        "omicsVersion":"1.0.0",
        "arn":"arn:aws:omics:us-west-2:123456789012:run/2101313",
        "status":"COMPLETED",
        "uuid":"153893cd-097a-40ec-aec7-838a97cd2b21",
        "runId": "1234567",
        "runName": "run name",
        "runOutputUri": "s3://amzn-s3-demo-bucket/run-output/2101313",
        "workflowId": "1234567",
        "workflowName": "workflow name"
    }
}
```

次の例は、タスクステータスの変更のイベントです。

```
{
```

```
    "version": "0",
    "id": "718d6817-c868-26d3-8ef0-0dc9b2ac73f4",
    "detail-type": "Task Status Change",
    "source": "aws.omics",
    "account": "123456789012",
    "time": "2024-10-30T09:05:44Z",
    "region": "us-west-2",
    "resources": ["arn:aws:omics:us-west-2:123456789012:task/8888888"],
    "detail": {
        "omicsVersion": "1.0.0",
        "arn": "arn:aws:omics:us-west-2:123456789012:task/8888888",
        "status": "COMPLETED",
        "runArn": "arn:aws:omics:us-west-2:123456789012:run/2101313",
        "runUuid": "153893cd-097a-40ec-aec7-838a97cd2b21",
        "runId": "1234567",
        "runName": "run name",
        "workflowId": "1234567",
        "workflowName": "workflow name"
    }
}
```

以下は、リードセットのステータス変更のイベントの例です。

```
{
  "version": "0",
  "id": "64ca0eda-9751-dc55-c41a-1bd50b4fc9b7",
  "detail-type": "Read Set Status Change",
  "source": "aws.omics",
  "account": "123456789012",
  "time": "2023-04-04T17:53:06Z",
  "region": "us-west-2",
  "resources": ["arn:aws:omics:us-west-2:123456789012:sequenceStore/1234567890/
readSet/3456789012"],
  "detail": {
     "omicsVersion": "1.0.0",
     "arn": "arn:aws:omics:us-west-2:123456789012:sequenceStore/1234567890/
readSet/3456789012",
     "sequenceStoreId" : "1234567890",
     "id": "3456789012",
     "status": "PROCESSING_UPLOAD"
  }
}
```

バリアントストアのインポートジョブに同様のイベントが作成されます。

```
{
    "version": "0",
    "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
    "detail-type": "Variant Store Status Change",
    "source": "aws.omics",
    "account": "123456789012",
    "time": "2015-12-22T18:43:48Z",
    "region": "us-east-1",
    "resources": ["arn:aws:omics:us-east-1:123456789012:myvariantstore2"],
    "detail": {
       "omicsVersion": "1.0.0",
       "arn": "arn:aws:omics:us-east-1:123456789012:myvariantstore2",
       "status": "CREATED",
