Add a permission의

AWS HealthOmics

버전 latest

AWS HealthOmics: Add a permission의

Table of Contents

AWS HealthOmics란 무엇인가요?

AWS HealthOmics 는 생물 정보학 워크플로 뒤에 있는 복잡한 인프라를 완전히 관리하여 임상 진단 테스트, 약물 발견 및 농업 연구를 가속화하는 HIPAA 적격 서비스입니다. HealthOmics는 업계 표준 워크플로 언어(WDL, Nextflow, CWL)를 지원하고 생물 정보 인프라를 원활하게 확장하여 매일 수만 건의 테스트 데이터를 지원하며,이 모든 것이 예측 가능한 샘플당 비용으로 이루어집니다. HealthOmics는 컴퓨팅 리소스 관리 및 워크플로 엔진 유지 관리와 같은 기술적 복잡성을 처리하므로 과학적 혁신에 전적으로 집중할 수 있습니다.

항목

중요 공지 사항

HealthOmics는 데이터 전송, 저장, 형식 지정 또는 표시와 워크플로 관리를 위한 인프라 및 구성 지원 제공에만 사용됩니다. HealthOmics는 전문적인 의학적 조언, 진단 또는 치료를 대체하지 않으며 질병 또는 건강 상태를 치료, 치료, 완화, 예방 또는 진단하기 위한 것이 아닙니다. 사용자는 임상 의사 결정을 알리기 위한 타사 제품과 관련된 경우를 AWS HealthOmics포함하여 사용의 일부로 인적 검토를 도입할 책임이 있습니다.

HealthOmics 기능

HealthOmics의 기본 사용 사례:

- 임상 진단 - 예측 가능한 비용과 테스트 볼륨에 따라 증가하는 완전 관리형 인프라를 사용하여 진단 테스트 워크플로를 구축하고 확장합니다.
- 약물 검색 - 수백만 명의 잠재적 후보에 걸쳐 신속한 반복을 가능하게 하는 대규모로 생물학적 파운데이션 모델을 오케스트레이션하여 치료 연구를 가속화합니다.

- 농업 연구 - 식품 보안 및 농업 생산성을 개선하는 AI 기반 워크플로를 통해 가뭄 방지 및 내충격성과 같은 작물 특성을 개선합니다.

HealthOmics의 주요 이점:

- 확장성 - 100,000개 이상의 동시 vCPUs에서 워크플로를 확장하여 인프라 관리 없이 샘플당 예측 가능한 비용으로 매일 수만 개의 테스트를 지원합니다.
- 인프라가 아닌 과학에 집중 -가 인프라 오케스트레이션 및 APIs를 사용합니다. AWS
- 규정 준수 유지 - 포괄적인 감사 추적, 데이터 출처 추적 및 즉시 사용 가능한 임상 워크플로용으로 설계된 HIPAA 적격 인프라 out-of-the-box는 규제 요구 사항을 충족하는 솔루션 개발을 지원합니다.

HealthOmics는 세 가지 주요 구성 요소로 구성됩니다.

- HealthOmics 워크플로 - 자동으로 프로비저닝되고 규모가 조정된 인프라에서 생물 정보학 계산을 실행합니다.
- HealthOmics 스토리지 - 기가베이스당 저렴한 비용으로 페타바이트의 유전체학 데이터를 효율적으로 저장하고 공유합니다.
- HealthOmics 분석 - 멀티오믹스 및 멀티모달 분석을 위한 게놈 데이터를 준비합니다.

이러한 구성 요소를 독립적으로 사용하거나 end-to-end 솔루션을 위해 결합합니다.

HealthOmics 개념

이 주제에서는이 가이드에서 사용한 HealthOmics의 용어를 이해하는 데 도움이 되도록 HealthOmics와 관련된 주요 개념 및 용어에 대한 정의를 다룹니다.

주제

- 워크플로
- 스토리지
- 분석

워크플로

HealthOmics 워크플로를 사용하면 게놈 데이터를 처리하고 분석할 수 있습니다.

- 워크플로 - 파라미터 및 도구에 대한 참조를 포함한 엔드 투 엔드 프로세스의 전체 정의입니다. 워크플로 정의는 WDL, Nextflow 또는 CWL로 표현할 수 있습니다. 생성된 각 워크플로에는 고유한 식별자가 있습니다.
- 실행 - 워크플로의 단일 호출입니다. 개별 실행은 정의된 입력 데이터를 사용하고 출력을 생성합니다. 생성된 각 실행에는 고유한 식별자가 있습니다.
- 작업 - 실행 내의 개별 프로세스입니다. HealthOmics 워크플로는 이러한 정의된 컴퓨팅 사양을 사용하여 작업을 실행합니다. 각 작업에는 고유한 식별자가 있습니다.
- 실행 그룹 - 최대 vCPU, 최대 기간 또는 최대 동시 실행을 설정하여 실행당 사용되는 컴퓨팅 리소스를 제한할 수 있는 실행 그룹입니다. 실행 그룹 내에서 실행에 대한 우선 순위를 지정하고 구성할 수 있습니다. 예를 들어 우선 순위가 낮은 실행보다 우선 순위가 높은 실행을 수행하여 우선 순위 대기열을 생성하도록 지정할 수 있습니다. 실행 그룹을 사용하는 것은 선택 사항이며 각 실행 그룹에는 고유한 식별자가 있습니다.

스토리지

데이터 스토리지는 게놈 시퀀스 및 관련 정보에 대한 시퀀스 저장소와 모든 참조 게놈에 대한 참조 저장소로 구분됩니다. 다음 용어는 HealthOmics와 관련된 구현을 설명합니다.

- 시퀀스 스토어 - 게놈 파일을 저장하기 위한 데이터 스토어입니다. HealthOmics 내에 하나 이상의 시퀀스 스토어가 있을 수 있습니다. 시퀀스 스토어에서 액세스 권한 및 AWS KMS 암호화를 설정하여 데이터에 액세스할 수 있는 사용자를 제어할 수 있습니다.
- 읽기 세트 - 읽기 세트는 FASTQ, BAM 또는 CRAM 형식으로 저장되는 게놈 읽기의 추상화입니다. 읽기 세트는 시퀀스 저장소로 가져오고 메타데이터로 주석을 달 수 있습니다. 속성 기반 액세스 제어(ABAC)를 사용하여 읽기 세트에 권한을 적용할 수 있습니다.
- 참조 - 유전체 참조는 유전체에서 특정 읽기 또는 읽기 그룹이 매핑되는 위치를 식별하기 위해 읽기와 함께 사용됩니다. FASTA 형식이며 참조 스토어에 저장됩니다.
- 참조 스토어 - 참조 유전체의 저장을 위한 데이터 스토어입니다. 각 계정 및 리전에 단일 참조 저장소를 가질 수 있습니다.

분석

HealthOmics Analytics를 사용하여 게놈 데이터를 변환하고 분석할 수 있습니다. 변형 저장소 또는 주석 저장소를 생성하여 쿼리에 대한 추가 정보를 포함합니다.

- 변형 저장소 - 변형 데이터를 모집단 규모로 저장하는 데이터 저장소입니다. 변형 저장소는 게놈 변형 호출 형식(gVCF)과 VCF 입력을 모두 지원합니다.

- 주석 저장소 - TSV/CSV, VCF 또는 일반 기능 형식(GFF3) 파일의 주석 데이터베이스와 같은 주석 데이터베이스를 나타내는 데이터 저장소입니다. 주석 저장소는 가져오기 중에 변형 저장소와 동일한 좌표계에 매핑됩니다.

관련 서비스

다음 서비스는 HealthOmics에서 작동합니다.

- Amazon Elastic Container Registry - 각 프라이빗 워크플로는 Amazon ECR 이미지(프라이빗 Amazon ECR 리포지토리)를 사용하여 워크플로를 실행하는 데 필요한 모든 실행 파일, 라이브러리 및 스크립트를 포함합니다.
- Amazon Simple Storage Service – Amazon S3는 저장 및 워크플로 데이터를 위한 파일 스토리지를 제공합니다.
- AWS Lake Formation - Lake Formation은 Analytics 데이터 스토어에 대한 데이터 액세스를 관리합니다.
- Amazon Athena - Athena를 사용하여 변형 저장소에 대한 쿼리를 수행합니다.
- Amazon SageMaker AI - SageMaker AI를 사용하여 Jupyter 노트북을 사용하여 HealthOmics 작업을 실행합니다.
- GitHub connections - 연결을 사용하여 외부 코드 리포지토리를 HealthOmics 워크플로에 연결합니다.

HealthOmics에 액세스하는 방법

관리 콘솔, CLI, SDKs 또는 API를 사용하여 AWS HealthOmics 기능에 액세스할 수 있습니다.

- AWS 관리 콘솔 - HealthOmics에 액세스하는 데 사용할 수 있는 웹 인터페이스를 제공합니다.
- AWS Command Line Interface (AWS CLI) - Windows, macOS 및 Linux AWS HealthOmics를 포함하여 다양한 AWS 서비스에 대한 명령을 제공합니다. 설치에 대한 자세한 내용은 단원을 AWS CLI참조하십시오AWS Command Line Interface.
- AWS SDKs- 다양한 프로그래밍 언어 및 플랫폼(Java, Python, Ruby, .NET, iOS 및 Android 포함)을 위한 라이브러리 및 샘플 코드로 구성된 SDKs(소프트웨어 개발 키트)를 AWS 제공합니다. SDKs는 프로그래밍 방식으로 HealthOmics를 사용하는 편리한 방법을 제공합니다. 자세한 내용은 AWS SDK 개발자 센터를 참조하세요.
- AWS API - API 작업을 사용하여 HealthOmics에 프로그래밍 방식으로 액세스하고 관리할 수 있습니다. 자세한 내용은 HealthOmics API 참조를 참조하세요.

AWS HealthOmics의 리전 및 엔드포인트

리전 및 엔드포인트의 전체 목록은 AWS 일반 참조를 참조하세요.

기본적으로 활성 상태인 AWS 리전 외에도 활성화해야 하는 옵트인 리전도 있습니다. 리전을 활성화하거나 비활성화하는 방법에 대한 자세한 내용은 AWS 계정 관리 안내서의 계정에서 사용할 수 있는 AWS 리전 지정을 참조하세요.

자세히 알아보기

다음 워크숍 및 자습서에서 HealthOmics에 대해 자세히 알아보세요.

- HealthOmics 워크숍 - HealthOmics 엔드 투 엔드 워크숍
- AWS 게놈 리소스 - 게놈과 관련된 퍼블릭 Amazon ECR 리포지토리
- Python 자습서 - HealthOmics 스토리지, 분석 및 워크플로를 다루는 GitHub의 Jupyter 노트북 자습서

다음을 AWS 제공하는 추가 HealthOmics 도구에 익숙해지세요.

- WDL 린터 - WDL용 HealthOmics 린터
- Nextflow linter – Nextflow용 HealthOmics linter
- HealthOmics Amazon ECR 헬퍼 도구 - HealthOmics용 Amazon ECR 헬퍼 도구
- GitHub의 HealthOmics 도구 - HealthOmics 작업 도구(전송 관리자, URI 구문 분석기, Omics 재실행, 분석기 실행).

AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경

신중한 고려 끝에 2025년 11월 7일부터 신규 고객에게 AWS HealthOmics 변형 저장소와 주석 저장소를 해지하기로 결정했습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다.

다음 섹션에서는 변형 저장소 및 분석 저장소를 새 솔루션으로 이동하는 데 도움이 되는 마이그레이션 옵션에 대해 설명합니다. 질문이나 우려 사항이 있는 경우 support.console.aws.amazon.com 지원 사례를 생성합니다.

주제

마이그레이션 옵션 개요

다음 마이그레이션 옵션은 변형 저장소 및 주석 저장소를 사용하는 대안을 제공합니다.

1. ETL 로직의 HealthOmics 제공 참조 구현을 사용합니다.

   스토리지에 S3 테이블 버킷을 사용하고 기존 AWS 분석 서비스를 계속 사용합니다.
2. 기존 AWS 서비스의 조합을 사용하여 솔루션을 생성합니다.

   ETL의 경우 사용자 지정 Glue ETL 작업을 작성하거나 EMR에서 오픈 소스 HAIL 또는 GLOW 코드를 사용하여 변형 데이터를 변환할 수 있습니다.

   스토리지에 S3 테이블 버킷 사용 및 기존 AWS 분석 서비스 계속 사용
3. 변형 및 주석 저장소 대안을 제공하는 AWS 파트너를 선택합니다.

ETL 로직의 마이그레이션 옵션

ETL 로직에 대해 다음 마이그레이션 옵션을 고려하세요.

1. HealthOmics는 현재 변형 저장소 ETL 로직을 참조 HealthOmics 워크플로로 제공합니다. 이 워크플로의 엔진을 사용하여 변형 저장소와 정확히 동일한 변형 데이터 ETL 프로세스에 전원을 공급할 수 있지만 ETL 로직을 완전히 제어할 수 있습니다.

   이 참조 워크플로는 요청에 따라 사용할 수 있습니다. 액세스를 요청하려면 support.console.aws.amazon.com 지원 사례를 생성합니다.
2. 변형 데이터를 변환하려면 사용자 지정 Glue ETL 작업을 작성하거나 EMR에서 오픈 소스 HAIL 또는 GLOW 코드를 사용할 수 있습니다.

스토리지를 위한 마이그레이션 옵션

서비스 호스팅 데이터 스토어를 대체하기 위해 Amazon S3 테이블 버킷을 사용하여 사용자 지정 테이블 스키마를 정의할 수 있습니다. 테이블 버킷에 대한 자세한 내용은 Amazon S3 사용 설명서의 테이블 버킷을 참조하세요.

Amazon S3의 완전 관리형 Iceberg 테이블에 테이블 버킷을 사용할 수 있습니다.

지원 사례를 제기하여 HealthOmics 팀에 변형 또는 주석 저장소의 데이터를 구성한 Amazon S3 테이블 버킷으로 마이그레이션하도록 요청할 수 있습니다.

Amazon S3 테이블 버킷에 데이터가 채워지면 변형 저장소와 주석 저장소를 삭제할 수 있습니다. 자세한 내용은 HealthOmics 분석 저장소 삭제를 참조하세요.

분석

데이터 분석의 경우 Amazon Amazon Athena EMR, Amazon Redshift 또는 Amazon Quick과 같은 AWS 분석 서비스를 계속 사용합니다.

AWS 파트너

사용자 지정 가능한 ETL, 테이블 스키마, 기본 제공 쿼리 및 분석 도구, 데이터와 상호 작용하기 위한 사용자 인터페이스를 제공하는 AWS 파트너와 협력할 수 있습니다.

예제

다음 예제에서는 VCF 및 GVCF 데이터를 저장하는 데 적합한 테이블을 생성하는 방법을 보여줍니다.

Athena DDL

Athena에서 다음 DDL 예제를 사용하여 VCF 및 GVCF 데이터를 단일 테이블에 저장하는 데 적합한 테이블을 생성할 수 있습니다. 이 예제는 변형 저장소 구조와 정확히 동일하지는 않지만 일반적인 사용 사례에 적합합니다.

테이블을 생성할 때 DATABASE_NAME 및 TABLE_NAME에 대한 고유한 값을 생성합니다.

```
 CREATE TABLE <DATABASE_NAME>. <TABLE_NAME> (
  sample_name string,
  variant_name string COMMENT 'The ID field in VCF files, '.' indicates no name',
  chrom string,
  pos bigint,
  ref string,
  alt array <string>,
  qual double,
  filter string,
  genotype string,
  info map <string, string>,
  attributes map <string, string>,
  is_reference_block boolean COMMENT 'Used in GVCF for non-variant sites')
PARTITIONED BY (bucket(128, sample_name), chrom)
LOCATION '{URL}/'
TBLPROPERTIES (
  'table_type'='iceberg',
  'write_compression'='zstd'
);
```

Python을 사용하여 테이블 생성(Athena 제외)

다음 Python 코드 예제에서는 Athena를 사용하지 않고 테이블을 생성하는 방법을 보여줍니다.

```
 import boto3
from pyiceberg.catalog import Catalog, load_catalog
from pyiceberg.schema import Schema
from pyiceberg.table import Table
from pyiceberg.table.sorting import SortOrder, SortField, SortDirection, NullOrder
from pyiceberg.partitioning import PartitionSpec, PartitionField
from pyiceberg.transforms import IdentityTransform, BucketTransform
from pyiceberg.types import (
    NestedField,
```

```
    StringType,
    LongType,
    DoubleType,
    MapType,
    BooleanType,
    ListType
)


def load_s3_tables_catalog(bucket_arn: str) -> Catalog:
    session = boto3.session.Session()
    region = session.region_name or 'us-east-1'

    catalog_config = {
        "type": "rest",
        "warehouse": bucket_arn,
        "uri": f"https://s3tables.{region}.amazonaws.com/iceberg",
        "rest.sigv4-enabled": "true",
        "rest.signing-name": "s3tables",
        "rest.signing-region": region
    }

    return load_catalog("s3tables", **catalog_config)


def create_namespace(catalog: Catalog, namespace: str) -> None:
    try:
        catalog.create_namespace(namespace)
        print(f"Created namespace: {namespace}")
    except Exception as e:
        if "already exists" in str(e):
            print(f"Namespace {namespace} already exists.")
        else:
            raise e


def create_table(catalog: Catalog, namespace: str, table_name: str, schema: Schema,
                 partition_spec: PartitionSpec = None, sort_order: SortOrder = None) ->
 Table:
    if catalog.table_exists(f"{namespace}.{table_name}"):
        print(f"Table {namespace}.{table_name} already exists.")
        return catalog.load_table(f"{namespace}.{table_name}")

    create_table_args = {
```

```
        "identifier": f"{namespace}.{table_name}",
        "schema": schema,
        "properties": {"format-version": "2"}
    }

    if partition_spec is not None:
        create_table_args["partition_spec"] = partition_spec
    if sort_order is not None:
        create_table_args["sort_order"] = sort_order

    table = catalog.create_table(**create_table_args)
    print(f"Created table: {namespace}.{table_name}")
    return table


def main(bucket_arn: str, namespace: str, table_name: str):
    # Schema definition
    genomic_variants_schema = Schema(
        NestedField(1, "sample_name", StringType(), required=True),
        NestedField(2, "variant_name", StringType(), required=True),
        NestedField(3, "chrom", StringType(), required=True),
        NestedField(4, "pos", LongType(), required=True),
        NestedField(5, "ref", StringType(), required=True),
        NestedField(6, "alt", ListType(element_id=1000, element_type=StringType(),
 element_required=True), required=True),
        NestedField(7, "qual", DoubleType()),
        NestedField(8, "filter", StringType()),
        NestedField(9, "genotype", StringType()),
        NestedField(10, "info", MapType(key_type=StringType(), key_id=1001,
 value_type=StringType(), value_id=1002)),
        NestedField(11, "attributes", MapType(key_type=StringType(), key_id=2001,
 value_type=StringType(), value_id=2002)),
        NestedField(12, "is_reference_block", BooleanType()),
        identifier_field_ids=[1, 2, 3, 4]
    )

    # Partition and sort specifications
    partition_spec = PartitionSpec(
        PartitionField(source_id=1, field_id=1001, transform=BucketTransform(128),
 name="sample_bucket"),
        PartitionField(source_id=3, field_id=1002, transform=IdentityTransform(),
 name="chrom")
    )
```

```
    sort_order = SortOrder(
        SortField(source_id=3, transform=IdentityTransform(),
 direction=SortDirection.ASC, null_order=NullOrder.NULLS_LAST),
        SortField(source_id=4, transform=IdentityTransform(),
 direction=SortDirection.ASC, null_order=NullOrder.NULLS_LAST)
    )

    # Connect to catalog and create table
    catalog = load_s3_tables_catalog(bucket_arn)
    create_namespace(catalog, namespace)
    table = create_table(catalog, namespace, table_name, genomic_variants_schema,
 partition_spec, sort_order)

    return table


if __name__ == "__main__":
    bucket_arn = 'arn:aws:s3tables:<REGION>:<ACCOUNT_ID>:bucket/<TABLE_BUCKET_NAME'
    namespace = "variant_db"
    table_name = "genomic_variants"

    main(bucket_arn, namespace, table_name)
```

HealthOmics 설정

설정하려면 AWS HealthOmics에 가입하고 AWS 계정, 관리 사용자를 생성하고, 추가 사용자의 액세스를 안전하게 관리합니다.

주제

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 이동하고 내 계정을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자AWS Management Console로에 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 User Guide의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자(콘솔)에 대한 가상 MFA 디바이스 활성화를 참조하세요.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 자격 증명 소스 IAM Identity Center 디렉터리 로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요.

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS 로그인 사용 설명서의 AWS 액세스 포털에 로그인을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

HealthOmics에 대한 IAM 권한 생성

HealthOmics를 사용하려면 다음 IAM 권한을 구성합니다.

- 계정의 사용자가 HealthOmics에 액세스하기 위한 IAM 자격 증명 기반 정책입니다.
- HealthOmics가 사용자를 대신하여 리소스에 액세스할 수 있는 IAM 서비스 역할입니다.
- 사용자 및 HealthOmics 서비스가 리소스에 액세스할 수 있는 다른 서비스(예: Lake Formation 및 Amazon ECR)의 권한.

HealthOmics에 대한 IAM 권한 구성에 대한 자세한 내용은 섹션을 참조하세요HealthOmics에 대한 IAM 권한.

외부 코드 리포지토리와 연결

를 사용하면를 통해 Git 기반 리포지토리를 사용하여 워크플로를 관리할 AWS HealthOmics수 있습니다 AWS CodeConnections. HealthOmics는이 연결을 사용하여 소스 코드 리포지토리에 액세스합니다.

외부 코드 리포지토리로 작업하기 전에 연결 설정 가이드에 따라 작업을 시작합니다 AWS CodeConnections. AWS 계정에 대한 적절한 IAM 정책 및 권한을 생성했는지 확인합니다. 지원되는 Git 공급자 목록 및 자세한 내용은 연결을 생성할 수 있는 타사 공급자는 무엇입니까?를 참조하세요.

연결 생성

선호하는 리포지토리 공급자와의 연결을 생성하려면 연결 생성 자습서를 따르세요.

HealthOmics에서 Amazon Q CLI 사용

Amazon Q CLI는와 자연어 상호 작용을 AWS HealthOmics제공하므로 대화형 명령을 사용하여 복잡한 게놈 워크플로 및 분석 작업을 수행할 수 있습니다. Amazon Q CLI를 사용하려면 Amazon Q가 리소스에 액세스할 수 있도록 HealthOmics 및 기타 서비스(예: CloudWatch, Amazon ECR 또는 Amazon S3)에 대한 IAM 권한을 구성해야 합니다.

HealthOmics Agentic 생성형 AI 자습서에서는 컨텍스트 파일을 구성하고 Amazon Q CLI가 AWS HealthOmics 워크플로를 생성, 실행 및 최적화할 수 있도록 하는 step-by-step 지침을 제공합니다.

HealthOmics 시작하기

HealthOmics를 시작하려면 HealthOmics에 대한 IAM 권한 및 역할을 올바르게 설정했는지 확인합니다.

HealthOmics 콘솔에서 Ready2Run 워크플로 사용

다음 연습에서는 Ready2Run 워크플로를 사용하는 방법을 보여줍니다. Ready2Run 워크플로는 워크플로를 실행하는 데 필요한 파라미터 및 도구 참조로 미리 구성됩니다. 워크플로 게시자는 샘플 데이터를 제공하므로 자체 데이터를 생성할 필요가 없습니다.

1. HealthOmics 콘솔을 엽니다.
2. 왼쪽 상단의 탐색 창(☰)을 선택하고 Ready2Run 워크플로를 선택합니다.
3. Ready2Run 워크플로 페이지에서 ESMFold for up to 800 residues 워크플로를 선택합니다. 콘솔에서 해당 워크플로의 세부 정보 페이지가 열립니다.
4. 세부 정보 탭은 워크플로에 대한 정보를 제공합니다. 워크플로를 시도하려면 페이지 오른쪽 상단에서 실행 시작을 선택합니다.
5. 실행 세부 정보 지정 페이지에서 실행 이름을 입력합니다.
6. 실행 출력에 대한 Amazon S3 위치를 입력하거나 선택합니다.
7. 메타데이터 보존 모드 실행에서 runmeta 데이터를 보존할지 또는 제거할지를 선택합니다.
8. 서비스 역할 패널에서 새 서비스 역할 생성 및 사용을 선택합니다.
9. 다음을 선택합니다.
10. 파라미터 값 추가 페이지에서 Ready2Run 테스트 데이터를 사용하여 워크플로 실행을 선택합니다.
11. 다음을 선택합니다.
12. 입력을 검토한 다음 실행 시작을 선택합니다.

Amazon Q CLI에 대한 프롬프트 예제

Amazon Q CLI는 자연어 명령을 AWS HealthOmics 사용하여에서 게놈 워크플로 및 분석 작업을 실행할 수 있습니다. 다음 예제 프롬프트를 사용하면 워크플로를 생성하고, 실행을 관리하고, 게놈 데이터를 분석할 수 있습니다. HealthOmics에 대한 자세한 내용과 예제 프롬프트는 GitHub의 HealthOmics Agentic 생성형 AI 자습서를 참조하세요.

- "WDL 1.1 워크플로 파일을 HealthOmics에서 실행되는 `main.wdl` 대로 생성합니다. 워크플로는 참조 유전체를 입력 및 fastq 파일 페어로 사용합니다. BWA를 사용하여 참조 유전체를 인덱싱한 다음 각 fastq 파일 쌍을 참조에 매핑합니다. 마지막으로 매핑된 각 BAM을 단일 BAM 파일에 병합하고이 파일을 베이 인덱스로 출력합니다."
- "워크플로를 패키징하고 HealthOmics에서 생성합니다."
- "Amazon S3 버킷의 실제 파일을 사용하도록 input.json 파일 업데이트"`omics-my-bucket-with-genome-data`(특정 Amazon S3 버킷 위치를 제공하거나 Amazon Q가 탐색하도록 허용)
- "Amazon ECR 리포지토리에서 적절한 컨테이너를 찾고 이를 사용할 수 있도록 input.json을 업데이트합니다."
- "워크플로를 실행할 때 사용할 적절한 IAM 역할을 찾거나 생성합니다."
- "내 워크플로에 대한 실행 캐시 생성"
- "HealthOmics에서 워크플로 실행"
- "실행 상태 확인"

> **Warning**
>
> Amazon Q CLI로 작업할 때는 계속하기 전에 생성된 모든 콘텐츠와 제안된 작업을 검토합니다. 응답 품질을 개선하고 워크플로의 요구 사항에 맞는 피드백을 제공합니다. 자세한 내용은 Amazon Q의 보안 고려 사항 및 모범 사례를 참조하세요.

HealthOmics의 프라이빗 워크플로

자체 워크플로 정의를 생성하려는 경우 프라이빗 워크플로를 사용합니다. 워크플로 정의는 워크플로에 대한 정보를 지정하고 워크플로 작업을 정의합니다. 실행은 워크플로의 단일 호출이고 작업은 실행 내의 단일 프로세스입니다.

HealthOmics는 워크플로 설명 언어(WDL), 공통 워크플로 언어(CWL) 또는 Nextflow에서 생성하는 워크플로 정의를 지원합니다.

HealthOmics 워크플로는 다음과 같은 선택적 기능을 제공합니다.

- Run groups - 실행 그룹에 프라이빗 워크플로를 추가하여 컴퓨팅 사용량을 제어할 수 있습니다. 실행 그룹은 최대 동시 실행 및 최대 실행 기간과 같은 리소스 제한 세트를 공유하는 워크플로 실행 모음입니다. 이러한 제한을 설정하여 실행 그룹이 사용하는 컴퓨팅 리소스를 제어합니다.
- Call caching - 호출 캐시를 사용하여 작업 출력을 저장하고 재사용할 수 있으므로 실행 시간이 단축되고 컴퓨팅 비용이 절감됩니다.
- Sharing workflows - 프라이빗 워크플로를 동일한 리전 AWS 계정 의 다른와 공유할 수 있습니다.
- Workflow versions - 프라이빗 워크플로의 버전을 생성할 수 있습니다. 워크플로 버전 관리는 사용자가 업데이트된 기능 사용을 시작할 시기를 선택할 수 있는 기능을 제공합니다. 워크플로 버전은 변경할 수 없으며 워크플로와 동일한 수준의 데이터 출처를 제공합니다.

워크플로에 대한 IAM 권한 구성에 대한 자세한 내용은 섹션을 참조하세요HealthOmics에 대한 IAM 권한.

HealthOmics 프라이빗 워크플로를 사용하는 방법에 대한 전체 예제는 HealthOmics Github 자습서 또는 HealthOmics.

주제

- HealthOmics에서 프라이빗 워크플로 생성
- HealthOmics의 워크플로 버전 관리
- HealthOmics 실행 사용
- HealthOmics 실행 그룹 사용
- HealthOmics 실행을 위한 호출 캐싱
- HealthOmics 워크플로 공유

HealthOmics에서 프라이빗 워크플로 생성

프라이빗 워크플로는 워크플로를 생성하기 전에 생성하고 구성하는 다양한 리소스에 따라 달라집니다.

- Workflow definition file: WDL, Nextflow또는 로 작성된 워크플로 정의 파일입니다CWL. 워크플로 정의는 워크플로를 사용하는 실행에 대한 입력 및 출력을 지정합니다. 또한 컴퓨팅 및 메모리 요구 사항을 포함하여 워크플로의 실행 및 실행 작업에 대한 사양도 포함되어 있습니다. 워크플로 정의 파일은 `.zip` 형식이어야 합니다. 자세한 내용은 워크플로 정의 파일을 참조하세요.
  - Amazon Q CLI를 사용하여 WDL, Nextflow 및 CWL에서 워크플로 정의 파일을 빌드하고 검증할 수 있습니다. 자세한 내용은 Amazon Q CLI에 대한 프롬프트 예제 및 GitHub의 HealthOmics Agentic 생성형 AI 자습서를 참조하세요.
- (Optional) Parameter template file:에 작성된 파라미터 템플릿 파일입니다JSON. 파일을 생성하여 실행 파라미터를 정의하거나 HealthOmics에서 파라미터 템플릿을 생성합니다. 자세한 내용은 HealthOmics 워크플로용 파라미터 템플릿 파일을 참조하세요.
- Amazon ECR container images: 워크플로에 대한 프라이빗 Amazon ECR 리포지토리를 생성합니다. 프라이빗 리포지토리에서 컨테이너 이미지를 생성하거나 지원되는 업스트림 레지스트리의 콘텐츠를 Amazon ECR 프라이빗 리포지토리와 동기화합니다.
- (Optional) Sentieon licenses: 프라이빗 워크플로에서 Sentieon 소프트웨어를 사용할 수 있는 Sentieon 라이선스를 요청합니다.

선택적으로 워크플로를 생성하기 전이나 후에 워크플로 정의에서 린터를 실행할 수 있습니다. 이 linter 주제에서는 HealthOmics에서 사용할 수 있는 린터에 대해 설명합니다.

주제

- Git 기반 리포지토리와 HealthOmics 워크플로 통합
- HealthOmics의 워크플로 정의 파일
- HealthOmics 워크플로용 파라미터 템플릿 파일
- 프라이빗 워크플로용 컨테이너 이미지
- HealthOmics 워크플로 README 파일
- 프라이빗 워크플로에 대한 Sentieon 라이선스 요청
- HealthOmics의 워크플로 린터
- HealthOmics 워크플로 작업

Git 기반 리포지토리와 HealthOmics 워크플로 통합

워크플로(또는 워크플로 버전)를 생성할 때 워크플로, 실행 및 작업에 대한 정보를 지정하는 워크플로 정의를 제공합니다. HealthOmics는 워크플로 정의를 .zip 아카이브(로컬 또는 Amazon S3 버킷에 저장됨) 또는 지원되는 Git 기반 리포지토리에서 검색할 수 있습니다.

Git 기반 리포지토리와 HealthOmics 통합을 통해 다음 기능을 사용할 수 있습니다.

- 퍼블릭, 프라이빗 및 자체 관리형 인스턴스에서 직접 워크플로를 생성합니다.
- 리포지토리의 워크플로 README 파일 및 파라미터 템플릿 통합.
- GitHub, GitLab 및 Bitbucket 리포지토리를 지원합니다.

Git 기반 리포지토리를 사용하면 워크플로 정의 파일 및 입력 파라미터 템플릿 파일을 다운로드하고.zip 아카이브를 생성한 다음 아카이브를 S3에 스테이징하는 수동 단계를 피할 수 있습니다. 이렇게 하면 다음 예제와 같은 시나리오의 워크플로 생성이 간소화됩니다.

1. nf-core와 같은 일반적인 오픈 소스 워크플로를 사용하여 빠르게 시작하려고 합니다. HealthOmics는 GitHub의 nf-core 리포지토리에서 모든 워크플로 정의 및 입력 파라미터 템플릿 파일을 자동으로 검색하고 이러한 파일을 사용하여 새 워크플로를 생성합니다.
2. GitHub의 퍼블릭 워크플로를 사용 중이며 일부 새 업데이트를 사용할 수 있게 됩니다. GitHub의 업데이트된 워크플로 정의를 소스로 사용하여 새 HealthOmics 워크플로 버전을 쉽게 생성할 수 있습니다. 워크플로 사용자는 원래 워크플로 또는 생성한 새 워크플로 버전 중에서 선택할 수 있습니다.
3. 팀이 공개되지 않은 독점 파이프라인을 구축하고 있습니다. 코드를 프라이빗 git 리포지토리에 보관하고이 워크플로 정의를 HealthOmics 워크플로에 사용합니다. 팀은 반복 워크플로 개발 수명 주기의 일부로 워크플로 정의를 자주 업데이트합니다. 프라이빗 리포지토리에서 필요에 따라 새 워크플로 버전을 쉽게 생성할 수 있습니다.

주제

지원되는 Git 기반 리포지토리

HealthOmics는 다음 Git 기반 공급자에 대한 퍼블릭 및 프라이빗 리포지토리를 지원합니다.

- GitHub
- GitLab
- Bitbucket

HealthOmics는 다음 Git 기반 공급자에 대해 자체 관리형 리포지토리를 지원합니다.

- GitHubEnterpriseServer
- GitLabSelfManaged

HealthOmics는 GitHub, GitLab 및 Bitbucket에 대한 교차 계정 연결 사용을 지원합니다. AWS Resource Access Manager를 통해 공유 권한을 설정합니다. 예제는 CodePipeline 사용 설명서의 공유 연결을 참조하세요.

외부 코드 리포지토리에 대한 연결 구성

AWS CodeConnection을 사용하여 워크플로를 Git 기반 리포지토리에 연결합니다. HealthOmics는이 연결을 사용하여 소스 코드 리포지토리에 액세스합니다.

> **Note**
>
> il-central-1 리전에서는 AWS CodeConnections 서비스를 사용할 수 없습니다. 이 리전의 경우 리포지토리에서 워크플로 또는 워크플로 버전을 생성하도록 서비스 us-east-1을 구성합니다.

연결 생성

연결을 생성하려면 먼저 개발자 콘솔 도구 사용 설명서의 연결 설정의 지침을 따르세요.

연결을 생성하려면 개발자 콘솔 도구 사용 설명서의 연결 생성의 지침을 따르세요.

연결에 대한 권한 부여 구성

공급자의 OAuth 흐름을 사용하여 연결을 승인해야 합니다. 연결 상태를 사용하기 `AVAILABLE` 전에 상태가 인지 확인합니다.

예제는 블로그 게시물 How To Create a AWS HealthOmics Workflows from Content in Git를 참조하세요.

자체 관리형 리포지토리 액세스

GitLab 자체 관리형 리포지토리에 대한 연결을 설정하려면 호스트를 생성할 때 관리자 개인 액세스 토큰을 사용합니다. 후속 연결 생성은 고객 계정으로 Oauth에 액세스합니다.

다음 예시에서는 GitLab 자체 관리형 리포지토리에 대한 연결을 설정합니다.

1. 관리자 사용자의 개인 액세스 토큰에 대한 액세스를 설정합니다.

   GitLab 자체 관리형 리포지토리에서 PAT를 설정하려면 GitLab Docs의 개인 액세스 토큰을 참조하세요. GitLab
2. 호스트 생성

   a. CodePipeline>설정>연결로 이동합니다.

   b. 호스트 탭을 선택한 다음 호스트 생성을 선택합니다.

   c. 다음 필드를 구성합니다.

      - 호스트의 이름을 입력합니다.
      - 공급자 유형에서 GitLab 자체 관리형을 선택합니다.
      - 호스트 URL을 입력합니다.
      - 호스트가 VPC에 정의된 경우 VPC 정보를 입력합니다.

   d. 호스트 생성을 선택하면 호스트가 PENDING 상태로 생성됩니다.

   e. 설정을 완료하려면 호스트 설정을 선택합니다.

   f. 관리자 사용자의 개인 액세스 토큰(PAT)을 입력한 다음 계속을 선택합니다.
3. 연결 생성

   a. 연결 탭에서 연결 생성을 선택합니다.

   b. 공급자 유형에서 GitLab 자체 관리형을 선택합니다.

   c. 연결 설정>연결 이름 입력에서 이전에 생성한 호스트 URL을 입력합니다.

   d. GitLab 자체 관리형 인스턴스에 VPC를 통해서만 액세스할 수 있는 경우 VPC 세부 정보를 구성합니다.

   e. 대기 중인 연결 업데이트를 선택합니다. 모달 창은 GitLab 로그인 페이지로 리디렉션합니다.

   f. 고객 계정의 사용자 이름과 암호를 입력하고 권한 부여 프로세스를 완료합니다.

g. 처음 설정하려면 Authorize AWS Connector for Gitlab Self Managed를 선택합니다.

외부 코드 리포지토리와 관련된 할당량

외부 코드 리포지토리와 HealthOmics 통합의 경우 리포지토리, 각 리포지토리 파일 및 각 README 파일의 최대 크기가 있습니다. 자세한 내용은 HealthOmics 워크플로 고정 크기 할당량을 참조하세요.

필수 IAM 권한

자격 증명 기반 IAM 정책에 다음 작업을 추가합니다.

```
"codeconnections:CreateConnection",
"codeconnections:GetConnection",
"codeconnections:GetHost",
"codeconnections:ListConnections",
"codeconnections:UseConnection"
```

HealthOmics의 워크플로 정의 파일

워크플로 정의를 사용하여 워크플로, 실행 및 실행의 작업에 대한 정보를 지정합니다. 워크플로 정의 언어를 사용하여 하나 이상의 파일에 워크플로 정의를 생성합니다. HealthOmics는 WDL, Nextflow 또는 CWL로 작성된 워크플로 정의를 지원합니다.

HealthOmics는 WDL 워크플로 정의에 대해 다음과 같은 선택을 지원합니다.

- WDL - 사양을 준수하는 WDL 엔진을 제공합니다.
- WDL lenient - Cromwell에서 마이그레이션된 워크플로를 처리하도록 설계되었습니다. 고객 Cromwell 지시문과 일부 규정 미준수 로직을 지원합니다. 자세한 내용은 WDL lenient의 암시적 유형 변환을 참조하세요.

각 워크플로 언어에 대한 자세한 내용은 아래의 언어별 세부 섹션을 참조하세요.

워크플로 정의에서 다음과 같은 유형의 정보를 지정합니다.