       "storeId": "6710c5f02610",
       "storeName": "myvariantstore2"
    }
}
```

以下は、インポートジョブのステータスの変更に関するイベントです。

```
{
    "version": "0",
    "id": "6a7e8feb-b491-4cf7-a9f1-bf3703467718",
    "detail-type": "Variant Import Job Status Change",
    "source": "aws.omics",
    "account": "123456789012",
    "time": "2015-12-22T18:43:48Z",
    "region": "us-east-1",
    "resources": ["arn:aws:omics:us-east-1:123456789012:my_variant_store/
b64ea9a3-459f-4b68-92c3-3ddb83209fe9"],
    "detail": {
       "omicsVersion": "1.0.0",
       "arn": "arn:aws:omics:us-east-1:123456789012:my_variant_store/
b64ea9a3-459f-4b68-92c3-3ddb83209fe9",
       "status": "COMPLETED",
       "jobId": "b64ea9a3-459f-4b68-92c3-3ddb83209fe9",
       "storeId": "a74869f91e20",
       "storeName": "my_variant_store"
    }
}
```

トラブルシューティング

以下のトピックは、HealthOmics ワークフローとデータストアを使用する際に発生する問題のトラブルシューティングに役立ちます。

トピック

ワークフローのトラブルシューティング

トピック

失敗した実行のトラブルシューティング方法を教えてください。

GetRun API オペレーションを使用して、失敗の理由を取得します。詳細については、「実行失敗の理由」を参照してください。

失敗したタスクのトラブルシューティング方法を教えてください。

タスク失敗メッセージのエラーコードを確認して、失敗を理解します。CloudWatch のタスクログを確認して、タスクの詳細なログ記録メッセージを確認します。詳細なログメッセージが表示されない場合は、ワークフローを修正して追加のログステートメントを出力できます。詳細については、「CloudWatch Logs による HealthOmics のモニタリング」を参照してください。

正常に完了した実行のエンジンログはどこにありますか？

HealthOmics は、失敗した実行に対してのみ CloudWatch にログを発行します。実行が正常に完了すると、HealthOmics はエンジンログを Amazon S3 バケットに配信します。詳細については、「Amazon S3 のログ」を参照してください。

ワークフローの入力パラメータサイズを減らすにはどうすればよいですか？

ワークフローには、最大 50 KB の入力パラメータを指定できます。ディレクトリのインポートまたはサンプルシートを使用して、このサイズ制約内にとどまることができます。詳細については、「実行パラメータサイズの管理」を参照してください。

実行が完了しないのはなぜですか？

コードに問題があり、プロセスが適切に終了していない場合、実行が応答しなくなるか、「スタック」する可能性があります。応答しない実行を防止してキャッチする方法の詳細については、「」を参照してください応答しない実行のガイダンス。

コールキャッシュの問題のトラブルシューティング

以下のトピックは、通話キャッシュで発生する問題のトラブルシューティングに役立ちます。

トピック

- 実行がキャッシュに保存されないのはなぜですか？
- タスクがキャッシュエントリを使用していないのはなぜですか？

実行がキャッシュに保存されないのはなぜですか？

1. GetRun API オペレーションレスポンスの cacheId フィールドをチェックして、キャッシュを使用するように実行が設定されていることを確認します。CLI を使用して、次のコマンドを実行します: `aws omics get-run –id <run_id>`。
2. 実行が成功した場合は、GetRun レスポンスで返されるキャッシュ動作が CACHE_ALWAYS であることを確認します。キャッシュ動作が CACHE_ON_FAILURE に設定されている場合、実行は失敗したときにのみキャッシュに保存されます。

タスクがキャッシュエントリを使用していないのはなぜですか？

/aws/omics/WorkflowLog CloudWatch ロググループで、実行キャッシュのログストリームを開きます: runCache/<cache_id>/<cache_uuid>。

1. 前の実行で、キャッシュされる予定のタスクのキャッシュエントリが作成されていることを確認します。キャッシュに保存された実行は、CACHE_ENTRY_CREATED のログメッセージで記録されます。
2. タスクの CACHE_MISS ログを見つけ、完了した を実行します。ログエントリがない場合は、キャッシュを使用するように実行が設定されていることを確認します。
3. キャッシュエントリが作成された場合は、CPU、メモリCPUs、GPUs、コンテナダイジェストが両方のタスクで同じであることを確認します。キャッシュエントリを作成したタスクのタスク ARN は、ログメッセージにあります。
4. 両方のタスクのコンピューティング要件が一致する場合は、タスク間で入力が変更されていないことを確認します。これを行うには、エンジンログを開きます。実行のステータスが FAILED の場合、ログは Cloudwatch Log Group /aws/omics/WorkflowLog になります。それ以外の場合、エンジンログは実行の出力ディレクトリにあります。

データストアのトラブルシューティング

トピック

- 読み取りセットで S3 GetObject が失敗するのはなぜですか？
- Athena で注釈ストアまたはバリアントストアが表示されないのはなぜですか？
- Athena のデータストアにアクセスできないのはなぜですか？

読み取りセットで S3 GetObject が失敗するのはなぜですか？

最も一般的に、障害の原因はアクセス許可がないことです。シーケンスストア S3 読み取りアクセス許可は、アクセスを許可するシーケンスストア S3 アクセスポリシーと、アクセスを許可するポリシーをアタッチする IAM プリンシパルの両方を必要とする双方向の設定です。ポリシー要件の詳細については、「」を参照してくださいAmazon S3 URIs を使用したデータアクセスのアクセス許可。次の設定が設定されていることを確認します。

- シーケンスストア S3 アクセスポリシーは、IAM プリンシパルまたはプリンシパルのアカウントのルートへのアクセスを明示的に許可しています。

- IAM プリンシパルに、アクセスするリソースにアクセス許可を明示的に付与するポリシーがあることを確認します。アクセス許可を定義するときは、IAM プリンシパルポリシーはアクセスポイント ARN を使用する必要があり、アクセスポイントエイリアスベースのパスを使用する必要はなく、ARN は 条件にあり、リソースの指定には使用されないことに注意してください。
- ストアでカスタマーマネージドキー (CMK-KMS) を使用している場合は、IAM プリンシパルにキーに対する kms:decrypt アクセス許可があることを確認します。アカウント間の使用状況の設定については、KMS クロスアカウントアクセスガイドを参照してください。

タグベースのアクセスコントロールを使用しているポリシーがある場合は、以下を確認してください。

- シーケンスストアがタグの同期を完了していることを確認します。そのためには、ストアのステータスは active ではなく である必要がありますupdating。
- 読み取りセットとポリシーのタグキーまたはキー値にタイプミスがないことを確認します。

Athena で注釈ストアまたはバリアントストアが表示されないのはなぜですか？

Lake Formation では、共有されたストアに基づいてリソースリンクを作成してください。アクセス許可を持つリソースリンクを作成すると、ストアが Athena に表示されます。詳細については、「HealthOmics を使用するように Lake Formation を設定する」を参照してください。

Athena のデータストアにアクセスできないのはなぜですか？

注釈ストアまたはバリアントストアが表示されていても、アクセスが拒否されたことを示すエラーメッセージが表示されている場合は、使用しているクエリエンジンのバージョンを確認してください。エンジンバージョン 3 を使用して実行されるクエリのみがサポートされています。Athena クエリエンジンのバージョンの詳細については、Amazon Athena ドキュメントを参照してください。

のクォータ AWS HealthOmics

AWS は、HealthOmics クォータのデフォルト値をアカウントに入力します。特に明記されていない限り、各クォータ値はリージョンごとの最大値です。

> **Important**
>
> ほとんどのサービスクォータと API クォータの引き上げをリクエストできます。詳細については、以下のトピックを参照してください。

トピック

HealthOmics サービスクォータ

次の表に、HealthOmics サービスクォータとそのデフォルト値を示します。各リージョンの現在のクォータを表示するには、Service Quotas コンソールを開きます。

> **Important**
>
> Service Quotas コンソールを使用して、調整可能なクォータの引き上げをリクエストできます。