- Language version - 워크플로 정의의 언어 및 버전입니다.
- Compute and memory - 워크플로의 작업에 대한 컴퓨팅 및 메모리 요구 사항입니다.
- Inputs - 워크플로 작업에 대한 입력의 위치입니다. 자세한 내용은 HealthOmics 실행 입력 단원을 참조하십시오.
- Outputs - 작업이 생성하는 출력을 저장할 위치입니다.

- Task resources - 각 작업에 대한 컴퓨팅 및 메모리 요구 사항입니다.
- Accelerators - 액셀러레이터와 같이 작업에 필요한 기타 리소스.

주제

HealthOmics 워크플로 정의 요구 사항

HealthOmics 워크플로 정의 파일은 다음 요구 사항을 충족해야 합니다.

- 작업은 입력/출력 파라미터, Amazon ECR 컨테이너 리포지토리, 메모리 또는 CPU 할당과 같은 런타임 사양을 정의해야 합니다.
- IAM 역할에 필요한 권한이 있는지 확인합니다.
  - 워크플로는 Amazon S3와 같은 AWS 리소스의 입력 데이터에 액세스할 수 있습니다.
  - 워크플로는 필요한 경우 외부 리포지토리 서비스에 액세스할 수 있습니다.
- 워크플로 정의에서 출력 파일을 선언합니다. 중간 실행 파일을 출력 위치에 복사하려면 이를 워크플로 출력으로 선언합니다.
- 입력 및 출력 위치는 워크플로와 동일한 리전에 있어야 합니다.
- HealthOmics 스토리지 워크플로 입력은 `ACTIVE` 상태여야 합니다. HealthOmics는 `ARCHIVED` 상태의 입력을 가져오지 않으므로 워크플로가 실패합니다. Amazon S3 객체 입력에 대한 자세한 내용은 섹션을 참조하세요HealthOmics 실행 입력.
- ZIP 아카이브에 단일 워크플로 정의 또는 'main'이라는 파일이 포함된 경우 워크플로의 main 위치는 선택 사항입니다.
  - 예시 경로: `workflow-definition/main-file.wdl`

- Amazon S3 또는 로컬 드라이브에서 워크플로를 생성하기 전에 워크플로 정의 파일과 하위 워크플로와 같은 종속성의 zip 아카이브를 생성합니다.
- 워크플로에서 Amazon ECR 컨테이너를 Amazon ECR 권한 검증을 위한 입력 파라미터로 선언하는 것이 좋습니다.

추가 Nextflow 고려 사항:

- /bin

  Nextflow 워크플로 정의에는 실행 가능한 스크립트가 있는 /bin 폴더가 포함될 수 있습니다. 이 경로에는 작업에 대한 읽기 전용 및 실행 가능한 액세스 권한이 있습니다. 이러한 스크립트에 의존하는 작업은 적절한 스크립트 인터프리터로 빌드된 컨테이너를 사용해야 합니다. 가장 좋은 방법은 인터프리터를 직접 호출하는 것입니다. 예제:

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

- includeConfig

  Nextflow 기반 워크플로 정의에는 파라미터 정의를 추상화하거나 리소스 프로파일을 처리하는 데 도움이 되는 nextflow.config 파일이 포함될 수 있습니다. 여러 환경에서 Nextflow 파이프라인의 개발 및 실행을 지원하려면 includeConfig 지시문을 사용하여 글로벌 구성에 추가하는 HealthOmics별 구성을 사용합니다. 이식성을 유지하려면 다음 코드를 사용하여 HealthOmics에서 실행할 때만 파일을 포함하도록 워크플로를 구성합니다.

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

- Reports

  HealthOmics는 엔진 생성 Dag, 추적 및 실행 보고서를 지원하지 않습니다. GetRun 및 GetRunTask API 호출을 조합하여 추적 및 실행 보고서에 대한 대안을 생성할 수 있습니다.

추가 CWL 고려 사항:

- Container image uri interpolation

  HealthOmics를 사용하면 DockerRequirement의 dockerPull 속성이 인라인 Javascript 표현식이 될 수 있습니다. 예제:

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

  이렇게 하면 컨테이너 이미지 URIs 워크플로의 입력 파라미터로 지정할 수 있습니다.

- Javascript expressions

  Javascript 표현식은 `strict mode` 규정을 준수해야 합니다.

- Operation process

  HealthOmics는 CWL 작업 프로세스를 지원하지 않습니다.

HealthOmics 워크플로 정의 언어에 대한 버전 지원

HealthOmics는 Nextflow, WDL 또는 CWL로 작성된 워크플로 정의 파일을 지원합니다. 다음 섹션에서는 이러한 언어에 대한 HealthOmics 버전 지원에 대한 정보를 제공합니다.

주제

- WDL 버전 지원
- CWL 버전 지원
- Nextflow 버전 지원

WDL 버전 지원

HealthOmics는 버전 1.0, 1.1 및 WDL 사양의 개발 버전을 지원합니다.

모든 WDL 문서에는 준수하는 사양의 버전(주요 및 부)을 지정하는 버전 문이 포함되어야 합니다. 버전에 대한 자세한 내용은 WDL 버전 관리를 참조하세요.

WDL 사양 버전 1.0 및 1.1은 `Directory` 유형을 지원하지 않습니다. 입력 또는 출력에 `Directory` 유형을 사용하려면 파일의 첫 번째 줄development에서 버전을 로 설정합니다.

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

CWL 버전 지원

HealthOmics는 CWL 언어 버전 1.0, 1.1 및 1.2를 지원합니다.

CWL 워크플로 정의 파일에서 언어 버전을 지정할 수 있습니다. CWL에 대한 자세한 내용은 CWL 사용 설명서를 참조하세요.

Nextflow 버전 지원

HealthOmics는 세 가지 Nextflow 안정 버전을 지원합니다. Nextflow는 일반적으로 6개월마다 안정적인 버전을 릴리스합니다. HealthOmics는 월별 '엣지' 릴리스를 지원하지 않습니다.

HealthOmics는 각 버전에서 릴리스된 기능을 지원하지만 미리 보기 기능은 지원하지 않습니다.

지원되는 버전

HealthOmics는 다음 Nextflow 버전을 지원합니다.

- Nextflow v22.04.01 DSL 1 및 DSL 2
- Nextflow v23.10.0 DSL 2(기본값)
- Nextflow v24.10.8 DSL 2

워크플로를 지원되는 최신 버전(v24.10.8)으로 마이그레이션하려면 Nextflow 업그레이드 가이드를 따르세요.

Nextflow 마이그레이션 가이드의 다음 섹션에 설명된 대로 Nextflow v23에서 v24로 마이그레이션할 때 몇 가지 주요 변경 사항이 있습니다.

- 24.04의 주요 변경 사항
- 24.10의 주요 변경 사항

Nextflow 버전 감지 및 처리

HealthOmics는 지정한 DSL 버전과 Nextflow 버전을 감지합니다. 이러한 입력을 기반으로 실행할 최적의 Nextflow 버전을 자동으로 결정합니다.

DSL 버전

HealthOmics는 워크플로 정의 파일에서 요청된 DSL 버전을 감지합니다. 예를 들어를 지정할 수 있습니다`nextflow.enable.dsl=2`.

HealthOmics는 기본적으로 DSL 2를 지원합니다. 워크플로 정의 파일에 지정된 경우 DSL 1과의 이전 버전과의 호환성을 제공합니다.

- DSL 2를 지정하면 Nextflow v22.04.0 또는 v24.10.8를 지정하지 않는 한 HealthOmics는 Nextflow v23.10.0을 실행합니다.
- DSL 1을 지정하면 HealthOmics는 Nextflow v22.04 DSL1(DSL 1을 실행하는 지원되는 유일한 버전)을 실행합니다.
- DSL 버전을 지정하지 않거나 HealthOmics가 어떤 이유로든 DSL 정보를 구문 분석할 수 없는 경우(예: 워크플로 정의 파일의 구문 오류) HealthOmics는 기본적으로 DSL 2로 설정되고 Nextflow v23.10.0을 실행합니다.
- 최신 Nextflow 버전 및 소프트웨어 기능을 활용하도록 워크플로를 DSL 1에서 DSL 2로 업그레이드하려면 DSL 1에서 마이그레이션을 참조하세요.

Nextflow 버전

HealthOmics는이 파일을 제공하는 경우 Nextflow 구성 파일(nextflow.config)에서 요청된 Nextflow 버전을 감지합니다. 포함된 구성으로 인한 예기치 않은 재정의를 방지하려면 파일 끝에 `nextflowVersion` 절을 추가하는 것이 좋습니다. 자세한 내용은 Nextflow 구성을 참조하세요.

다음 구문을 사용하여 Nextflow 버전 또는 버전 범위를 지정할 수 있습니다.

```
// 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'
```

HealthOmics는 Nextflow 버전 정보를 다음과 같이 처리합니다.

- =를 사용하여 HealthOmics가 지원하는 정확한 버전을 지정하는 경우 HealthOmics는 해당 버전을 사용합니다.
- !를 사용하여 지원되지 않는 정확한 버전 또는 버전 범위를 지정하는 경우 HealthOmics는 예외를 발생시키고 실행에 실패합니다. 버전 요청을 엄격하게 적용하고 요청에 지원되지 않는 버전이 포함된 경우 빠르게 실패하려면이 옵션을 사용하는 것이 좋습니다.
- 버전 범위를 지정하면 범위에 v24.10.8가 포함되지 않는 한 HealthOmics는 해당 범위에서 지원되는 최신 버전을 사용합니다. 이 경우 HealthOmics는 이전 버전을 선호합니다. 예를 들어 범위가 v23.10.0과 v24.10.8를 모두 포함하는 경우 HealthOmics는 v23.10.0을 선택합니다.
- 요청된 버전이 없거나 요청된 버전이 유효하지 않거나 어떤 이유로든 구문 분석할 수 없는 경우:
  - DSL 1을 지정한 경우 HealthOmics는 Nextflow v22.04를 실행합니다.
  - 그렇지 않으면 HealthOmics는 Nextflow v23.10.0을 실행합니다.

각 실행에 HealthOmics가 사용한 Nextflow 버전에 대한 다음 정보를 검색할 수 있습니다.

- 실행 로그에는 HealthOmics가 실행에 사용한 실제 Nextflow 버전에 대한 정보가 포함되어 있습니다.
- HealthOmics는 요청된 버전과 직접 일치하지 않거나 지정한 버전과 다른 버전을 사용해야 하는 경우 실행 로그에 경고를 추가합니다.
- GetRun API 작업에 대한 응답에는 HealthOmics가 실행에 사용한 실제 Nextflow 버전이 포함된 필드(`engineVersion`)가 포함됩니다. 예제:

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

HealthOmics 작업에 대한 컴퓨팅 및 메모리 요구 사항

HealthOmics는 omics 인스턴스에서 프라이빗 워크플로 작업을 실행합니다. HealthOmics는 다양한 유형의 작업을 수용할 수 있는 다양한 인스턴스 유형을 제공합니다. 각 인스턴스 유형에는 고정 메모리 및 vCPU 구성(및 가속 컴퓨팅 인스턴스 유형에 대한 고정 GPU 구성)이 있습니다. omics 인스턴스 사용 비용은 인스턴스 유형에 따라 다릅니다. 자세한 내용은 HealthOmics 요금 페이지를 참조하세요.

워크플로의 태스크의 경우 워크플로 정의 파일에서 필요한 메모리와 vCPUs를 지정합니다. 워크플로 작업이 실행되면 HealthOmics는 요청된 메모리 및 vCPUs. 예를 들어 작업에 64GiB의 메모리와 8개의 vCPUs 필요한 경우 HealthOmics는를 선택합니다`omics.r.2xlarge`.

인스턴스 유형을 검토하고 요구 사항에 가장 적합한 인스턴스와 일치하도록 요청된 vCPUs 및 메모리 크기를 설정하는 것이 좋습니다. 작업 컨테이너는 인스턴스 유형에 추가 vCPUs 및 메모리가 있더라도 워크플로 정의 파일에 지정한 vCPUs 수와 메모리 크기를 사용합니다.

다음 목록에는 vCPU 및 메모리 할당에 대한 추가 정보가 포함되어 있습니다.

- 컨테이너 리소스 할당은 하드 제한입니다. 작업의 메모리가 부족하거나 추가 vCPUs를 사용하려고 하면 작업에서 오류 로그가 생성되고 종료됩니다.
- 컴퓨팅 또는 메모리 요구 사항을 지정하지 않으면 HealthOmics는 vCPU 1omics.c.large개와 메모리 1GiB가 있는 구성을 선택하고 기본값으로 설정합니다.
- 요청할 수 있는 최소 구성은 vCPU 1개와 메모리 1GiB입니다.
- 지원되는 인스턴스 유형을 초과하는 vCPUs, 메모리 또는 GPUs를 지정하면 HealthOmics에서 오류 메시지가 발생하고 워크플로가 검증에 실패합니다.
- 분수 단위를 지정하면 HealthOmics는 가장 가까운 정수로 반올림합니다.
- HealthOmics는 관리 및 로깅 에이전트를 위해 소량의 메모리(5%)를 예약하므로 작업의 애플리케이션에서 전체 메모리 할당을 항상 사용할 수 있는 것은 아닙니다.
- HealthOmics는 지정한 컴퓨팅 및 메모리 요구 사항에 맞게 인스턴스 유형과 일치하며 하드웨어 생성을 혼합하여 사용할 수 있습니다. 따라서 동일한 작업에 대한 작업 실행 시간에 약간의 차이가 있을 수 있습니다.

이 주제에서는 HealthOmics가 지원하는 인스턴스 유형에 대한 세부 정보를 제공합니다.

주제

> **Note**
>
> 표준, 컴퓨팅 및 메모리 최적화 인스턴스의 경우 인스턴스에 더 높은 처리량이 필요한 경우 인스턴스 대역폭 크기를 늘립니다. vCPUs(크기 4xl 이하)인 Amazon EC2 인스턴스는 처리량 버스팅이 발생할 수 있습니다. Amazon EC2 인스턴스 처리량에 대한 자세한 내용은 Amazon EC2 사용 가능한 인스턴스 대역폭을 참조하세요.

표준 인스턴스 유형

표준 인스턴스 유형의 경우 구성은 컴퓨팅 성능과 메모리의 균형을 목표로 합니다.

HealthOmics는 미국 서부(오레곤) 및 미국 동부(버지니아 북부) 리전에서 32xlarge 및 48xlarge 인스턴스를 지원합니다.

Instance	vCPUs 수	Memory
omics.m.large	2	8GiB
omics.m.xlarge	4	16GiB
omics.m.2xlarge	8	32GiB
omics.m.4xlarge	16	64GiB
omics.m.8xlarge	32	128GiB
omics.m.12xlarge	48	192GiB
omics.m.16xlarge	64	256GiB
omics.m.24xlarge	96	384 GiB
omics.m.32xlarge	128	512GiB
omics.m.48xlarge	192	768GiB

컴퓨팅 최적화 인스턴스

컴퓨팅 최적화 인스턴스 유형의 경우 구성의 컴퓨팅 성능과 메모리가 더 적습니다.

HealthOmics는 미국 서부(오레곤) 및 미국 동부(버지니아 북부) 리전에서 32xlarge 및 48xlarge 인스턴스를 지원합니다.

Instance	vCPUs 수	Memory
omics.c.large	2	4GiB
omics.c.xlarge	4	8GiB

Instance	vCPUs 수	Memory
omics.c.2xlarge	8	16GiB
omics.c.4xlarge	16	32GiB
omics.c.8xlarge	32	64GiB
omics.c.12xlarge	48	96GiB
omics.c.16xlarge	64	128GiB
omics.c.24xlarge	96	192GiB
omics.c.32xlarge	128	256GiB
omics.c.48xlarge	192	384 GiB

메모리 최적화 인스턴스

메모리 최적화 인스턴스 유형의 경우 구성의 컴퓨팅 성능과 메모리가 더 적습니다.

HealthOmics는 미국 서부(오레곤) 및 미국 동부(버지니아 북부) 리전에서 32xlarge 및 48xlarge 인스턴스를 지원합니다.

Instance	vCPUs 수	Memory
omics.r.large	2	16GiB
omics.r.xlarge	4	32GiB
omics.r.2xlarge	8	64GiB
omics.r.4xlarge	16	128GiB
omics.r.8xlarge	32	256GiB
omics.r.12xlarge	48	384 GiB
omics.r.16xlarge	64	512GiB

Instance	vCPUs 수	Memory
omics.r.24xlarge	96	768GiB
omics.r.32xlarge	128	1024GiB
omics.r.48xlarge	192	1536GiB

가속 컴퓨팅 인스턴스

HealthOmics가 작업에 가속 컴퓨팅 인스턴스를 할당하도록 워크플로의 각 작업에 대해 GPU 리소스를 선택적으로 지정할 수 있습니다. 워크플로 정의 파일에서 GPU 정보를 지정하는 방법에 대한 자세한 내용은 섹션을 참조하세요HealthOmics 워크플로 정의의 태스크 액셀러레이터.

여러 인스턴스 유형을 지원하는 태스크 액셀러레이터를 지정하는 경우 HealthOmics는 가용성에 따라 인스턴스 유형을 선택합니다. 두 개 이상의 인스턴스 유형을 사용할 수 있는 경우 HealthOmics는 저렴한 인스턴스를 선호합니다. 리전에서 사용 가능한 최신 세대 인스턴스를 기본 설정하는 nvidia-t4-a10g-l4 태스크 액셀러레이터는 예외입니다.

이스라엘(텔아비브) 리전에서는 G4 인스턴스가 지원되지 않습니다. 아시아 태평양(싱가포르) 리전에서는 G5 인스턴스가 지원되지 않습니다.

주제

- G6 및 G6e 인스턴스 유형
- G4 및 G5 인스턴스

G6 및 G6e 인스턴스 유형

HealthOmics는 다음과 같은 G6 가속 컴퓨팅 인스턴스 구성을 지원합니다. 모든 omics.g6 인스턴스는 Nvidia L4 GPUs 사용합니다.

HealthOmics는 미국 서부(오레곤) 및 미국 동부(버지니아 북부) 리전에서 G6 및 G6e 인스턴스를 지원합니다.

Instance	vCPUs 수	Memory	GPUs 수	GPU 메모리	
omics.g6.xlarge	4	16GiB	1	24GiB	

Instance	vCPUs 수	Memory	GPUs 수	GPU 메모리
omics.g6.2xlarge	8	32GiB	1	24GiB
omics.g6.4xlarge	16	64GiB	1	24GiB
omics.g6.8xlarge	32	128GiB	1	24GiB
omics.g6.12xlarge	48	192GiB	4	96GiB
omics.g6.16xlarge	64	256GiB	1	24GiB
omics.g6.24xlarge	96	384 GiB	4	96GiB

모든 omics.g6e 인스턴스는 Nvidia L40s GPUs 사용합니다.

Instance	vCPUs 수	Memory	GPUs 수	GPU 메모리
omics.g6e.xlarge	4	32GiB	1	48GiB
omics.g6e.2xlarge	8	64GiB	1	48GiB
omics.g6e.4xlarge	16	128GiB	1	48GiB
omics.g6e.8xlarge	32	256GiB	1	48GiB
omics.g6e.12xlarge	48	384 GiB	4	192GiB

Instance	vCPUs 수	Memory	GPUs 수	GPU 메모리
omics.g6e .16xlarge	64	512GiB	1	48GiB
omics.g6e .24xlarge	96	768GiB	4	192GiB

G4 및 G5 인스턴스

HealthOmics는 다음과 같은 G4 및 G5 가속 컴퓨팅 인스턴스 구성을 지원합니다.

모든 omics.g5 인스턴스는 Nvidia Tesla A10G GPUs 사용합니다.

Instance	vCPUs 수	Memory	GPUs 수	GPU 메모리
omics.g5. xlarge	4	16GiB	1	24GiB
omics.g5. 2xlarge	8	32GiB	1	24GiB
omics.g5. 4xlarge	16	64GiB	1	24GiB
omics.g5. 8xlarge	32	128GiB	1	24GiB
omics.g5. 12xlarge	48	192GiB	4	96GiB
omics.g5. 16xlarge	64	256GiB	1	24GiB
omics.g5. 24xlarge	96	384 GiB	4	96GiB

모든 omics.g4dn 인스턴스는 Nvidia Tesla T4 GPUs 사용합니다.

Instance	vCPUs 수	Memory	GPUs 수	GPU 메모리	
omics.g4dn.xlarge	4	16GiB	1	16GiB	
omics.g4dn.2xlarge	8	32GiB	1	16GiB	
omics.g4dn.4xlarge	16	64GiB	1	16GiB	
omics.g4dn.8xlarge	32	128GiB	1	16GiB	
omics.g4dn.12xlarge	48	192GiB	4	64GiB	
omics.g4dn.16xlarge	64	256GiB	1	24GiB	

HealthOmics 워크플로 정의의 작업 출력

워크플로 정의에서 작업 출력을 지정합니다. 기본적으로 HealthOmics는 워크플로가 완료되면 모든 중간 작업 파일을 삭제합니다. 중간 파일을 내보내려면 이를 출력으로 정의합니다.

호출 캐싱을 사용하는 경우 HealthOmics는 출력으로 정의한 중간 파일을 포함하여 작업 출력을 캐시에 저장합니다.

다음 주제에는 각 워크플로 정의 언어에 대한 작업 정의 예제가 포함되어 있습니다.

주제

- WDL에 대한 작업 출력
- Nextflow에 대한 작업 출력
- CWL에 대한 작업 출력

WDL에 대한 작업 출력

WDL로 작성된 워크플로 정의의 경우 최상위 워크플로 outputs 섹션에서 출력을 정의합니다.

HealthOmics

주제

- STDOUT에 대한 작업 출력
- STDERR에 대한 작업 출력
- 파일에 대한 작업 출력
- 파일 배열에 대한 작업 출력

STDOUT에 대한 작업 출력

이 예제에서는 STDOUT 콘텐츠를 작업 출력 파일에 에코SayHello하는 라는 작업을 생성합니다. WDL stdout 함수는 파일에서 STDOUT 콘텐츠(이 예제에서는 입력 문자열 Hello World!)를 캡처합니다stdout_file.

HealthOmics는 모든 STDOUT 콘텐츠에 대한 로그를 생성하므로 출력은 작업에 대한 다른 STDERR 로깅 정보와 함께 CloudWatch Logs에도 표시됩니다.

```
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()
    }
}
```

STDERR에 대한 작업 출력

이 예제에서는 STDERR 콘텐츠를 작업 출력 파일에 에코SayHello하는 라는 작업을 생성합니다. WDL stderr 함수는 파일에서 STDERR 콘텐츠(이 예에서는 입력 문자열 Hello World!)를 캡처합니다stderr_file.

HealthOmics는 모든 STDERR 콘텐츠에 대한 로그를 생성하므로 출력은 작업에 대한 다른 STDERR 로깅 정보와 함께 CloudWatch Logs에 표시됩니다.

```
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()
    }
}
```

파일에 대한 작업 출력

이 예제에서 SayHello 작업은 두 개의 파일(message.txt 및 info.txt)을 생성하고 이러한 파일을 명명된 출력(message_file 및 info_file)으로 명시적으로 선언합니다.

```
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"
    }
}
```

파일 배열에 대한 작업 출력

이 예제에서 GenerateGreetings 작업은 파일 배열을 작업 출력으로 생성합니다. 작업은 입력 배열의 각 멤버에 대해 하나의 인사말 파일을 동적으로 생성합니다names. 런타임까지 파일 이름을 알 수 없으므로 출력 정의는 WDL glob() 함수를 사용하여 패턴과 일치하는 모든 파일을 출력합니다*_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")
    }
 }
```

Nextflow에 대한 작업 출력

Nextflow에 작성된 워크플로 정의의 경우 publishDir 명령을 정의하여 작업 콘텐츠를 출력 Amazon S3 버킷으로 내보냅니다. publishDir 값을 로 설정합니다`/mnt/workflow/pubdir`.

HealthOmics가 Amazon S3로 파일을 내보내려면 파일이이 디렉터리에 있어야 합니다.

작업이 후속 작업에 대한 입력으로 사용할 출력 파일 그룹을 생성하는 경우 이러한 파일을 디렉터리에 그룹화하고 디렉터리를 작업 출력으로 내보내는 것이 좋습니다. 각 개별 파일을 열거하면 기본 파일 시스템에서 I/O 병목 현상이 발생할 수 있습니다. 예:

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

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

CWL에 대한 작업 출력

CWL로 작성된 워크플로 정의의 경우 작업을 사용하여 `CommandLineTool` 작업 출력을 지정할 수 있습니다. 다음 섹션에서는 다양한 유형의 출력을 정의하는 `CommandLineTool` 작업의 예를 보여줍니다.

주제

- STDOUT에 대한 작업 출력
- STDERR에 대한 작업 출력
- 파일에 대한 작업 출력
- 파일 배열에 대한 작업 출력

STDOUT에 대한 작업 출력

이 예제에서는 STDOUT 콘텐츠를 라는 텍스트 출력 파일로 에코하는 `CommandLineTool` 작업을 생성합니다output.txt. 예를 들어 다음 입력을 입력하면 결과 작업 출력은 output.txt 파일의 Hello World!입니다.

```
{
```

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

outputs 명령은 출력 이름이 example_out 이고 유형이 임을 지정합니다stdout. 다운스트림 작업이 이 작업의 출력을 소비하려면 출력을 라고 합니다example_out.

HealthOmics는 모든 STDERR 및 STDOUT 콘텐츠에 대한 로그를 생성하므로 출력은 작업에 대한 다른 STDERR 로깅 정보와 함께 CloudWatch Logs에도 표시됩니다.

```
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
```

STDERR에 대한 작업 출력

이 예제에서는 STDERR 콘텐츠를 라는 텍스트 출력 파일로 에코하는 CommandLineTool 작업을 생성합니다stderr.txt. 작업은가 (STDOUT 대신) STDERR에 echo 쓰baseCommand도록를 수정합니다.

outputs 명령은 출력 이름이 stderr_out 이고 유형이 임을 지정합니다stderr.

HealthOmics는 모든 STDERR 및 STDOUT 콘텐츠에 대한 로그를 생성하므로 작업에 대한 다른 STDERR 로깅 정보와 함께 출력이 CloudWatch Logs에 표시됩니다.

```
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
```

파일에 대한 작업 출력

이 예제에서는 입력 파일에서 압축된 tar 아카이브를 생성하는 CommandLineTool 작업을 생성합니다. 아카이브의 이름을 입력 파라미터(archive_name)로 제공합니다.

outputs 명령은 archive_file 출력 유형이 임을 지정File하고 입력 파라미터에 대한 참조를 사용하여 출력 파일에 archive_name 바인딩합니다.

```
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
```

파일 배열에 대한 작업 출력

이 예제에서 CommandLineTool 작업은 touch 명령을 사용하여 파일 배열을 생성합니다. 명령은 files-to-create 입력 파라미터의 문자열을 사용하여 파일의 이름을 지정합니다. 명령은 파일 배열을 출력합니다. 배열에는 작업 디렉터리에서 glob 패턴과 일치하는 모든 파일이 포함됩니다. 이 예제에서는 모든 파일과 일치하는 와일드카드 패턴("*")을 사용합니다.

```
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 워크플로 정의의 태스크 리소스

워크플로 정의에서 각 작업에 대해 다음을 정의합니다.

- 작업의 컨테이너 이미지입니다. 자세한 내용은 프라이빗 워크플로용 컨테이너 이미지 단원을 참조하십시오.
- 작업에 필요한 CPUs 및 메모리 수입니다. 자세한 내용은 HealthOmics 작업에 대한 컴퓨팅 및 메모리 요구 사항 단원을 참조하십시오.

HealthOmics는 작업별 스토리지 사양을 무시합니다. HealthOmics는 실행 중인 모든 작업이 액세스할 수 있는 실행 스토리지를 제공합니다. 자세한 내용은 HealthOmics 워크플로에서 스토리지 유형 실행 단원을 참조하십시오.

WDL

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

WDL 워크플로의 경우 HealthOmics는 서비스 오류로 인해 실패한 작업에 대해 최대 2회의 재시도를 시도합니다(API 요청은 5XX HTTP 상태 코드를 반환함). 작업 재시도에 대한 자세한 내용은 섹션을 참조하세요작업 재시도.

WDL 정의 파일에서 작업에 대해 다음 구성을 지정하여 재시도 동작을 옵트아웃할 수 있습니다.

```
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 워크플로 정의의 태스크 액셀러레이터

워크플로 정의에서 선택적으로 작업에 대한 GPU 액셀러레이터 사양을 지정할 수 있습니다. HealthOmics는 지원되는 인스턴스 유형과 함께 다음 액셀러레이터 사양 값을 지원합니다.

액셀러레이터 사양	Healthomics 인스턴스 유형				
nvidia-tesla-t4	G4				
nvidia-tesla-t4-a10g	G4 및 G5				
nvidia-tesla-a10g	G5				
nvidia-t4-a10g-l4	G4, G5 및 G6				

액셀러레이터 사양	Healthomics 인스턴스 유형				
nvidia-l4-a10g	G5 및 G6				
nvidia-l4	G6				
nvidia-l40s	G6e				

여러 인스턴스 유형을 지원하는 액셀러레이터 유형을 지정하는 경우 HealthOmics는 사용 가능한 용량을 기반으로 인스턴스 유형을 선택합니다. 두 인스턴스 유형을 모두 사용할 수 있는 경우 HealthOmics는 저렴한 인스턴스를 선호합니다. 단, 사용 가능한 최신 세대 인스턴스에 대한 기본 설정을 제공하는 nvidia-t4-a10g-l4 태스크 액셀러레이터는 예외입니다.

인스턴스 유형에 대한 자세한 내용은 섹션을 참조하세요가속 컴퓨팅 인스턴스.

다음 예제에서 워크플로 정의는를 액셀러레이터nvidia-l4로 지정합니다.

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"
```

WDL 워크플로 정의 세부 정보

다음 주제에서는 HealthOmics에서 WDL 워크플로 정의에 사용할 수 있는 유형 및 명령에 대한 세부 정보를 제공합니다.

주제

WDL lenient의 암시적 유형 변환

HealthOmics는 input.json 파일 및 워크플로 정의에서 암시적 유형 변환을 지원합니다. 암시적 유형 캐스팅을 사용하려면 워크플로를 생성할 때 워크플로 엔진을 WDL lenient로 지정합니다. WDL lenient는 Cromwell에서 마이그레이션된 워크플로를 처리하도록 설계되었습니다. 고객 Cromwell 지시문과 일부 비준수 로직을 지원합니다.

WDL lenient는 WDL의 제한된 예외 목록에 있는 다음 항목에 대한 유형 변환을 지원합니다.

- Float to Int. 여기서 강제로 인해 정밀도가 손실되지 않습니다(예: 1.0이 1에 매핑됨).
- Int/Float에 대한 문자열로, 강제로 인해 정밀도가 손실되지 않습니다.

- Map[W, X]를 Array[Pair[Y, Z]]에 매핑합니다.이 경우 W는 Y에 강제 적용되고 X는 Z에 강제 적용됩니다.
- Array[Pair[W, X]]를 Map[Y, Z]으로 바꿉니다.이 경우 W는 Y로 강제 적용되고 X는 Z로 강제 적용됩니다(예: 1.0을 1로 매핑).

암시적 유형 캐스팅을 사용하려면 워크플로 또는 워크플로 버전을 생성할 때 워크플로 엔진을 WDL_LENIENT로 지정합니다.

콘솔에서 워크플로 엔진 파라미터의 이름은 Language입니다. API에서 워크플로 엔진 파라미터의 이름은 엔진입니다. 자세한 내용은 프라이빗 워크플로 생성 또는 워크플로 버전 생성을 참조하세요.

input.json의 네임스페이스 정의

HealthOmics는 input.json에서 정규화된 변수를 지원합니다. 예를 들어 워크플로 SumWorkflow에서 number1 및 number2라는 두 개의 입력 변수를 선언하는 경우:

```
workflow SumWorkflow {
  input {
    Int number1
    Int number2
  }
}
```

input.json에서 정규화된 변수로 사용할 수 있습니다.

```
{
    "SumWorkflow.number1": 15,
    "SumWorkflow.number2": 27
}
```

WDL의 기본 유형

다음 표는 WDL의 입력이 일치하는 기본 유형에 매핑되는 방법을 보여줍니다. HealthOmics는 유형 강제에 대한 제한된 지원을 제공하므로 명시적 유형을 설정하는 것이 좋습니다.

기본 유형

WDL 유형	JSON 유형	WDL 예	JSON 키 및 값의 예	참고
`Boolean`	`boolean`	`Boolean b`	`"b": true`	값은 소문자여야 하며 따옴표가 없어야 합니다.
`Int`	`integer`	`Int i`	`"i": 7`	따옴표가 없어야 합니다.
`Float`	`number`	`Float f`	`"f": 42.2`	따옴표가 없어야 합니다.
`String`	`string`	`String s`	`"s": "characters"`	URI인 JSON 문자열은 가져올 WDL 파일에 매핑되어야 합니다.
`File`	`string`	`File f`	`"f": "s3://amzn-s3-demo-bucket1/path/to/file"`	워크플로에 제공된 IAM 역할에 이러한 객체에 대한 읽기 액세스 권한이 있는 한 Amazon S3 및 HealthOmics 스토리지 URIs를 가져옵니다. 다른 URI 체계는 지원되지 않습니다 (예: `file://`, `https://`및 `ftp://`). URI는 객체를 지정해야 합니다. 디렉터리일 수 없습니다.

WDL 유형	JSON 유형	WDL 예	JSON 키 및 값의 예	참고
				즉, 로 끝날 수 없습니다/.
`Directory`	`string`	`Directory d`	`"d": "s3://bucket/path/"`	`Directory` 유형은 WDL 1.0 또는 1.1에 포함되지 않으므로 WDL 파일의 헤더`version development` `nt` 에를 추가해야 합니다. URI는 Amazon S3 URI여야 하며 접두사가 '/'로 끝나야 합니다. 디렉터리의 모든 콘텐츠는 단일 다운로드로 워크플로에 재귀적으로 복사됩니다. 에는 워크플로와 관련된 파일만 포함되어야 `Directory` 합니다.

WDL의 복잡한 유형

다음 표는 WDL의 입력이 일치하는 복합 JSON 유형에 매핑되는 방법을 보여줍니다. WDL의 복잡한 유형은 기본 유형으로 구성된 데이터 구조입니다. 목록과 같은 데이터 구조는 배열로 변환됩니다.

복잡한 유형

WDL 유형	JSON 유형	WDL 예	JSON 키 및 값의 예	참고
Array	array	Array[Int] nums	“nums": [1, 2, 3]	배열의 멤버는 WDL 배열 유형의 형식을 따라야 합니다.
Pair	object	Pair[String, Int] str_to_i	“str_to_i": {"left": "0", "right": 1}	페어의 각 값은 일치하는 WDL 유형의 JSON 형식을 사용해야 합니다.
Map	object	Map[Int, String] int_to_string	"int_to_string": { 2: "hello", 1: "goodbye" }	맵의 각 항목은 일치하는 WDL 유형의 JSON 형식을 사용해야 합니다.
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-s3-demo-bucket1/NA12878.bam.bai"`	구조체 멤버의 이름은 JSON 객체 키의 이름과 정확히 일치해야 합니다. 각 값은 일치하는 WDL 유형의 JSON 형식을 사용해야 합니다.

WDL 유형	JSON 유형	WDL 예	JSON 키 및 값의 예	참고
			}	
`Object`	해당 사항 없음	해당 사항 없음	해당 사항 없음	WDL `Object` 유형은 오래된 유형이므로 `Struct` 모든 경우에 로 교체해야 합니다.

WDL의 지침

HealthOmics는 HealthOmics 지원합니다.

GPU 리소스 구성

HealthOmics는 지원되는 모든 GPU 인스턴스acceleratorCount에서 런타임 속성 acceleratorType 및를 지원합니다. HealthOmics는 액셀러레이터와 동일한 기능을 gpuCount가진 gpuType 및 라는 별칭도 지원합니다. WDL 정의에 두 명령이 모두 포함된 경우 HealthOmics는 액셀러레이터 값을 사용합니다.

다음 예제에서는 이러한 지시문을 사용하는 방법을 보여줍니다.

```
runtime {
    gpuCount: 2
    gpuType: "nvidia-tesla-t4"
}
```

서비스 오류에 대한 작업 재시도 구성

HealthOmics는 서비스 오류(5XX HTTP 상태 코드)로 인해 실패한 작업에 대해 최대 2회의 재시도를 지원합니다. 최대 재시도 횟수(1 또는 2)를 구성하고 서비스 오류에 대한 재시도를 옵트아웃할 수 있습니다. 기본적으로 HealthOmics는 최대 2회의 재시도를 시도합니다.

다음 예제에서는 서비스 오류에 대한 재시도를 옵트아웃`preemptible`하도록를 설정합니다.

```
{
  preemptible: 0
}
```

HealthOmics의 작업 재시도에 대한 자세한 내용은 섹션을 참조하세요작업 재시도.

메모리 부족에 대한 작업 재시도 구성

HealthOmics는 메모리가 부족하여 실패한 작업에 대한 재시도를 지원합니다(컨테이너 종료 코드 137, 4XX HTTP 상태 코드). HealthOmics는 각 재시도에 대해 메모리 양을 두 배로 늘립니다.

기본적으로 HealthOmics는 이러한 유형의 실패에 대해 재시도하지 않습니다. `maxRetries` 지시문을 사용하여 최대 재시도 횟수를 지정합니다.

다음 예제에서는를 3`maxRetries`으로 설정하여 HealthOmics가 최대 4회의 작업 완료 시도(최초 시도 + 3회의 재시도)를 시도하도록 합니다.

```
runtime {
    maxRetries: 3
}
```

> **Note**
>
> 메모리 부족에 대한 작업 재시도에는 GNU findutils 4.2.3 이상이 필요합니다. 기본 HealthOmics 이미지 컨테이너에는이 패키지가 포함되어 있습니다. WDL 정의에서 사용자 지정 이미지를 지정하는 경우 이미지에 GNU findutils 4.2.3 이상이 포함되어 있는지 확인합니다.

반환 코드 구성

returnCodes 속성은 작업의 성공적인 실행을 나타내는 반환 코드 또는 반환 코드 세트를 지정하는 메커니즘을 제공합니다. WDL 엔진은 WDL 정의의 런타임 섹션에서 지정한 반환 코드를 준수하고 그에 따라 작업 상태를 설정합니다.

```
runtime {
    returnCodes: 1
}
```

HealthOmics는 또한 returnCodes와 동일한 기능을 가진 continueOnReturnCode라는 별칭을 지원합니다. 두 속성을 모두 지정하면 HealthOmics는 returnCodes 값을 사용합니다.

WDL의 작업 메타데이터

HealthOmics는 WDL 작업에 대해 다음과 같은 메타데이터 옵션을 지원합니다.

휘발성 속성을 사용하여 작업 수준 캐싱 비활성화

휘발성 속성을 사용하면 WDL 워크플로의 특정 작업에 대한 통화 캐싱을 비활성화할 수 있습니다. 작업이 휘발성으로 표시되면 실행에 대해 캐싱이 활성화된 경우에도 항상 실행되고 캐시된 결과를 사용하지 않습니다.