サービスクォータの詳細については、「Service Quotas ユーザーガイド」の「クォータの引き上げのリクエスト」を参照してください。 Service Quotas Service Quotas コンソールで使用できないクォータについては、クォータ引き上げフォームを使用します。

名前	デフォルト	引き上げ可能	説明
分析 - 最大注釈ストア	サポートされている各リージョン: 10	あり	現在の AWS リージョンの注釈ストアの最大数
分析 - 最大同時バリアントまたは注釈ストアのインポートジョブ	サポートされている各リージョン : 5	あり	現在の AWS リージョンでの同時インポートジョブの最大数
分析 - バリアントストアのインポートジョブあたりの最大ファイル数	サポートされている各リージョン: 1,000	あり	現在の AWS リージョンにおけるバリアントインポートジョブあたりのファイルの最大数
Analytics - 注釈ストアあたりの最大共有数	サポートされている各リージョン: 10	あり	現在の AWS リージョンにおける注釈ストアあたりの最大共有数
分析 - バリアントストアあたりの最大共有数	サポートされている各リージョン: 10	あり	現在の AWS リージョンにおけるバリアントストアあたりの最大共有数
分析 - バリアントインポートジョブの各ファイルの最大サイズ	サポートされている各リージョン: 20 GB	あり	現在の AWS リージョンのバリアントインポートジョブ内の 1 つのファイルの最大サイズ
分析 - 注釈インポートジョブの各ファイルの最大サイズ	サポートされている各リージョン: 20 GB	あり	現在の AWS リージョンの注釈インポートジョブ内の 1 つのファイルの最大サイズ

名前	デフォルト	引き上げ可能	説明
分析 - 最大バリアントストア	サポートされている各リージョン: 10	あり	現在の AWS リージョンのバリアントストアの最大数
Analytics - 注釈ストアあたりの最大バージョン	サポートされている各リージョン: 10	あり	現在の AWS リージョンの注釈ストアあたりのバージョンの最大数
ストレージ - 最大同時読み取りセットアクティベーションジョブ	サポートされている各リージョン: 25	あり	現在の AWS リージョンでの同時読み取りセットアクティベーションジョブの最大数
ストレージ - 最大同時シーケンスとリファレンスストアのエクスポートジョブ	サポートされている各リージョン：5	あり	現在の AWS リージョンのシーケンスまたはリファレンスストアからの同時エクスポートジョブの最大数
ストレージ - 最大同時シーケンスまたはリファレンスストアのインポートジョブ	サポートされている各リージョン：5	あり	現在の AWS リージョンのシーケンスまたはリファレンスストアの同時インポートジョブの最大数
ストレージ - シーケンスストアあたりの最大読み取りセット	サポートされている各リージョン: 1,000,000	あり	現在の AWS リージョンのシーケンスストア内の読み取りセットの最大数

名前	デフォルト	引き上げ可能	説明
ストレージ - リファレンスストアあたりの最大リファレンス	サポートされている各リージョン: 50	可能	現在の AWS リージョンのリファレンスストア内のリファレンスの最大数
ストレージ - 最大シーケンスストア	サポートされている各リージョン: 20	可能	現在の AWS リージョンのシーケンスストアの最大数
ワークフロー - 最大アクティブ GPUs	サポートされている各リージョン: 12	あり	現在の AWS リージョンでの同時アクティブ GPUs の最大数。us-east-1 および us-west-2 では、最大 500 の値に対するクォータ引き上げリクエストが自動的に承認されます。
ワークフロー - 動的実行ストレージを使用した最大同時アクティブ実行数	サポートされている各リージョン: 50	可能	現在の AWS リージョンで動的実行ストレージを使用するアクティブな実行の最大数。最大 200 の値に対するクォータ引き上げリクエストは自動的に承認されます。

名前	デフォルト	引き上げ可能	説明
ワークフロー - 静的実行ストレージを使用した最大同時アクティブ実行数	サポートされている各リージョン: 10	あり	現在の AWS リージョンで静的実行ストレージを使用するアクティブな実行の最大数。最大 50 個の値に対するクォータ引き上げリクエストは自動的に承認されます。
ワークフロー - 実行あたりの最大同時タスク数	サポートされている各リージョン: 25	あり	現在の AWS リージョンで実行される各実行の同時タスクの最大数。us-east-1 および us-west-2 では、最大 100 の値に対するクォータ引き上げリクエストが自動的に承認されます。
ワークフロー - 最大実行期間	サポートされている各リージョン: 604,800 秒	あり	現在の AWS リージョンにおけるワークフローの最大実行期間。
ワークフロー - 最大実行数 (アクティブまたは非アクティブ)	サポートされている各リージョン: 5,000	あり	現在の AWS リージョンでの実行 (アクティブまたは非アクティブ) の最大数。
ワークフロー - ワークフローあたりの最大共有数	サポートされている各リージョン: 100	可能	現在の AWS リージョンにおけるワークフローあたりの最大共有数

名前	デフォルト	引き上げ可能	説明
ワークフロー - 実行あたりの最大静的実行ストレージ容量	サポートされている各リージョン: 9,600	あり	現在の AWS リージョンでの実行ごとの最大静的実行ストレージ容量 (GiB)。us-east-1 および us-west-2 では、最大 50,000 の値に対するクォータ引き上げリクエストが自動的に承認されます。
ワークフロー - 最大ワークフロー	サポートされている各リージョン: 1,000	あり	現在の AWS リージョンのワークフローの最大数。
ワークフロー - StartRun オペレーションのトランザクション/秒 (TPS)	サポートされている各リージョン: 0.1	あり	現在の AWS リージョンにおける StartRun オペレーションの 1 秒あたりの最大トランザクション数 (TPS)。最大 1 の値に対するクォータ引き上げリクエストは自動的に承認されます。

HealthOmics 固定サイズクォータ

に加えてHealthOmics サービスクォータ、HealthOmics には固定サイズを持つクォータが含まれています。これらの値の増加をリクエストすることはできません。

特に明記されていない限り、各クォータにはリージョンあたりの最大値が一覧表示されます。

トピック

HealthOmics 分析の固定サイズクォータ

次の表は、分析クォータでサポートされている最大値を示しています。これらの値は調整できません。

名前	説明	最大値	調整可能 はい/いいえ
Analytics - 注釈ストアのインポートジョブあたりの最大ファイル数	注釈インポートジョブあたりのファイルの最大数。	1	なし

HealthOmics ストレージの固定サイズクォータ

次の表は、ストレージファイルでサポートされている最大値を示しています。これらの値は調整できません。

名前	説明	最大値	調整可能 はい/いいえ
ストレージ - 最大 S3 アクセスリソースポリシーサイズ	S3 アクセスリソースポリシーの最大サイズ	15 KB	なし
ストレージ - 最大伝播セットレベルタグ	S3 オブジェクトに伝播するストアあたりの設定レベルのタグキーの最大数	5	なし
ストレージ - アクティベーションジョブあ	アクティベーションジョブあたりの読み	20	なし

名前	説明	最大値	調整可能 はい/いいえ
たりの最大読み取りセット	取りセットの最大数。		
ストレージ - エクスポートジョブあたりの最大読み取りセット	エクスポートジョブあたりの読み取りセットの最大数。	100	なし
ストレージ - インポートジョブあたりの最大読み取りセット	インポートジョブあたりの読み取りセットの最大数。	100	なし
ストレージ - 最大リファレンスストア	リファレンスストアの最大数。	1	なし
ストレージ - 直接アップロードの最大パーツサイズ	シーケンスストアへの直接アップロードの最大パーツサイズ。	100 MB	なし
ストレージ - 直接アップロード用のファイル内の最大パート数	シーケンスストアに直接アップロードするためのファイル内のパートの最大数。	10,000	なし
ストレージ - 最大リファレンスサイズ	リファレンスストアにインポートできるリファレンスファイルの最大サイズ。	15 GB	なし
ストレージ - 読み取りセットの最大ソースサイズ	シーケンスストアにインポートできる読み取りセット内の 1 つのソースファイルの最大サイズ。	976 GB	なし

HealthOmics ワークフローの固定サイズクォータ

次の表は、ワークフロークォータでサポートされている最大値を示しています。これらの値は調整できません。

名前	説明	最大サイズ	調整可能 はい/いいえ
ワークフロー - 最大実行グループ	実行グループの最大数。	1,000	なし
ワークフロー - 最大実行キャッシュ	1 つのアカウントに対して作成できる実行キャッシュの最大数。 1 つ以上の実行が同じ実行キャッシュを共有できます。HealthOmics がアカウントごとにキャッシュできる実行数にはクォータはありません。	1,000	なし
ワークフロー - 最大ワークフローバージョン	ワークフローあたりのワークフローバージョンの最大数。	1,000	なし
ワークフロー - CPU インスタンスコンテナサイズ	CPU インスタンスの最大コンテナイメージサイズ。	45 GiB	なし
ワークフロー - GPU インスタンスコンテナサイズ	GPU インスタンスの最大コンテナイメージサイズ。	95 GiB	なし

名前	説明	最大サイズ	調整可能 はい/いいえ
GPU インスタンス / dev/shm 共有メモリ	GPU インスタンスあたりの共有メモリの最大量。	GPU あたり 8 GB	なし
ワークフロー - パラメータファイルの実行	実行パラメータファイルの最大サイズ。	50,000 バイト	なし
ワークフロー - ワークフローパラメータテンプレートファイル	ワークフローパラメータテンプレートファイルの最大エントリ数と最大ファイルサイズ。このクォータは、コンソールまたは API を使用して作成したワークフローに適用されます。	1,000 エントリ、400 KB	なし