작업 정의의 메타 섹션에 휘발성 속성을 추가합니다.

```
task my_volatile_task {
    meta {
        volatile: true
    }

    input {
        String input_file
    }

    command {
        echo "Processing ${input_file}" > output.txt
    }

    output {
        File result = "output.txt"
    }
}
```

WDL 워크플로 정의 예제

다음 예제에서는 WDLBAM에서에서 로 변환CRAM하기 위한 프라이빗 워크플로 정의를 보여줍니다. BAM 워크플로CRAM에 대한는 두 가지 작업을 정의하고 genomes-in-the-cloud 컨테이너의 도구를 사용합니다.이 작업은 예제에 나와 있으며 공개적으로 사용할 수 있습니다.

다음 예제에서는 Amazon ECR 컨테이너를 파라미터로 포함하는 방법을 보여줍니다. 이렇게 하면 HealthOmics가 실행을 시작하기 전에 컨테이너에 대한 액세스 권한을 확인할 수 있습니다.

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

다음 예제에서는 파일이 Amazon S3 버킷에 있을 때 실행에 사용할 파일을 지정하는 방법을 보여줍니다.

```
{
    "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"
}
```

시퀀스 스토어에서 파일을 지정하려면 다음 예제와 같이 시퀀스 스토어의 URI를 사용하여를 지정합니다.

```
{
    "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"
}
```

그런 다음 다음 예제와 같이 WDL에서 워크플로를 정의할 수 있습니다.

```
 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}"
    }
}
```

Nextflow 워크플로 정의 세부 정보

HealthOmics는 Nextflow DSL1 및 DSL2를 지원합니다. 자세한 내용은 Nextflow 버전 지원을 참조하세요.

Nextflow DSL2는 Groovy 프로그래밍 언어를 기반으로 하므로 파라미터는 동적이며 Groovy와 동일한 규칙을 사용하여 강제 유형을 지정할 수 있습니다. 입력 JSON에서 제공하는 파라미터와 값은 워크플로의 파라미터(`params`) 맵에서 사용할 수 있습니다.

주제

- nf-schema 및 nf-validation 플러그인 사용
- 스토리지 URIs 지정
- Nextflow 지시문
- 작업 콘텐츠 내보내기

nf-schema 및 nf-validation 플러그인 사용

> **Note**
>
> 플러그인에 대한 HealthOmics 지원 요약:
>
> - v22.04 - 플러그인을 지원하지 않음
> - v23.10 - `nf-schema` 및 지원 `nf-validation`
> - v24.10 - 지원 `nf-schema`

HealthOmics는 Nextflow 플러그인에 대해 다음과 같은 지원을 제공합니다.

- Nextflow v23.10의 경우 HealthOmics는 nf-validation@1.1.1 플러그인을 사전 설치합니다.
- Nextflow v23.10 이상의 경우 HealthOmics는 nf-schema@2.3.0 플러그인을 사전 설치합니다.
- 워크플로 실행 중에는 추가 플러그인을 검색할 수 없습니다. HealthOmics는 `nextflow.config` 파일에서 지정한 다른 플러그인 버전을 무시합니다.
- Nextflow v24 이상의 경우 `nf-schema`는 더 이상 사용되지 않는 `nf-validation` 플러그인의 새 버전입니다. 자세한 내용은 Nextflow GitHub 리포지토리의 nf-schema를 참조하세요.

스토리지 URIs 지정

Amazon S3 또는 HealthOmics URI를 사용하여 Nextflow 파일 또는 경로 객체를 구성하는 경우 읽기 액세스 권한이 부여되는 한 워크플로에서 일치하는 객체를 사용할 수 있습니다. Amazon S3 URIs. 예시는 Amazon S3 입력 파라미터 형식 섹션을 참조하세요.

HealthOmics는 Amazon S3 URIs 또는 HealthOmics 스토리지 URIs에서 glob 패턴 사용을 부분적으로 지원합니다. 워크플로 정의에서 `path` 또는 `file` 채널 생성에 Glob 패턴을 사용합니다. 예상되는 동작과 정확한 사례는 단원을 참조하십시오Amazon S3 입력에서 Glob 패턴의 Nextflow 처리.

Nextflow 지시문

Nextflow 구성 파일 또는 워크플로 정의에서 Nextflow 지시문을 구성합니다. 다음 목록은 HealthOmics가 구성 설정을 적용하는 데 사용하는 우선 순위의 순서를 가장 낮은 우선 순위부터 가장 높은 우선 순위까지 보여줍니다.

1. 구성 파일의 전역 구성입니다.
2. 워크플로 정의의 작업 섹션입니다.
3. 구성 파일의 작업별 선택기입니다.

주제

- 를 사용한 작업 재시도 전략 errorStrategy
- 를 사용하여 작업 재시도 maxRetries
- 를 사용하여 작업 재시도 옵트아웃 omicsRetryOn5xx
- time 지시문을 사용한 작업 기간

를 사용한 작업 재시도 전략 `errorStrategy`

`errorStrategy` 지시문을 사용하여 작업 오류에 대한 전략을 정의합니다. 기본적으로 작업이 오류 표시(종료 상태가 0이 아님)와 함께 반환되면 작업이 중지되고 HealthOmics가 전체 실행을 종료합니다. 를 `errorStrategy`로 설정하면 `retry` HealthOmics는 실패한 작업을 한 번 재시도합니다. 재시도 횟수를 늘리려면 섹션을 참조하세요를 사용하여 작업 재시도 `maxRetries`.

```
process {
    label 'my_label'
    errorStrategy 'retry'

    script:
    """
    your-command-here
    """
}
```

HealthOmics가 실행 중에 작업 재시도를 처리하는 방법에 대한 자세한 내용은 섹션을 참조하세요작업 재시도.

를 사용하여 작업 재시도 **maxRetries**

기본적으로 HealthOmics는 실패한 작업의 재시도를 시도하지 않거나를 구성한 경우 한 번의 재시도를 시도합니다errorStrategy. 최대 재시도 횟수를 늘리려면를 retry errorStrategy로 설정하고 maxRetries 명령을 사용하여 최대 재시도 횟수를 구성합니다.

다음 예제에서는 글로벌 구성에서 최대 재시도 횟수를 3으로 설정합니다.

```
process {
    errorStrategy = 'retry'
    maxRetries = 3
}
```

다음 예제에서는 워크플로 정의maxRetries의 작업 섹션에서를 설정하는 방법을 보여줍니다.

```
process myTask {
    label 'my_label'
    errorStrategy 'retry'
    maxRetries 3

    script:
    """
    your-command-here
    """
}
```

다음 예제에서는 이름 또는 레이블 선택기를 기반으로 Nextflow 구성 파일에서 작업별 구성을 지정하는 방법을 보여줍니다.

```
process {
    withLabel: 'my_label' {
        errorStrategy = 'retry'
        maxRetries = 3
    }

    withName: 'myTask' {
        errorStrategy = 'retry'
        maxRetries = 3
    }
}
```

를 사용하여 작업 재시도 옵트아웃 **omicsRetryOn5xx**

Nextflow v23 및 v24의 경우 HealthOmics는 서비스 오류(5XX HTTP 상태 코드)로 인해 작업이 실패한 경우 작업 재시도를 지원합니다. 기본적으로 HealthOmics는 실패한 작업을 최대 2회 재시도합니다.

서비스 오류에 대한 작업 재시도를 옵트아웃omicsRetryOn5xx하도록를 구성할 수 있습니다. HealthOmics의 작업 재시도에 대한 자세한 내용은 섹션을 참조하세요작업 재시도.

다음 예제에서는 작업 재시도를 옵트아웃하도록 전역 구성omicsRetryOn5xx에서를 구성합니다.

```
process {
    omicsRetryOn5xx = false
}
```

다음 예제에서는 워크플로 정의omicsRetryOn5xx의 작업 섹션에서를 구성하는 방법을 보여줍니다.

```
process myTask {
    label 'my_label'
    omicsRetryOn5xx = false

    script:
    """
    your-command-here
    """
}
```

다음 예제에서는 이름 또는 레이블 선택기를 기반으로 Nextflow 구성 파일에서 작업별 구성omicsRetryOn5xx으로를 설정하는 방법을 보여줍니다.

```
process {
    withLabel: 'my_label' {
        omicsRetryOn5xx = false
    }

    withName: 'myTask' {
        omicsRetryOn5xx = false
    }
}
```

`time` 지시문을 사용한 작업 기간

HealthOmics는 조정 가능한 할당량(참조HealthOmics 서비스 할당량)을 제공하여 실행의 최대 기간을 지정합니다. Nextflow v23 및 v24 워크플로의 경우 Nextflow `time` 명령을 사용하여 최대 작업 기간을 지정할 수도 있습니다.

새 워크플로 개발 중에 최대 작업 기간을 설정하면 런어웨이 작업과 장기 실행 작업을 포착하는 데 도움이 됩니다.

Nextflow 시간 명령에 대한 자세한 내용은 Nextflow 참조의 시간 명령을 참조하세요.

HealthOmics는 Nextflow 시간 명령에 대해 다음과 같은 지원을 제공합니다.

1. HealthOmics는 시간 명령에 대해 1분 단위를 지원합니다. 60초에서 최대 실행 기간 값 사이의 값을 지정할 수 있습니다.
2. 60 미만의 값을 입력하면 HealthOmics는 값을 60초로 반올림합니다. 60보다 큰 값의 경우 HealthOmics는 가장 가까운 분으로 내림합니다.
3. 워크플로가 작업에 대한 재시도를 지원하는 경우 HealthOmics는 시간 초과 시 작업을 재시도합니다.
4. 작업이 시간 초과(또는 마지막 재시도 시간이 초과)되면 HealthOmics는 작업을 취소합니다. 이 작업의 지속 시간은 1~2분입니다.
5. 작업 시간 초과 시 HealthOmics는 실행 및 작업 상태를 실패로 설정하고 실행 중인 다른 작업(시작 중, 보류 중 또는 실행 중 상태의 작업)을 취소합니다. HealthOmics는 제한 시간 이전에 완료한 작업의 출력을 지정된 S3 출력 위치로 내보냅니다.
6. 작업이 보류 중 상태로 소비하는 시간은 작업 기간에 포함되지 않습니다.
7. 실행이 실행 그룹의 일부이고 실행 그룹이 작업 타이머보다 빨리 시간 초과되면 실행 및 작업이 실패 상태로 전환됩니다.

`ms`, , 또는 단위 중 하나 이상을 사용하여 제한 시간을 지정합니다`smhd`.

다음 예제에서는 Nextflow 구성 파일에서 전역 구성을 지정하는 방법을 보여줍니다. 전역 제한 시간을 1시간 30분으로 설정합니다.

```
process {
    time = '1h30m'
}
```

다음 예제에서는 워크플로 정의의 작업 섹션에서 시간 지시문을 지정하는 방법을 보여줍니다. 이 예제에서는 제한 시간을 3일, 5시간 및 4분으로 설정합니다. 이 값은 구성 파일의 전역 값보다 우선하지만 구성 `my_label` 파일의에 대한 작업별 시간 명령보다 우선하지는 않습니다.

```
process myTask {
    label 'my_label'
    time '3d5h4m'

    script:
    """
    your-command-here
    """
}
```

다음 예제에서는 이름 또는 레이블 선택기를 기반으로 Nextflow 구성 파일에서 작업별 시간 지시문을 지정하는 방법을 보여줍니다. 이 예제에서는 글로벌 작업 제한 시간 값을 30분으로 설정합니다. 작업의 경우 값을 2시간`myTask`으로 설정하고 레이블이 인 작업의 경우 값을 3시간으로 설정합니다`my_label`. 선택기와 일치하는 작업의 경우 이러한 값이 워크플로 정의의 전역 값 및 값보다 우선합니다.

```
process {
    time = '30m'

    withLabel: 'my_label' {
        time = '3h'
    }

    withName: 'myTask' {
        time = '2h'
    }
}
```

작업 콘텐츠 내보내기

Nextflow에 작성된 워크플로의 경우 publishDir 명령을 정의하여 작업 콘텐츠를 출력 Amazon S3 버킷으로 내보냅니다. 다음 예제와 같이 publishDir 값을 로 설정합니다`/mnt/workflow/pubdir`. Amazon S3로 파일을 내보내려면 파일이이 디렉터리에 있어야 합니다.

```
 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 워크플로 정의 세부 정보

공통 워크플로 언어 또는 CWL로 작성된 워크플로는 WDL 및 Nextflow로 작성된 워크플로와 유사한 기능을 제공합니다. Amazon S3 또는 HealthOmics 스토리지 URIs 입력 파라미터로 사용할 수 있습니다.

하위 워크플로의 secondaryFile에서 입력을 정의하는 경우 기본 워크플로에 동일한 정의를 추가합니다.

HealthOmics 워크플로는 작업 프로세스를 지원하지 않습니다. CWL 워크플로의 운영 프로세스에 대한 자세한 내용은 CWL 설명서를 참조하세요.

사용하는 각 컨테이너에 대해 별도의 CWL 워크플로를 정의하는 것이 좋습니다. 고정 Amazon ECR URI로 dockerPull 항목을 하드코딩하지 않는 것이 좋습니다.

주제

HealthOmics를 사용하도록 CWL 워크플로 변환

HealthOmics를 사용하도록 기존 CWL 워크플로 정의를 변환하려면 다음을 변경합니다.

- 모든 Docker 컨테이너 URIs Amazon ECR URIs.
- 모든 워크플로 파일이 기본 워크플로에서 입력으로 선언되고 모든 변수가 명시적으로 정의되었는지 확인합니다.
- 모든 JavaScript 코드가 엄격한 모드 수신 거부인지 확인합니다.

를 사용하여 작업 재시도 옵트아웃 **omicsRetryOn5xx**

HealthOmics는 서비스 오류(5XX HTTP 상태 코드)로 인해 작업이 실패한 경우 작업 재시도를 지원합니다. 기본적으로 HealthOmics는 실패한 작업을 최대 2회 재시도합니다. HealthOmics의 작업 재시도에 대한 자세한 내용은 섹션을 참조하세요작업 재시도.

서비스 오류에 대한 작업 재시도를 옵트아웃하려면 워크플로 정의에서 omicsRetryOn5xx 명령을 구성합니다. 요구 사항 또는 힌트에 따라이 지시문을 정의할 수 있습니다. 이식성을 위한 힌트로 지시문을 추가하는 것이 좋습니다.

```
requirements:
  ResourceRequirement:
    omicsRetryOn5xx: false

hints:
  ResourceRequirement:
    omicsRetryOn5xx: false
```

요구 사항은 힌트를 재정의합니다. 태스크 구현이 인클로징 워크플로의 요구 사항에서 제공하는 힌트에 리소스 요구 사항을 제공하는 경우 인클로징 요구 사항이 우선합니다.

워크플로의 다른 수준에서 동일한 작업 요구 사항이 나타나는 경우 HealthOmics는의 가장 구체적인 항목requirements(또는에 항목이 없는 hints경우)을 사용합니다requirements. 다음 목록은 HealthOmics가 구성 설정을 적용하는 데 사용하는 우선 순위의 순서를 가장 낮은 우선 순위부터 가장 높은 우선 순위까지 보여줍니다.

- 워크플로 수준
- 단계 수준
- 워크플로 정의의 작업 섹션

다음 예제에서는 워크플로의 다양한 수준에서 omicsRetryOn5xx 명령을 구성하는 방법을 보여줍니다. 이 예제에서는 워크플로 수준 요구 사항이 워크플로 수준 힌트를 재정의합니다. 작업 및 단계 수준의 요구 사항 구성은 힌트 구성을 재정의합니다.

```
class: Workflow
# Workflow-level requirement and hint
requirements:
  ResourceRequirement:
    omicsRetryOn5xx: false
```

```
hints:
  ResourceRequirement:
    omicsRetryOn5xx: false  # The value in requirements overrides this value

steps:
  task_step:
    # Step-level requirement
    requirements:
      ResourceRequirement:
        omicsRetryOn5xx: false
    # Step-level hint
    hints:
      ResourceRequirement:
        omicsRetryOn5xx: false
    run:
      class: CommandLineTool
      # Task-level requirement
      requirements:
        ResourceRequirement:
          omicsRetryOn5xx: false
      # Task-level hint
      hints:
        ResourceRequirement:
          omicsRetryOn5xx: false
```

워크플로 단계 반복

HealthOmics는 워크플로 단계 반복을 지원합니다. 루프를 사용하여 지정된 조건이 충족될 때까지 워크플로 단계를 반복적으로 실행할 수 있습니다. 이는 작업을 여러 번 반복해야 하거나 특정 결과가 달성될 때까지 반복 프로세스에 유용합니다.

참고: 루프 기능을 사용하려면 CWL 버전 1.2 이상이 필요합니다. 1.2 이전의 CWL 버전을 사용하는 워크플로는 루프 작업을 지원하지 않습니다.

CWL 워크플로에서 루프를 사용하려면 루프 요구 사항을 정의합니다. 다음 예제에서는 루프 요구 사항 구성을 보여줍니다.

```
requirements:
  - class: "http://commonwl.org/cwltool#Loop"
    loopWhen: $(inputs.counter < inputs.max)
    loop:
```

```
      counter:
        loopSource: result
        valueFrom: $(self)
    outputMethod: last
```

loopWhen 필드는 루프가 종료되는 시기를 제어합니다. 이 예제에서는 카운터가 최대값보다 작은 한 루프가 계속됩니다. loop 필드는 반복 간에 입력 파라미터를 업데이트하는 방법을 정의합니다. 는 이전 반복에서 다음 반복으로 피드되는 출력을 loopSource 지정합니다. 로 설정된 outputMethod 필드는 최종 반복의 출력만 last 반환합니다.

메모리가 증가한 상태에서 작업 재시도

HealthOmics는 out-of-memory 작업 실패의 자동 재시도를 지원합니다. 태스크가 코드 137(out-of-memory)로 종료되면 HealthOmics는 지정된 승수를 기반으로 메모리 할당을 늘려 새 태스크를 생성합니다.

> **Note**
>
> HealthOmics는 out-of-memory 실패를 최대 3회 또는 메모리 할당이 1536GiB에 도달할 때까지 중 먼저 도달하는 쪽까지 재시도합니다.

다음 예제에서는 out-of-memory 재시도를 구성하는 방법을 보여줍니다.

```
hints:
  ResourceRequirement:
    ramMin: 4096
  http://arvados.org/cwl#OutOfMemoryRetry:
    memoryRetryMultiplier: 2.5
```

out-of-memory으로 인해 작업이 실패하면 HealthOmics는 공식을 사용하여 재시도 메모리 할당을 계산합니다previous_run_memory × memoryRetryMultiplier. 위 예제에서 메모리가 4,096MB인 작업이 실패하면 재시도는 4,096 × 2.5 = 10,240MB의 메모리를 사용합니다.

memoryRetryMultiplier 파라미터는 재시도에 할당할 추가 메모리의 양을 제어합니다.

- 기본값: 값을 지정하지 않으면 기본값은 입니다2(메모리의 두 배).
- 유효한 범위: 보다 큰 양수여야 합니다1. 값이 잘못되면 4XX 검증 오류가 발생합니다.
- 최소 유효 값: 의미 있는 메모리 증가를 보장하고 과도한 재시도를 방지하기 1.5 위해 1와 사이의 값이 로 1.5 자동 증가합니다.

예제

다음은 CWL로 작성된 워크플로의 예입니다.

```
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
```

다음 파일은 `copy.cwl` 작업을 정의합니다.

```
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)"
```

다음은 GPU 요구 사항이 있는 CWL로 작성된 워크플로의 예입니다.

```
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: []
```

워크플로 정의 예

다음 예제에서는 WDL, Nextflow 및 CWL에서 동일한 워크플로 정의를 보여줍니다.

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
```

HealthOmics 워크플로용 파라미터 템플릿 파일

파라미터 템플릿은 워크플로의 입력 파라미터를 정의합니다. 입력 파라미터를 정의하여 워크플로를 보다 유연하고 다양하게 만들 수 있습니다. 예를 들어 참조 유전체 파일의 Amazon S3 위치에 대한 파라미터를 정의할 수 있습니다. 파라미터 템플릿은 Git 기반 리포지토리 서비스 또는 로컬 드라이브를 통해 제공할 수 있습니다. 그런 다음 사용자는 다양한 데이터 세트를 사용하여 워크플로를 실행할 수 있습니다.

워크플로에 대한 파라미터 템플릿을 생성하거나 HealthOmics에서 파라미터 템플릿을 생성할 수 있습니다.

파라미터 템플릿은 JSON 파일입니다. 파일에서 각 입력 파라미터는 워크플로 입력의 이름과 일치해야 하는 명명된 객체입니다. 실행을 시작할 때 필요한 모든 파라미터에 값을 제공하지 않으면 실행이 실패합니다.

입력 파라미터 객체에는 다음 속성이 포함됩니다.

- description -이 필수 속성은 콘솔이 실행 시작 페이지에 표시하는 문자열입니다. 이 설명은 실행 메타데이터로도 유지됩니다.
- optional -이 선택적 속성은 입력 파라미터가 선택 사항인지 여부를 나타냅니다. optional 필드를 지정하지 않으면 입력 파라미터가 필요합니다.

다음 예제 파라미터 템플릿은 입력 파라미터를 지정하는 방법을 보여줍니다.

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

파라미터 템플릿 생성

HealthOmics는 워크플로 정의를 구문 분석하여 입력 파라미터를 감지하여 파라미터 템플릿을 생성합니다. 워크플로에 파라미터 템플릿 파일을 제공하는 경우 파일의 파라미터가 워크플로 정의에서 감지된 파라미터를 재정의합니다.

다음 섹션에 설명된 대로 CWL, WDL 및 Nextflow 엔진의 구문 분석 로직에는 약간의 차이가 있습니다.

주제

CWL에 대한 파라미터 감지

CWL 워크플로 엔진에서 구문 분석 로직은 다음과 같이 가정합니다.

- null이 가능한 지원되는 모든 유형은 선택적 입력 파라미터로 표시됩니다.
- null이 아닌 지원 유형은 필수 입력 파라미터로 표시됩니다.
- 기본값이 있는 모든 파라미터는 선택적 입력 파라미터로 표시됩니다.
- 설명은 `main` 워크플로 정의의 `label` 섹션에서 추출됩니다. `label`를 지정하지 않으면 설명이 비어 있습니다(빈 문자열).

다음 표에는 CWL 보간 예제가 나와 있습니다. 각 예제의 파라미터 이름은 입니다x. 파라미터가 필요한 경우 파라미터 값을 제공해야 합니다. 파라미터가 선택 사항인 경우 값을 제공할 필요가 없습니다.

이 표에는 기본 유형에 대한 CWL 보간 예제가 나와 있습니다.

입력	입력/출력 예제	필수
`x:` `  type: int`	1 또는 2 또는 ...	예
`x:` `  type: int`	기본값은 2입니다. 유효한 입력은 1 또는 2 또는 ...입니다.	아니요

입력	입력/출력 예제	필수
`default: 2`		
`x:` `  type: int?`	유효한 입력은 없음, 1 또는 2 또는 ...입니다.	아니요

`x:`
`  type: int?`
`  default: 2`
	기본값은 2입니다. 유효한 입력은 없음, 1 또는 2 또는 ...입니다.	아니요

다음 표에는 복잡한 유형에 대한 CWL 보간 예제가 나와 있습니다. 복합 유형은 기본 유형의 모음입니다.

입력	입력/출력 예제	필수

`x:`
`  type: array`
`  items: int`
	[] 또는 [1,2,3]	예

`x:`
`  type: array?`
`  items: int`
	없음 또는 [] 또는 [1,2,3]	아니요

`x:`
`  type: array`
`  items: int?`
	[] 또는 [없음, 3, 없음]	예

`x:`
`  type: array?`
`  items: int?`
	[없음] 또는 없음 또는 [1,2,3] 또는 [없음, 3]이지만 []는 아님	아니요

WDL에 대한 파라미터 감지

WDL 워크플로 엔진에서 구문 분석 로직은 다음과 같은 가정을 합니다.

- null이 가능한 지원되는 모든 유형은 선택적 입력 파라미터로 표시됩니다.
- Null링할 수 없는 지원되는 유형의 경우:
  - 리터럴 또는 표현식이 할당된 모든 입력 변수는 선택적 파라미터로 표시됩니다. 예:

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

  - 입력 파라미터에 값 또는 표현식이 할당되지 않은 경우 필수 파라미터로 표시됩니다.
- 설명은 `main` 워크플로 정의의 `parameter_meta`에서 추출됩니다. `parameter_meta`를 지정하지 않으면 설명이 비어 있습니다(빈 문자열). 자세한 내용은 파라미터 메타데이터에 대한 WDL 사양을 참조하세요.

다음 표에는 WDL 보간 예제가 나와 있습니다. 각 예제의 파라미터 이름은 입니다x. 파라미터가 필요한 경우 파라미터 값을 제공해야 합니다. 파라미터가 선택 사항인 경우 값을 제공할 필요가 없습니다.

이 표에는 기본 유형에 대한 WDL 보간 예제가 나와 있습니다.

입력	입력/출력 예제	필수
정수 x	1 또는 2 또는 ...	예
정수 x = 2	2	아니요
정수 x = 1+2	3	아니요
정수 x = y+z	y+z	아니요
정수? x	없음 또는 1 또는 2 또는 ...	예
정수? x = 2	없음 또는 2	아니요
정수? x = 1+2	없음 또는 3	아니요
정수? x = y+z	없음 또는 y+z	아니요

다음 표에는 복잡한 유형에 대한 WDL 보간 예제가 나와 있습니다. 복합 유형은 기본 유형의 모음입니다.

입력	입력/출력 예제	필수		
배열[Int] x	[1,2,3] 또는 []	예		
배열[Int]+ x	[1]이지만 []는 아님	예		
배열[Int]? x	없음 또는 [] 또는 [1,2,3]	아니요		
배열[Int?] x	[] 또는 [없음, 3, 없음]	예		
배열[Int?]=? x	[없음] 또는 없음 또는 [1,2,3] 또는 [없음, 3]이지만 []는 아님	아니요		
구조 샘플 {문자열 a, 정수 y} 입력의 뒷부분: 샘플 mySample	`String a = mySample.a Int y = mySample.y`	예		
구조 샘플 {문자열 a, 정수 y} 입력의 후반부: 샘플? mySample	`if (defined( mySample)) { String a = mySample.a Int y = mySample.y }`	아니요		

Nextflow에 대한 파라미터 감지

Nextflow의 경우 HealthOmics는 `nextflow_schema.json` 파일을 구문 분석하여 파라미터 템플릿을 생성합니다. 워크플로 정의에 스키마 파일이 포함되지 않은 경우 HealthOmics는 기본 워크플로 정의 파일을 구문 분석합니다.

주제

스키마 파일 구문 분석

구문 분석이 올바르게 작동하려면 스키마 파일이 다음 요구 사항을 충족하는지 확인합니다.

- 스키마 파일의 이름은 이며 기본 워크플로 파일`nextflow_schema.json`과 동일한 디렉터리에 있습니다.
- 스키마 파일은 다음 스키마 중 하나에 정의된 유효한 JSON입니다.
  - json-schema.org/draft/2020-12/schema.
  - json-schema.org/draft-07/schema.

HealthOmics는 `nextflow_schema.json` 파일을 구문 분석하여 파라미터 템플릿을 생성합니다.

- 스키마에 properties 정의된 모든를 추출합니다.
- 속성에 사용할 수 있는 description 경우 속성을 포함합니다.
- 속성의 required 필드를 기반으로 각 파라미터가 선택 사항인지 필수인지 식별합니다.

다음 예제에서는 정의 파일과 생성된 파라미터 파일을 보여줍니다.

```
{
    "$schema": "https://json-schema.org/draft/2020-12/schema",
    "type": "object",
    "$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" }
    ]
}
```

생성된 파라미터 템플릿:

```
{
    "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
    }
}
```

기본 파일 구문 분석

워크플로 정의에 nextflow_schema.json 파일이 포함되지 않은 경우 HealthOmics는 기본 워크플로 정의 파일을 구문 분석합니다.

HealthOmics는 기본 워크플로 정의 파일 및 nextflow.config 파일에 있는 params 표현식을 분석합니다. 기본값이 params 있는 모든는 선택 사항으로 표시됩니다.

구문 분석이 올바르게 작동하려면 다음 요구 사항에 유의하세요.

- HealthOmics는 기본 워크플로 정의 파일만 구문 분석합니다. 모든 파라미터를 캡처하려면 모든 하위 모듈과 가져온 워크플로에 params 모든 파라미터를 연결하는 것이 좋습니다.
- 구성 파일은 선택 사항입니다. 정의한 경우 이름을 지정nextflow.config하고 기본 워크플로 정의 파일과 동일한 디렉터리에 배치합니다.

다음 예제에서는 정의 파일과 생성된 파라미터 템플릿을 보여줍니다.

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

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

생성된 파라미터 템플릿:

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

nextflow.config에 정의된 기본값의 경우 HealthOmics는 다음 예제`params {}`와 같이 내에 선언된 `params` 할당 및 파라미터를 수집합니다. 할당 문에서는 문 왼쪽에 나타나`params`야 합니다.

```
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}"
```

생성된 파라미터 템플릿:

```
{
    // 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
    }
}
```

중첩 파라미터

nextflow_schema.json 및 모두 중첩된 파라미터를 nextflow.config 허용합니다. 그러나 HealthOmics 파라미터 템플릿에는 최상위 파라미터만 필요합니다. 워크플로에서 중첩 파라미터를 사용하는 경우 JSON 객체를 해당 파라미터의 입력으로 제공해야 합니다.

스키마 파일의 중첩 파라미터

HealthOmics는 nextflow_schema.json 파일을 구문 분석할 params 때 중첩된를 건너뜁니다. 예를 들어 다음 nextflow_schema.json 파일을 정의하는 경우:

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

HealthOmics는 파라미터 템플릿을 생성할 input_num 때 input_file 및를 무시합니다.

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

이 워크플로를 실행하면 HealthOmics는 다음과 유사한 `input.json` 파일을 예상합니다.

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

구성 파일의 중첩 파라미터

HealthOmics는 `nextflow.config` 파일에 중첩된 params를 수집하지 않으며 구문 분석 중에 이를 건너뜁니다. 예를 들어 다음 `nextflow.config` 파일을 정의하는 경우:

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

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

HealthOmics는 파라미터 템플릿을 생성할 `params.group.delta` 때 `params.nested.beta` 및를 무시합니다.

```
{
    "alpha": {
```

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

Nextflow 보간의 예

다음 표에는 기본 파일의 파라미터에 대한 Nextflow 보간 예제가 나와 있습니다.

파라미터	필수
params.input_file	예
params.input_file = "s3://bucket/data.json"	아니요
params.nested.input_file	N/A
params.nested.input_file = "s3://bucket/data.json"	N/A

다음 표에는 `nextflow.config` 파일의 파라미터에 대한 Nextflow 보간 예제가 나와 있습니다.

파라미터	필수
`params.input_file = "s3://bucket/data.json"`	아니요
`params { input_file = "s3://bucket/data.json" }`	아니요
`params { nested {`	N/A

파라미터	필수
`input_file = "s3://bucket/data.json" }}`	
`input_file = params.input_file`	N/A

프라이빗 워크플로용 컨테이너 이미지

HealthOmics는 Amazon ECR 프라이빗 리포지토리에서 호스팅되는 컨테이너 이미지를 지원합니다. 컨테이너 이미지를 생성하여 프라이빗 리포지토리에 업로드할 수 있습니다. Amazon ECR 프라이빗 레지스트리를 풀스루 캐시로 사용하여 업스트림 레지스트리의 콘텐츠를 동기화할 수도 있습니다.

Amazon ECR 리포지토리는 서비스를 호출하는 계정과 동일한 AWS 리전에 있어야 합니다. 소스 이미지 리포지토리가 적절한 권한을 제공하는 한 다른가 컨테이너 이미지를 소유할 AWS 계정 수 있습니다. 자세한 내용은 교차 계정 Amazon ECR 액세스에 대한 정책 단원을 참조하십시오.

실행이 시작되기 전에 액세스를 확인할 수 있도록 Amazon ECR 컨테이너 이미지 URIs를 워크플로의 파라미터로 정의하는 것이 좋습니다. 또한 리전 파라미터를 변경하여 새 리전에서 워크플로를 더 쉽게 실행할 수 있습니다.

> **Note**
>
> HealthOmics는 ARM 컨테이너를 지원하지 않으며 퍼블릭 리포지토리에 대한 액세스를 지원하지 않습니다.

HealthOmics가 Amazon ECR에 액세스하도록 IAM 권한을 구성하는 방법에 대한 자세한 내용은 섹션을 참조하세요HealthOmics 리소스 권한.

주제

- 타사 컨테이너 레지스트리와 동기화
- Amazon ECR 컨테이너 이미지에 대한 일반 고려 사항
- HealthOmics 워크플로의 환경 변수
- Amazon ECR 컨테이너 이미지에서 Java 사용

- Amazon ECR 컨테이너 이미지에 작업 입력 추가

타사 컨테이너 레지스트리와 동기화

Amazon ECR 풀스루 캐시 규칙을 사용하여 지원되는 업스트림 레지스트리의 리포지토리를 Amazon ECR 프라이빗 리포지토리와 동기화할 수 있습니다. 자세한 내용은 Amazon ECR 사용 설명서의 업스트림 레지스트리 동기화를 참조하세요.

풀스루 캐시는 캐시를 생성할 때 프라이빗 레지스트리에 이미지 리포지토리를 자동으로 생성하고 업스트림 이미지가 변경될 때 캐시된 이미지와 자동으로 동기화됩니다.

HealthOmics는 다음 업스트림 레지스트리에 대한 풀스루 캐시를 지원합니다.

- Amazon ECR 퍼블릭
- Kubernetes 컨테이너 이미지 레지스트리
- Quay
- Docker Hub
- Microsoft Azure 컨테이너 레지스트리
- GitHub 컨테이너 레지스트리
- GitLab 컨테이너 레지스트리

HealthOmics는 업스트림 Amazon ECR 프라이빗 리포지토리에 대한 풀스루 캐시를 지원하지 않습니다.

Amazon ECR 풀스루 캐시 사용의 이점은 다음과 같습니다.

1. 컨테이너 이미지를 Amazon ECR로 수동으로 마이그레이션하거나 타사 리포지토리에서 업데이트를 동기화할 필요가 없습니다.
2. 워크플로는 프라이빗 리포지토리의 동기화된 컨테이너 이미지에 액세스하며, 이는 퍼블릭 레지스트리에서 런타임에 콘텐츠를 다운로드하는 것보다 더 안정적입니다.
3. Amazon ECR 풀스루 캐시는 예측 가능한 URI 구조를 사용하기 때문에 HealthOmics 서비스는 Amazon ECR 프라이빗 URI를 업스트림 레지스트리 URI와 자동으로 매핑할 수 있습니다. 워크플로 정의에서 URI 값을 업데이트하고 바꿀 필요는 없습니다.

주제

- 풀스루 캐시 구성

- 레지스트리 매핑
- 이미지 매핑

풀스루 캐시 구성

Amazon ECR은 각 리전 AWS 계정 에서에 대한 레지스트리를 제공합니다. 워크플로를 실행하려는 리전과 동일한 리전에서 Amazon ECR 구성을 생성해야 합니다.

다음 섹션에서는 풀스루 캐시의 구성 작업에 대해 설명합니다.

구성 작업

- 풀스루 캐시 규칙 생성
- 업스트림 레지스트리에 대한 레지스트리 권한
- 리포지토리 생성 템플릿
- 워크플로 생성

풀스루 캐시 규칙 생성

캐시하려는 이미지가 있는 각 업스트림 레지스트리에 대해 Amazon ECR 풀스루 캐시 규칙을 생성합니다. 규칙은 업스트림 레지스트리와 Amazon ECR 프라이빗 리포지토리 간의 매핑을 지정합니다.

인증이 필요한 업스트림 레지스트리의 경우 AWS Secrets Manager를 사용하여 자격 증명을 제공합니다.

> **Note**
>
> 활성 실행이 프라이빗 리포지토리를 사용하는 동안에는 풀스루 캐시 규칙을 변경하지 마십시오. 실행이 실패하거나 더 심각하게는 예상치 못한 이미지를 사용하는 파이프라인이 발생할 수 있습니다.

자세한 내용은 Amazon Elastic Container Registry 사용 설명서의 풀스루 캐시 규칙 생성을 참조하세요.

콘솔을 사용하여 풀스루 캐시 규칙 생성

풀스루 캐시를 구성하려면 Amazon ECR 콘솔을 사용하여 다음 단계를 따릅니다.

1. Amazon ECR 콘솔 열기: https://console.aws.amazon.com/ecr

2. 왼쪽 메뉴의 프라이빗 레지스트리에서 기능 및 설정을 확장한 다음 캐시 풀스루를 선택합니다.
3. 풀스루 캐시 페이지에서 규칙 추가를 선택합니다.
4. 업스트림 레지스트리 패널에서 프라이빗 레지스트리와 동기화할 업스트림 레지스트리를 선택한 후 다음을 선택합니다.
5. 업스트림 레지스트리에 인증이 필요한 경우 보안 인증 정보가 포함된 SageMaker AI 보안 암호를 지정하는 새 페이지가 콘솔에 열립니다. 다음을 선택합니다.
6. 네임스페이스 지정의 캐시 네임스페이스 패널에서 특정 리포지토리 접두사를 사용하거나 접두사 없이 프라이빗 리포지토리를 생성할지 여부를 선택합니다. 접두사를 사용하도록 선택한 경우 캐시 리포지토리 접두사에 접두사 이름을 지정합니다.
7. 업스트림 네임스페이스 패널에서 특정 리포지토리 접두사를 사용하거나 접두사 없이 업스트림 리포지토리에서 가져올지 여부를 선택합니다. 접두사를 사용하도록 선택한 경우 Upstream 리포지토리 접두사에 접두사 이름을 지정합니다.

   네임스페이스 예제 패널에는 예제 풀 요청, 업스트림 URL 및 생성된 캐시 리포지토리의 URL이 표시됩니다.
8. 다음을 선택합니다.
9. 구성을 검토하고 생성을 선택하여 규칙을 생성합니다.

자세한 내용은 풀스루 캐시 규칙 생성(AWS 관리 콘솔)을 참조하세요.

CLI를 사용하여 풀스루 캐시 규칙 생성

Amazon ECR create-pull-through-cache-rule 명령을 사용하여 풀스루 캐시 규칙을 생성합니다. 인증이 필요한 업스트림 레지스트리의 경우 Secrets Manager 보안 암호에 자격 증명을 저장합니다.

다음 섹션에서는 지원되는 각 업스트림 레지스트리의 예를 제공합니다.