ワークフロー - ワークフロー定義ファイルサイズ - API	API オペレーションまたは AWS SDK を使用してワークフローを作成するときのワークフロー定義ファイルの最大サイズ。	100 MB	なし
ワークフロー - ワークフロー定義ファイルサイズ - コンソール (直接アップロード)	コンソールを使用してワークフローを作成するときに、直接アップロードとして指定できるワークフロー定義ファイルの最大サイズ。	4.4 MB	なし

名前	説明	最大サイズ	調整可能 はい/いいえ
ワークフロー - ワークフロー定義ファイルサイズ - コンソール (Amazon S3 からアップロード)	コンソールを使用してワークフローを作成するときに、Amazon S3 からのアップロードとして提供できるワークフロー定義ファイルの最大サイズ。	100 MB	なし
ワークフロー - リポジトリサイズ	外部コードリポジトリの最大サイズ。	1 GiB	なし
ワークフロー - リポジトリの個々のファイルサイズ	外部コードリポジトリからの個々のファイルの最大サイズ。	100 MiB	なし
ワークフロー - README ファイルサイズ	README ファイルの最大サイズ。	500 KiB	なし

実行パラメータファイルのサイズを減らす方法の提案については、「」を参照してください実行パラメータサイズの管理。

HealthOmics Ready2Run ワークフロー固定サイズクォータ

各 Ready2Run ワークフローには最大入力ファイルサイズがあります。次の表では、ファイルサイズ単位をギビバイト (GiB) で示しています。これらの最大ファイルサイズは調整できません。

Ready2Run ワークフロー名	最大入力ファイルサイズ (GiB)	調整可能 (はい/いいえ)
AlphaFold for 601-1200	1	なし
最大 600 個のリテンション用の AlphaFold	1	なし
2x150 用の Bases2Fastq	1,000	なし

Ready2Run ワークフロー名	最大入力ファイルサイズ (GiB)	調整可能 (はい/いいえ)
2x300 用の Bases2Fastq	1,000	なし
Bases2Fastq for 2x75	500	なし
ESMFold 最大 800 個のリテンション	1	なし
GATK-BP fq2bam	64	なし
30x ゲノムの GATK-BP Germline bam2vcf	39	なし
30x ゲノム用の GATK-BP Germline fq2vcf	64	なし
GATK-BP Somatic WES bam2vcf	86	なし
NVIDIA Parabricks BAM2FQ2BAM WGS 最大 30X	80	なし
最大 50X NVIDIA Parabricks BAM2FQ2BAM WGS	120	なし
NVIDIA Parabricks BAM2FQ2BAM WGS 最大 5X	20	なし
NVIDIA Parabricks FQ2BAM WGS、最大 30X	71	なし
NVIDIA Parabricks FQ2BAM WGS、最大 50X	137	なし
NVIDIA Parabricks FQ2BAM WGS、最大 5X	13	なし

Ready2Run ワークフロー名	最大入力ファイルサイズ (GiB)	調整可能 (はい/いいえ)
NVIDIA Parabricks Germline DeepVariant WGS、最大 30X	71	なし
NVIDIA Parabricks Germline DeepVariant WGS、最大 50X	137	なし
NVIDIA Parabricks Germline DeepVariant WGS、最大 5X	12	なし
NVIDIA Parabricks Germline HaplotypeCaller WGS、最大 30X	71	なし
NVIDIA Parabricks Germline HaplotypeCaller WGS、最大 50X	137	なし
NVIDIA Parabricks Germline HaplotypeCaller WGS、最大 5X	13	なし
NVIDIA Parabricks Somatic Mutect2 WGS、最大 50X	196	なし
KallistoBUStools を使用した scRNAseq	119	なし
Salmon Alevin-fry を使用した scRNAseq	119	なし
scRNAseq と STARsolo	119	なし
最大 300 倍の Sentieon Germline BAM WES	9	なし
最大 32 倍の Sentieon Germline BAM WGS	18	なし

Ready2Run ワークフロー名	最大入力ファイルサイズ (GiB)	調整可能 (はい/いいえ)
最大 100 倍の Sentieon Germline FASTQ WES	5	なし
最大 300 倍の Sentieon Germline FASTQ WES	26	なし
最大 32 倍の Sentieon Germline FASTQ WGS	51	なし
ONT 用 Sentieon LongRead	25	なし