Amazon ECR 퍼블릭의 경우

다음 예제에서는 Amazon ECR 퍼블릭 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 `ecr-public`으로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가 `ecr-public/`*`upstream-repository-name`*의 명명 체계를 사용하게 됩니다.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix ecr-public \
     --upstream-registry-url public.ecr.aws \
     --region us-east-1
```

Kubernetes Container Registry의 경우

다음 예제에서는 Kubernetes 퍼블릭 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 kubernetes으로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가 kubernetes/*upstream-repository-name*의 명명 체계를 사용하게 됩니다.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix kubernetes \
     --upstream-registry-url registry.k8s.io \
     --region us-east-1
```

Quay의 경우

다음 예제에서는 Quay 퍼블릭 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 quay로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가 quay/*upstream-repository-name*의 명명 체계를 사용하게 됩니다.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix quay \
     --upstream-registry-url quay.io \
     --region us-east-1
```

Docker Hub의 경우

다음 예제에서는 Docker Hub 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 docker-hub으로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가 docker-hub/*upstream-repository-name*의 명명 체계를 사용하게 됩니다. Docker Hub 보안 인증 정보가 포함된 보안 암호의 전체 Amazon 리소스 이름(ARN)을 지정해야 합니다.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix docker-hub \
     --upstream-registry-url registry-1.docker.io \
     --credential-arn arn:aws:secretsmanager:us-east-1:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-1
```

GitHub Container Registry의 경우

다음 예제에서는 GitHub 컨테이너 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 github으로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가

github/*upstream-repository-name*의 명명 체계를 사용하게 됩니다. GitHub 컨테이너 레지스트리 보안 인증 정보가 포함된 보안 암호의 전체 Amazon 리소스 이름(ARN)을 지정해야 합니다.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix github \
     --upstream-registry-url ghcr.io \
     --credential-arn arn:aws:secretsmanager:us-east-1:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-1
```

Microsoft Azure 컨테이너 레지스트리의 경우

다음 예제에서는 Microsoft Azure 컨테이너 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 azure으로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가 azure/*upstream-repository-name*의 명명 체계를 사용하게 됩니다. Microsoft Azure 컨테이너 레지스트리 보안 인증 정보가 포함된 보안 암호의 전체 Amazon 리소스 이름(ARN)을 지정해야 합니다.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix azure \
     --upstream-registry-url myregistry.azurecr.io \
     --credential-arn arn:aws:secretsmanager:us-east-1:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-1
```

GitLab 컨테이너 레지스트리의 경우

다음 예제에서는 GitLab 컨테이너 레지스트리에 대한 풀스루 캐시 규칙을 생성합니다. 리포지토리 접두사를 gitlab으로 지정합니다. 이렇게 하면 풀스루 캐시 규칙을 통해 생성한 각 리포지토리가 gitlab/*upstream-repository-name*의 명명 체계를 사용하게 됩니다. GitLab 컨테이너 레지스트리 보안 인증 정보가 포함된 보안 암호의 전체 Amazon 리소스 이름(ARN)을 지정해야 합니다.

```
aws ecr create-pull-through-cache-rule \
     --ecr-repository-prefix gitlab \
     --upstream-registry-url registry.gitlab.com \
     --credential-arn arn:aws:secretsmanager:us-east-1:111122223333:secret:ecr-
pullthroughcache/example1234 \
     --region us-east-1
```

자세한 내용은 Amazon ECR 사용 설명서의 풀스루 캐시 규칙(CLI) 생성을 참조하세요.

get-run-task CLI 명령을 사용하여 특정 작업에 사용되는 컨테이너 이미지에 대한 정보를 검색할 수 있습니다.

```
aws omics get-run-task --id 1234567 --task-id <task_id>
```

출력에는 컨테이너 이미지에 대한 다음 정보가 포함됩니다.

```
"imageDetails": {
   "image": "string",
   "imageDigest": "string",
   "sourceImage": "string",
         ...
}
```

업스트림 레지스트리에 대한 레지스트리 권한

레지스트리 권한을 사용하여 HealthOmics가 풀스루 캐시를 사용하고 컨테이너 이미지를 Amazon ECR 프라이빗 레지스트리로 가져올 수 있도록 허용합니다. 실행에 사용되는 컨테이너를 제공하는 Amazon ECR 레지스트리 정책을 레지스트리에 추가합니다.

다음 정책은 HealthOmics 서비스가 지정된 풀스루 캐시 접두사(들)로 리포지토리를 생성하고 이러한 리포지토리로 업스트림 풀을 시작할 수 있는 권한을 부여합니다.

1. Amazon ECR 콘솔에서 왼쪽 메뉴를 열고 프라이빗 레지스트리에서 레지스트리 권한을 확장한 다음 명령문 생성을 선택합니다.
2. 오른쪽 상단에서 JSON을 선택합니다. 다음과 유사한 정책을 입력합니다.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPTCinRegPermissions",
            "Effect": "Allow",
            "Principal": {
                "Service": "omics.amazonaws.com"
            },
            "Action": [
                "ecr:CreateRepository",
```

```
                "ecr:BatchImportUpstreamImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-east-1:123456789012:repository/ecr-public/*",
                "arn:aws:ecr:us-east-1:123456789012:repository/docker-hub/*"
            ]
        }
    ]
}
```

리포지토리 생성 템플릿

HealthOmics에서 풀스루 캐싱을 사용하려면 Amazon ECR 리포지토리에 리포지토리 생성 템플릿이 있어야 합니다. 템플릿은 사용자 또는 Amazon ECR이 업스트림 레지스트리에 대한 프라이빗 리포지토리를 생성할 때의 구성 설정을 정의합니다.

각 템플릿에는 Amazon ECR이 새 리포지토리를 특정 템플릿과 일치시키는 데 사용하는 리포지토리 네임스페이스 접두사가 포함되어 있습니다. 템플릿은 리소스 기반 액세스 정책, 태그 불변성, 암호화 및 수명 주기 정책을 포함한 모든 리포지토리 설정에 대한 구성을 지정합니다.

자세한 내용은 Amazon Elastic Container Registry 사용 설명서의 리포지토리 생성 템플릿을 참조하세요.

리포지토리 생성 템플릿을 생성하는 방법:

1. Amazon ECR 콘솔에서 왼쪽 메뉴를 열고 프라이빗 레지스트리에서 기능 및 설정을 확장한 다음 리포지토리 생성 템플릿을 선택합니다.
2. 템플릿 생성을 선택합니다.
3. 템플릿 세부 정보에서 풀스루 캐시를 선택합니다.
4. 이 템플릿을 특정 접두사에 적용할지 아니면 다른 템플릿과 일치하지 않는 모든 리포지토리에 적용할지 선택합니다.

   특정 접두사를 선택하는 경우 접두사에 네임스페이스 접두사 값을 입력합니다. PTC 규칙을 생성할 때이 접두사를 지정했습니다.
5. 다음을 선택합니다.
6. 리포지토리 생성 구성 추가 페이지에서 리포지토리 권한을 입력합니다. 샘플 정책 설명 중 하나를 사용하거나 다음 예제와 유사한 명령문을 입력합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PTCRepoCreationTemplate",
            "Effect": "Allow",
            "Principal": {
                "Service": "omics.amazonaws.com"
            },
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        }
    ]
}
```

7. 선택적으로 수명 주기 정책 및 태그와 같은 리포지토리 설정을 추가할 수 있습니다. Amazon ECR은 지정된 접두사를 사용하는 풀스루 캐시용으로 생성된 모든 컨테이너 이미지에 이러한 규칙을 적용합니다.
8. 다음을 선택합니다.
9. 구성을 검토하고 다음을 선택합니다.

워크플로 생성

새 워크플로 또는 워크플로 버전을 생성할 때 레지스트리 매핑을 검토하고 필요한 경우 업데이트합니다. 자세한 내용은 프라이빗 워크플로 생성을 참조하세요.

레지스트리 매핑

프라이빗 Amazon ECR 레지스트리의 접두사와 업스트림 레지스트리 이름 간에 매핑하도록 레지스트리 매핑을 정의합니다.

Amazon ECR 레지스트리 매핑에 대한 자세한 내용은 Amazon ECR에서 풀스루 캐시 규칙 생성을 참조하세요.

다음 예제에서는 Docker Hub, Quay 및 Amazon ECR Public에 대한 레지스트리 매핑을 보여줍니다.

```
{
    "registryMappings": [
        {
            "upstreamRegistryUrl": "registry-1.docker.io",
            "ecrRepositoryPrefix": "docker-hub"
        },
        {
            "upstreamRegistryUrl": "quay.io",
            "ecrRepositoryPrefix": "quay"
        },
        {
            "upstreamRegistryUrl": "public.ecr.aws",
            "ecrRepositoryPrefix": "ecr-public"
        }
    ]
}
```

이미지 매핑

프라이빗 Amazon ECR 워크플로에 정의된 이미지 이름과 업스트림 레지스트리의 이미지 이름 간에 매핑하도록 이미지 매핑을 정의합니다.

풀스루 캐시를 지원하는 레지스트리에서 이미지 매핑을 사용할 수 있습니다. HealthOmics 않는 업스트림 레지스트리에서 이미지 매핑을 사용할 수도 있습니다. 업스트림 레지스트리를 프라이빗 리포지토리와 수동으로 동기화해야 합니다.

Amazon ECR 이미지 매핑에 대한 자세한 내용은 Amazon ECR에서 풀스루 캐시 규칙 생성을 참조하세요.

다음 예제에서는 프라이빗 Amazon ECR 이미지에서 퍼블릭 게놈 이미지 및 최신 Ubuntu 이미지로의 매핑을 보여줍니다.

```
{
    "imageMappings": [
        {
            "sourceImage": "public.ecr.aws/aws-genomics/broadinstitute/gatk:4.6.0.2",
            "destinationImage": "123456789012.dkr.ecr.us-east-1.amazonaws.com/
broadinstitute/gatk:4.6.0.2"
        },
        {
            "sourceImage": "ubuntu:latest",
```

```
            "destinationImage": "123456789012.dkr.ecr.us-east-1.amazonaws.com/custom/
ubuntu:latest",
        }
    ]
}
```

Amazon ECR 컨테이너 이미지에 대한 일반 고려 사항

- 아키텍처

  HealthOmics는 x86_64 컨테이너를 지원합니다. 로컬 시스템이 Apple Mac과 같은 ARM 기반인 경우 다음과 같은 명령을 사용하여 x86_64 컨테이너 이미지를 빌드합니다.

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

- 진입점 및 쉘

  HealthOmics 워크플로 엔진은 워크플로 작업에 사용되는 컨테이너 이미지에 대한 명령 재정의로 bash 스크립트를 삽입합니다. 따라서 컨테이너 이미지는 bash 쉘이 기본값이 되도록 지정된 ENTRYPOINT 없이 빌드해야 합니다.

- 탑재된 경로

  공유 파일 시스템은 /tmp에서 컨테이너 작업에 마운트됩니다. 이 위치의 컨테이너 이미지에 내장된 모든 데이터 또는 도구는 재정의됩니다.

  워크플로 정의는 /mnt/workflow의 읽기 전용 탑재를 통해 태스크에 사용할 수 있습니다.

- 이미지 크기

  최대 컨테이너 이미지 크기는 HealthOmics 워크플로 고정 크기 할당량 섹션을 참조하세요.

HealthOmics 워크플로의 환경 변수

HealthOmics는 컨테이너에서 실행되는 워크플로에 대한 정보가 있는 환경 변수를 제공합니다. 워크플로 작업의 로직에서 이러한 변수의 값을 사용할 수 있습니다.

모든 HealthOmics 워크플로 변수는 `AWS_WORKFLOW_` 접두사로 시작합니다. 이 접두사는 보호된 환경 변수 접두사입니다. 워크플로 컨테이너에서 자체 변수에이 접두사를 사용하지 마세요.

HealthOmics는 다음과 같은 워크플로 환경 변수를 제공합니다.

AWS_REGION

이 변수는 컨테이너가 실행 중인 리전입니다.

AWS_WORKFLOW_RUN

이 변수는 현재 실행의 이름입니다.

AWS_WORKFLOW_RUN_ID

이 변수는 현재 실행의 실행 식별자입니다.

AWS_WORKFLOW_RUN_UUID

이 변수는 현재 실행의 실행 UUID입니다.

AWS_WORKFLOW_TASK

이 변수는 현재 작업의 이름입니다.

AWS_WORKFLOW_TASK_ID

이 변수는 현재 작업의 작업 식별자입니다.

AWS_WORKFLOW_TASK_UUID

이 변수는 현재 작업의 작업 UUID입니다.

다음 예제에서는 각 환경 변수의 일반적인 값을 보여줍니다.

```
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 컨테이너 이미지에서 Java 사용

워크플로 태스크가 GATK와 같은 Java 애플리케이션을 사용하는 경우 컨테이너에 대한 다음 메모리 요구 사항을 고려하세요.

- Java 애플리케이션은 스택 메모리와 힙 메모리를 사용합니다. 기본적으로 최대 힙 메모리는 컨테이너에서 사용 가능한 총 메모리의 백분율입니다. 이 기본값은 특정 JVM 배포 및 JVM 버전에 따라 달

라지므로 JVM 관련 설명서를 참조하거나 Java 명령줄 옵션(예: `-Xmx`)을 사용하여 힙 메모리 최대 값을 명시적으로 설정합니다.

- JVM 스택에도 메모리가 필요하므로 최대 힙 메모리를 컨테이너 메모리 할당의 100%로 설정하지 마세요. 메모리는 JVM 가비지 수집기 및 컨테이너에서 실행되는 기타 운영 체제 프로세스에도 필요합니다.
- GATK와 같은 일부 Java 애플리케이션은 기본 메서드 호출 또는 메모리 매핑 파일과 같은 기타 최적화를 사용할 수 있습니다. 이러한 기법에는 JVM 최대 힙 파라미터로 제어되지 않는 "오프 힙"으로 수행되는 메모리 할당이 필요합니다.

  Java 애플리케이션이 오프 힙 메모리를 할당한다는 것을 알고 있는 경우(또는 의심하는 경우) 태스크 메모리 할당에 오프 힙 메모리 요구 사항이 포함되어 있는지 확인합니다.

  이러한 오프 힙 할당으로 인해 컨테이너의 메모리가 부족해지는 경우 일반적으로 JVM이이 메모리를 제어하지 않기 때문에 Java OutOfMemory 오류가 표시되지 않습니다.

Amazon ECR 컨테이너 이미지에 작업 입력 추가

워크플로 작업을 실행하는 데 필요한 모든 실행 파일, 라이브러리 및 스크립트를 작업을 실행하는 데 사용되는 Amazon ECR 이미지에 추가합니다.

작업 컨테이너 이미지 외부에 있는 스크립트, 바이너리 및 라이브러리를 사용하지 않는 것이 좋습니다. 이는 `bin` 디렉터리를 `nf-core` 워크플로 패키지의 일부로 사용하는 워크플로를 사용할 때 특히 중요합니다. 이 디렉터리는 워크플로 작업에 사용할 수 있지만 읽기 전용 디렉터리로 마운트됩니다. 이 디렉터리의 필수 리소스는 작업 이미지에 복사해야 하며 런타임 시 또는 작업에 사용되는 컨테이너 이미지를 빌드할 때 사용할 수 있어야 합니다.

HealthOmics가 지원하는 컨테이너 이미지의 최대 크기는 HealthOmics 워크플로 고정 크기 할당량 섹션을 참조하세요. HealthOmics

HealthOmics 워크플로 README 파일

워크플로에 대한 지침, 다이어그램 및 필수 정보가 포함된 README.md 파일을 업로드할 수 있습니다. 각 워크플로 버전은 언제든지 업데이트할 수 있는 README 파일 하나를 지원합니다.

README 요구 사항은 다음과 같습니다.

- README 파일은 마크다운(.md) 형식이어야 합니다.
- 최대 파일 크기: 500KiB

주제

기존 README 사용

Git 리포지토리에서 내보낸 READMEs에는 일반적으로 리포지토리 외부에서 작동하지 않는 상대 링크가 포함되어 있습니다. HealthOmics Git 통합은 콘솔에서 적절한 렌더링을 위해 이를 절대 링크로 자동 변환하므로 수동 URL 업데이트가 필요하지 않습니다.

Amazon S3 또는 로컬 드라이브에서 가져온 READMEs 경우 이미지와 링크는 퍼블릭 URLs 사용하거나 적절한 렌더링을 위해 상대 경로를 업데이트해야 합니다.

> **Note**
>
> HealthOmics 콘솔에 표시하려면 이미지를 공개적으로 호스팅해야 합니다. GitHub Enterprise Server 또는 GitLab Self-Managed리포지토리에 저장된 이미지는 렌더링할 수 없습니다.

렌더링 조건

HealthOmics 콘솔은 절대 경로를 사용하여 공개적으로 액세스할 수 있는 이미지와 링크를 보간합니다. 프라이빗 리포지토리에서 URLs을 렌더링하려면 사용자가 리포지토리에 액세스할 수 있어야 합니다. 사용자 지정 도메인을 사용하는 GitHub Enterprise Server 또는 GitLab Self-Managed리포지토리의 경우 HealthOmics는 이러한 프라이빗 리포지토리에 저장된 상대 링크를 확인하거나 이미지를 렌더링할 수 없습니다.

다음 표에는 AWS 콘솔 README 뷰에서 지원하는 마크다운 요소가 나와 있습니다.

요소	AWS 콘솔
알림	예, 하지만 아이콘은 없습니다.
배지	예
기본 텍스트 형식 지정	예
코드 블록	예, 하지만 구문 강조 표시 및 복사 버튼 기능이 없습니다.

요소	AWS 콘솔
축소 가능한 섹션	예
제목	예
이미지 형식	예
이미지(클릭 가능)	예
줄 바꿈	예
Mermaid 다이어그램	만 그래프를 열고, 그래프 위치를 이동하고, 코드를 복사할 수 있습니다.
견적	예
Subscript 및 Superscript	예
테이블	예, 하지만 텍스트 정렬을 지원하지 않습니다.
텍스트 정렬	예

이미지 및 링크 URLs 사용

소스 공급자에 따라 페이지 및 이미지URLs을 다음 형식으로 구성합니다.

- `{username}`: 리포지토리가 호스팅되는 사용자 이름입니다.
- `{repo}`: 리포지토리 이름입니다.
- `{ref}`: 소스 참조(브랜치, 태그 및 커밋 ID).
- `{path}`: 리포지토리의 페이지 또는 이미지에 대한 파일 경로입니다.

소스 공급자	페이지 URL	이미지 URL
GitHub	`https://github.com/{username}/{repo}/blob/{ref}/{path}`	`https://github.com/{username}/{repo}/blob/{ref}/{path}?raw=true`

소스 공급자	페이지 URL	이미지 URL
		`https://raw.githubusercontent.com/{username}/{repo}/{ref}/{path}`
GitLab	`https://gitlab.com/{username}/{repo}/-/blob/{ref}/{path}`	`https://gitlab.com/{username}/{repo}/-/raw/{ref}/{path}`
Bitbucket	`https://bitbucket.org/{username}/{repo}/src/{ref}/{path}`	`https://bitbucket.org/{username}/{repo}/raw/{ref}/{path}`

GitHub, GitLab및는 퍼블릭 리포지토리에 연결되는 페이지 및 이미지 URLs을 모두 Bitbucket 지원합니다. 다음 표에는 프라이빗 리포지토리에 대한 이미지 및 링크 URLs 렌더링에 대한 각 소스 공급자의 지원이 나와 있습니다.

프라이빗 리포지토리 지원		
소스 공급자	페이지 URL	이미지 URL
GitHub	리포지토리에 액세스할 수 있는 경우에만	아니요
GitLab	리포지토리에 액세스할 수 있는 경우에만	아니요
Bitbucket	리포지토리에 액세스할 수 있는 경우에만	아니요

프라이빗 워크플로에 대한 Sentieon 라이선스 요청

프라이빗 워크플로에서 Sentieon 소프트웨어를 사용하는 경우 Senieon 라이선스가 필요합니다. 다음 단계에 따라 Sentieon 소프트웨어에 대한 라이선스를 요청하고 설정합니다.

- Sentieon 라이선스 요청
  - Sentieon 지원 그룹(support@sentieon.com)에 이메일을 보내 소프트웨어 라이선스를 요청합니다.
    - 이메일에 AWS 정식 사용자 ID를 입력합니다.
    - 다음 지침에 따라 AWS 정식 사용자 ID를 찾습니다.
- HealthOmics 서비스 역할을 업데이트하여 해당 리전의 Sentieon 라이선스 서버 프록시 및 Sentieon Omics 버킷에 대한 액세스 권한을 부여합니다. 다음 예제에서는에서 액세스 권한을 부여합니다`us-east-1`. 필요한 경우이 텍스트를 해당 리전으로 바꿉니다.

  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/*"
            ]
        }
    ]
 }
```

- AWS 지원 사례를 생성하여 Sentieon 라이선스 서버 프록시에 액세스합니다.
  - 지원 사례를 생성하려면 support.console.aws.amazon.com 이동합니다.
  - 지원 사례에 AWS 계정 및 리전을 제공합니다. 계정이 라이선스 서버 프록시의 허용 목록에 추가됩니다.
- Sentieon 컨테이너와 Sentieon 라이선스 스크립트를 사용하여 프라이빗 워크플로를 빌드합니다.
  - 프라이빗 워크플로 내에서 Sentieon 도구를 사용하는 방법에 대한 추가 지침은 GitHub의 Sentieon-Amazon-Omics를 참조하세요.
- Sentieon 소프트웨어 버전 202112.07 이상은 HealthOmics 라이선스 서버 프록시를 지원합니다. 202112.07 이전의 Sentieon 소프트웨어 버전을 사용하려면 Sentieon 지원팀에 문의하십시오.

HealthOmics의 워크플로 린터

워크플로를 생성한 후에는 첫 번째 실행을 시작하기 전에 워크플로에서 린터를 실행하는 것이 좋습니다. 린터는 실행이 실패할 수 있는 오류를 감지합니다.

WDL의 경우 워크플로를 생성할 때 HealthOmics가 자동으로 린터를 실행합니다. 린터 출력은 get-workflow 응답의 `statusMessage` 필드에서 사용할 수 있습니다. 다음 CLI 명령을 사용하여 상태 출력을 검색합니다(생성한 WDL 워크플로의 워크플로 ID 사용).

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

HealthOmics는 워크플로를 생성하기 전에 워크플로 정의에서 실행할 수 있는 린터를 제공합니다. HealthOmics로 마이그레이션하려는 기존 파이프라인에서 이러한 린터를 실행합니다.

- WDL - WDL 린터를 실행하는 퍼블릭 Amazon ECR 이미지입니다.

- Nextflow - Nextflow에 대한 Linter 규칙을 실행하는 퍼블릭 Amazon ECR 이미지입니다. GitHub에서 이 린터의 소스 코드에 액세스할 수 있습니다.
- CWL - 사용할 수 없음

HealthOmics 워크플로 작업

프라이빗 워크플로를 생성하려면 다음이 필요합니다.

- Workflow definition file: WDL, Nextflow또는 로 작성된 워크플로 정의 파일입니다CWL. 워크플로 정의는 워크플로를 사용하는 실행에 대한 입력 및 출력을 지정합니다. 또한 컴퓨팅 및 메모리 요구 사항을 포함하여 워크플로의 실행 및 실행 작업에 대한 사양도 포함되어 있습니다. 워크플로 정의 파일은 `.zip` 형식이어야 합니다. 자세한 내용은 HealthOmics의 워크플로 정의 파일을 참조하세요. HealthOmics
  - Amazon Q CLI를 사용하여 WDL, Nextflow 및 CWL에서 워크플로 정의 파일을 빌드하고 검증할 수 있습니다. 자세한 내용은 Amazon Q CLI에 대한 프롬프트 예제 및 GitHub의 HealthOmics Agentic 생성형 AI 자습서를 참조하세요.
- (Optional) Parameter template file:에 작성된 파라미터 템플릿 파일입니다JSON. 파일을 생성하여 실행 파라미터를 정의하거나 HealthOmics가 파라미터 템플릿을 생성합니다. 자세한 내용은 HealthOmics 워크플로용 파라미터 템플릿 파일을 참조하세요.

- Amazon ECR container images: 워크플로에 사용되는 각 컨테이너에 대해 프라이빗 Amazon ECR 리포지토리를 생성합니다. 워크플로용 컨테이너 이미지를 생성하여 프라이빗 리포지토리에 저장하거나 지원되는 업스트림 레지스트리의 콘텐츠를 ECR 프라이빗 리포지토리와 동기화합니다.
- (Optional) Sentieon licenses: 프라이빗 워크플로에서 Sentieon 소프트웨어를 사용할 수 있는 Sentieon 라이선스를 요청합니다.

4MiB(압축)보다 큰 워크플로 정의 파일의 경우 워크플로 생성 중에 다음 옵션 중 하나를 선택합니다.

- Amazon Simple Storage Service 폴더에 업로드하고 위치를 지정합니다.
- 외부 리포지토리(최대 크기 1GiB)에 업로드하고 리포지토리 세부 정보를 지정합니다.

워크플로를 생성한 후 `UpdateWorkflow` 작업으로 다음 워크플로 정보를 업데이트할 수 있습니다.

- 명칭
- 설명
- 기본 스토리지 유형
- 기본 스토리지 용량(워크플로 ID 사용)
- README.md 파일

워크플로의 다른 정보를 변경하려면 새 워크플로 또는 워크플로 버전을 생성합니다.

워크플로 버전 관리를 사용하여 워크플로를 구성하고 구조화합니다. 버전은 반복 워크플로 업데이트의 도입을 관리하는 데도 도움이 됩니다. 버전에 대한 자세한 내용은 워크플로 버전 생성을 참조하십시오.

주제

- 프라이빗 워크플로 생성
- 프라이빗 워크플로 업데이트
- 프라이빗 워크플로 삭제
- 워크플로 상태 확인
- 워크플로 정의에서 게놈 파일 참조

프라이빗 워크플로 생성

HealthOmics 콘솔, AWS CLI 명령 또는 AWS SDKs 중 하나를 사용하여 워크플로를 생성합니다.

> **Note**
>
> 워크플로 이름에 개인 식별 정보(PII)를 포함하지 마십시오. 이러한 이름은 CloudWatch 로그에 표시됩니다.

워크플로를 생성할 때 HealthOmics는 워크플로에 범용 고유 식별자(UUID)를 할당합니다. 워크플로 UUID는 워크플로 및 워크플로 버전에서 고유한 글로벌 고유 식별자(guid)입니다. 데이터 출처를 위해 워크플로 UUID를 사용하여 워크플로를 고유하게 식별하는 것이 좋습니다.

워크플로 작업에서 외부 도구(실행 파일, 라이브러리 또는 스크립트)를 사용하는 경우 이러한 도구를 컨테이너 이미지에 빌드합니다. 컨테이너 이미지를 호스팅하는 옵션은 다음과 같습니다.

- ECR 프라이빗 레지스트리에서 컨테이너 이미지를 호스팅합니다. 이 옵션의 사전 조건:
  - ECR 프라이빗 리포지토리를 생성하거나 기존 리포지토리를 선택합니다.
  - 에 설명된 대로 ECR 리소스 정책을 구성합니다Amazon ECR 권한.
  - 컨테이너 이미지를 프라이빗 리포지토리에 업로드합니다.
- 컨테이너 이미지를 지원되는 타사 레지스트리의 콘텐츠와 동기화합니다. 이 옵션의 사전 조건:
  - ECR 프라이빗 레지스트리에서 각 업스트림 레지스트리에 대한 풀스루 캐시 규칙을 구성합니다. 자세한 내용은 이미지 매핑 단원을 참조하십시오.
  - 에 설명된 대로 ECR 리소스 정책을 구성합니다Amazon ECR 권한.
  - 리포지토리 생성 템플릿을 생성합니다. 템플릿은 Amazon ECR이 업스트림 레지스트리에 대한 프라이빗 리포지토리를 생성하는 시기에 대한 설정을 정의합니다.
  - 접두사 매핑을 생성하여 워크플로 정의의 컨테이너 이미지 참조를 ECR 캐시 네임스페이스에 다시 매핑합니다.

워크플로를 생성할 때 워크플로, 실행 및 작업에 대한 정보가 포함된 워크플로 정의를 제공합니다. HealthOmics는 워크플로 정의를 로컬 또는 Amazon S3 버킷 또는 지원되는 Git 기반 리포지토리에 저장된 .zip 아카이브로 검색할 수 있습니다.

주제

- 콘솔을 사용하여 워크플로 생성
- CLI를 사용하여 워크플로 생성
- SDK를 사용하여 워크플로 생성

콘솔을 사용하여 워크플로 생성

워크플로를 생성하는 단계

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 프라이빗 워크플로를 선택합니다.
3. 프라이빗 워크플로 페이지에서 워크플로 생성을 선택합니다.
4. 워크플로 정의 페이지에서 다음 정보를 제공합니다.
    1. 워크플로 이름:이 워크플로의 고유한 이름입니다. AWS HealthOmics 콘솔 및 CloudWatch 로그에서 실행을 구성하려면 워크플로 이름을 설정하는 것이 좋습니다.
    2. 설명(선택 사항):이 워크플로에 대한 설명입니다.
5. 워크플로 정의 패널에서 다음 정보를 제공합니다.
    1. 워크플로 언어(선택 사항): 워크플로의 사양 언어를 선택합니다. 그렇지 않으면 HealthOmics가 워크플로 정의에서 언어를 결정합니다.
    2. 워크플로 정의 소스에서 Git 기반 리포지토리, Amazon S3 위치 또는 로컬 드라이브에서 정의 폴더를 가져오도록 선택합니다.
        a. 리포지토리 서비스에서 가져오기의 경우:

> **Note**
>
> HealthOmics는 GitHub, , , GitLab, Bitbucket에 대한 퍼블릭 및 프라이빗 리포지토리 GitHub self-managed를 지원합니다GitLab self-managed.

        i. 연결을 선택하여 AWS 리소스를 외부 리포지토리에 연결합니다. 연결을 생성하려면 섹션을 참조하세요외부 코드 리포지토리와 연결.

> **Note**
>
> TLV 리전의 고객은 워크플로를 생성하려면 IAD (us-east-1) 리전에서 연결을 생성해야 합니다.

        ii. 전체 리포지토리 ID에 리포지토리 ID를 user-name/repo-name으로 입력합니다. 이 리포지토리의 파일에 액세스할 수 있는지 확인합니다.

iii. 소스 참조(선택 사항)에 리포지토리 소스 참조(브랜치, 태그 또는 커밋 ID)를 입력합니다. HealthOmics는 소스 참조가 지정되지 않은 경우 기본 브랜치를 사용합니다.

iv. 파일 패턴 제외에 파일 패턴을 입력하여 특정 폴더, 파일 또는 확장자를 제외합니다. 이렇게 하면 리포지토리 파일을 가져올 때 데이터 크기를 관리하는 데 도움이 됩니다. 패턴은 최대 50개이며 패턴은 glob 패턴 구문을 따라야 합니다. 예제:

A. `tests/`

B. `*.jpeg`

C. `large_data.zip`

b. S3에서 정의 폴더 선택의 경우:

i. 압축된 워크플로 정의 폴더가 포함된 Amazon S3 위치를 입력합니다. Amazon S3 버킷은 워크플로와 동일한 리전에 있어야 합니다.

ii. 계정이 Amazon S3 버킷을 소유하지 않은 경우 S3 버킷 소유자의 AWS 계정 ID에 버킷 소유자의 계정 ID를 입력합니다. S3 이 정보는 HealthOmics가 버킷 소유권을 확인할 수 있도록 하기 위해 필요합니다.

c. 로컬 소스에서 정의 폴더 선택의 경우:

i. 압축된 워크플로 정의 폴더의 로컬 드라이브 위치를 입력합니다.

3. 기본 워크플로 정의 파일 경로(선택 사항): 압축된 워크플로 정의 폴더 또는 리포지토리에서 파일로의 `main` 파일 경로를 입력합니다. 워크플로 정의 폴더에 파일이 하나만 있거나 기본 파일의 이름이 "main"인 경우에는이 파라미터가 필요하지 않습니다.

6. README 파일(선택 사항) 패널에서 README 파일의 소스를 선택하고 다음 정보를 제공합니다.

- 리포지토리 서비스에서 가져오기의 README 파일 경로에 리포지토리 내의 README 파일 경로를 입력합니다.
- S3에서 파일 선택의 S3의 README 파일에 README 파일의 Amazon S3 URI를 S3입력합니다. Amazon S3
- 로컬 소스에서 파일 선택: README - 선택 사항에서 파일 선택을 선택하여 업로드할 마크다운(.md) 파일을 선택합니다.

7. 기본 실행 스토리지 구성 패널에서이 워크플로를 사용하는 실행에 대한 기본 실행 스토리지 유형 및 용량을 제공합니다.

1. 실행 스토리지 유형: 정적 또는 동적 스토리지를 임시 실행 스토리지의 기본값으로 사용할지 여부를 선택합니다. 기본값은 정적 스토리지입니다.

2. 스토리지 용량 실행(선택 사항): 정적 실행 스토리지 유형의 경우이 워크플로에 필요한 기본 실행 스토리지 양을 입력할 수 있습니다. 이 파라미터의 기본값은 1200GiB입니다. 실행을 시작할 때 이러한 기본값을 재정의할 수 있습니다.

8. 태그(선택 사항):이 워크플로에 최대 50개의 태그를 연결할 수 있습니다.
9. 다음을 선택합니다.
10. 워크플로 파라미터 추가(선택 사항) 페이지에서 파라미터 소스를 선택합니다.

    1. 워크플로 정의 파일에서 구문 분석의 경우 HealthOmics는 워크플로 정의 파일에서 워크플로 파라미터를 자동으로 구문 분석합니다.
    2. Git 리포지토리에서 파라미터 템플릿 제공에서 리포지토리의 파라미터 템플릿 파일 경로를 사용합니다.
    3. 로컬 소스에서 JSON 파일 선택에서 파라미터를 지정하는 로컬 소스에서 JSON 파일을 업로드합니다.
    4. 워크플로 파라미터를 수동으로 입력하려면 파라미터 이름과 설명을 수동으로 입력합니다.

11. 파라미터 미리 보기 패널에서이 워크플로 버전의 파라미터를 검토하거나 변경할 수 있습니다. JSON 파일을 복원하면 로컬 변경 사항이 손실됩니다.
12. 다음을 선택합니다.
13. 컨테이너 URI 다시 매핑 페이지의 매핑 규칙 패널에서 워크플로에 대한 URI 매핑 규칙을 정의할 수 있습니다.

    매핑 파일 소스에서 다음 옵션 중 하나를 선택합니다.

    - 없음 - 매핑 규칙이 필요하지 않습니다.
    - S3에서 JSON 파일 선택 - 매핑 파일의 S3 위치를 지정합니다.
    - 로컬 소스에서 JSON 파일 선택 - 로컬 디바이스에서 매핑 파일 위치를 지정합니다.
    - 수동으로 매핑 입력 - 매핑 패널에 레지스트리 매핑 및 이미지 매핑을 입력합니다.

14. 콘솔에 매핑 패널이 표시됩니다. 매핑 소스 파일을 선택한 경우 콘솔에 파일의 값이 표시됩니다.

    a. 레지스트리 매핑에서 매핑을 편집하거나 매핑을 추가할 수 있습니다(최대 20개의 레지스트리 매핑).

       각 레지스트리 매핑에는 다음 필드가 포함됩니다.

       - 업스트림 레지스트리 URL - 업스트림 레지스트리의 URI입니다.

- ECR 리포지토리 접두사 - Amazon ECR 프라이빗 리포지토리에서 사용할 리포지토리 접두사입니다.
- (선택 사항) 업스트림 리포지토리 접두사 - 업스트림 레지스트리에 있는 리포지토리의 접두사입니다.
- (선택 사항) ECR 계정 ID - 업스트림 컨테이너 이미지를 소유한 계정의 계정 ID입니다.

b. 이미지 매핑에서 이미지 매핑을 편집하거나 매핑을 추가할 수 있습니다(최대 100개의 이미지 매핑).

각 이미지 매핑에는 다음 필드가 포함됩니다.

- 소스 이미지 - 업스트림 레지스트리에서 소스 이미지의 URI를 지정합니다.
- 대상 이미지 - 프라이빗 Amazon ECR 레지스트리에서 해당 이미지의 URI를 지정합니다.

15. 다음을 선택합니다.
16. 워크플로 구성을 검토한 다음 워크플로 생성을 선택합니다.

CLI를 사용하여 워크플로 생성

워크플로 파일과 파라미터 템플릿 파일이 로컬 시스템에 있는 경우 다음 CLI 명령을 사용하여 워크플로를 생성할 수 있습니다.

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

`create-workflow` 작업은 다음 응답을 반환합니다.

```
{
  "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"
}
```

워크플로를 생성할 때 사용할 선택적 파라미터

워크플로를 생성할 때 선택적 파라미터를 지정할 수 있습니다. 구문 세부 정보는 AWS HealthOmics API 참조의 CreateWorkflow를 참조하세요.

주제

- 워크플로 정의 Amazon S3 위치 지정
- Git 기반 리포지토리에서 워크플로 정의 사용
- Readme 파일 지정
- main 정의 파일 지정
- 실행 스토리지 유형 지정
- GPU 구성 지정
- 풀스루 캐시 매핑 파라미터 구성

워크플로 정의 Amazon S3 위치 지정

워크플로 정의 파일이 Amazon S3 폴더에 있는 경우 다음 예제와 같이 `definition-uri` 파라미터를 사용하여 위치를 지정합니다. 계정이 Amazon S3 버킷을 소유하지 않은 경우 소유자의 AWS 계정 ID를 입력합니다.

```
aws omics create-workflow  \
  --name Test  \
  --definition-uri s3://omics-bucket/workflow-definition/  \
  --owner-id  123456789012
    ...
```

Git 기반 리포지토리에서 워크플로 정의 사용

지원되는 Git 기반 리포지토리의 워크플로 정의를 사용하려면 요청에 `definition-repository` 파라미터를 사용합니다. 입력 소스가 두 개 이상 포함된 경우 요청이 실패하므로 다른 `definition` 파라미터를 제공하지 마십시오.

`definition-respository` 파라미터에는 다음 필드가 포함됩니다.

- connectionArn - AWS 리소스를 외부 리포지토리에 연결하는 코드 연결의 ARN입니다.
- fullRepositoryId - 리포지토리 ID를 로 입력합니다`owner-name/repo-name`. 이 리포지토리의 파일에 액세스할 수 있는지 확인합니다.

- sourceReference (선택 사항) - 리포지토리 참조 유형(BRANCH, TAG 또는 COMMIT)과 값을 입력합니다.

  소스 참조를 지정하지 않으면 HealthOmics는 기본 브랜치에서 최신 커밋을 사용합니다.
- excludeFilePatterns (선택 사항) - 파일 패턴을 입력하여 특정 폴더, 파일 또는 확장자를 제외합니다. 이렇게 하면 리포지토리 파일을 가져올 때 데이터 크기를 관리하는 데 도움이 됩니다. 최대 50개의 패턴을 제공합니다. 패턴은 glob 패턴 구문을 따라야 합니다. 예제:
  - `tests/`
  - `*.jpeg`
  - `large_data.zip`

Git 기반 리포지토리에서 워크플로 정의를 지정할 때를 사용하여 파라미터 템플릿 파일을 `parameter-template-path` 지정합니다. 이 파라미터를 제공하지 않으면 HealthOmics는 파라미터 템플릿 없이 워크플로를 생성합니다.

다음 예제는 Git 기반 프라이빗 리포지토리의 콘텐츠와 관련된 파라미터를 보여줍니다.

```
aws omics create-workflow \
    --name custom-variant \
    --description "Custom variant calling pipeline" \
    --engine "WDL" \
    --definition-repository '{
            "connectionArn": "arn:aws:codeconnections:us-
east-1:123456789012:connection/abcd1234-5678-90ab-cdef-1234567890ab",
            "fullRepositoryId": "myorg/my-genomics-workflows",
            "sourceReference": {
                "type": "BRANCH",
                "value": "main"
            },
            "excludeFilePatterns": ["tests/**", "*.log"]
      }' \
    --main "workflows/variant-calling/main.wdl" \
    --parameter-template-path "parameters/variant-calling-params.json" \
    --readme-path "docs/variant-calling-README.md" \
    --storage-type "DYNAMIC" \
```

자세한 예제는 블로그 게시물 How To Create a AWS HealthOmics Workflows from Content in Git를 참조하세요.

Readme 파일 지정

다음 파라미터 중 하나를 사용하여 README 파일 위치를 지정할 수 있습니다.

- readme-markdown - 로컬 시스템의 문자열 입력 또는 파일입니다.
- readme-uri - S3에 저장된 파일의 URI입니다.
- readme-path - 리포지토리의 README 파일 경로입니다.

readme-path는 정의 리포지토리와 함께 사용해야 합니다. README 파라미터를 지정하지 않으면 HealthOmics는 루트 수준 README.md 파일을 리포지토리에 가져옵니다(있는 경우).

다음 예제에서는 readme-path 및 readme-uri를 사용하여 README 파일 위치를 지정하는 방법을 보여줍니다.

```
# Using README from repository
aws omics create-workflow \
    --name "documented-workflow" \
    --definition-repository '...' \
    --readme-path "docs/workflow-guide.md"

# Using README from S3
aws omics create-workflow \
    --name "s3-readme-workflow" \
    --definition-repository '...' \
    --readme-uri "s3://my-bucket/workflow-docs/readme.md"
```

자세한 내용은 HealthOmics 워크플로 README 파일 단원을 참조하십시오.

main 정의 파일 지정

여러 워크플로 정의 파일을 포함하는 경우 `main` 파라미터를 사용하여 워크플로의 기본 정의 파일을 지정합니다.

```
aws omics create-workflow  \
  --name Test  \
  --main multi_workflow/workflow2.wdl  \
    ...
```

실행 스토리지 유형 지정

기본 실행 스토리지 유형(DYNAMIC 또는 STATIC) 및 실행 스토리지 용량(정적 스토리지에 필요)을 지정할 수 있습니다. 실행 스토리지 유형에 대한 자세한 내용은 섹션을 참조하세요HealthOmics 워크플로에서 스토리지 유형 실행.

```
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  \
```

GPU 구성 지정

액셀러레이터 파라미터를 사용하여 가속 컴퓨팅 인스턴스에서 실행되는 워크플로를 생성합니다. 다음 예제에서는 accelerators 파라미터를 사용하는 방법을 보여줍니다. 워크플로 정의에서 GPU 구성을 지정합니다. 가속 컴퓨팅 인스턴스을(를) 참조하세요.

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

풀스루 캐시 매핑 파라미터 구성

Amazon ECR 풀스루 캐시 매핑 기능을 사용하는 경우 기본 매핑을 재정의할 수 있습니다. 컨테이너 설정 파라미터에 대한 자세한 내용은 섹션을 참조하세요프라이빗 워크플로용 컨테이너 이미지.

다음 예제에서 파일에는이 콘텐츠가 mappings.json 포함되어 있습니다.

```
{
    "registryMappings": [
        {
            "upstreamRegistryUrl": "registry-1.docker.io",
            "ecrRepositoryPrefix": "docker-hub"
        },
        {
            "upstreamRegistryUrl": "quay.io",
            "ecrRepositoryPrefix": "quay",
            "accountId": "123412341234"
        },
        {
```

```
            "upstreamRegistryUrl": "public.ecr.aws",
            "ecrRepositoryPrefix": "ecr-public"
        }
    ],
    "imageMappings": [{
            "sourceImage": "docker.io/library/ubuntu:latest",
            "destinationImage": "healthomics-docker-2/custom/ubuntu:latest",
            "accountId": "123412341234"
        },
        {
            "sourceImage": "nvcr.io/nvidia/k8s/dcgm-exporter",
            "destinationImage": "healthomics-nvidia/k8s/dcgm-exporter"
        }
    ]
}
```

create-workflow 명령에서 매핑 파라미터를 지정합니다.

```
aws omics create-workflow  \
     ...
--container-registry-map-file file://mappings.json
    ...
```

매핑 파라미터 파일의 S3 위치를 지정할 수도 있습니다.

```
aws omics create-workflow  \
     ...
--container-registry-map-uri s3://amzn-s3-demo-bucket1/test.zip
    ...
```

SDK를 사용하여 워크플로 생성

SDKs. 다음 예제에서는 Python SDK를 사용하여 워크플로를 생성하는 방법을 보여줍니다.

```
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={ ... }
)
```

프라이빗 워크플로 업데이트

HealthOmics 콘솔, AWS CLI 명령 또는 AWS SDKs.

> **Note**
>
> 워크플로 이름에 개인 식별 정보(PII)를 포함하지 마세요. 이러한 이름은 CloudWatch 로그에 표시됩니다.

주제

- 콘솔을 사용하여 워크플로 업데이트
- CLI를 사용하여 워크플로 업데이트
- SDK를 사용하여 워크플로 업데이트

콘솔을 사용하여 워크플로 업데이트

워크플로를 업데이트하는 단계

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 프라이빗 워크플로를 선택합니다.
3. 프라이빗 워크플로 페이지에서 업데이트할 워크플로를 선택합니다.
4. 워크플로 페이지에서:
   - 워크플로에 버전이 있는 경우 기본 버전을 선택해야 합니다.
   - 작업 목록에서 선택한 편집을 선택합니다.
5. 워크플로 편집 페이지에서 다음 값 중 하나를 변경할 수 있습니다.
   - 워크플로 이름입니다.
   - 워크플로 설명.
   - 워크플로의 기본 실행 스토리지 유형입니다.

- 기본 실행 스토리지 용량(실행 스토리지 유형이 정적 스토리지인 경우). 기본 실행 스토리지 구성에 대한 자세한 내용은 섹션을 참조하세요콘솔을 사용하여 워크플로 생성.

6. 변경 사항 저장을 선택하여 변경 사항을 적용합니다.

CLI를 사용하여 워크플로 업데이트

다음 예제와 같이 워크플로 이름과 설명을 업데이트할 수 있습니다. 기본 실행 스토리지 유형(STATIC 또는 DYNAMIC)과 실행 스토리지 용량(정적 스토리지 유형의 경우)을 변경할 수도 있습니다. 실행 스토리지 유형에 대한 자세한 내용은 섹션을 참조하세요HealthOmics 워크플로에서 스토리지 유형 실행.

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

`update-workflow` 요청에 대한 응답을 받지 못합니다.

SDK를 사용하여 워크플로 업데이트

SDKs.

다음 예제에서는 Python SDK를 사용하여 워크플로를 업데이트하는 방법을 보여줍니다.

```
import boto3

omics = boto3.client('omics')

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

프라이빗 워크플로 삭제

워크플로가 더 이상 필요하지 않은 경우 HealthOmics 콘솔, AWS CLI 명령 또는 AWS SDKs. 다음 기준을 충족하는 워크플로를 삭제할 수 있습니다.

- 상태는 ACTIVE 또는 FAILED입니다.

- 활성 공유가 없습니다.
- 모든 워크플로 버전을 삭제했습니다.

워크플로를 삭제해도 워크플로를 사용하는 진행 중인 실행에는 영향을 주지 않습니다.

주제

- 콘솔을 사용하여 워크플로 삭제
- CLI를 사용하여 워크플로 삭제
- SDK를 사용하여 워크플로 삭제

콘솔을 사용하여 워크플로 삭제

워크플로 삭제

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 프라이빗 워크플로를 선택합니다.
3. 프라이빗 워크플로 페이지에서 삭제할 워크플로를 선택합니다.
4. 워크플로 페이지의 작업 목록에서 선택한 삭제를 선택합니다.
5. 워크플로 삭제 모달에 "확인"을 입력하여 삭제를 확인합니다.
6. 삭제를 선택합니다.

CLI를 사용하여 워크플로 삭제

다음 예제에서는 AWS CLI 명령을 사용하여 워크플로를 삭제하는 방법을 보여줍니다. 예제를 실행하려면를 삭제하려는 워크플로의 ID*workflow id*로 바꿉니다.

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

HealthOmics는 `delete-workflow` 요청에 대한 응답을 보내지 않습니다.

SDK를 사용하여 워크플로 삭제

SDKs.

다음 예제에서는 Python SDK를 사용하여 워크플로를 삭제하는 방법을 보여줍니다.

```
import boto3

omics = boto3.client('omics')

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

워크플로 상태 확인

워크플로를 생성한 후 다음과 같이 get-workflow를 사용하여 워크플로의 상태를 확인하고 다른 세부 정보를 볼 수 있습니다.

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

응답에는 다음과 같이 상태를 포함한 워크플로 세부 정보가 포함됩니다.

```
{
    "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"
  }
```

상태가 로 전환된 후이 워크플로를 사용하여 실행을 시작할 수 있습니다`ACTIVE`.

워크플로 정의에서 게놈 파일 참조

HealthOmics 참조 스토어 객체는 다음과 같은 URI를 사용하여 참조할 수 있습니다. 필요한 *reference ID* 경우 자체 *reference store ID*, 및 *account ID*를 사용합니다.

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

일부 워크플로에는 참조 유전체에 대한 SOURCE 및 INDEX 파일이 모두 필요합니다. 이전 URI는 기본 짧은 양식이며 기본적으로 SOURCE 파일로 설정됩니다. 두 파일 중 하나를 지정하려면 다음과 같이 긴 URI 양식을 사용할 수 있습니다.

```
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
```

시퀀스 읽기 세트를 사용하면 다음과 같이 유사한 패턴을 갖게 됩니다.

```
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 기반 읽기 세트와 같은 일부 읽기 세트에는 페어링된 읽기가 포함될 수 있습니다. 다음 예제에서는 이를 SOURCE1 및 SOURCE2라고 합니다. BAM 및 CRAM과 같은 형식에는 SOURCE1 파일만 있습니다. 일부 읽기 세트에는 bai 또는 파일과 같은 INDEX crai 파일이 포함됩니다. 앞의 URI는 기본 짧은 양식이며 SOURCE1 파일로 기본 설정됩니다. 다음과 같이 긴 URI 양식을 사용하여 정확한 파일 또는 인덱스를 지정할 수 있습니다.

```
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
```

다음은 두 개의 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>"
}
```

시작 실행 요청에를 AWS CLI 추가하여의 입력 JSON 파일을 참조`--inputs file://<input_file.json>`합니다.

HealthOmics의 워크플로 버전 관리

워크플로를 변경해야 하는 경우 새 워크플로 또는 새 워크플로 버전을 생성할 수 있습니다. 실행 로직에 영향을 주지 않는 허용된 구성 변경을 제외하고 버전은 변경할 수 없습니다.

워크플로 버전은 다음과 같은 이점을 제공합니다.

- 버전은 관련된 논리적 워크플로 그룹을 형성합니다. 각 워크플로 버전에 사용자 정의 이름을 추가하여 보다 쉽게 관리할 수 있습니다(특히 버전이 많은 워크플로의 경우).
- 여러 버전의 워크플로를 동시에 실행할 수 있습니다.
- 워크플로의 모든 버전은 동일한 워크플로 ID와 기본 ARN을 공유하므로 워크플로를 수정한 후 파이프라인 관리를 간소화할 수 있습니다.
- 워크플로 버전은 워크플로와 동일한 수준의 데이터 출처를 제공합니다. 버전은 변경할 수 없으며 HealthOmics는 각 워크플로 버전에 대해 고유한 ARN을 생성합니다. 버전 ARN에는 다음 예제와 같이 워크플로 ID와 버전 이름이 포함됩니다.

  ```
  arn:aws:omics:us-west-2:123456789012:workflow/1234567/version/
  myUniqueVersionName
  ```
- 공유 워크플로를 소유한 경우 구독자(이전 버전을 계속 사용할 수 있는 사용자)를 중단하지 않고 워크플로를 업데이트할 수 있습니다. 구독자는 모든 워크플로 버전에 액세스할 수 있습니다. 새 버전을 생성하는 경우 워크플로를 다시 공유할 필요가 없습니다.
- 워크플로 실행을 시작할 때 워크플로 버전을 지정할 수 있습니다.
  - 사용자는 프로덕션 실행을 위해 안정적인 버전을 유지하고 테스트 실행을 위해 최신 버전을 사용해 볼 수 있습니다.
  - 새 버전에 문제가 발생하면 사용자는 이전 버전의 워크플로로 되돌릴 수 있습니다.
  - 공유 워크플로의 구독자는 사용할 버전을 선택할 수 있습니다.

주제

기본 워크플로 버전

워크플로의 버전을 하나 이상 생성한 후 HealthOmics는 원래 워크플로를 기본 버전으로 취급합니다. 실행을 시작할 때 선택적으로 실행에 대한 워크플로 버전을 지정할 수 있습니다. 실행을 시작할 때 버전을 지정하지 않으면 HealthOmics는 기본 버전을 사용합니다.

콘솔에서 HealthOmics는 기본 버전 레이블이 있는 원래 워크플로를 나타냅니다. 콘솔은 워크플로 버전을 하나 이상 생성한 후에만이 레이블을 사용합니다. 원본 워크플로는 항상 기본 버전으로 유지됩니다. 다른 버전을 기본값으로 할당할 수 없습니다.

워크플로와 연결된 다른 버전이 있는 경우 워크플로의 기본 버전을 삭제할 수 없습니다. 자세한 내용은 프라이빗 워크플로 삭제 단원을 참조하십시오.

워크플로 버전 생성

워크플로의 새 버전을 생성할 때 새 버전의 구성 값을 지정해야 합니다. 워크플로에서 구성 값을 상속하지 않습니다.

버전을 생성할 때이 워크플로에 고유한 버전 이름을 제공합니다. HealthOmics가 버전을 생성한 후에는 이름을 변경할 수 없습니다.

버전 이름은 문자 또는 숫자로 시작해야 하며 대문자 및 소문자, 숫자, 하이픈, 마침표 및 밑줄을 포함할 수 있습니다. 최대 길이는 64자입니다. 예를 들어 버전 1, 버전 2, 버전 3과 같은 간단한 이름 지정 체계를 사용할 수 있습니다. 워크플로 버전을 2.7.0, 2.7.1, 2.7.2와 같은 자체 내부 버전 관리 규칙과 일치시킬 수도 있습니다.

선택적으로 버전 설명 필드를 사용하여이 버전에 대한 메모를 추가합니다. 예를 들어 Fix for syntax error in workflow definition입니다.

> **Note**
>
> 버전 이름에 개인 식별 정보(PII)를 포함하지 마십시오. 버전 이름은 워크플로 버전 ARN에 표시됩니다.

HealthOmics는 워크플로 버전에 고유한 ARN을 할당합니다. ARN은 워크플로 ID와 버전 이름의 조합에 따라 고유합니다.

> **Warning**
>
> 워크플로 버전을 삭제한 후 HealthOmics를 사용하면 다른 워크플로 버전의 버전 이름을 재사용할 수 있습니다. 버전 이름을 재사용하지 않는 것이 가장 좋습니다. 이름을 재사용하는 경우 워크플로와 각 버전에는 출처에 사용할 수 있는 고유한 UUID가 있습니다.

주제

콘솔을 사용하여 워크플로 버전 생성

워크플로 버전을 생성하는 단계

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(☰)을 엽니다. 프라이빗 워크플로를 선택합니다.
3. 프라이빗 워크플로 페이지에서 새 버전의 워크플로를 선택합니다.
4. 워크플로 세부 정보 페이지에서 새 버전 생성을 선택합니다.
5. 버전 생성 페이지에서 다음 정보를 제공합니다.
    1. 버전 이름: 워크플로 전체에서 고유한 워크플로 버전의 이름을 입력합니다.
    2. 버전 설명(선택 사항): 설명 필드를 사용하여이 버전에 대한 메모를 추가할 수 있습니다.
6. 워크플로 정의 패널에서 다음 정보를 제공합니다.
    1. 워크플로 언어(선택 사항): 워크플로 버전의 사양 언어를 선택합니다. 그렇지 않으면 HealthOmics가 워크플로 정의에서 언어를 결정합니다.
    2. 워크플로 정의 소스에서 Git 기반 리포지토리, Amazon S3 위치 또는 로컬 드라이브에서 정의 폴더를 가져오도록 선택합니다.
        a. 리포지토리 서비스에서 가져오기의 경우:

> **Note**
>
> HealthOmics는 GitHub, , , GitLab, Bitbucket에 대한 퍼블릭 및 프라이빗 리포지토리 GitHub self-managed를 지원합니다GitLab self-managed.

i. 연결을 선택하여 AWS 리소스를 외부 리포지토리에 연결합니다. 연결을 생성하려면 섹션을 참조하세요외부 코드 리포지토리와 연결.

> **Note**
>
> TLV 리전의 고객은 워크플로를 생성하려면 IAD (us-east-1) 리전에서 연결을 생성해야 합니다.

ii. 전체 리포지토리 ID에 리포지토리 ID를 user-name/repo-name으로 입력합니다. 이 리포지토리의 파일에 액세스할 수 있는지 확인합니다.

iii. 소스 참조(선택 사항)에 리포지토리 소스 참조(브랜치, 태그 또는 커밋 ID)를 입력합니다. HealthOmics는 소스 참조가 지정되지 않은 경우 기본 브랜치를 사용합니다.

iv. 파일 패턴 제외에 파일 패턴을 입력하여 특정 폴더, 파일 또는 확장자를 제외합니다. 이렇게 하면 리포지토리 파일을 가져올 때 데이터 크기를 관리하는 데 도움이 됩니다. 패턴은 최대 50개이며 패턴은 glob 패턴 구문을 따라야 합니다. 예:

A. `tests/`

B. `*.jpeg`

C. `large_data.zip`

b. S3에서 정의 폴더 선택의 경우:

i. 압축된 워크플로 정의 폴더가 포함된 Amazon S3 위치를 입력합니다. Amazon S3 버킷은 워크플로와 동일한 리전에 있어야 합니다.

ii. 계정이 Amazon S3 버킷을 소유하지 않은 경우 S3 버킷 소유자의 AWS 계정 ID에 버킷 소유자의 계정 ID를 입력합니다. S3 이 정보는 HealthOmics가 버킷 소유권을 확인할 수 있도록 하기 위해 필요합니다.

c. 로컬 소스에서 정의 폴더 선택의 경우:

i. 압축된 워크플로 정의 폴더의 로컬 드라이브 위치를 입력합니다.

3. 기본 워크플로 정의 파일 경로(선택 사항): 압축된 워크플로 정의 폴더 또는 리포지토리에서 파일로의 `main` 파일 경로를 입력합니다. 워크플로 정의 폴더에 파일이 하나만 있거나 기본 파일의 이름이 "main"인 경우에는이 파라미터가 필요하지 않습니다.

7. README 파일(선택 사항) 패널에서 README 파일의 소스를 선택하고 다음 정보를 제공합니다.

   - 리포지토리 서비스에서 가져오기의 README 파일 경로에 리포지토리 내의 README 파일 경로를 입력합니다.
   - S3에서 파일 선택의 경우 S3의 README 파일에 README 파일의 Amazon S3 URI를 입력합니다.
   - 로컬 소스에서 파일 선택: README - 선택 사항에서 파일 선택을 선택하여 업로드할 마크다운 (.md) 파일을 선택합니다.

8. 기본 실행 스토리지 구성 패널에서이 워크플로를 사용하는 실행에 대한 기본 실행 스토리지 유형 및 용량을 제공합니다.

   1. 실행 스토리지 유형: 정적 또는 동적 스토리지를 임시 실행 스토리지의 기본값으로 사용할지 여부를 선택합니다. 기본값은 정적 스토리지입니다.
   2. 스토리지 용량 실행(선택 사항): 정적 실행 스토리지 유형의 경우이 워크플로에 필요한 기본 실행 스토리지 양을 입력할 수 있습니다. 이 파라미터의 기본값은 1200GiB입니다. 실행을 시작할 때 이러한 기본값을 재정의할 수 있습니다.

9. 태그(선택 사항): 최대 50개의 태그를이 워크플로 버전과 연결할 수 있습니다.

10. 다음을 선택합니다.

11. 워크플로 파라미터 추가(선택 사항) 페이지에서 파라미터 소스를 선택합니다.

    1. 워크플로 정의 파일에서 구문 분석의 경우 HealthOmics는 워크플로 정의 파일에서 워크플로 파라미터를 자동으로 구문 분석합니다.
    2. Git 리포지토리의 파라미터 템플릿 제공에서 리포지토리의 파라미터 템플릿 파일 경로를 사용합니다.
    3. 로컬 소스에서 JSON 파일 선택에서 파라미터를 지정하는 로컬 소스에서 JSON 파일을 업로드합니다.
    4. 워크플로 파라미터를 수동으로 입력하려면 파라미터 이름과 설명을 수동으로 입력합니다.

12. 파라미터 미리 보기 패널에서이 워크플로 버전의 파라미터를 검토하거나 변경할 수 있습니다. JSON 파일을 복원하면 로컬 변경 사항이 손실됩니다.

13. 컨테이너 URI 다시 매핑 페이지의 매핑 규칙 패널에서 워크플로에 대한 URI 매핑 규칙을 정의할 수 있습니다.

매핑 파일 소스에서 다음 옵션 중 하나를 선택합니다.

- 없음 - 매핑 규칙이 필요하지 않습니다.
- S3에서 JSON 파일 선택 - 매핑 파일의 S3 위치를 지정합니다.
- 로컬 소스에서 JSON 파일 선택 - 로컬 디바이스에서 매핑 파일 위치를 지정합니다.
- 수동으로 매핑 입력 - 매핑 패널에 레지스트리 매핑 및 이미지 매핑을 입력합니다.

14. 콘솔에 매핑 패널이 표시됩니다. 매핑 소스 파일을 선택한 경우 콘솔에 파일의 값이 표시됩니다.

    a. 레지스트리 매핑에서 매핑을 편집하거나 매핑을 추가할 수 있습니다(최대 20개의 레지스트리 매핑).

       각 레지스트리 매핑에는 다음 필드가 포함됩니다.

       - 업스트림 레지스트리 URL - 업스트림 레지스트리의 URI입니다.
       - ECR 리포지토리 접두사 - Amazon ECR 프라이빗 리포지토리에서 사용할 리포지토리 접두사입니다.
       - (선택 사항) 업스트림 리포지토리 접두사 - 업스트림 레지스트리에 있는 리포지토리의 접두사입니다.
       - (선택 사항) ECR 계정 ID - 업스트림 컨테이너 이미지를 소유한 계정의 계정 ID입니다.

    b. 이미지 매핑에서 이미지 매핑을 편집하거나 매핑을 추가할 수 있습니다(최대 100개의 이미지 매핑).

       각 이미지 매핑에는 다음 필드가 포함됩니다.

       - 소스 이미지 - 업스트림 레지스트리에서 소스 이미지의 URI를 지정합니다.
       - 대상 이미지 - 프라이빗 Amazon ECR 레지스트리에서 해당 이미지의 URI를 지정합니다.

15. 다음을 선택합니다.
16. 버전 구성을 검토한 다음 버전 생성을 선택합니다.

버전이 생성되면 콘솔이 워크플로 세부 정보 페이지로 돌아가 워크플로 및 버전 테이블에 새 버전을 표시합니다.

CLI를 사용하여 워크플로 버전 생성

`CreateWorkflowVersion` API 작업을 사용하여 워크플로 버전을 생성할 수 있습니다. 선택적 파라미터의 경우 HealthOmics는 다음 기본값을 사용합니다.

파라미터	Default
Engine	워크플로 정의에서 결정됨
스토리지 유형	STATIC
스토리지 용량(정적 스토리지용)	1200 GiB
기본	워크플로 정의 폴더의 내용을 기반으로 결정됩니다. 자세한 내용은 HealthOmics 워크플로 정의 요구 사항을 참조하세요.
액셀러레이터	없음
태그	없음

다음 CLI 예제에서는 정적 스토리지를 기본 실행 스토리지로 사용하여 워크플로 버전을 생성합니다.

```
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
```

워크플로 정의 파일이 Amazon S3 폴더에 있는 경우 대신 `definition-uri` 파라미터를 사용하여 위치를 입력합니다`definition-zip`. 자세한 내용은 AWS HealthOmics API 참조의 CreateWorkflowVersion을 참조하세요.

`create-workflow-version` 요청에 대해 다음과 같은 응답을 받습니다.

```
{
  "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"
}
```

SDK를 사용하여 워크플로 버전 생성

SDKs.

다음 예제에서는 Python SDK를 사용하여 워크플로 버전을 생성하는 방법을 보여줍니다.

```
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={ ... }
)
```

워크플로 버전의 상태 확인

워크플로 버전을 생성한 후 다음과 같이 get-workflow-version을 사용하여 워크플로의 상태를 확인하고 다른 세부 정보를 볼 수 있습니다.

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

응답은 다음과 같이 상태를 포함한 워크플로 세부 정보를 제공합니다.

```
{
```

```
  "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"
}
```

이 워크플로 버전으로 실행을 시작하려면 상태가 로 전환되어야 합니다ACTIVE.

워크플로 버전 업데이트

프라이빗 워크플로 버전에 대한 설명과 기본 실행 스토리지 구성을 업데이트할 수 있습니다. 워크플로 버전의 다른 정보를 변경하려면 새 버전을 생성합니다.

주제

- 콘솔을 사용하여 워크플로 버전 업데이트
- CLI를 사용하여 워크플로 버전 업데이트
- SDK를 사용하여 워크플로 버전 업데이트

콘솔을 사용하여 워크플로 버전 업데이트

워크플로 버전을 업데이트하려면

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 프라이빗 워크플로를 선택합니다.
3. 프라이빗 워크플로 페이지에서 워크플로를 선택합니다.
4. 워크플로 페이지의 업데이트할 워크플로 버전을 선택하고 작업 목록에서 선택한 편집을 선택합니다.

   - 기본 버전을 선택하면 콘솔에서 워크플로 편집 페이지가 열립니다. 자세한 내용은 프라이빗 워크플로 업데이트 단원을 참조하십시오.
   - 사용자 정의 버전을 선택하면 콘솔에서 버전 편집 페이지가 열립니다.
5. 버전 편집 페이지에서 다음 정보를 제공합니다.

   - 버전 설명(선택 사항) -이 버전에 대한 설명입니다.
6. 기본 실행 스토리지 구성 패널에서이 워크플로 버전을 사용하는 실행에 대해 다음과 같은 기본값을 제공합니다. 실행을 시작할 때 기본값을 재정의할 수 있습니다.

- 스토리지 유형 실행에서 정적 또는 동적을 선택합니다.
- 정적 실행 스토리지의 경우이 워크플로 버전을 사용하는 실행의 기본 스토리지 용량 실행을 선택합니다. 이 파라미터의 기본값은 1200GiB입니다.

7. 변경 사항 저장을 선택합니다.

콘솔이 워크플로 세부 정보 페이지로 돌아가고 업데이트된 워크플로 버전이 포함된 페이지 배너가 표시됩니다.

CLI를 사용하여 워크플로 버전 업데이트

다음 CLI 명령을 사용하여 워크플로 버전의 파라미터를 업데이트할 수 있습니다. 워크플로 ID와 버전 이름의 조합은 버전을 고유하게 식별합니다.

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

`update-workflow-version` 요청에 대한 응답을 받지 못합니다.

SDK를 사용하여 워크플로 버전 업데이트

SDKs. 다음 python SDK 예제에서는 워크플로 버전의 스토리지 유형과 설명을 업데이트하는 방법을 보여줍니다.

```
import boto3

omics = boto3.client('omics')

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

워크플로 버전 삭제

콘솔, CLI 또는 SDKs. 워크플로 버전을 삭제해도 워크플로 버전을 사용하는 진행 중인 실행에는 영향을 주지 않습니다.

는 삭제할 수 없습니다기본 워크플로 버전. 모든 사용자 정의 버전을 삭제한 다음 워크플로를 삭제합니다.

주제

- 콘솔을 사용하여 워크플로 버전 삭제
- CLI를 사용하여 워크플로 버전 삭제
- SDK를 사용하여 워크플로 버전 삭제

콘솔을 사용하여 워크플로 버전 삭제

워크플로 버전을 삭제하려면

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 프라이빗 워크플로를 선택합니다.
3. 프라이빗 워크플로 페이지에서 워크플로를 선택합니다.
4. 워크플로 페이지의 삭제할 워크플로 버전을 선택하고 작업 목록에서 선택한 삭제를 선택합니다.
5. 워크플로 버전 삭제 모달에 "확인"을 입력하여 삭제를 확인합니다.
6. 삭제를 선택합니다.

콘솔에 삭제된 워크플로 버전이 포함된 페이지 배너가 표시됩니다.

CLI를 사용하여 워크플로 버전 삭제

다음 CLI 명령을 사용하여 사용자 정의 워크플로 버전을 삭제할 수 있습니다. 워크플로 ID와 버전 이름의 조합은 버전을 고유하게 식별합니다.

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

`delete-workflow-version` 요청에 대한 응답을 받지 못합니다.

SDK를 사용하여 워크플로 버전 삭제

SDKs.

다음 예제에서는 Python SDK를 사용하여 워크플로를 삭제하는 방법을 보여줍니다.

```
import boto3

omics = boto3.client('omics')

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

HealthOmics 실행 사용

워크플로를 생성한 후 워크플로를 사용하여 실행을 시작할 수 있습니다.

실행을 시작하면 HealthOmics는 실행 중에 사용할 워크플로 엔진에 임시 실행 스토리지를 할당합니다. 데이터 격리 및 보안을 보장하기 위해 HealthOmics는 각 실행 시작 시 스토리지를 프로비저닝하고 실행 종료 시 프로비저닝을 해제합니다.

HealthOmics는 워크플로 실행 및 작업과 관련된 여러 할당량을 제공합니다. 기본값은 예기치 않은 비용 오버런을 방지하기 위해 의도적으로 보수적입니다. 이 할당량의 증가를 요청할 수 있습니다. 자세한 내용은 HealthOmics 서비스 할당량 단원을 참조하십시오.

실행을 시작하면 HealthOmics는 실행 ID와 실행 UID를 실행에 할당합니다. 계정의 실행에는 고유한 실행 IDs. 그러나 HealthOmics는 삭제된 실행 IDs 재사용하므로 실행과 삭제된 실행의 실행 ID가 동일할 수 있습니다. 또한 드물지만 공유 워크플로가 계정의 실행과 동일한 실행 ID를 가질 수 있습니다.

run uuid는 계정 간 실행을 식별하거나 동일한 실행 ID를 가진 계정에서 두 실행을 구별하는 데 사용할 수 있는 글로벌 고유 식별자(guid)입니다.

> **Note**
>
> 데이터 출처를 위해를 사용하여 실행run uuid을 고유하게 식별하는 것이 좋습니다. 또한 run uuid는 내부 랩 정보 관리 시스템(LIMs) 또는 샘플 추적 시스템에 연결하는 데 가장 적합한 식별자입니다.

Amazon Q CLI를 사용하여 실행을 최적화하고 실행 성능을 분석할 수 있습니다. 자세한 내용은 Amazon Q CLI에 대한 프롬프트 예제 및 GitHub의 HealthOmics Agentic 생성형 AI 자습서를 참조하세요.

주제

HealthOmics 워크플로에서 스토리지 유형 실행

실행을 시작하면 HealthOmics는 실행 중에 사용할 워크플로 엔진에 임시 실행 스토리지를 할당합니다. HealthOmics는 임시 실행 스토리지를 파일 시스템으로 제공합니다.

지정된 워크플로 또는 워크플로 실행에 대해 동적 또는 정적 실행 스토리지를 선택할 수 있습니다. 기본적으로 HealthOmics는 DYNAMIC 실행 스토리지를 제공합니다.

> **Note**
>
> 스토리지 사용량을 실행하면 계정에 요금이 발생합니다. 정적 및 동적 실행 스토리지에 대한 요금 정보는 HealthOmics 요금을 참조하세요.

다음 섹션에서는 사용할 실행 스토리지 유형을 결정할 때 고려해야 할 정보를 제공합니다.

동적 실행 스토리지

더 빠른 시작 시간이 필요한 실행, 스토리지 요구 사항을 미리 모르는 실행, 반복 개발 테스트 주기 등 대부분의 실행에는 동적 실행 스토리지를 사용하는 것이 좋습니다.

실행에 필요한 스토리지 또는 처리량을 추정할 필요가 없습니다. HealthOmics는 실행 중 파일 시스템 사용률에 따라 스토리지 크기를 동적으로 늘리거나 줄입니다. 또한 HealthOmics는 워크플로의 요구 사항에 따라 처리량을 동적으로 조정합니다. 파일 시스템의 스토리지 부족 오류로 인해 실행이 실패하지 않습니다.

동적 실행 스토리지는 정적 실행 스토리지보다 더 빠른 프로비저닝/프로비저닝 해제 시간을 제공합니다. 더 빠른 설정은 대부분의 워크플로에서 장점이며 개발/테스트 주기에서도 장점입니다.

실행이 완료되면(성공 경로 또는 실패 경로) getRun API 작업은 storageCapacity 필드에서 실행에 사용된 최대 스토리지를 반환합니다. 로그 omics 그룹에 있는 실행 매니페스트 로그에서도이 정보를 찾을 수 있습니다. 2시간 이내에 완료되는 동적 스토리지 실행의 경우 최대 스토리지 값을 사용하지 못할 수 있습니다.

동적 실행 스토리지의 경우 실행은 NFS 프로토콜을 사용하는 파일 시스템을 프로비저닝합니다. NFS는 CREATE, DELETE 및 RENAME 파일 작업을 멱등성이 없는 것으로 취급하므로 코드가 정상적으로 처리해야 하는 이러한 작업에 대한 경합 조건이 발생할 수 있습니다. 예를 들어 존재하지 않는 파일을 삭제하려고 해도 코드가 실패해서는 안 됩니다. 동적 실행 스토리지를 채택하기 전에 멱등성이 없는 파일 작업에 복원력이 있도록 워크플로 코드를 조정하는 것이 좋습니다. 멱등성이 없는 작업의 안전한 처리를 위한 코드 예제을(를) 참조하세요.

멱등성이 없는 작업의 안전한 처리를 위한 코드 예제

다음 python 예제에서는 파일이 없는 경우 실패하지 않고 파일을 삭제하는 방법을 보여줍니다.

```
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")
```

다음 예제에서는 Bash 쉘을 사용합니다. 파일이 없더라도 안전하게 제거하려면 다음을 사용합니다.

```
rm -f my_file
```

파일을 안전하게 이동(이름 변경)하려면 파일이 현재 디렉터리에 old_name 있는 경우에만 이동 명령을 실행합니다.

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

디렉터리를 생성하려면 다음 명령을 사용합니다.

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

정적 실행 스토리지

정적 실행 스토리지의 경우 실행은 Lustre 프로토콜을 사용하는 파일 시스템을 프로비저닝합니다. 이 프로토콜은 기본적으로 멱등성이 아닌 파일 작업에 복원력이 있습니다. 멱등성이 없는 파일 작업을 처리하기 위해 워크플로 코드를 조정할 필요가 없습니다.

HealthOmics는 고정된 양의 실행 스토리지를 할당합니다. 실행을 시작할 때이 값을 지정합니다. 값을 지정하지 않으면 기본 실행 스토리지는 1200GiB입니다. StartRun API 요청에서 스토리지 크기 값을 지정하면 시스템은 값을 1200GiB의 가장 가까운 배수로 반올림합니다. 해당 스토리지 크기를 사용할 수 없는 경우 2,400GiB의 가장 가까운 배수로 반올림됩니다.

정적 실행 스토리지의 경우 HealthOmics는 다음과 같은 처리량 값을 프로비저닝합니다.

- 프로비저닝된 스토리지 용량의 TiB당 200MB/s의 기준 처리량.
- 프로비저닝된 스토리지 용량의 TiB당 최대 1,300MB/s의 버스트 처리량.

지정된 스토리지 크기가 너무 낮으면 파일 시스템의 스토리지 부족 오류와 함께 실행이 실패합니다. 정적 실행 스토리지는 스토리지 요구 사항이 알려진 예측 가능한 워크플로에 적합합니다.

정적 실행 스토리지는 작업 동시성이 높은 급증하는 대규모 워크로드(예: 병렬로 처리되는 대량의 RNASeq 샘플)에 적합합니다. 동적 실행 스토리지보다 GiB당 파일 시스템 처리량이 높고 GiB당 비용이 낮습니다.

필요한 정적 실행 스토리지 계산

기본 파일 시스템 설치는 정적 파일 시스템 용량의 7%를 사용하기 때문에 정적 실행 스토리지(동적 실행 스토리지와 비교)를 사용할 때 워크플로에 추가 용량이 필요합니다.

동적 실행 스토리지 워크플로를 실행하여 실행에 사용되는 최대 스토리지를 측정하는 경우 다음 계산을 사용하여 필요한 최소 정적 스토리지 양을 결정합니다.

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

예제:

```
    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.
```

따라서이 실행에는 1200GiB(정적 실행 스토리지의 최소 용량)로 충분합니다.

HealthOmics 실행에 대한 보존 모드 실행

실행이 완료되면 HealthOmics는 실행 메타데이터를 CloudWatch에 보관합니다. 기본적으로 CloudWatch는 CloudWatch 보존 정책을 변경하지 않는 한 실행 데이터를 무기한 유지합니다. 실행 출력은 삭제할 때까지 Amazon S3에도 저장됩니다.

조정 가능한 중 하나는 리전HealthOmics 서비스 할당량maximum number of runs (active and inactive)의 입니다. HealthOmics는 콘솔 및 API 작업(ListRuns 및 GetRun)에서 사용할 수 있도록 최대 수의 실행에 대해 실행 메타데이터를 유지합니다. GetRun 실행을 시작할 때 실행의 보존 동작을 나타내도록 실행 보존 모드 파라미터를 설정할 수 있습니다. 파라미터는 REMOVE 및 RETAIN 값을 지원합니다.

보존 모드가 REMOVE로 설정된 새 실행의 경우 HealthOmics가 최대 실행 수를 이미 저장한 후 실행을 추가하려고 하면 제거 모드를 설정한 가장 오래된 실행의 메타데이터가 자동으로 제거됩니다. 이 제거는 CloudWatch 또는 Amazon S3에 저장된 데이터에 영향을 주지 않습니다.

RETAIN은 실행 보존 모드의 기본값입니다. 이 모드에서 실행되는 경우 시스템은 실행 메타데이터를 삭제하지 않습니다. HealthOmics가 모두 RETAIN으로 설정된 최대 실행 수에 도달하면 일부 실행을 삭제할 때까지 추가 실행을 생성할 수 없습니다.

동시에 최대 실행 횟수를 초과하는 배치를 실행하려는 경우 실행 보존 모드를 제거로 설정해야 합니다. 그렇지 않으면 HealthOmics가 최대값 이후에 다음 실행을 시작하려고 하면 배치가 실패합니다.