PacBio HiFi 用 Sentieon LongRead	58	なし
Sentieon Somatic WES	50	なし
Sentieon Somatic WGS	113	なし
最大 40 倍の Ultima Genomics DeepVariant	91	なし

HealthOmics API クォータ

HealthOmics には、API オペレーションに関連する以下のクォータがあります。示されている場合、クォータは調整可能です。引き上げをリクエストするには、クォータ引き上げフォームを使用します。

リストされている各 API オペレーションについて、クォータは各リージョンにおけるその API オペレーションの 1 秒あたりの最大トランザクション数 (TPS) です。

トピック

- 一般的な API クォータ
- Storage API クォータ
- ワークフロー API クォータ
- Analytics API クォータ

一般的な API クォータ

次の表に、複数のカテゴリ (ストレージ、ワークフロー、分析) に適用される一般的な API オペレーションを示します。

API オペレーション	デフォルトの最大 TPS	調整可能 (はい/いいえ)
AcceptShare、CreateShare、DeleteShare、GetShare、ListShares	1 TPS	あり

Storage API クォータ

次の表に、ストレージ API オペレーションを示します。

ストレージ API オペレーション	デフォルトの最大 TPS	調整可能 (はい/いいえ)
CreateSequenceStore、UpdateSequenceStore、DeleteSequenceStore、CreateReferenceStore、DeleteReferenceStore	1 TPS	あり
BatchDeleteReadSet、DeleteReference	1 TPS	あり
CreateMultipartReadSetUpload、CompleteMultipartReadSetUpload、AbortMultipartReadSetUpload	1 TPS	なし
GetS3AccessPolicy, PutS3AccessPolicy, DeleteS3AccessPolicy	1 TPS	あり

ストレージ API オペレーション	デフォルトの最大 TPS	調整可能 (はい/いいえ)
GetReference	10 TPS	あり
UploadReadSetPart	10 TPS	あり
GetReadSet	30 TPS	あり
GetSequenceStore、ListSequenceStores	5 TPS	あり
GetReadSetMetadata、ListReadSets	5 TPS	あり
StartReadSetImportJob、GetReadSetImportJob、ListReadSetImportJobs	5 TPS	あり
StartReadSetExportJob、GetReadSetExportJob、ListReadSetExportJobs	5 TPS	あり
ListReferenceStores	5 TPS	あり
StartReferencetImportJob、GetReferenceImportJob、ListReferenceImportJobs	5 TPS	あり
ListReferences、GetReferenceMetadata	5 TPS	あり
StartReadsetActivationJob	5 TPS	あり
ListReadsetActivationJobs、GetReadSetActivationJob	5 TPS	あり
ListMultipartReadSetUploads、ListReadSetUploadParts	5 TPS	あり

ストレージ API オペレーション	デフォルトの最大 TPS	調整可能 (はい/いいえ)
TagResource、UntagResource、ListTagsForResource	5 TPS	あり

ワークフロー API クォータ

次の表に、ワークフロー API オペレーションを示します。

ワークフロー API オペレーション	デフォルトの最大 TPS	調整可能 (はい/いいえ)
StartRun	0.1 TPS	あり
CreateWorkflow	5 TPS	あり
CancelRun、DeleteRun、GetRun、GetRunTask、ListRunTasks、ListRuns	10 TPS	あり
CreateRunGroup、DeleteRunGroup、GetRunGroup、ListRunGroups、UpdateRunGroup	10 TPS	あり
CreateRunCache、UpdateRunCache、DeleteRunCache、GetRunCache、ListRunCaches	10 TPS	あり
DeleteWorkflow、GetWorkflow、ListWorkflows、UpdateWorkflow	10 TPS	あり

Analytics API クォータ

次の表に、分析 API オペレーションを示します。