REMOVE 보존 모드 사용에 대한 추가 고려 사항:

- REMOVE를 보존 모드로 처음 사용하기 시작할 때는 RETAIN 모드를 사용하는 하나 이상의 실행을 삭제하여 슬롯을 확보하는 것이 좋습니다. 추가 REMOVE 실행을 시작하면 자동 제거가 인계되므로 새 실행에 사용할 수 있는 슬롯이 충분합니다.
- 아카이브된 실행(또는 실행 세트)을 다시 실행하려면 HealthOmics CLI 다시 실행 도구를 사용합니다. 이 도구를 사용하는 방법에 대한 자세한 내용과 예제는 HealthOmics 도구 GitHub 리포지토리의 Omics 재실행을 참조하세요. GitHub
- 각 실행에 대해 고유한 이름을 구성하는 것이 좋습니다. HealthOmics가 실행을 제거한 후에는 콘솔 또는 API를 사용하여 실행 이름 또는 실행 ID를 찾을 수 없습니다. 그러나 CloudWatch를 사용하여 실행 이름을 검색할 수 있으므로 고유한 이름을 사용하여 최상의 검색 결과를 얻을 수 있습니다.
- CloudWatch start-query 명령을 사용하여 보관된 실행에 대한 정보를 가져올 수 있습니다. 실행 이름이 고유하지 않으면 쿼리가 여러 매니페스트를 반환할 수 있습니다. 시작 시간 및 종료 시간 파라미터는 검색의 시간 범위를 정의합니다.

```
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 명령은 쿼리 ID를 반환합니다. 쿼리 ID를 get-query-results 명령에 전달하면 쿼리 결과가 반환됩니다.

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

HealthOmics 실행 입력

워크플로 정의가 워크플로 또는 워크플로 작업에 대한 입력 파일을 지정하는 경우 HealthOmics는 워크플로 실행 전용 스크래치 볼륨으로 파일을 스테이징합니다. 이러한 입력 파일은 읽기 전용이므로 작업이 워크플로의 다른 작업에 대한 잠재적 입력을 수정하지 못합니다. 디렉터리 가져오기의 경우 디렉터리도 읽기 전용입니다.

많은 유전체학 애플리케이션은 인덱스 파일이 시퀀스 파일(예: bam 파일의 컴패니언 bai 파일)과 함께 위치한다고 가정합니다. 인덱스 파일을 포함하려면 워크플로 정의에 작업 입력으로 지정합니다.

주제

- 실행 파라미터 크기 관리

- Amazon S3 입력 파라미터 형식
- Amazon S3 입력 아카이브 상태

실행 파라미터 크기 관리

실행을 시작할 때 실행 파라미터 JSON 객체 또는 파일에 실행 입력을 지정합니다. 워크플로에 대해 최대 50KB의 실행 파라미터를 지정할 수 있습니다. 다음 기법을 사용하여이 크기 제약 조건을 유지할 수 있습니다.

- 디렉터리 가져오기 사용

  많은 수의 입력 파일을 지정하려면 각 파일 위치에 대한 파라미터를 지정하는 대신 하나의 파라미터를 모든 파일이 포함된 Amazon S3 위치로 지정합니다. 자세한 내용은 다음 주제(Amazon S3 입력 파라미터 형식)를 참조하세요.
- 샘플 시트 사용

  샘플 시트는 fastq.gz 주소용 열 1개(또는 쌍 읽기용 열 2개)와 샘플 이름과 같은 메타데이터용 추가 열이 있는 CSV 또는 TSV 파일입니다. 샘플 시트를 각 입력 파일의 파라미터 대신 실행 입력 파라미터로 지정합니다.

  워크플로는 샘플 시트가 워크플로의 데이터 구조에 매핑되는 방법을 정의합니다. WDL 및 CWL의 샘플 시트에 대한 코드를 작성할 수 있지만 NextFlow에서는 더 일반적입니다. 예제는 nf-core GitHub 사이트의 샘플 시트를 참조하세요.

Amazon S3 입력 파라미터 형식

Amazon S3 위치를 수락하는 입력 파라미터의 경우 파라미터는 파일 하나 또는 전체 파일 디렉터리의 위치를 지정할 수 있습니다. 디렉터리를 사용하면 다음과 같은 이점이 있습니다.

- 편의성 - 디렉터리 이름을 파라미터로 지정합니다. 각 파일 이름은 나열하지 않습니다.
- 압축 - 입력 파라미터 최대 파일 크기는 50KB입니다. 입력 파일 이름의 긴 목록을 제공하는 경우이 최대값을 초과할 수 있습니다.

Amazon S3는 플랫 객체 스토리지 시스템이므로 디렉터리를 지원하지 않습니다. 각 파일에 동일한 객체 키 접두사를 지정하여 파일을 "디렉터리"로 그룹화합니다. Amazon S3 객체 키 접두사에 대한 자세한 내용은 접두사를 사용하여 객체 구성을 참조하세요.

HealthOmics는 다음과 같이 입력 파라미터 값을 해석합니다.

- Amazon S3 위치가 슬래시로 끝나지 않거나 glob 패턴을 사용하지 않는 경우 HealthOmics는 파라미터 값이 하나의 Amazon S3 객체에 대한 키가 될 것으로 예상합니다.

  예를 들어 file1.fastq를 입력`s3://myfiles/runs/inputs/a/file1.fastq`하도록 지정합니다.

- Amazon S3 위치가 슬래시로 끝나는 경우 HealthOmics는 파라미터 값을 Amazon S3 접두사로 해석합니다. 해당 접두사가 있는 모든 Amazon S3 객체를 로드합니다.

  예를 들어 키가이 접두사로 시작하는 모든 객체를 로드`s3://myfiles/runs/inputs/a/`하도록를 지정할 수 있습니다.

- Nextflow의 경우 HealthOmics는 입력 파라미터에서 Amazon S3 URIs에 대한 glob 패턴을 병렬로 지원합니다.

  예를 들어 키가이 접두사로 시작하는 모든 .gz 파일을 입력`"s3://myfiles/runs/inputs/a/*.gz"`하도록를 지정할 수 있습니다.

Amazon S3 입력에서 Glob 패턴의 Nextflow 처리

글롭 패턴	HealthOmics 일치 동작	참고
s3://bucket/directory/*.txt	접두사 s3://bucket/directory/ 아래의 깊이에 있는 모든 `.txt` 객체와 일치합니다. 예를 들어는 s3://bucket/directory/abc.txt 또는 s3://bucket/directory/subDir/123.txt 등과 일치합니다.	
s3://bucket/directory/**/*.txt	접두사 s3://bucket/directory/ 아래의 깊이에 있는 모든 `.txt` 객체와 일치합니다. 예를 들어는 s3://bucket/directory/abc.txt 또는 s3://bucket/directory/subDir/123.txt 등과 일치합니다.	S3에서 `**`는와 동일합니다`*`.
s3://bucket/directory/{a,b}.txt	s3://bucket/directory/a.txt, s3://bucket/directory/b.txt	

글롭 패턴	HealthOmics 일치 동작	참고
s3://bucket/directory/?.txt	파일 이름이 단일 문자이고 뒤에가 오는 접두사 루트의 객체와 일치합니다`.txt`. 예를 들어 s3://bucket/directory/a.txt와 일치하지만 s3://bucket/directory/someDir/a.txt 또는 s3://bucket/directory/someDir/subDir/a.txt와는 일치하지 않습니다.	
s3://bucket/directory/[0-9].txt	s3://bucket/directory/0.txt, s3://bucket/directory/1.txt, ... ,s3://bucket/directory/9.txt	
s3://bucket/directory/[0-9].txt	s3://bucket/directory/1.txt, s3://bucket/directory/2.txt, s3://bucket/directory/3.txt	
s3://bucket/directory/[0-9].txt	s3://bucket/directory/b.txt, s3://bucket/directory/c.txt, ... ,s3://bucket/directory/Y.txt	

Amazon S3 입력에서 이중 슬래시의 언어별 처리

HealthOmics는 Amazon S3 URIs에서 이중 슬래시를 처리할 때 각 워크플로 엔진에 대한 기본 엔진 동작을 유지하므로 워크플로를 HealthOmics로 마이그레이션할 때 워크플로를 변경할 필요가 없습니다. 다음 섹션에서는 각 엔진이 다양한 시나리오를 처리하는 방법을 설명합니다.

WDL

입력 파라미터에 URI 중간 또는 끝에 이중 슬래시가 포함된 경우 WDL 엔진은 이중 슬래시를 유지합니다.

입력 파라미터	예상 위치	
s3://myfiles/runs/inputs//file1.fastq	s3://myfiles/runs/inputs//file1.fastq	

입력 파라미터	예상 위치	
s3://myfiles/runs/inputs//	s3://myfiles/runs/inputs//	

다음 흐름

입력 파라미터에 URI 중간에 이중 슬래시가 포함된 경우 Nextflow 엔진은 이중 슬래시를 유지합니다. URI 끝에 있는 이중 슬래시의 경우 Nextflow 엔진은 이를 단일 슬래시로 해석합니다.

입력 파라미터	예상 위치	
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

입력 파라미터에 URI 중간 또는 끝에 이중 슬래시가 포함된 경우 CWL 엔진은 이중 슬래시를 유지합니다.

입력 파라미터	예상 위치	
s3://myfiles//runs/inputs//file1.fastq	s3://myfiles//runs/inputs//file1.fastq	
s3://myfiles//runs/inputs//	s3://myfiles//runs/inputs//	

Amazon S3 입력 아카이브 상태

HealthOmics는 Amazon S3 S3 객체를 검색할 수 있습니다. 다음 아카이브된 스토리지 상태에 있는 객체restore의 경우 HealthOmics에서 사용할 수 있도록 할 객체입니다.

- Amazon S3 Glacier의 Flexible Retrieval 또는 Deep Archive 스토리지 클래스.
- Intelligent Tiering의 Archived Access 또는 Deep Archive Access 계층.

객체 복원에 대한 자세한 내용은 Amazon S3 사용 설명서의 아카이브된 객체 복원을 참조하세요.

HealthOmics 워크플로에서 수명 주기 실행

실행 상태를 모니터링하여 실행 진행 상황을 추적할 수 있습니다. HealthOmics는 실행이 수명 주기를 진행함에 따라 실행 상태를 업데이트합니다.

다음 방법 중 하나를 사용하여 실행 상태를 검색할 수 있습니다.

- HealthOmics 콘솔은 Runs 페이지에 각 실행의 상태를 표시합니다.
- GetRun API 작업은 현재 실행 상태를 반환합니다.
- EventBridge 이벤트를 사용하여 실행 상태를 모니터링할 수 있습니다. 자세한 내용은 에서 EventBridge 사용 AWS HealthOmics 단원을 참조하십시오.

주제

- 실행 상태 값
- 작업 재시도
- 실행 상태의 요금 영향

실행 상태 값

실행을 시작하면 HealthOmics는 실행 상태를 로 설정합니다Pending. 실행이 수명 주기를 진행하면 HealthOmics는 현재 진행 상황을 반영하도록 상태 값을 업데이트합니다.

> **Note**
>
> 실행 중이 아닌 다른 실행 상태에서는 요금이 발생하지 않습니다. 자세한 내용은 다음 단원을 참조하십시오.

HealthOmics는 다음과 같은 실행 상태 값을 지원합니다.

보류중

실행이 대기열에 있으며 시작을 기다리고 있습니다. 실행은 일반적으로 시작하기 전에 짧은 기간 동안 보류 중 상태로 유지됩니다.

- 여러 작업을 동시에 제출하면 실행이 보류 중 상태로 더 오래 유지될 수 있습니다.
- 계정이 최대 동시 실행 수에 도달한 후에도 실행은 보류 중으로 유지됩니다.
- 실행이 리소스 최대값에 도달한 실행 그룹의 일부인 경우 실행은 보류 중으로 유지됩니다.
- 대기 중인 특정 실행이 다른 실행보다 먼저 시작되도록 실행 우선 순위를 조정할 수 있습니다. 실행 우선 순위에 대한 자세한 내용은 섹션을 참조하세요실행 우선 순위.

Starting(시작 중)

HealthOmics는 실행을 생성하고 실행에 필요한 리소스(예: 임시 실행 스토리지 및 엔진 노드)를 프로비저닝합니다.

- HealthOmics는 실행 시작 시 임시 실행 스토리지를 프로비저닝하고 실행이 중지 중일 때 실행 스토리지를 프로비저닝 해제합니다.

실행 중

가져오기 프로세스, 각 작업 처리 및 내보내기 프로세스 중에 실행은 실행 중 상태로 유지됩니다.

- HealthOmics는 입력 파일을 임시 실행 스토리지 파일 시스템으로 가져옵니다. 작업에서 워크플로의 다른 작업에 대한 입력을 수정하지 못하도록 입력 파일은 읽기 전용입니다.
- 파일을 내보내는 동안 HealthOmics는 실행 스토리지 파일 시스템에서 S3 위치로 출력 파일을 내보냅니다.
- HealthOmics는 실행 상태가 실행 중일 때 실행 로그와 작업 로그를 CloudWatch에 실시간으로 전달합니다. 자세한 내용은 CloudWatch의 로그그 단원을 참조하십시오.

Stopping

내보내기 프로세스가 완료되면 실행이 중지 중 상태로 전환됩니다.

- HealthOmics는 모든 리소스(실행 스토리지 파일 시스템 및 엔진 노드 포함)의 프로비저닝을 해제합니다.

완료됨

HealthOmics가 리소스 프로비저닝 해제를 완료한 후 실행이 완료됨으로 전환됩니다.

- HealthOmics는 모든 실행 작업을 완료하고 오류 없이 출력 데이터를 내보냈습니다.
- 실행 출력은 지정된 Amazon S3 URI 출력 위치에서 사용할 수 있습니다. WDL 및 CWL의 경우 HealthOmics는에 대한 정보를 제공하는 실행 출력 요약 파일을 생성합니다HealthOmics 실행 출력.
- 최종 실행 매니페스트 로그 및 엔진 로그(해당하는 경우)는 CloudWatch에서 사용할 수 있습니다.

- 작업 재시도를 지원하는 실행의 경우 완료됨 상태의 실행에는 실패한 작업이 하나 이상 포함될 수 있습니다. 실패한 각 작업에 대해 작업 재시도가 성공한 한 HealthOmics는 실행을 완료됨으로 전환합니다. HealthOmics는 각 재시도에 새 작업 ID를 할당하므로 실행에는 실패한 시도와 완료된 시도에 대한 작업 IDs가 포함됩니다.

Failed

HealthOmics에서 하나 이상의 오류가 발생하여 모든 실행 작업을 완료하지 못했습니다.

- HealthOmics가 리소스 프로비저닝을 해제하는 동안 실패한 실행은 중지 중 상태로 전환됩니다.

취소됨

사용자가 실행 취소 요청을 시작했습니다.

- HealthOmics는 실행 중인 모든 작업을 중지하고 모든 리소스를 프로비저닝 해제합니다.
- HealthOmics는 사용자가 실행을 취소할 때 실행 출력 데이터를 내보내지 않습니다. 취소된 실행의 중간 파일에 액세스할 수 없습니다.
- 취소 전에 실행 상태에서 실행이 사용한 작업 및 리소스에 대해 계정에 요금이 부과됩니다.
- 보류 중 또는 시작 상태에서 실행을 취소하는 경우 요금이 부과되지 않습니다.

작업 재시도

HealthOmics는 서비스 오류(5XX HTTP 상태 코드)로 인해 실패한 작업에 대한 작업 재시도를 지원합니다.

실행의 모든 작업이 결국 완료되는 경우 재시도가 필요하더라도 HealthOmics는 실행을 완료됨으로 전환합니다. HealthOmics는 각 재시도에 새 작업 ID를 할당하므로 실행에는 실패한 시도와 완료된 시도에 대한 작업 IDs가 포함됩니다.

기본 재시도 동작은 워크플로에서 사용하는 정의 언어에 따라 달라집니다. Nextflow의 기본값은 재시도 없음입니다. WDL 및 CWL의 경우 HealthOmics는 실패한 작업을 최대 두 번 재시도하지만 특정 작업 또는 워크플로의 모든 작업에 대한 작업 재시도를 옵트아웃할 수 있습니다. 작업 재시도는 간헐적인 서비스 오류를 해결하는 데 유용합니다. 그러나 멱등성이 있는 작업을 옵트아웃하는 것을 고려할 수 있습니다.

각 워크플로 정의 언어에 대한 자세한 내용은 다음 주제를 참조하세요.

- WDL - 워크플로 정의에서 작업 재시도 동작을 구성합니다. WDL 작업 재시도 동작 구성을 참조하세요.

- Nextflow - Nextflow 구성 파일 또는 워크플로 정의에서 작업 재시도 동작을 구성합니다. Nextflow 작업 재시도 동작 구성을 참조하세요.
- CWL - 워크플로 정의에서 작업 재시도 동작을 구성합니다. CWL 작업 재시도 동작 구성을 참조하세요.

실행 상태의 요금 영향

실행 상태가 실행 중일 때 계정에 요금이 발생할 수 있습니다. 다른 실행 상태에서는 요금이 발생하지 않습니다. 예를 들어 실행이 시작 중 또는 중지 중일 때는 리소스에 대한 요금이 부과되지 않습니다.

실행 중 상태의 실행에는 다음과 같은 결제 영향이 있습니다.

- 실행 상태가 실행 중일 때 계정에 실행 스토리지 파일 시스템 사용에 대한 요금이 발생합니다. 실행 스토리지 유형에 대한 자세한 내용은 섹션을 참조하세요HealthOmics 워크플로에서 스토리지 유형 실행.
- 계정에는 워크플로 정의의 각 작업에 대해 지정한 컴퓨팅 및 메모리 리소스와 작업 기간에 따라 실행 중인 작업에 대한 요금이 부과됩니다. 자세한 내용은 HealthOmics 작업에 대한 컴퓨팅 및 메모리 요구 사항 단원을 참조하십시오.
- 각 작업의 최소 결제 임계값은 1분입니다. 작업을 1분 미만으로 실행하면 최소 1분 사용량에 대한 요금이 발생합니다. 가능하면 작은 작업을 그룹화하여 비용을 최적화합니다. 또한 태스크를 그룹화하면 여러 순차적 태스크의 가동을 방지하여 실행 시간이 단축됩니다.

HealthOmics 요금에 대한 자세한 내용은 HealthOmics 요금을 참조하세요.

HealthOmics 실행 출력

WDL 또는 CWL 실행이 완료되면 출력에는 실행에서 생성된 모든 출력을 나열하는 출력 요약 파일(JSON 형식)이 포함됩니다. 다음과 같은 목적으로 출력 요약 파일을 사용할 수 있습니다.

- 실행이 생성한 출력 파일을 프로그래밍 방식으로 결정합니다.
- 실행이 모든 예상 출력을 생성했는지 확인합니다.

주제

- WDL에 대한 출력 요약 실행
- CWL에 대한 출력 요약 실행

WDL에 대한 출력 요약 실행

WDL 실행이 완료되면 HealthOmics는 라는 출력 요약 파일을 생성합니다output.json.

워크플로의 각 출력에 대해 파일에 해당 키/값 페어가 있습니다. 키에는 형식의 워크플로 이름과 출력 이름이 포함됩니다`WorkflowName.output_name`. 파일 출력의 경우 값은 파일이 저장되는 S3의 출력 위치를 가리키는 S3 URI입니다. 배열[파일] 출력의 경우 값은 S3 URIs.

다음 예제에서는 라는 워크플로의 output.json 파일을 보여줍니다BWAMappingWorkflow.

```
{
  "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"
  ]
}
```

워크플로가 파일 유형이 아닌 출력(예: 문자열, 정수, 부동 소수점 또는 부울)을 생성하는 경우 필드 값은 JSON 기본값입니다. 예:

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

```
}
```

CWL에 대한 출력 요약 실행

CWL 실행이 완료되면 HealthOmics는 다음 위치에 라는 출력 요약 파일을 생성합니다outputs.json.

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

출력 요약 파일에는 출력 목록이 포함됩니다. 각 출력은 키/값 페어이며, 여기서 키는 출력의 이름입니다. 값은 다음 속성을 포함하는 객체입니다.

- 위치 - 출력 파일의 정규화된 경로입니다.
- basename - 경로의 파일 이름 부분입니다.
- class - 출력 유형으로, 일반적으로 파일입니다.
- size - 바이트 단위의 파일 크기

다음 예제에서 output.json 파일에는 두 개의 출력 파일 목록이 있습니다.

```
{
  "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
  }
}
```

실행 실패 이유

실행이 실패하면 GetRun API 작업을 사용하여 실패 이유를 검색합니다.

실패 이유를 검토하여 실행이 실패한 이유를 해결할 수 있습니다. 다음 표에는 오류에 대한 설명과 함께 각 실패 이유가 나열되어 있습니다.

실패 이유	오류 설명
ASSUME_ROLE_FAILED	HealthOmics에는 역할을 수임할 권한이 없습니다. 역할에 대한 신뢰 관계에서 HealthOmics 보안 주체를 지정합니다.
CANNOT_START_CONTAINER_ERROR	워크플로 작업을 시작할 수 없음: ##, ID: 이미지를 사용하는 *ID* 컨테이너: ### ##. 이미지가 유효한지 확인하고 다시 시도하세요.
CANNOT_START_CONTAINER_SIZE_ERROR	워크플로 작업을 시작할 수 없음: ##, ID: 이미지를 사용하는 *ID* 컨테이너: ### ##. 이미지 크기가 45GiB(GPU 인스턴스의 경우 95GiB) 미만인지 확인하고 다시 시도하세요.
ECR_PERMISSION_ERROR	HealthOmics에는 이미지 URI에 액세스할 수 있는 권한이 없습니다. Amazon ECR 프라이빗 리포지토리가 존재하고 HealthOmics 서비스 보안 주체에 대한 액세스 권한을 부여했는지 확인합니다.
내보내기_실패	내보내기에 실패했습니다. 출력 버킷이 존재하고 실행 역할에 버킷에 대한 쓰기 권한이 있는지 확인합니다.
FILE_SYSTEM_OUT_OF_SPACE	파일 시스템에 충분한 공간이 없습니다. 파일 시스템 크기를 늘리고 다시 실행합니다.
이미지_검증_실패	이미지 ### ### 확인할 수 없습니다. 문제를 해결하려면 이미지를 가져온 다음 ECR 리포지토리로 다시 푸시합니다.
가져오기_실패	가져오기에 실패했습니다. 입력 파일이 존재하고 실행 역할이 입력에 액세스할 수 있는지 확인합니다.
INACTIVE_OMICS_스토리지_리소스	HealthOmics 스토리지 URI가 ACTIVE 상태가 아닙니다. 읽기 세트를 활성화하고 다시 시도하세요. 읽기 세트 활성화에 대한 자세한 내용은 섹션을 참조하세요HealthOmics에서 읽기 세트 활성화.
INPUT_URI_NOT_FOUND	제공된 URI가 존재하지 않습니다. *uri*. URI 경로가 존재하는지 확인하고 역할이 객체에 액세스할 수 있는지 확인합니다.
인스턴스_예약_실패	워크플로 실행을 완료하기에 인스턴스 용량이 충분하지 않습니다. 기다렸다가 워크플로 실행을 다시 시도합니다.

실패 이유	오류 설명
INVALID_ECR_IMAGE_URI	Amazon ECR 이미지 URI 구조가 유효하지 않습니다. 유효한 URI를 입력하고 다시 시도하세요.
INVALID_TASK_RESOURCE_VALUE	요청된 GPU, CPU 또는 메모리가 사용 가능한 컴퓨팅 용량에 비해 너무 높거나 작업 *ID*의 최소값인 1보다 작습니다.
INVALID_URI_INPUT	URI 구조는 유효한 *uri*가 아닙니다. URI 구조를 확인하고 다시 시도하세요.
MODIFIED_INPUT_RESOURCE	제공된 URI *uri*는 실행이 시작된 후 수정되었습니다. 실행을 다시 시도합니다.
OUT_OF_MEMORY_ERROR	워크플로 작업 *ID*에 메모리가 부족합니다. 워크플로 정의의 메모리 값을 늘리고 실행을 다시 시도합니다.
RUN_TASK_FAILED	작업이 실패하여 실행에 실패했습니다. 작업 실패를 디버깅하려면 GetRunTask API 작업과 Amazon CloudWatch Logs 스트림을 사용합니다.
RUN_TIMED_OUT	*#* 분 후 실행 제한 시간입니다.
SERVICE_ERROR	서비스에서 일시적인 오류가 발생했습니다. 워크플로 실행을 다시 시도합니다.
TASK_TIMED_OUT	*#* 초 후에 작업 *ID*가 시간 초과되었습니다.
UNSUPPORTED_INPUT_SIZE	총 입력 크기가 너무 큽니다. 입력 크기를 줄이고 다시 시도하세요.
WORKFLOW_RUN_FAILED	워크플로 실행에 실패했습니다. CloudWatch Logs 엔진 로그 스트림: *ID*를 검토하여 실패를 디버깅합니다.
워크플로_VER_검증_실패	HealthOmics는 요청된 Nextflow 버전: *##* --를 지원하지 않습니다. 지원되는 최신 버전은 *##*입니다. Nextflow 버전을 지원되는 버전으로 수정하고 다시 시도하세요.

실패 이유	오류 설명
지원되지 않는_GPU_INSTANCE_TYPE	요청된 인스턴스 유형은 *##*에서 지원되지 않습니다. 이 리전에서 지원되는 GPU 인스턴스 유형으로 실행을 재시도합니다. 사용 가능한 인스턴스 유형은 *GPU #### ##*입니다.

응답하지 않는 실행에 대한 지침

새 워크플로를 개발할 때 코드에 문제가 있고 태스크가 프로세스를 제대로 종료하지 못하면 실행 또는 특정 태스크가 "정지" 또는 "중단"될 수 있습니다. 이는 장기간 태스크를 실행하는 것이 정상이므로 문제를 해결하고 포착하기 어려울 수 있습니다. 응답하지 않는 실행을 방지하고 식별하려면 다음 섹션의 권장 모범 사례를 따르세요.

응답하지 않는 실행을 방지하기 위한 모범 사례

- 작업 코드에서 열린 모든 파일을 닫는지 확인합니다. 파일을 너무 많이 열면 워크플로 엔진 내에서 스레드 문제가 발생할 수 있습니다.
- 작업이 종료되면 워크플로 작업에서 생성된 백그라운드 프로세스가 종료되어야 합니다. 그러나 백그라운드 프로세스가 완전히 종료되지 않으면 작업 코드에서 해당 프로세스를 명시적으로 종료해야 합니다.
- 프로세스가 종료하지 않고 반복되지 않는지 확인합니다. 이로 인해 응답하지 않는 실행이 발생할 수 있으며 워크플로 정의 코드를 변경해야 해결할 수 있습니다.
- 작업에 적절한 메모리 및 CPU 할당을 제공합니다. CloudWatch 로그를 분석하거나 워크플로의 성공적으로 완료된 실행분석기 실행에서를 사용하여 최적의 컴퓨팅 할당이 있는지 확인합니다. Run Analyzer `headroom` 파라미터를 사용하여 추가 헤드룸을 포함하여 프로세스에 완료할 충분한 리소스가 있는지 확인합니다. 백그라운드 운영 체제 프로세스를 고려하기 위해 할당된 메모리 및 CPU에 최소 5%의 헤드룸을 포함합니다.
  - 또한 인스턴스에 더 높은 처리량이 필요한 경우 인스턴스 대역폭 크기를 늘립니다. vCPUs(크기 4xl 이하)인 Amazon EC2 인스턴스는 처리량 버스팅이 발생할 수 있습니다. Amazon EC2 인스턴스 처리량에 대한 자세한 내용은 Amazon EC2 사용 가능한 인스턴스 대역폭을 참조하세요.
- 실행에 올바른 파일 시스템 크기를 사용하고 있는지 확인합니다. 정적 실행 스토리지를 사용하는 응답하지 않는 실행의 경우 정적 실행 스토리지 할당을 늘려 파일 시스템에서 더 높은 IO 처리량 및 스토리지 용량을 활성화하는 것이 좋습니다. 실행 매니페스트를 분석하여 최대 파일 시스템 스토리지를 확인하고, Run Analyzer를 사용하여 파일 시스템 할당을 늘려야 하는지 확인합니다.

응답하지 않는 실행을 포착하는 모범 사례

- 새 워크플로를 개발할 때 최대 실행 시간 제한이 설정된 실행 그룹을 사용하여 런어웨이 코드를 포착합니다. 예를 들어 실행을 완료하는 데 1시간이 걸리는 경우 2~3시간(또는 사용 사례에 따라 다른 기간) 후에 시간이 초과되는 실행 그룹에 실행을 배치하여 런어웨이 작업을 포착합니다. 또한 처리 시간의 차이를 고려하기 위해 버퍼를 적용합니다.
- 최대 런타임 제한이 서로 다른 일련의 실행 그룹을 설정합니다. 예를 들어 몇 시간 후에 실행을 종료하는 실행 그룹에 짧은 실행을 할당하고 예상 워크플로 기간에 따라 며칠 후에 종료되는 긴 실행 그룹을 할당할 수 있습니다.
- HealthOmics의 기본 최대 실행 기간 서비스 제한은 604,800초 또는 7일로, 할당량 도구의 요청을 통해 조정할 수 있습니다. 1주일에 가까운 실행이 있는 경우에만이 할당량의 서비스 한도 증가를 요청합니다. 단기 실행과 장기 실행이 혼합되어 있고 실행 그룹을 사용하지 않는 경우 최대 실행 기간 서비스 한도가 더 높은 별도의 계정에 장기 실행을 배치하는 것이 좋습니다.
- 응답하지 않을 수 있다고 의심되는 작업이 있는지 CloudWatch 로그를 검사합니다. 태스크가 일반적으로 일반 로그 문을 출력하고 장기간 출력하지 않은 경우 태스크가 중단되거나 고정될 수 있습니다.

응답하지 않는 실행이 발생할 경우 수행할 작업

- 추가 비용이 발생하지 않도록 실행을 취소합니다.
- 작업 로그를 검사하여 프로세스가 올바르게 종료되지 않았는지 확인합니다.
- 엔진 로그를 검사하여 비정상적인 엔진 동작을 식별합니다.
- 응답하지 않는 실행의 작업 및 엔진 로그를 성공적으로 완료된 동일한 실행의 작업 및 엔진 로그와 비교합니다. 이렇게 하면 응답하지 않는 동작을 일으켰을 수 있는 차이를 식별하는 데 도움이 될 수 있습니다.
- 근본 원인을 확인할 수 없는 경우 지원 사례를 제기하고 다음을 포함합니다.
  - 중단된 실행의 ARN과 성공적으로 완료된 동일한 실행의 ARN입니다.
  - 엔진 로그(실행이 취소되거나 실패하면 사용 가능)
  - 응답하지 않는 작업에 대한 작업 로그입니다. 문제를 해결하기 위해 워크플로의 모든 작업에 작업 로그가 필요하지는 않습니다.

HealthOmics 실행의 작업 수명 주기

작업은 실행 내의 단일 프로세스입니다. HealthOmics는 워크플로의 각 작업을 작업의 필수 리소스에 가장 적합한 omics 컴퓨팅 인스턴스 유형에 매핑합니다. 워크플로 정의에서 필요한 리소스를 지정합니다. 자세한 내용은 단원을 참조하십시오HealthOmics 작업에 대한 컴퓨팅 및 메모리 요구 사항.

HealthOmics는 작업에서 사용할 임시 실행 스토리지를 제공합니다. HealthOmics는 작업 입력 파일을 임시 실행 스토리지에 읽기 전용 파일로 복사합니다. HealthOmics는 태스크가 작업 디렉터리의 입력 파일에 액세스할 수 있도록 심볼 링크를 제공합니다. 작업은 워크플로 정의 파일에서 선언한 파일에만 액세스할 수 있습니다.

작업 상태 값

작업 상태를 모니터링하여 작업 진행 상황을 추적할 수 있습니다. 실행을 시작하면 HealthOmics는 실행의 각 작업에 Pending 대해 작업 상태를 로 설정합니다. 작업이 시작되고 수명 주기 동안 진행되면 HealthOmics는 현재 진행 상황을 반영하도록 상태 값을 업데이트합니다.

다음 방법 중 하나를 사용하여 작업 상태를 검색할 수 있습니다.

- HealthOmics 콘솔은 Run details 페이지에 실행 중인 각 작업의 상태를 표시합니다.
- GetRunTask API 작업은 작업 상태를 반환합니다.
- EventBridge 이벤트를 사용하여 작업 상태를 모니터링할 수 있습니다. 자세한 내용은 에서 EventBridge 사용 AWS HealthOmics 단원을 참조하십시오.

GetRunTask API 작업을 사용하여 작업의 현재 상태를 검색할 수 있습니다. HealthOmics 콘솔은 Run details 페이지에 실행 중인 각 작업의 상태를 표시합니다.

HealthOmics는 다음 작업 상태 값을 지원합니다.

보류중

작업이 대기열에 있으며 시작을 기다리고 있습니다. 작업은 시작하기 전에 잠시 보류 상태로 유지됩니다.

- 계정이 최대 동시 작업 수에 도달한 후에도 작업은 보류 상태로 유지됩니다.
- 실행이 리소스 최대값에 도달한 실행 그룹의 일부인 경우 작업은 보류 상태로 유지됩니다.
- 대기 중인 특정 실행과 해당 작업이 다른 대기 중인 실행 전에 시작되도록 실행 우선 순위를 조정할 수 있습니다. 실행 우선 순위에 대한 자세한 내용은 섹션을 참조하세요. 실행 우선 순위

Starting(시작 중)

HealthOmics는 작업을 생성하고 워크플로 작업 노드와 같은 작업에 필요한 리소스를 프로비저닝합니다.

실행 중

HealthOmics가 작업을 처리하는 동안 작업 상태는 실행 중입니다.

Stopping

작업 처리를 완료하고 출력 데이터를 내보낸 후 작업이 중지 중으로 전환됩니다.

- HealthOmics는 워크플로 작업 노드의 프로비저닝을 해제합니다.

완료됨

HealthOmics는 작업 처리를 완료하고 출력 데이터를 실행 스토리지 파일 시스템으로 전송했습니다.

Failed

HealthOmics에서 작업을 처리하는 동안 오류가 발생하여 완료하지 못했습니다.

- 작업은 중지 중 상태(HealthOmics가 리소스 프로비저닝 해제)로 전환된 다음 실패 상태로 전환됩니다.
- 오류가 서비스 오류(5XX HTTP 상태 코드)이고 워크플로가이 작업에 대한 재시도를 지원하는 경우 HealthOmics는 작업을 다시 처리하려고 시도합니다. HealthOmics는 재시도에 새 작업 ID를 할당합니다.

취소됨

HealthOmics는 사용자가 실행을 취소하도록 요청한 후 작업을 중지합니다.

- 작업은 중지 중 상태(HealthOmics가 리소스 프로비저닝 해제)로 전환된 다음 취소됨 상태로 전환됩니다.

워크플로 작업 문제 해결

다음은 작업 문제 해결을 위한 모범 사례 및 고려 사항입니다.

- 작업 로그는 작업에서 STDERR 생성되는 STDOUT 및에 의존합니다. 작업에 사용되는 애플리케이션이 둘 중 하나를 생성하지 않으면 작업 로그가 없습니다. 디버깅을 지원하려면 verbose 모드에서 애플리케이션을 사용합니다.
- 작업에서 실행 중인 명령을 보간된 값과 함께 보려면 `set -x` Bash 명령을 사용합니다. 이렇게 하면 작업이 올바른 입력을 사용하고 있는지 확인하고 오류로 인해 작업이 의도한 대로 실행되지 않았을 수 있는 위치를 식별할 수 있습니다.
- echo 명령을 사용하여 변수 값을 STDOUT 또는에 출력합니다STDERR. 이렇게 하면 예상대로 설정되어 있는지 확인할 수 있습니다.
- 와 같은 명령을 사용하여 입력이 존재하고 예상 크기인지 `ls -l <name_of_input_file>` 확인합니다. 그렇지 않으면 버그로 인해 빈 출력을 생성하는 이전 작업의 문제가 발견될 수 있습니다.

- 작업 스크립트`df -Ph . | awk 'NR==2 {print $4}'`에서 명령을 사용하여 현재 작업에 사용할 수 있는 공간을 결정하고 추가 스토리지 할당으로 워크플로를 실행해야 하는 상황을 식별할 수 있습니다.

작업 스크립트에 위의 명령 중 하나를 포함하면 작업 컨테이너에도 이러한 명령이 포함되어 있고 컨테이너 환경`path`의에 있다고 가정합니다.

프라이빗 HealthOmics 워크플로에 대한 최적화 실행

총 비용, 총 실행 시간 또는 둘 다에 대해 실행을 최적화할 수 있습니다. HealthOmics는 최적화 결정을 실행하는 데 도움이 되는 데이터와 도구를 제공합니다. 실행 최적화는 Ready2Run 워크플로에는 적용되지 않습니다. 서비스가 이러한 워크플로에 대한 리소스 프로비저닝을 관리하는 방법을 제어할 수 없기 때문입니다.

첫 번째 단계는 실행 중인 작업에 대한 현재 작업 리소스 사용량과 비용을 파악한 다음 실행 비용 및 성능을 최적화하는 방법을 적용하는 것입니다.

주제

- 분석기 실행
- 실행 비용 결정
- 런타임 사용량 결정
- 실행을 최적화하는 방법
- 실행 간 파일 크기 차이의 영향
- 리소스 동시성을 최적화하는 방법

분석기 실행

HealthOmics는 Run Analyzer라는 오픈 소스 도구를 제공합니다. 이 도구는 실행에 대한 작업 수준 리소스 사용 정보를 추출하고 비용 및 실행 성능에 대한 최적화 기회를 제안합니다.

> **Note**
>
> 실행 분석기는 도구를 실행할 때 AWS 정가를 기준으로 작업 비용과 잠재적 비용 절감을 추정합니다. 최적화 권장 사항을 평가하고 사용 사례에 적합한 권장 사항을 구현합니다. 채택한 최적화를 테스트하여 실행에 적합한지 확인합니다.

Run Analyzer는 다음 작업을 수행합니다.

- 메모리 및 컴퓨팅 병목 현상을 평가합니다.
- 메모리 또는 CPU용으로 과다 프로비저닝된 작업을 식별하고 비용을 절감할 수 있는 새 인스턴스 크기를 권장합니다.
- 개별 작업에 대한 예상 비용을 계산하고 권장 사항을 적용할 경우 잠재적 비용 절감을 계산합니다.
- 작업 종속성과 처리 순서를 확인할 수 있도록 작업에 대한 타임라인 보기를 제공합니다. 타임라인은 장기 실행 작업을 식별하는 데도 도움이 됩니다.
- 실행 스토리지의 파일 시스템 크기에 대한 권장 사항을 제공합니다.
- 대용량 컨테이너 로드로 인해 프로비저닝 시간이 느려질 수 있는 영역을 식별할 수 있도록 작업 프로비저닝 시간을 표시합니다.
- 이 도구에는 최적화 권장 사항의 공격성을 제어하는 데 사용할 수 있는 입력 파라미터(헤드룸)가 포함되어 있습니다.

다음 섹션에는 Run Analyzer를 사용하여 실행을 최적화하기 위한 특정 제안 사항이 포함되어 있습니다.

실행 비용 결정

다음 방법 및 지침을 사용하여 실행 비용을 결정할 수 있습니다.

- 결제 기간의 총 실행 비용을 보려면 다음 단계를 따르세요.

  1. Billing and Cost Management 콘솔을 열고 청구서를 선택합니다.
  2. 서비스별 요금에서 Omics를 확장합니다.
  3. 리전을 확장한 다음 omics 인스턴스 유형, 실행 스토리지 유형 및 Ready2Run 워크플로별로 항목화된 모든 실행의 비용을 확인합니다.

- 각 실행에 대한 정보가 포함된 비용 보고서를 생성하려면 다음 단계를 따릅니다.

  1. Billing and Cost Management 콘솔을 열고 데이터 내보내기를 선택합니다.
  2. 생성을 선택하여 새 데이터 내보내기를 생성합니다.
  3. 데이터 내보내기의 내보내기 이름을 입력합니다. 다른 필드를 기본값으로 유지하여 CUR(비용 및 사용량) 보고서를 생성합니다.
  4. 시간 세분화에서 시간별 또는 일별을 선택합니다.
  5. 데이터 내보내기 스토리지 설정에서 다음 구성 단계를 수행합니다.

a. 데이터 내보내기를 위해 Amazon S3 버킷을 구성합니다.

b. 파일 버전 관리에서 기존 내보내기 파일을 덮어쓸지 아니면 매번 새 파일을 생성할지 선택합니다.

시스템은 향후 24시간 이내에 첫 번째 보고서를 생성하고 하루에 한 번 후속 보고서를 생성합니다.

6. 데이터 내보내기를 생성하는 방법에 대한 자세한 내용은 데이터 내보내기 사용 설명서AWS 의 데이터 내보내기 생성을 참조하세요.

- 실행에 태그를 지정하여 팀 또는 프로젝트와 같은 범주별로 비용을 모니터링하고 최적화할 수 있습니다. 태그를 사용하는 경우 다음 단계에 따라 태그 범주별로 실행 비용을 확인합니다.

  1. Billing and Cost Management 콘솔을 열고 Cost Explorer를 선택합니다.
  2. 보고서 파라미터 > 그룹화 기준에서 차원으로 태그를 선택하고 원하는 태그 이름을 선택합니다.

- 작업에 대한 리소스 사용량을 보려면 CloudWatch에서 실행 매니페스트 로그를 확인합니다. 자세한 내용은 CloudWatch Logs를 사용하여 HealthOmics 모니터링 단원을 참조하십시오.
- 분석기 실행 도구를 사용하여 실행에 대한 작업 리소스 사용 정보를 추출합니다.

런타임 사용량 결정

다음 방법을 사용하여 런타임 사용량을 조사할 수 있습니다.

- 콘솔의 실행 페이지에서 실행의 총 실행 시간을 볼 수 있습니다.
- 실행 세부 정보 페이지에서 다음 항목을 볼 수 있습니다.
  - 실행의 총 실행 시간을 봅니다.
  - 실행 중인 각 작업의 실행 시간을 확인합니다.
  - 링크 중 하나를 선택하여 Amazon S3에서 로그를 보거나 CloudWatch에서 실행 로그를 보거나 매니페스트 로그를 실행합니다.
- 작업 실행 목록에서 작업에 대한 로그 보기 링크를 선택하여 CloudWatch에서 작업 로그를 봅니다.
- `listRuns` API 작업에 대한 응답에는 실행 시작 시간과 중지 시간이 포함되므로 총 실행 시간을 계산할 수 있습니다.
- 이 분석기 실행 도구는 타임라인 보기에 작업 기간을 표시합니다. 이 도구는 예상 순서와 일치시킬 수 있는 작업 처리 시퀀스의 시각적 표현을 제공합니다.

실행을 최적화하는 방법

HealthOmics는 데이터 스테이징(예: 데이터 가져오기 및 데이터 내보내기)을 수행하는 리소스를 자동으로 프로비저닝, 관리 및 최적화합니다. HealthOmics는 워크플로에 대한 워크플로 엔진도 시작하고 실행합니다. 그러나 다양한 실행 구성을 설정하여 실행 시작 시간, 작업 시작 시간 및 전체 작업 실행 시간에 영향을 미칠 수 있습니다. 워크플로 정의 및 설계에 대한 전반적인 접근 방식은 작업 실행 시간에도 영향을 미칩니다. 다음 목록은 실행 및 작업 성능에 영향을 미칠 수 있는 요소를 설명합니다.

스토리지 유형 실행

실행 스토리지 유형은 실행 성능 및 실행 프로비저닝 시간에 영향을 미칩니다. 동적 실행 스토리지는 실행 스토리지 요구 사항에 따라 동적으로 확장되므로 더 빠르게 프로비저닝되고 메모리가 부족해지지 않습니다. 동적 실행 스토리지는 문제를 해결하기 위해 워크플로를 시작하고 중지할 수 있는 개발 중인 워크플로에도 적합합니다.

정적 실행 스토리지는 더 긴 파일 시스템 프로비저닝 시간이 필요하지만 일반적으로 실행의 작업 동시성이 높거나 9.6TiB 이상의 파일 시스템 용량이 필요한 경우 일부 실행을 더 빠르게 완료할 수 있습니다. 정적 실행 스토리지는 I/O 요구 사항이 높은 장기 실행 워크플로에 적합합니다.

지정된 실행에 대한 각 실행 스토리지 유형의 비용 대비 성능을 평가하는 데 도움이 되도록 A/B 테스트를 시도하여 어떤 실행 스토리지 유형이 더 나은 성능을 제공하는지 확인할 수 있습니다. 또한 개발 주기에 동적 실행 스토리지를 사용하는 것을 고려한 다음 대규모 프로덕션 실행에 정적 실행 스토리지를 사용하세요.

실행 스토리지 유형에 대한 자세한 내용은 HealthOmics 워크플로에서 스토리지 유형 실행

오버프로비저닝 실행 정적 스토리지

워크플로 작업 계산이 I/O에 의해 제한되는 경우 정적 실행 스토리지를 오버프로비저닝하는 것이 좋습니다. 스토리지 비용은 크기에 따라 증가하지만 파일 시스템의 최대 처리량도 증가합니다. 비용이 많이 드는 컴퓨팅 작업에 I/O 병목 현상이 발생하는 경우 파일 시스템 크기를 늘려 작업 실행 시간을 줄이면 전체 비용이 절감될 수 있습니다.

컨테이너 이미지 크기 축소

각 작업이 시작되면 HealthOmics는 작업에 지정한 컨테이너를 로드합니다. 컨테이너가 클수록 로드하는 데 시간이 더 오래 걸립니다. 컨테이너를 최대한 작게 최적화하여 새 작업을 시작하는 효율성을 개선합니다. 컨테이너에 대용량 데이터 세트를 추가하는 경우 데이터 세트를 S3에 저장하고 워크플로가 S3에서 데이터를 가져오도록 하는 것이 좋습니다. HealthOmics가 지원하는 최대 컨테이너 크기는 섹션을 참조하세요HealthOmics 워크플로 고정 크기 할당량.

태스크 크기

작은 순차 태스크를 단일 태스크로 결합하여 태스크 프로비저닝 시간을 절약할 수 있습니다. 또한 HealthOmics에는 최소 1분의 작업 기간 요금이 부과되므로 작업을 결합하면 비용이 절감될 수 있습니다. 결합된 작업 내에서 Unix 파이프를 사용하여 파일 직렬화 및 역직렬화의 I/O 비용을 방지할 수 있습니다.

파일 압축

워크플로 중간 파일을 과도하게 압축하지 마세요. 대부분의 유전체학 형식은 “gzip” 또는 “block gzip” 압축을 사용합니다. 작업 입력 파일을 압축 해제하고 작업 출력 파일을 다시 압축하면 전체 작업 CPU 사용량의 많은 비율을 소비할 수 있습니다. 일부 유전체 애플리케이션에서는 출력을 직렬화할 때 압축 수준을 설정할 수 있습니다. 압축 수준을 줄이면 CPU 시간을 줄일 수 있지만 파일이 클수록 디스크에 쓰는 데 걸리는 시간이 늘어납니다. 작업 및 애플리케이션에 따라 실행 시간이 가장 짧은 중간 파일에 대한 최적의 압축 수준을 찾을 수 있습니다. 먼저 출력 파일이 가장 큰 태스크를 대상으로 지정하는 것이 좋습니다. 압축 수준이 2이면 여러 시나리오에 적합합니다. 사용 사례에 대해이 수준으로 시작하고 다른 압축 수준을 시도하여 결과를 비교할 수 있습니다.

스레드 수

작업 정의에서 스레드를 지정하는 경우 스레드 수를 요청된 vCPUs 수와 동일한 값으로 설정합니다.

컴퓨팅 및 메모리 지정

작업에서 메모리 또는 컴퓨팅 리소스를 지정하지 않으면 HealthOmics는 가장 작은 인스턴스 유형(`omics.c.large`)을 기본 로 할당합니다. HealthOmics에서 더 큰 인스턴스 유형을 할당하도록 하려면 메모리 및 컴퓨팅 요구 사항을 명시적으로 선언합니다.

HealthOmics는 사용자가 요청하는 vCPUs, 메모리 및 GPU 리소스 수를 할당합니다. 예를 들어 15vCPUs 및 33GiB를 요청하면 HealthOmics는 작업에 omics.m.4xl 인스턴스(16vCPUs, 64GB)를 할당하지만 작업은 15개의 vCPUs와 33GiB만 사용할 수 있습니다. 따라서 omics 인스턴스와 일치하는 vCPUs 및 메모리 리소스를 요청하는 것이 좋습니다.

여러 샘플을 한 번의 실행으로 일괄 처리

파일 시스템 프로비저닝은 실행 시작 시 시간이 걸리므로 여러 샘플을 동일한 실행으로 일괄 처리하여 프로비저닝 시간을 절약할 수 있습니다. 이 접근 방식을 결정하기 전에 다음 요소를 고려하세요.

- 하나의 잘못된 샘플로 인해 워크플로가 실패할 수 있으므로 샘플을 일괄 처리하면 실패한 워크플로 수가 증가할 수 있습니다. 워크플로가 대부분 성공할 것이라고 확신하지 못하는 경우 샘플당 하나의 실행이 더 나은 접근 방식이 될 수 있습니다.

- HealthOmics는 전체 워크플로에 하나의 실행 스토리지 파일 시스템을 할당합니다. 샘플 배치의 경우 모든 샘플을 처리하기에 충분한 양의 실행 스토리지를 지정해야 합니다.
- 워크플로당 최대 실행 스토리지 양이 있으므로가 배치에 추가할 수 있는 샘플 수를 제한할 수 있습니다.
- 최소 실행 스토리지 크기는 1.2TiB이므로 워크플로에서 각 샘플의 최소 스토리지보다 훨씬 적은 스토리지를 사용하는 경우 일괄 처리로 인해 비용이 절감될 수 있습니다.
- 실행 스토리지는 여러 동시 연결을 처리할 수 있으므로 동일한 실행 스토리지를 사용하는 여러 작업이 있더라도 I/O 병목 현상이 발생하지 않습니다.
- 각 실행에는 고유한 태그 세트가 있습니다. 예산 책정 또는 추적을 위한 정보로 워크플로에 태그를 지정하는 경우 별도의 실행을 사용하는 것이 더 좋을 수 있습니다.
- IAM 역할은 전체 실행에 적용됩니다. 각 사용자는 샘플 배치의 모든 데이터에 액세스할 수 있습니다. 워크플로를 분리하면 더 세분화된 권한을 사용할 수 있습니다.
- HealthOmics는 최대 동시 워크플로 수와 워크플로의 최대 동시 작업 수에 대한 계정 수준 할당량을 설정합니다. 이러한 할당량 증가를 요청하는 방법에 대한 자세한 내용은 섹션을 참조하세요HealthOmics 서비스 할당량.

컨테이너 이미지에 파라미터 사용

워크플로에 URIs 포함하는 대신 컨테이너 이미지를 파라미터화합니다. 실행 파라미터인 경우 HealthOmics는 실행이 시작되기 전에 실행이 컨테이너에 액세스할 수 있는지 확인합니다. 그렇지 않으면 완료된 작업에 대해 요금이 발생했을 때 실행 중에 작업이 실패합니다. 또한 파라미터화된 입력이므로 HealthOmics는 실행 매니페스트에 체크섬을 생성하여 실행 출처를 개선합니다.

린터 사용

새 워크플로를 실행하기 전에 린터를 사용하여 일반적인 워크플로 오류를 찾습니다. 자세한 내용은 HealthOmics의 워크플로 린터 단원을 참조하십시오.

EventBridge를 사용하여 문제에 플래그 지정

EventBridge 사용자 지정 알림을 사용하여 비즈니스 로직과 관련된 이상을 파악합니다.

시퀀스 스토어 사용

소스 데이터에 시퀀스 스토어를 사용하여 스토리지 비용을 절감하는 것이 좋습니다. 자세한 내용은 HealthOmics.

실행 간 파일 크기 차이의 영향

사용자는 종종 작은 테스트 데이터 세트를 사용하여 실행을 설계하고 테스트한 다음 프로덕션 실행에서 파일 크기가 크게 달라지는 다양한 데이터를 접합니다. 실행을 최적화할 때이 차이를 고려해야 합니다.

다음 목록은 파일 크기에 상당한 차이가 있는 최적화를 위한 권장 사항을 설명합니다.

테스트 데이터의 다양한 파일 크기

개발 중에 대표적인 분산량이 있는 테스트 데이터를 사용하세요.

Run Analyzer 사용

다양한 샘플에서 Run Analyzer 도구를 사용하여 데이터 크기의 차이를 고려하세요.

실행 분석기를 사용하여 프로덕션 데이터 샘플의 실행 간 차이를 이해할 수 있습니다. Run Analyzer에서 `--batch` 모드를 사용하여 실행 배치에 대한 통계를 생성하고 데이터 세트에서 이상값을 처리하는 데 필요한 최대 컴퓨팅 리소스를 분석합니다.

예를 들어 실행 분석기에 배치 모드의 전체 데이터 흐름 셀을 제공하여 전체 흐름 셀의 최대 vCPU 및 메모리 사용률을 이해할 수 있습니다.

입력 데이터 세트의 크기 분산 감소

샘플 크기의 차이가 높으면 HealthOmics 업스트림의 샘플을 분기하고 각 배치에 대해 서로 다른 파일 시스템 크기를 선택하여 실행 스토리지 비용을 절감할 수 있습니다.

WDL에서 `size` 함수를 사용하여 대규모 샘플과 소규모 샘플의 개별 작업에 대한 리소스 할당을 분기합니다. 가장 많은 영향을 미치려면 가장 비용이 많이 드는 작업에이 전략을 적용합니다.

Nextflow에서 파일 크기 또는 파일 이름을 기반으로 리소스 할당을 계층화하는 데 조건부 리소스를 사용합니다. 자세한 내용은 Nextflow GitHub 사이트의 조건부 프로세스 리소스를 참조하세요.

너무 빨리 최적화하지 마세요.

상당한 성능 튜닝 노력에 투자하기 전에 워크플로 코드와 로직을 마무리합니다. 코드를 변경하면 필요한 리소스에 상당한 영향을 미칠 수 있습니다. 개발 프로세스에서 실행을 너무 빨리 최적화하는 경우 과도하게 최적화하거나 나중에 워크플로 정의가 변경되면 다시 최적화해야 할 수 있습니다.

Run Analyzer 도구를 주기적으로 다시 실행

시간 경과에 따라 워크플로 정의를 변경하거나 샘플 분산이 변경되는 경우 주기적으로 Run Analyzer 도구를 실행하여 추가 최적화를 수행할 수 있습니다.

리소스 동시성을 최적화하는 방법

HealthOmics는 대규모 실행을 처리할 때 비용을 제어하고 관리하는 데 도움이 되는 다음과 같은 기능을 제공합니다.

- 실행 그룹을 사용하여 비용 및 리소스 사용량을 제어합니다. 동시 실행 수, vCPUs, GPUs 및 작업당 총 실행 시간에 대해 실행 그룹의 최대값을 설정할 수 있습니다. 별도의 팀 또는 그룹이 동일한 계정을 사용하는 경우 각 팀에 대해 별도의 실행 그룹을 생성할 수 있습니다. 실행 그룹 최대값을 구성하여 팀당 리소스 사용량과 비용을 제어할 수 있습니다. 자세한 내용은 HealthOmics 실행 그룹 사용 단원을 참조하십시오.
- 개발 중에 최대값이 더 낮은 별도의 실행 그룹을 구성하여 런어웨이 작업을 포착할 수 있습니다.
- Service Quotas는 과도한 리소스 요청으로부터 계정을 보호하는 데도 도움이 됩니다. 할당량 값 증가를 요청하는 방법을 포함하여 Service Quotas에 대한 자세한 내용은 섹션을 참조하세요. HealthOmics 서비스 할당량

HealthOmics에서 작업 실행

실행을 시작, 재실행, 복제, 취소 또는 삭제할 수 있습니다.

- Start - HealthOmics는 지정한 구성 설정을 사용하여 새 실행을 생성한 다음 실행을 시작합니다.
- Rerun - HealthOmics는 지정한 실행과 중복되는 새 실행을 생성합니다. HealthOmics rerun 도구를 사용하여 삭제된 실행을 다시 실행할 수 있습니다.
- Clone - 콘솔을 사용하여 기존 실행을 복제할 수 있습니다. 콘솔이 실행 복제 페이지를 열고 기존 실행의 값을 사용하여 구성 필드를 미리 채웁니다. 필요에 따라 값을 수정하고 복제된 실행을 시작할 수 있습니다.
- Cancel - 아직 완료되지 않은 실행을 취소할 수 있습니다. 실행을 취소하면 HealthOmics는 실행 출력을 저장하지 않습니다.
- Delete - 완료된 실행을 수동으로 삭제하거나 HealthOmics의 실행 보존 모드를 설정하여 가장 오래된 실행을 자동으로 삭제할 수 있습니다. 보존 모드에 대한 자세한 내용은 섹션을 참조하세요the section called “보존 모드 실행”.

주제

- HealthOmics에서 실행 시작
- HealthOmics에서 실행 다시 실행
- HealthOmics에서 실행 복제

- HealthOmics에서 실행 취소
- HealthOmics에서 실행 삭제

HealthOmics에서 실행 시작

실행을 시작할 때 HealthOmics가 실행 중에 사용하도록 할당하는 리소스를 지정합니다.

실행 스토리지 유형 및 스토리지 양(정적 스토리지의 경우)을 지정합니다. 데이터 격리 및 보안을 보장하기 위해 HealthOmics는 각 실행 시작 시 스토리지를 프로비저닝하고 실행 종료 시 스토리지를 프로비저닝 해제합니다. 자세한 내용은 HealthOmics 워크플로에서 스토리지 유형 실행 섹션을 참조하세요.

출력 파일의 Amazon S3 위치를 지정합니다. 대량의 워크플로를 동시에 실행하는 경우 버킷 제한이 발생하지 않도록 각 워크플로에 대해 별도의 Amazon S3 출력 URIs를 사용합니다. 자세한 내용은 Amazon S3 사용 설명서의 접두사를 사용하여 객체 구성 및 Amazon S3 성능 최적화 백서의 스토리지 연결 수평 조정을 참조하세요.

실행 우선 순위를 지정할 수도 있습니다. 우선 순위가 실행에 미치는 영향은 실행이 실행 그룹과 연결되어 있는지 여부에 따라 달라집니다. 자세한 내용은 실행 우선 순위 섹션을 참조하세요.

워크플로에 하나 이상의 버전이 있는 경우 실행을 시작할 때 버전을 지정할 수 있습니다. 버전을 지정하지 않으면 HealthOmics가 기본 워크플로 버전을 시작합니다.

HealthOmics API를 사용할 때 각 실행에 고유한 요청 ID를 제공할 수 있습니다. 요청 ID는 HealthOmics가 중복 요청을 식별하는 데 사용하는 멱등성 토큰입니다. 및는 한 번만 실행을 시작합니다.

> **Note**
>
> 실행을 시작할 때 IAM 서비스 역할을 지정합니다. 선택적으로 콘솔에서 서비스 역할을 생성할 수 있습니다. 자세한 내용은 에 대한 서비스 역할 AWS HealthOmics 단원을 참조하십시오.

주제

- HealthOmics 실행 파라미터
- 콘솔을 사용하여 실행 시작
- API를 사용하여 실행 시작
- 실행에 대한 정보 가져오기

HealthOmics 실행 파라미터

실행을 시작할 때 실행 파라미터 JSON 파일에 실행 입력을 지정하거나 파라미터 값을 인라인으로 입력할 수 있습니다. 실행 파라미터 JSON 파일의 크기 관리에 대한 자세한 내용은 섹션을 참조하세요실행 파라미터 크기 관리.

HealthOmics는 파라미터 값에 대해 다음과 같은 JSON 유형을 지원합니다.

JSON 유형	예제 키 및 값	참고
부울	"b":true	값은 따옴표가 아니며 모두 소문자입니다.
정수	"i":7	값이 따옴표로 묶여 있지 않습니다.
number	"f":42.3	값이 따옴표로 묶여 있지 않습니다.
문자열	"s":"문자"	값은 따옴표로 묶입니다. 텍스트 값 및 URIs에 문자열 유형을 사용합니다. URI 대상은 예상 입력 유형이어야 합니다.
배열	"a":[1,2,3]	값이 따옴표로 묶여 있지 않습니다. 배열 멤버는 각각 입력 파라미터로 정의된 유형을 가져야 합니다.
객체	"o":{"left":"a", "right":1}	WDL에서 객체는 WDL 페어, 맵 또는 구조체에 매핑됩니다.

콘솔을 사용하여 실행 시작

실행을 시작하려면

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 실행을 선택합니다.
3. 실행 페이지에서 실행 시작을 선택합니다.

4. 실행 세부 정보 패널에서 다음 정보를 제공합니다.

   - 워크플로 소스 - 소유 워크플로 또는 공유 워크플로를 선택합니다.
   - 워크플로 ID -이 실행과 연결된 워크플로 ID입니다.
   - 워크플로 버전(선택 사항) -이 실행에 사용할 워크플로 버전을 선택합니다. 버전을 선택하지 않으면 실행은 워크플로 기본 버전을 사용합니다.
   - 실행 이름 -이 실행의 고유한 이름입니다.
   - 실행 우선 순위(선택 사항) -이 실행의 우선 순위입니다. 숫자가 클수록 우선 순위가 높아지고 우선 순위가 가장 높은 작업이 먼저 실행됩니다.
   - 스토리지 유형 실행 - 워크플로에 지정된 기본 실행 스토리지 유형을 재정의하려면 여기에서 스토리지 유형을 지정합니다. 정적 스토리지는 실행에 고정된 양의 스토리지를 할당합니다. 동적 스토리지는 실행 중인 각 작업에 대해 필요에 따라 확장 및 축소됩니다.
   - 스토리지 용량 실행 - 정적 실행 스토리지의 경우 실행에 필요한 스토리지 양을 지정합니다. 이 항목은 워크플로에 지정된 기본 실행 스토리지 양을 재정의합니다.
   - S3 출력 대상 선택 - 실행 출력이 저장될 S3 위치입니다.
   - 출력 버킷 소유자의 계정 ID(선택 사항) - 계정이 출력 버킷을 소유하지 않은 경우 버킷 소유자의 AWS 계정 ID를 입력합니다. 이 정보는 HealthOmics가 버킷 소유권을 확인할 수 있도록 하기 위해 필요합니다.
   - 메타데이터 보존 모드 실행 - 모든 실행에 대해 메타데이터를 보존할지 아니면 계정이 최대 실행 수에 도달하면 시스템에서 가장 오래된 실행 메타데이터를 제거하도록 할지 선택합니다. 자세한 내용은 HealthOmics 실행에 대한 보존 모드 실행 단원을 참조하십시오.
5. 서비스 역할에서 기존 서비스 역할을 사용하거나 새 서비스 역할을 생성할 수 있습니다.
6. (선택 사항) 태그의 경우 실행에 최대 50개의 태그를 할당할 수 있습니다.
7. 다음을 선택합니다.
8. 파라미터 값 추가 페이지에서 실행 파라미터를 제공합니다. 파라미터를 지정하는 JSON 파일을 업로드하거나 값을 수동으로 입력할 수 있습니다.
9. 다음을 선택합니다.
10. 그룹 실행 패널에서 선택적으로이 실행에 대한 실행 그룹을 지정할 수 있습니다. 자세한 내용은 HealthOmics 실행 그룹 사용 단원을 참조하십시오.
11. 캐시 실행 패널에서 선택적으로이 실행에 대한 실행 캐시를 지정할 수 있습니다. 자세한 내용은 콘솔을 사용하여 실행 캐시로 실행 구성 단원을 참조하십시오.
12. Review and start run(실행 검토 및 시작)을 선택합니다.
13. 실행 구성을 검토한 후 실행 시작을 선택합니다.

API를 사용하여 실행 시작

실행 시작 API 작업을 사용하여 실행을 생성하고 시작합니다.

다음 예제에서는 워크플로 ID와 서비스 역할을 지정합니다. 이 예제에서는 보존 모드를 로 설정합니다REMOVE. 보존 모드에 대한 자세한 내용은 섹션을 참조하세요HealthOmics 실행에 대한 보존 모드 실행.

```
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
```

이에 대한 응답으로 다음 출력을 얻습니다. uuid는 실행에 고유하며와 함께 출력 데이터가 기록되는 위치를 추적하는 데 사용할 outputUri 수 있습니다.

```
{
    "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"
}
```

파라미터 파일 포함

워크플로의 파라미터 템플릿이 필요한 파라미터를 선언하는 경우 워크플로 실행을 시작할 때 입력의 로컬 JSON 파일을 제공할 수 있습니다. JSON 파일에는 각 입력 파라미터의 정확한 이름과 파라미터 값이 포함됩니다.

start-run 요청에를 AWS CLI 추가하여의 입력 JSON 파일을 참조--parameters file://<input_file.json>합니다. 실행 파라미터에 대한 자세한 내용은 섹션을 참조하세요HealthOmics 실행 입력.

요청 ID 제공

각 실행requestId에 고유한를 제공할 수 있습니다. 요청 ID는 HealthOmics가 중복 요청을 포착하는 데 사용하는 멱등성 토큰입니다. 요청 ID가 이전 실행과 중복되는 경우 실행이 시작되지 않습니다.

실행 시작 오케스트레이션에 인프라(예: Lambda 함수 또는 단계 함수)를 사용하는 경우 모범 사례는 각 StartRun 요청에 고유한 요청 ID를 제공하는 것입니다. 이렇게 하면 인프라가 이미 시작된 실행을 실수로 시작하면 HealthOmics가 중복 실행을 시작하지 않습니다. 예를 들어 인프라가 업스트림 오류 복구를 시도하는 경우 중복 요청인 실행을 시작하려고 시도하는 스크립트를 다시 실행할 수 있습니다.

워크플로 버전 선택

실행에 대한 워크플로 버전을 지정할 수 있습니다. 버전을 지정하지 않으면 HealthOmics는 기본 워크플로 버전으로 실행을 시작합니다.

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

실행 스토리지 유형 재정의

워크플로에 설정된 기본 실행 스토리지 유형을 재정의할 수 있습니다.

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

GPU 워크플로 실행

다음 예제와 같이 GPU 워크플로 ID를 지정할 수도 있습니다.

```
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
```

실행에 대한 정보 가져오기

다음과 같이 get-run API와 함께 응답의 ID를 사용하여 실행 상태를 확인할 수 있습니다.

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

이 API 작업의 응답은 워크플로 실행의 상태를 알려줍니다. 가능한 상태는 PENDING, STARTING, RUNNING및 입니다COMPLETED. 실행이 인 경우 출력 Amazon S3 버킷outfile.txt의 실행 ID 뒤에 라는 폴더에서 라는 출력 파일을 찾을 COMPLETED수 있습니다.

또한 get-run API 작업은 워크플로가 Ready2Run 또는 인지 여부, 워크플로 엔진 및 액셀러레이터 세부 정보와 같은 다른 세부 정보도 반환PRIVATE합니다. 다음 예제는 GPU 액셀러레이터가 있고 실행에 태그가 할당되지 않은 WDL에 설명된 프라이빗 워크플로 실행을 위한 get-run에 대한 응답을 보여줍니다.

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

다음과 같이 list-runs API 작업을 사용하여 모든 실행의 상태를 볼 수 있습니다.

```
aws omics list-runs
```

특정 실행에 대해 완료된 모든 작업을 보려면 list-run-tasks API를 사용합니다.

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

특정 작업의 세부 정보를 가져오려면 get-run-task API를 사용합니다.

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

실행이 완료되면 메타데이터가 스트림 아래의 CloudWatch로 전송됩니다**manifest/run/<run ID>/<run UUID>**.

다음은 매니페스트의 예입니다.

```
{
    "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 로그에 메타데이터가 없으면 실행 메타데이터가 삭제되지 않습니다.

HealthOmics에서 실행 다시 실행

아직 삭제하지 않은 실행의 경우 콘솔 또는 API를 사용하여 실행을 다시 실행합니다. 삭제한 실행의 경우 HealthOmics rerun 도구를 사용합니다.

주제

- 콘솔을 사용하여 실행 다시 실행
- API를 사용하여 실행 다시 실행
- 재실행 도구 사용

콘솔을 사용하여 실행 다시 실행

콘솔에서 다음 단계에 따라 실행을 다시 실행합니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 실행을 선택합니다.
3. 실행 페이지에서 다시 실행할 실행을 선택합니다.
4. 테이블 위의 작업 메뉴에서 재실행을 선택합니다.

API를 사용하여 실행 다시 실행

StartRun API 작업을 사용하여 기존 실행을 다시 실행합니다. 다음과 같은 필수 입력을 제공합니다.

- 서비스 역할 ARN(`roleArn`).
- 복제할 실행의 ID(`runId`).
- 실행이 실행 출력()을 저장하는 Amazon S3 위치입니다`outputUri`.

```
aws omics start-run
     --run-id run id \
     --role-arn arn:aws:iam::1234567892012:role/service-role/
OmicsWorkflow-20221004T164236 \
     --output-uri s3://workflow-output-b6f2fce1
```

재실행 도구 사용

삭제된 실행의 경우 HealthOmics rerun 도구를 다운로드하고 사용하여 실행을 다시 실행할 수 있습니다. 도구는 CloudWatch Logs 매니페스트에서 실행 정보를 검색합니다. HealthOmics rerun Tool GitHub 리포지토리에서 도구를 다운로드합니다. HealthOmics GitHub

다음 예제에서는 rerun 도구를 사용하는 방법을 보여줍니다.

```
aws-healthomics-rerun 9876543
```

CloudWatch에 실행이 있는 경우 다음 예제 출력과 유사한 응답을 받습니다. 워크플로가 더 이상 존재하지 않으면 오류 메시지가 표시됩니다.

```
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": {}
}
```

HealthOmics에서 실행 복제

HealthOmics 콘솔을 사용하여 기존 실행을 복제할 수 있습니다. 복제는 복제된 실행의 구성 값을 사용하여 새 실행을 생성합니다. 이러한 기본값을 수정하고 다른 선택적 입력을 추가할 수 있습니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(☰)을 엽니다. 실행을 선택합니다.
3. 실행 페이지에서 복제할 실행을 선택합니다.
4. 테이블 위의 작업 메뉴에서 실행 복제를 선택합니다. 콘솔에서 실행 복제 양식이 열립니다. 콘솔이 복제된 실행의 모든 관련 값으로 양식을 채운다는 점을 제외하면 양식은 실행 시작과 동일합니다.

   콘솔은 실행 복제본에 대한 새 실행 ID를 생성하고이 실행 ID를 실행 이름에 접미사로 추가합니다.

   양식 페이지를 진행하면서 필요에 따라 구성 값을 조정할 수 있습니다.
5. 실행 구성을 검토한 후 실행 시작을 선택합니다.

HealthOmics에서 실행 취소

상태가 PENDING, STARTING, RUNNING또는 인 경우 실행을 취소할 수 있습니다STOPPING.

> **Note**
>
> 실행을 취소하면 HealthOmics는 실행 출력을 저장하지 않습니다.

콘솔에서 다음 단계에 따라 실행을 취소합니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 실행을 선택합니다.
3. 실행 페이지에서 취소할 실행을 선택합니다.
4. 콘솔에서 실행 세부 정보 페이지가 열립니다. 페이지 상단의 상태 배너에서 실행 중지를 선택합니다.
5. 실행을 중지하려면 확인을 입력합니다.

API를 사용하여 실행을 취소하려면 CancelRun API 작업을 사용합니다.

다음 예제에서는를 사용하여 실행을 취소하는 방법을 보여줍니다 AWS CLI . 예제를 실행하려면를 취소하려는 실행의 ID*run id*로 바꿉니다. 성공하면 응답이 없습니다.

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

HealthOmics에서 실행 삭제

실행이 더 이상 필요하지 않은 경우 AWS CLI, API 또는 콘솔을 사용하여 삭제할 수 있습니다. 상태가 COMPLETED 또는 인 경우 실행을 삭제할 수 있습니다CANCELED.

콘솔에서 다음 단계에 따라 실행을 삭제합니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 실행을 선택합니다.
3. 실행 페이지에서 삭제할 실행을 하나 이상 선택합니다.
4. 테이블 위의 작업 메뉴에서 삭제를 선택합니다.

5. 모달 양식에 확인을 입력하여 삭제를 확인합니다.

다음 AWS CLI 명령은 실행을 삭제합니다. 예제를 실행하려면를 삭제하려는 실행의 ID*run id*로 바꿉니다. 실행이 성공적으로 삭제되면 응답이 없습니다.

```
aws omics delete-run --id run id
```

HealthOmics 실행 그룹 사용

선택적으로 실행 그룹을 생성하여 그룹에 추가하는 실행의 컴퓨팅 리소스를 제한할 수 있습니다. 그룹을 실행하면 다음과 같은 이점을 얻을 수 있습니다.

- 서비스 제한을 초과하지 않도록 실행을 대기열에 넣습니다.
- 최대 실행 기간을 설정하여 런어웨이 작업을 캡처합니다.
- 가장 중요한 실행이 먼저 완료되도록 각 실행의 우선 순위를 관리합니다.

최대 동시 vCPU, GPU 또는 실행을 설정하는 경우 최대에 도달하면 실행 작업이 대기열에 추가됩니다. 최대 실행 기간을 설정하면 최대 기간을 초과하면 실행이 실패합니다.

실행 우선 순위 설정을 사용하여 실행 그룹 내에서 우선 순위를 설정합니다.

서비스 제한이 실행 그룹 제한보다 우선합니다. 예를 들어 실행 그룹 최대값을 리전의 서비스 최대값보다 높은 값으로 설정하면 HealthOmics가 서비스 최대값을 적용합니다.

주제

- 실행 우선 순위
- 콘솔을 사용하여 실행 그룹 생성
- CLI를 사용하여 실행 그룹 생성
- 콘솔을 사용하여 실행 그룹 삭제
- CLI를 사용하여 실행 그룹 삭제

실행 우선 순위

실행 우선 순위를 사용하여 실행 그룹에서 실행 우선 순위를 설정할 수 있습니다.

여러 실행의 우선 순위가 동일한 경우 먼저 시작된 실행의 우선 순위가 더 높습니다.

실행 그룹에 없는 실행에 대한 우선 순위를 설정할 수도 있습니다. 우선 순위는 실행 그룹에 없는 다른 모든 실행의 우선 순위와 비교됩니다.

실행을 시작할 때 실행 우선 순위를 설정합니다. 자세한 내용은 HealthOmics에서 실행 시작 단원을 참조하십시오.

콘솔을 사용하여 실행 그룹 생성

실행 그룹 생성

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 그룹 실행을 선택합니다.
3. 실행 그룹 페이지에서 실행 그룹 생성을 선택합니다.
4. 실행 그룹 세부 정보 생성 페이지에서 다음 정보를 제공합니다.
   - 실행 그룹 이름 -이 실행 그룹의 고유한 이름입니다.
   - 동시 실행의 최대 vCPU - 실행 그룹의 모든 활성 실행에서 동시에 실행할 수 있는 최대 vCPUs 수입니다.
   - 최대 GPUs- 실행 그룹의 모든 활성 실행에서 동시에 실행할 수 있는 최대 GPUs 수입니다.
   - 실행당 최대 실행 시간(분) - 각 실행의 최대 시간(분)입니다. 실행이 최대 실행 시간을 초과하면 실행이 자동으로 실패합니다.
   - 최대 동시 실행 - 동시에 실행할 수 있는 최대 실행 수입니다.
5. (선택 사항) 실행 그룹에 최대 50개의 태그를 추가할 수 있습니다.
6. 실행 그룹 생성을 선택합니다.

CLI를 사용하여 실행 그룹 생성

실행 그룹을 생성하려면 create-run-group API 작업을 사용하여 라는 실행 그룹을 생성합니다`TestRunGroup`. 다음 예제에서는 최대 20CPUs, 10GPUs, 5개의 실행 및 600분의 최대 실행 기간을 설정합니다.

```
aws omics create-run-group --name TestRunGroup \
--max-cpus 20 \
--max-gpus 10 \
```

```
--max-duration 600 \
--max-runs 5
```

이 API 작업의 응답에는 새로 생성된의 ID가 포함됩니다RunGroup.

```
{
    "arn": "arn:aws:omics:us-west-2:12345678901:runGroup/2839621",
    "id": "2839621",
    "tags": {}
}
```

실행 그룹에 대한 추가 정보를 얻으려면 다음 예제와 같이 get-run-group API 작업과 함께이 ID를 사용합니다.

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

응답에는 실행 그룹에 대한 제한 설정과 할당된 태그가 포함됩니다.

```
{
    "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 작업을 사용하여 생성된 모든 실행 그룹을 볼 수도 있습니다.

```
aws omics list-run-groups
```

콘솔을 사용하여 실행 그룹 삭제

상태가 PENDING, STARTINGRUNNING, 또는 인 실행 그룹과 연결된 실행이 없는 경우 실행 그룹을 삭제할 수 있습니다STOPPING.

실행 그룹을 삭제하려면 다음 단계를 따릅니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 그룹 실행을 선택합니다.
3. 그룹 실행 페이지에서 삭제할 실행 그룹을 선택하고 xx에서 삭제를 선택합니다.

CLI를 사용하여 실행 그룹 삭제

상태가 PENDING, STARTINGRUNNING, 또는 인 실행 그룹과 연결된 실행이 없는 경우 실행 그룹을 삭제할 수 있습니다STOPPING.

다음 예제에서는 AWS CLI 를 사용하여 실행 그룹을 삭제하는 방법을 보여줍니다. 응답을 받지 못합니다. 예제를 실행하려면를 삭제하려는 실행 그룹의 ID*run group id*로 바꿉니다.