Analytics API オペレーション	デフォルトの最大 TPS	調整可能 (はい/いいえ)
CreateVariantStore、DeleteVariantStore、GetVariantStore、ListVariantStores、UpdateVariantStore	1 TPS	なし
StartVariantImportJob、CancelVariantImportJob、GetVariantImportJob、ListVariantImportJobs	1 TPS	なし
CreateAnnotationStore、DeleteAnnotationStore、GetAnnotationStore、ListAnnotationStores、UpdateAnnotationStore	1 TPS	なし
StartAnnotationImportJob、ListAnnotationImportJobs、GetAnnotationImportJob、CancelAnnotationImportJob	1 TPS	なし

HealthOmics ユーザーガイドのドキュメント履歴

次の表に、HealthOmics のドキュメントリリースを示します。

変更	説明	日付
新しい README とリポジトリの統合機能	外部コードリポジトリと README ファイルからワークフローを作成するサポートが追加されました。	2025 年 7 月 24 日
新機能	HealthOmics で Nextflow 自動パラメータ補間のサポートが追加されました。詳細については、HealthOmics ワークフローのパラメータテンプレートファイル」を参照してください。	2025 年 6 月 27 日
新機能	HealthOmics でワークフローバージョニングのサポートが追加されました。詳細については、HealthOmics でのワークフローのバージョニング」を参照してください。	2025 年 4 月 18 日
新機能	HealthOmics は、動的実行ストレージのエラスティックスループットを追加しました。詳細については、HealthOmics でストレージタイプを実行する」を参照してください。	2025 年 4 月 16 日
新機能	HealthOmics は、Sequence Store S3 ロケーションの属性ベースのアクセスコントロールを追加し、最大 5 つの読み	2024 年 11 月 22 日

	込みセットタグを Sequence Store S3 オブジェクトに同期する機能を追加しました。詳細については、HealthOmics シーケンスストアの作成」を参照してください。	
新機能	HealthOmics は、プライベートワークフローの再開とも呼ばれるコールキャッシュのサポートを追加しました。詳細については、「キャッシュの呼び出し」を参照してください。	2024 年 11 月 20 日
新機能	HealthOmics は、シーケンスストア入力ジョブとリードセット間のマッピングに役立つ新しい API フィールドを追加しました。	2024 年 8 月 29 日
新機能	HealthOmics で Nextflow バージョン管理のサポートが追加されました。詳細については、「Nextflow バージョン」を参照してください。	2024 年 8 月 14 日
新機能	HealthOmics は、共有ワークフローと動的実行ストレージのサポートを追加しました。	2024 年 4 月 30 日
新機能	HealthOmics は、リファレンスストアとシーケンスストアへの Amazon S3 アクセスのサポートと、SHA256 ETags のサポートを追加しました。	2024 年 4 月 22 日

新機能	HealthOmics がシーケンスストアにエンティティタグ (ETags) を追加しました。	2023 年 10 月 6 日
新機能	HealthOmics に注釈ストアのバージョニングと分析ストアの共有が追加されました。	2023 年 8 月 15 日
新機能	HealthOmics は、共通ワークフロー言語 (CWL) を HealthOmics ワークフローでサポートされている言語として追加しました。	2023 年 6 月 30 日
新機能	HealthOmics は、新しい Ready2Run ワークフロー、ワークフローの GPU サポート、注釈ストアのデータ解析、HealthOmics ストレージへの直接アップロード、EventBridge との統合を追加しました。	2023 年 5 月 15 日
新しいマネージドポリシー	HealthOmics は、フルアクセスを提供する新しい管理ポリシーを追加しました。詳細については、「AWS 管理ポリシー」を参照してください。	2023 年 2 月 23 日
新しいマネージドポリシー	HealthOmics は、アクセスを読み取り専用に制限する新しい管理ポリシーを追加しました。詳細については、「AWS 管理ポリシー」を参照してください。	2022 年 11 月 29 日
初回リリース	HealthOmics ユーザーガイドの初回リリース	2022 年 11 月 29 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。