```
aws omics delete-run-group --id run group id
```

HealthOmics 실행을 위한 호출 캐싱

AWS HealthOmics 는 프라이빗 워크플로에 대해 재개라고도 하는 통화 캐싱을 지원합니다. 호출 캐싱은 실행이 완료된 후 완료된 워크플로 작업의 출력을 저장합니다. 후속 실행은 작업 출력을 다시 계산하는 대신 캐시의 작업 출력을 사용할 수 있습니다. 호출 캐싱은 컴퓨팅 리소스 사용량을 줄여 실행 기간을 단축하고 컴퓨팅 비용을 절감합니다.

실행이 완료된 후 캐시된 작업 출력 파일에 액세스할 수 있습니다. 고급 작업 디버깅 및 문제 해결을 수행하려면 워크플로 정의에서 이러한 파일을 작업 출력으로 지정하여 중간 작업 파일을 캐싱할 수 있습니다.

호출 캐싱을 사용하여 실패한 실행에서 완료된 작업 결과를 저장할 수 있습니다. 다음 실행은 완료된 작업을 다시 계산하는 대신 마지막으로 성공적으로 완료된 작업에서 시작됩니다.

HealthOmics가 작업에 대해 일치하는 캐시 항목을 찾지 못하면 실행이 실패하지 않습니다. HealthOmics는 작업과 해당 종속 작업을 다시 계산합니다.

통화 캐싱 문제 해결에 대한 자세한 내용은 섹션을 참조하세요통화 캐싱 문제 해결.

주제

- 통화 캐싱 작동 방식
- 실행 캐시 생성
- 실행 캐시 업데이트

- 실행 캐시 삭제
- 실행 캐시의 내용
- 엔진별 캐싱 기능
- 실행 캐시 사용

통화 캐싱 작동 방식

통화 캐싱을 사용하려면 실행 캐시를 생성하고 캐시된 데이터에 연결된 Amazon S3 위치를 갖도록 구성합니다. 실행을 시작할 때 실행 캐시를 지정합니다. 실행 캐시는 하나의 워크플로에만 국한되지 않습니다. 여러 워크플로에서 실행되는는 동일한 캐시를 사용할 수 있습니다.

실행의 내보내기 단계에서 시스템은 완료된 작업 출력을 Amazon S3 위치로 내보냅니다. 중간 작업 파일을 내보내려면 워크플로 정의에서 이러한 파일을 작업 출력으로 선언합니다. 또한 호출 캐싱은 메타데이터를 내부적으로 저장하고 각 캐시 항목에 대해 고유한 해시를 생성합니다.

실행 중인 각 작업에 대해 워크플로 엔진은이 작업에 일치하는 캐시 항목이 있는지 여부를 감지합니다. 일치하는 캐시 항목이 없는 경우 HealthOmics는 작업을 계산합니다. 일치하는 캐시 항목이 있는 경우 엔진은 캐시된 결과를 검색합니다.

캐시 항목을 일치시키기 위해 HealthOmics는 기본 워크플로 엔진에 포함된 해싱 메커니즘을 사용합니다. HealthOmics는 이러한 기존 해시 구현을 확장하여 S3 eTags ECR 컨테이너 다이제스트와 같은 HealthOmics 변수를 고려합니다.

HealthOmics는 다음 워크플로 언어 버전에 대한 통화 캐싱을 지원합니다.

- WDL 버전 1.0, 1.1 및 개발 버전
- Nextflow 버전 23.10 및 24.10
- 모든 CWL 버전

> **Note**
>
> HealthOmics는 Ready2Run 워크플로에 대한 통화 캐싱을 지원하지 않습니다.

주제

- 공동 책임 모델

- 작업에 대한 캐싱 요구 사항
- 캐시 성능 실행
- 캐시 데이터 보존 및 무효화 이벤트

공동 책임 모델

작업과 실행이 통화 캐싱에 적합한 후보인지 여부를 결정하는 AWS 것은 사용자와 간에 공동 책임입니다. 호출 캐싱은 모든 작업이 멱등성일 때 최상의 결과를 달성합니다(동일한 입력을 사용하여 작업을 반복 실행하면 동일한 결과가 생성됨).

그러나 작업에 비결정적 요소(예: 난수 생성 또는 시스템 시간)가 포함된 경우 동일한 입력을 사용하여 작업을 반복적으로 실행하면 출력이 다를 수 있습니다. 이는 다음과 같은 방법으로 통화 캐싱의 효과에 영향을 미칠 수 있습니다.

- HealthOmics가 작업 실행이 현재 실행에 대해 생성하는 출력과 동일하지 않은 캐시 항목(이전 실행에서 생성)을 사용하는 경우, 실행은 캐싱 없이 동일한 실행과 다른 결과를 산출할 수 있습니다.
- HealthOmics는 비결정적 작업 출력으로 인해 일치해야 하는 작업에 대해 일치하는 캐시 항목을 찾지 못할 수 있습니다. 유효한 캐시 항목을 찾지 못하면 실행이 작업을 불필요하게 다시 계산하므로 통화 캐싱 사용으로 인한 비용 절감 이점이 줄어듭니다.

다음은 호출 캐싱 결과에 영향을 미치는 비결정적 결과를 초래할 수 있는 알려진 작업 동작입니다.

- 난수 생성기 사용.
- 시스템 시간에 대한 종속성입니다.
- 동시성 사용(레이스 조건이 출력 분산을 일으킬 수 있음).
- 작업 입력 파라미터에 지정된 것 이상으로 로컬 또는 원격 파일 가져오기.

비결정적 동작을 유발할 수 있는 다른 시나리오는 Nextflow 설명서 사이트의 비결정적 프로세스 입력을 참조하세요.

작업이 비결정적 출력을 생성한다고 의심되는 경우 비결정적 특정 작업을 캐싱하지 않도록 워크플로 엔진 기능을 사용하는 것이 좋습니다. 지원되는 각 워크플로 언어의 개별 작업에 대한 캐싱을 옵트아웃하는 방법에 대한 지침은 섹션을 참조하세요엔진별 캐싱 기능.

비효율적인 호출 캐싱 또는 예상과 다른 출력으로 인해 위험이 발생할 수 있는 환경에서 호출 캐싱을 활성화하기 전에 특정 워크플로 및 작업 요구 사항을 철저히 검토하는 것이 좋습니다. 예를 들어 호출

캐싱의 잠재적 제한 사항을 신중하게 고려하여 호출 캐싱이 임상 사용 사례에 적합한지 여부를 결정해야 합니다.

작업에 대한 캐싱 요구 사항

HealthOmics는 다음 요구 사항을 충족하는 작업에 대한 작업 출력을 캐싱합니다.

- 작업은 컨테이너를 정의해야 합니다. HealthOmics는 컨테이너가 없는 작업의 출력을 캐싱하지 않습니다.
- 작업은 하나 이상의 출력을 생성해야 합니다. 워크플로 정의에서 작업 출력을 지정합니다.
- 워크플로 정의는 동적 값을 사용해서는 안 됩니다. 예를 들어 실행마다 값이 증가하는 작업에 파라미터를 전달하면 HealthOmics는 작업 출력을 캐싱하지 않습니다.

> **Note**
>
> 한 실행에서 여러 작업이 동일한 컨테이너 이미지를 사용하는 경우 HealthOmics는 이러한 모든 작업에 동일한 이미지 버전을 제공합니다. HealthOmics가 이미지를 가져온 후에는 실행 기간 동안 컨테이너 이미지에 대한 모든 업데이트를 무시합니다. 이 접근 방식은 예측 가능하고 일관된 환경을 제공하며 실행 중 배포된 컨테이너 이미지 업데이트로 인해 발생할 수 있는 잠재적 문제를 방지합니다.

캐시 성능 실행

실행에 대한 호출 캐싱을 켜면 실행 성능에 다음과 같은 영향을 미칠 수 있습니다.

- 첫 번째 실행 중에 HealthOmics는 실행 중인 작업에 대한 캐시 데이터를 저장합니다. 호출 캐싱은 내보내기 데이터의 양을 증가시키기 때문에이 실행의 내보내기 시간이 길어질 수 있습니다.
- 후속 실행에서는 캐시에서 실행을 재개할 때 처리 단계 수를 줄이고 실행 시간을 줄일 수 있습니다.
- 중간 파일을 출력으로 선언하기로 선택하면이 데이터가 더 상세할 수 있으므로 내보내기 시간이 더 길어질 수 있습니다.

캐시 데이터 보존 및 무효화 이벤트

실행 캐시의 주요 목적은 실행 중인 작업의 계산을 최적화하는 것입니다. 작업에 유효한 일치하는 캐시 항목이 있는 경우 HealthOmics는 작업을 다시 계산하는 대신 캐시 항목을 사용합니다. 그렇지 않으면

HealthOmics는 기본 서비스 동작으로 되돌아갑니다. 즉, 작업과 해당 종속 작업을 다시 계산합니다. 이 접근 방식을 사용하면 캐시 누락으로 인해 실행이 실패하지 않습니다.

실행 캐시 크기를 관리하는 것이 좋습니다. 시간이 지남에 따라 워크플로 엔진 또는 HealthOmics 서비스 업데이트 또는 실행 또는 실행 작업의 변경으로 인해 캐시 항목이 더 이상 유효하지 않을 수 있습니다. 다음 섹션에서는 추가 세부 정보를 제공합니다.

주제

- 매니페스트 버전 업데이트 및 데이터 최신성
- 캐시 동작 실행
- 컨트롤 실행 캐시 크기

매니페스트 버전 업데이트 및 데이터 최신성

HealthOmics 서비스는 주기적으로 일부 또는 모든 실행 캐시 항목을 무효화하는 새로운 기능 또는 워크플로 엔진 업데이트를 도입할 수 있습니다. 이 경우 실행에 일회성 캐시 누락이 발생할 수 있습니다.

HealthOmics는 각 캐시 항목에 대해 JSON 매니페스트 파일을 생성합니다. 2025년 2월 12일 이후에 시작된 실행의 경우 매니페스트 파일에 버전 파라미터가 포함됩니다. 서비스 업데이트가 캐시 항목을 무효화하는 경우 HealthOmics는 버전 번호를 증가시켜 제거할 레거시 캐시 항목을 식별할 수 있습니다.

다음 예제에서는 버전이 2로 설정된 매니페스트 파일을 보여줍니다.

```
{
     "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,
}
```

더 이상 유효하지 않은 캐시 항목이 있는 실행의 경우 캐시를 다시 빌드하여 새 유효한 항목을 생성합니다. 각 실행에 대해 다음 단계를 수행합니다.

1. 캐시 보존을 CACHE Always로 설정하여 실행을 한 번 시작합니다. 이 실행은 새 캐시 항목을 생성합니다.
2. 후속 실행의 경우 캐시 보존을 이전 설정(CACHE Always 또는 CACHE ON FAILURE)으로 설정합니다.

더 이상 유효하지 않은 캐시 항목을 정리하려면 캐시 Amazon S3 버킷에서 이러한 캐시 항목을 삭제하면 됩니다. HealthOmics는 이러한 캐시 항목을 재사용하지 않습니다. 유효하지 않은 항목을 보존하도록 선택하면 실행에 영향을 주지 않습니다.

> **Note**
>
> 호출 캐싱은 캐시에 지정된 Amazon S3 위치에 작업 출력 데이터를 저장하므로 요금이 발생합니다 AWS 계정.

캐시 동작 실행

실행 캐시 동작을 설정하여 실패한 실행(실패 시 캐시) 또는 모든 실행(항상 캐시)에 대한 작업 출력을 저장할 수 있습니다. 실행 캐시를 생성할 때이 캐시를 사용하는 모든 실행에 대해 기본 캐시 동작을 설정합니다. 실행을 시작할 때 기본 동작을 재정의할 수 있습니다.

Cache on failure는 여러 작업이 성공적으로 완료된 후 실패하는 워크플로를 디버깅하는 경우에 유용합니다. 해시에서 고려하는 모든 고유 변수가 이전 실행과 동일한 경우, 마지막으로 성공적으로 완료된 작업에서 후속 실행이 재개됩니다.

Cache always는 성공적으로 완료된 워크플로에서 작업을 업데이트하는 경우에 유용합니다. 다음 단계를 따르는 것이 좋습니다.

1. 새 실행을 생성합니다. 캐시 동작을 항상 캐시로 설정하고 실행을 시작합니다.
2. 실행이 완료되면 워크플로에서 작업을 업데이트하고 항상 캐시 동작 세트로 새 실행을 시작합니다. 이 실행은 업데이트된 작업과 업데이트된 작업에 종속된 후속 작업을 처리합니다. 다른 모든 작업은 캐시된 결과를 사용합니다.
3. 업데이트된 작업에 대한 개발이 완료될 때까지 필요에 따라 2단계를 반복합니다.

4. 향후 실행 시 필요에 따라 업데이트된 작업을 사용합니다. 이러한 실행에 새 입력 또는 다른 입력을 사용하려는 경우 후속 실행을 실패 시 캐시로 전환해야 합니다.

> **Note**
>
> 동일한 테스트 데이터 세트를 사용하는 동안에는 캐시를 항상 모드로 설정하는 것이 좋지만 실행 배치에는 사용하지 않는 것이 좋습니다. 대규모 배치 실행에 대해이 모드를 설정하면 시스템에서 대량의 데이터를 Amazon S3로 내보낼 수 있으므로 내보내기 시간과 스토리지 비용이 늘어날 수 있습니다.

컨트롤 실행 캐시 크기

HealthOmics는 실행 캐시 데이터를 삭제하거나 자동 보관하거나 캐시 데이터 관리를 위한 Amazon S3 정리 규칙을 적용하지 않습니다. Amazon S3 스토리지 비용을 절감하고 실행 캐시 크기를 관리할 수 있도록 정기적인 캐시 정리를 수행하는 것이 좋습니다. 파일을 직접 삭제하거나 실행 캐시 버킷에서 데이터 보존/복제 정책을 설정할 수 있습니다.

예를 들어 90일 후에 객체를 만료하도록 Amazon S3 수명 주기 정책을 구성하거나 각 개발 프로젝트가 끝날 때 캐시 데이터를 수동으로 정리할 수 있습니다.

다음 정보는 캐시 데이터 크기를 관리하는 데 도움이 될 수 있습니다.

- Amazon S3를 확인하여 캐시에 있는 데이터의 양을 확인할 수 있습니다. HealthOmics는 캐시 크기를 모니터링하거나 보고하지 않습니다.
- 유효한 캐시 항목을 삭제해도 후속 실행은 실패하지 않습니다. HealthOmics는 작업과 해당 종속 작업을 다시 계산합니다.
- HealthOmics가 작업에 대해 일치하는 항목을 찾을 수 없도록 캐시 이름 또는 디렉터리 구조를 수정하면 HealthOmics가 작업을 다시 계산합니다.

캐시 항목이 여전히 유효한지 확인해야 하는 경우 캐시 매니페스트 버전 번호를 확인합니다. 자세한 내용은 매니페스트 버전 업데이트 및 데이터 최신성 단원을 참조하십시오.

실행 캐시 생성

실행 캐시를 생성할 때 캐시 데이터에 대한 Amazon S3 위치를 지정합니다. 이 데이터는 즉시 액세스할 수 있어야 합니다. 호출 캐싱은 Glacier에 보관된 객체(예: " 및 GDA 스토리지 클래스)를 검색하지 않습니다.

캐시 데이터의 Amazon S3 버킷을 다른이 소유한 경우 실행 캐시를 생성할 때 해당 계정 ID를 AWS 계정제공합니다.

콘솔을 사용하여 실행 캐시 생성

콘솔에서 다음 단계에 따라 실행 캐시를 생성합니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 캐시 실행을 선택합니다.
3. 캐시 실행 페이지에서 실행 캐시 생성을 선택합니다.
4. 실행 캐시 생성 페이지의 캐시 세부 정보 실행 패널에서 다음 필드를 구성합니다.
   a. 실행 캐시의 이름을 입력합니다.
   b. (선택 사항) 설명을 입력합니다.
   c. 캐시된 출력의 S3 위치를 입력합니다. 워크플로와 동일한 리전에서 버킷을 선택합니다.
   d. (선택 사항) 버킷 소유자 AWS 계정 의를 입력하여 버킷 소유권을 확인합니다. 값을 입력하지 않으면 기본값은 계정 ID입니다.
   e. 캐시 동작에서 기본 동작(실패한 실행 또는 모든 실행에 대한 출력을 캐싱할지 여부)을 구성합니다. 실행을 시작할 때 선택적으로 기본 동작을 재정의할 수 있습니다.
5. (선택 사항) 하나 이상의 태그를 실행 캐시와 연결합니다.
6. 실행 캐시 생성을 선택합니다. 콘솔은 캐시 실행 테이블에 새 실행 캐시를 표시합니다.

CLI를 사용하여 실행 캐시 생성

create-run-cache CLI 명령을 사용하여 실행 캐시를 생성합니다. 기본 캐시 동작은 입니다CACHE_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"
```

생성에 성공하면 다음 필드가 포함된 응답을 받게 됩니다.

```
{
```

```
  "arn": "string",
  "id": "string",
  "status": "ACTIVE"
  "tags": {}
  }
```

실행 캐시 업데이트

캐시 이름, 설명, 태그 또는 캐시 동작을 변경할 수 있지만 캐시의 S3 위치는 변경할 수 없습니다.

콘솔을 사용하여 실행 캐시 업데이트

콘솔에서 다음 단계에 따라 실행 캐시를 업데이트합니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 캐시 실행을 선택합니다.
3. 캐시 실행 테이블에서 업데이트할 실행 캐시를 선택한 다음 편집을 선택합니다.
4. 캐시 실행 세부 정보 패널에서 실행 캐시 이름, 설명 및 캐시 동작 필드를 업데이트할 수 있습니다.
5. (선택 사항) 하나 이상의 새 태그를 실행 캐시에 연결하거나 기존 태그를 제거합니다.
6. 실행 캐시 저장을 선택합니다.

CLI를 사용하여 실행 캐시 업데이트

update-run-cache CLI 명령을 사용하여 실행 캐시를 업데이트합니다.

```
aws omics update-run-cache \
      --name "workflow 123 run cache" \
      --id "workflow id" \
      --description "my run cache" \
      --cache-behavior "CACHE_ALWAYS"
```

업데이트에 성공하면 데이터 필드가 없는 응답을 받게 됩니다.

실행 캐시 삭제

사용 중인 활성 실행이 없는 경우 실행 캐시를 삭제할 수 있습니다. 실행 캐시를 사용하는 실행이 있는 경우 실행이 완료될 때까지 기다리거나 실행을 취소할 수 있습니다.

실행 캐시를 삭제하면 리소스와 메타데이터는 제거되지만 Amazon S3의 데이터는 삭제되지 않습니다. 캐시를 삭제한 후에는 캐시를 다시 연결하거나 후속 실행에 사용할 수 없습니다.

캐싱된 데이터는 검사를 위해 Amazon S3에 남아 있습니다. 표준 S3 Delete 작업을 사용하여 이전 캐시 데이터를 제거할 수 있습니다. 또는 Amazon S3 수명 주기 정책을 생성하여 더 이상 사용하지 않는 캐시된 데이터를 만료시킵니다.

콘솔을 사용하여 실행 캐시 삭제

콘솔에서 다음 단계에 따라 실행 캐시를 삭제합니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 캐시 실행을 선택합니다.
3. 캐시 실행 테이블에서 삭제할 실행 캐시를 선택합니다.
4. 캐시 실행 테이블 메뉴에서 삭제를 선택합니다.
5. 모달 대화 상자에서 나중에 참조할 수 있도록 Amazon S3 캐시 데이터 링크를 저장한 다음 실행 캐시를 삭제할지 확인합니다.

   Amazon S3 링크를 사용하여 캐시된 데이터를 검사할 수 있지만 데이터를 다른 실행 캐시에 다시 연결할 수는 없습니다. 검사를 마쳤으면 캐시 데이터를 삭제합니다.

CLI를 사용하여 실행 캐시 삭제

delete-run-cache CLI 명령을 사용하여 실행 캐시를 삭제합니다.

```
aws omics delete-run-cache \
      --id "my cache id"
```

삭제에 성공하면 데이터 필드가 없는 응답을 받게 됩니다.

실행 캐시의 내용

HealthOmics는 실행 캐시를 S3 버킷에 다음 구조로 구성합니다.

```
s3://{cache.S3location}/{cache.uuid}/runID/taskID/{cacheentry.uuid}/
```

cache.uuid는 캐시의 전역적으로 고유한 ID입니다. cacheentry.uuid는 캐시된 작업에 대한 전역적으로 고유한 uuid입니다. HealthOmics는 캐시 및 작업에 uuid를 할당합니다.

모든 워크플로 엔진에 대해 캐시에는 다음 파일이 포함됩니다.

- {cacheentryuuid}.json 파일 - HealthOmics는 캐시의 모든 항목 목록 및 캐시 버전을 포함하여 캐시에 대한 정보가 포함된이 매니페스트 파일을 생성합니다.
- 작업 출력 파일 - 각 작업 출력은 작업에서 정의한 하나 이상의 파일로 구성됩니다.

Nextflow를 사용하는 워크플로의 경우 Nextflow 엔진은 캐시에 다음과 같은 추가 파일을 생성합니다.

- command.out 파일 -이 파일에는 작업 실행 stdout 콘텐츠가 포함되어 있습니다.
- .exitcode 파일 -이 파일에는 작업 종료 코드(정수)가 포함되어 있습니다.

> **Note**
>
> 고급 문제 해결을 위해 실행 캐시의 중간 작업 파일에 액세스하려면 워크플로 정의에서 이러한 파일을 작업 출력으로 선언합니다.

엔진별 캐싱 기능

HealthOmics는 워크플로 엔진 간에 호출 캐싱을 일관되게 구현하려고 합니다. 각 워크플로 엔진이 특정 사례를 처리하는 방식에 따라 몇 가지 차이점이 있습니다.

- 다음 흐름
  - 다양한 Nextflow 버전 간의 캐싱은 보장되지 않습니다. 예를 들어 v23.10.0에서 작업을 실행한 다음 v24.10.8에서 동일한 작업을 실행하는 경우 HealthOmics는 두 번째 실행을 캐시 누락으로 간주할 수 있습니다.
  - 캐시 false 명령을 사용하여 개별 작업에 대한 캐싱을 끌 수 있습니다. 이 명령에 대한 자세한 내용은 Nextflow 사양의 프로세스를 참조하세요.
  - HealthOmics는 Nextflow lenient 모드를 사용하지만 딥 캐싱 모드는 지원하지 않습니다.
  - 작업의 입력에 대한 S3 경로에서 glob 패턴을 사용하는 경우 캐싱은 각 개별 S3 객체를 평가합니다. 새 객체를 추가하면 HealthOmics는 새 객체를 사용하는 작업만 다시 계산합니다.
  - HealthOmics는 작업 재시도를 캐싱하지 않습니다. 이 동작은 Nextflow의 기본 동작과 일치합니다.
- WDL
  - HealthOmics는 WDL 워크플로의 개발 버전을 사용할 때 입력에 대한 새로운 “디렉터리” 유형을 지원합니다. 호출 캐싱의 경우 디렉터리의 객체가 변경되면 HealthOmics는 디렉터리를 입력하는 모든 작업을 다시 계산합니다.
  - HealthOmics는 작업 수준 캐싱을 지원하지만 워크플로 수준 캐싱은 지원하지 않습니다.

  - 휘발성 속성을 사용하여 개별 작업에 대한 캐싱을 비활성화할 수 있습니다. 자세한 내용은 휘발성 속성을 사용하여 작업 수준 캐싱 비활성화 단원을 참조하십시오.
- CWL
  - 작업의 상수 출력은 매니페스트에서 명시적으로 표시되지 않습니다. HealthOmics는 상수 출력을 중간 파일로 캐싱합니다.
  - WorkReuse 기능을 사용하여 개별 작업의 캐싱을 제어할 수 있습니다.

실행 캐시 사용

기본적으로 실행은 실행 캐시를 사용하지 않습니다. 실행에 캐시를 사용하려면 실행을 시작할 때 실행 캐시와 실행 캐시 동작을 지정합니다.

실행이 완료되면 콘솔, CloudWatch Logs 또는 API 작업을 사용하여 캐시 적중을 추적하거나 캐시 문제를 해결할 수 있습니다. 자세한 내용은 통화 캐싱 정보 추적 및 통화 캐싱 문제 해결 섹션을 참조하세요.

실행에서 하나 이상의 작업이 비결정적 출력을 생성하는 경우 실행에 호출 캐싱을 사용하지 않거나 캐싱에서 이러한 특정 작업을 옵트아웃하는 것이 좋습니다. 자세한 내용은 공동 책임 모델 단원을 참조하십시오.

> **Note**
>
> 실행을 시작할 때 IAM 서비스 역할을 제공합니다. 통화 캐싱을 사용하려면 서비스 역할에 실행 캐시 Amazon S3 위치에 액세스할 수 있는 권한이 필요합니다. 자세한 내용은 에 대한 서비스 역할 AWS HealthOmics 단원을 참조하십시오.

Amazon Q CLI를 사용하여 실행 캐시 데이터를 분석하고 관리할 수 있습니다. 자세한 내용은 Amazon Q CLI에 대한 프롬프트 예제 및 GitHub의 HealthOmics Agentic 생성형 AI 자습서를 참조하세요.

주제

- 콘솔을 사용하여 실행 캐시로 실행 구성
- CLI를 사용하여 실행 캐시로 실행 구성
- 실행 캐시에 대한 오류 사례
- 통화 캐싱 정보 추적

콘솔을 사용하여 실행 캐시로 실행 구성

콘솔에서 실행을 시작할 때 실행에 대한 실행 캐시를 구성합니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 실행을 선택합니다.
3. 실행 페이지에서 시작할 실행을 선택합니다.
4. 실행 시작을 선택하고에 설명된 대로 실행 시작의 1단계와 2단계를 완료합니다콘솔을 사용하여 실행 시작.
5. 실행 시작의 3단계에서 기존 실행 캐시 선택을 선택합니다.
6. 캐시 ID 실행 드롭다운 목록에서 캐시를 선택합니다.
7. 기본 실행 캐시 동작을 재정의하려면 실행에 대한 캐시 동작을 선택합니다. 자세한 내용은 캐시 동작 실행 단원을 참조하십시오.
8. 실행 시작의 4단계로 계속합니다.

CLI를 사용하여 실행 캐시로 실행 구성

실행 캐시를 사용하는 실행을 시작하려면 start-run CLI 명령에 cache-id 파라미터를 추가합니다. 선택적으로 `cache-behavior` 파라미터를 사용하여 실행 캐시에 대해 구성한 기본 동작을 재정의할 수 있습니다. 다음 예제에서는 명령의 캐시 필드만 보여줍니다.

```
aws omics start-run \
        ...
      --cache-id "xxxxxx"    \
      --cache-behavior  CACHE_ALWAYS
```

작업이 성공하면 데이터 필드가 없는 응답을 받게 됩니다.

실행 캐시에 대한 오류 사례

다음 시나리오의 경우 캐시 동작이 항상 캐시로 설정된 실행의 경우에도 HealthOmics는 작업 출력을 캐시하지 않을 수 있습니다.

- 첫 번째 작업이 성공적으로 완료되기 전에 실행에 오류가 발생하면 내보낼 캐시 출력이 없습니다.
- 내보내기 프로세스가 실패하면 HealthOmics는 작업 출력을 Amazon S3 캐시 위치에 저장하지 않습니다.

- filesystem out of space 오류로 인해 실행이 실패하는 경우 호출 캐싱은 작업 출력을 저장하지 않습니다.
- 실행을 취소하면 호출 캐싱이 작업 출력을 저장하지 않습니다.
- 실행에 실행 제한 시간이 발생하는 경우 실패 시 캐시를 사용하도록 실행을 구성한 경우에도 호출 캐싱은 작업 출력을 저장하지 않습니다.

통화 캐싱 정보 추적

콘솔, CLI 또는 CloudWatch Logs를 사용하여 통화 캐싱 이벤트(예: 캐시 적중 실행)를 추적할 수 있습니다.

주제

- 콘솔을 사용하여 캐시 적중 추적
- CLI를 사용하여 통화 캐싱 추적
- CloudWatch Logs를 사용하여 통화 캐싱 추적

콘솔을 사용하여 캐시 적중 추적

실행 세부 정보 페이지의 실행 작업 테이블에는 각 작업에 대한 캐시 적중 정보가 표시됩니다. 테이블에는 연결된 캐시 항목에 대한 링크도 포함되어 있습니다. 다음 절차에 따라 실행에 대한 캐시 적중 정보를 봅니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 실행을 선택합니다.
3. 실행 페이지에서 검사할 실행을 선택합니다.
4. 실행 세부 정보 페이지에서 작업 실행 탭을 선택하여 작업 테이블을 표시합니다.
5. 작업에 캐시 적중이 있는 경우 캐시 적중 열에는 Amazon S3의 실행 캐시 항목 위치에 대한 링크가 포함됩니다.
6. 실행 캐시 항목을 검사할 링크를 선택합니다.

CLI를 사용하여 통화 캐싱 추적

get-run CLI 명령을 사용하여 실행이 통화 캐시를 사용했는지 확인합니다.

```
aws omics get-run --id 1234567
```

응답에서 cacheId 필드가 설정된 경우 실행은 해당 캐시를 사용합니다.

list-run-tasks CLI 명령을 사용하여 실행 시 캐시된 각 작업의 캐시 데이터 위치를 검색합니다.

```
aws omics list-run-tasks --id 1234567
```

응답에서 작업의 cacheHit 필드가 true인 경우 cacheS3Uri 필드는 해당 작업의 캐시 데이터 위치를 제공합니다.

get-run-task CLI 명령을 사용하여 특정 작업에 대한 캐시 데이터 위치를 검색할 수도 있습니다.

```
aws omics get-run-task --id 1234567 --task-id <task_id>
```

CloudWatch Logs를 사용하여 통화 캐싱 추적

HealthOmics는 /aws/omics/WorkflowLog CloudWatch 로그 그룹에 캐시 활동 로그를 생성합니다. 실행 캐시마다 runCache/<cache_id>/<cache_uuid> 로그 스트림이 있습니다.

통화 캐싱을 사용하는 실행의 경우 HealthOmics는 다음 이벤트에 대한 CloudWatch Logs 항목을 생성합니다.

- 캐시 항목 생성(CACHE_ENTRY_CREATED)
- 캐시 항목 일치(CACHE_HIT)
- 캐시 항목과 일치하지 않음(CACHE_MISS)

이러한 로그에 대한 자세한 내용은 섹션을 참조하세요CloudWatch의 로그그.

/aws/omics/WorkflowLog 로그 그룹에서 다음 CloudWatch Insights 쿼리를 사용하여이 캐시에 대한 실행당 캐시 적중 횟수를 반환합니다.

```
filter @logStream like 'runCache/<CACHE_ID>/'
 fields @timestamp, @message
 filter logMessage like 'CACHE_HIT'
 parse "run: *," as run
 stats count(*) as cacheHits by run
```

다음 쿼리를 사용하여 각 실행에서 생성된 캐시 항목 수를 반환합니다.

```
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 워크플로 공유

프라이빗 워크플로의 소유자는 워크플로를 동일한 리전의와 공유할 수 AWS 계정 있습니다. 워크플로를 둘 이상과 공유하려면 동일한 워크플로의 여러 공유를 AWS 계정생성합니다.

소유자는 공유를 삭제하여 공유 워크플로에 대한 액세스를 취소할 수 있습니다.

> **Note**
>
> HealthOmics는 구독자의 계정에서 워크플로가 실행되는 동안 공유 워크플로가 Amazon ECR 리포지토리에 자동으로 액세스할 수 있도록 허용합니다. 공유 워크플로에 대한 추가 리포지토리 액세스 권한을 부여할 필요가 없습니다.

워크플로를 공유하면 구독자는 모든 워크플로 버전을 사용할 수 있습니다. 공유 워크플로에 대한 버전 수준 액세스 제어가 필요한 경우 워크플로 버전을 사용하는 대신 별도의 워크플로를 생성하는 것이 좋습니다.

주제

- 공유 워크플로 구독
- 워크플로 공유 상태 모니터링
- 콘솔을 사용하여 프라이빗 워크플로 공유
- CLI를 사용하여 프라이빗 워크플로 공유
- 콘솔을 사용하여 공유 워크플로 수락
- 콘솔을 사용하여 공유 워크플로 실행
- API를 사용하여 공유 워크플로 실행

공유 워크플로 구독

공유 워크플로를 구독하려면 다음 전체 단계에 따라 워크플로를 수락하고 사용합니다.

1. 콘솔 또는 API를 사용하여 공유를 수락합니다. 현재 리전을 공유 요청과 동일한 리전으로 설정합니다.

- 콘솔에서 공유 요청을 찾으려면 모든 리소스 공유 페이지로 이동한 다음 나와 공유 탭을 선택합니다.

2. 콘솔 또는 API를 사용하여 공유 워크플로에 대한 실행을 생성합니다.
   - 콘솔에서 워크플로 세부 정보 페이지를 찾으려면 나와 공유됨(1단계 참조)으로 이동한 다음 공유 워크플로의 리소스 링크를 선택합니다.
3. 워크플로에 대한 자체 입력 데이터를 제공합니다.
4. 공유 워크플로는에서 실행됩니다 AWS 계정.

공유 워크플로의 구독자는 시스템에서 다음 워크플로 작업을 수행하지 못하도록 차단합니다.

- 공유 워크플로 내보내기
- 공유 워크플로 다시 실행
  - 공유 워크플로에 대한 새 실행을 생성합니다.
- 워크플로를 다시 공유합니다.
- 워크플로에 태그를 할당합니다.
- 워크플로 삭제.
  - 워크플로가 더 이상 필요하지 않으면 워크플로 공유를 삭제합니다.

리소스 공유에 대한 자세한 내용은 섹션을 참조의 교차 계정 리소스 공유 AWS HealthOmics하세요.

워크플로 공유 상태 모니터링

HealthOmics는 워크플로 공유의 각 상태 변경에 대해 이벤트를 EventBridge로 전송합니다. 특정 상태 변경에 대한 알림을 받으려면 워크플로 공유 상태 변경 이벤트를 모니터링하도록 EventBridge 규칙을 설정합니다. 예제:

- 워크플로 공유 요청을 수신할 때마다, 그리고 사용자가 워크플로 공유를 취소할 때마다 알림이 필요합니다.
- 워크플로 공유 요청을 시작한 후 사용자가 요청을 수락하거나 거부할 때 알림을 받으려고 합니다.

이벤트 사용에 대한 자세한 내용은 섹션을 참조하세요에서 EventBridge 사용 AWS HealthOmics.

콘솔을 사용하여 프라이빗 워크플로 공유

콘솔에서 프라이빗 워크플로를 워크플로와 동일한 리전 AWS 계정 의와 공유할 수 있습니다.

프라이빗 워크플로를 공유하려면

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(☰)을 엽니다. 프라이빗 워크플로를 선택합니다.
3. 프라이빗 워크플로 페이지의 워크플로 테이블에서 공유할 워크플로를 선택하고 공유를 선택합니다.
4. 워크플로 공유 페이지의 세부 정보 공유 패널에서 공유에 대한 설명이 포함된 이름을 입력하고 구독 AWS 계정 자의를 입력합니다.
5. 리소스 공유를 선택합니다. 콘솔은 모든 리소스 공유 페이지에 리소스 공유를 표시합니다.

공유의 초기 상태는 보류 중입니다. 구독자가 공유를 수락하면 상태가 활성으로 변경됩니다.

CLI를 사용하여 프라이빗 워크플로 공유

Create-share API 작업을 사용하여 워크플로 공유를 생성합니다. 보안 주체 구독자는 워크플로 AWS 계정 에 액세스할 수 있는 사용자의 입니다.

```
aws omics create-share \
  --resource-arn "arn:aws:omics:us-west-2:555555555555:workflow/123456" \
  --principal-subscriber "123456789012" \
  --name "my_Share-123"
```

생성에 성공하면 공유 ID 및 상태가 포함된 응답을 받게 됩니다.

```
{
"shareId": "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a",
"name": "my_Share-123",
"status": "PENDING"
}
```

구독자가 `accept-share` API 작업을 사용하여 수락할 때까지 공유는 보류 상태로 유지됩니다.

다른 API 사용 예제는 섹션을 참조의 교차 계정 리소스 공유 AWS HealthOmics하세요.

콘솔을 사용하여 공유 워크플로 수락

콘솔을 사용하여 제공된 워크플로 공유를 수락할 수 있습니다. 콘솔을 워크플로와 동일한 리전으로 설정해야 합니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(☰)을 엽니다. 모든 리소스 공유를 선택한 다음 나와 공유 탭을 선택합니다.
3. 나와 공유된 리소스 테이블에서 워크플로 공유를 선택한 다음 수락을 선택합니다.

워크플로를 수락한 후 공유 워크플로의 리소스 링크를 선택하여 세부 정보를 확인합니다.

콘솔을 사용하여 공유 워크플로 실행

워크플로 공유를 수락한 후 워크플로에서 실행을 시작할 수 있습니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(☰)을 엽니다. 모든 리소스 공유를 선택한 다음 나와 공유 탭을 선택합니다.
3. 나와 공유된 리소스 테이블에서 공유 워크플로의 리소스 링크를 선택합니다.
4. 워크플로 세부 정보 페이지에서 실행 생성을 선택합니다.

   콘솔에서 워크플로 유형(공유)과 워크플로 ID가 미리 채워진 실행 생성 페이지가 열립니다.
5. 실행 생성 양식에서 나머지 필드를 구성합니다. 자세한 내용은 콘솔을 사용하여 실행 시작 섹션을 참조하세요.

API를 사용하여 공유 워크플로 실행

get-workflow를 사용하여 공유 워크플로의 ARN을 검색합니다.

```
aws omics get-workflow --id 1234567 \
--workflow-owner-id 55555555555
```

워크플로를 실행할 때 워크플로 소유자의 AWS 계정 ID와 공유 워크플로의 ARN을 제공합니다.

```
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
```

HealthOmics에서 Ready2Run 워크플로

Ready2Run 워크플로는 타사 게시자가 게시하는 사전 구성된 워크플로입니다. Sentieon Inc와 같은 일부 게시자는 구독 기반 워크플로를 제공합니다. 다른 Ready2Run 워크플로에는 구독이 필요하지 않으며 일부 워크플로는 NF-Core 워크플로와 같은 오픈 소스입니다.

Ready2Run 워크플로는 다음 시나리오에 적합합니다.

- 기본 인프라를 설정할 필요 없이 파이프라인 출력 분석과 결과 생성에 집중하려고 합니다.
- 설정된 워크플로를 사용하여 결과를 복제하려고 합니다.
- 소프트웨어 개발자는 애플리케이션을 HealthOmics SDK와 직접 통합하려고 합니다.

HealthOmics는 Ready2Run 워크플로에 대한 버전 관리를 지원합니다. 버전을 제공하는 Ready2Run 워크플로의 경우 실행을 시작할 때 버전 이름을 지정할 수 있습니다.

모든 Ready2Run 워크플로는 문제 해결에 사용할 수 있는 CloudWatch 로그를 포함한 로그를 제공합니다.

> **Note**
>
> Sentieon Ready2Run 워크플로는 구독 기반입니다. 계정에서 처음으로 Sentieon Ready2Run 워크플로를 실행하면 Sentieon은에 대해 2주 평가 라이선스를 자동으로 생성합니다 AWS 계정. 라이선스는 모든 Sentieon Ready2Run 워크플로에 유효합니다. 평가 기간이 종료된 후 영구 라이선스를 요청하거나 평가 라이선스에 대한 확장을 요청할 수 있습니다. 세부 정보는 Subscribing to Sentieon Ready2Run workflows 섹션을 참조하세요.

주제

HealthOmics에서 사용 가능한 Ready2Run 워크플로

다음 표에는 HealthOmics에서 사용할 수 있는 Ready2Run 워크플로가 나열되어 있습니다.

HealthOmics 콘솔에 로그인하여 입력 파라미터 및 워크플로 다이어그램을 포함하여 이러한 워크플로에 대한 자세한 정보를 볼 수 있습니다. Ready2Run 워크플로에 대한 요금 정보는 HealthOmics 요금을 참조하세요.

> **Note**
>
> 각 Ready2Run 워크플로에는 최대 입력 파일 크기가 있습니다. 이러한 최대 파일 크기는 조정할 수 없습니다.

워크플로 이름	게시자	구독이 필요합니까?	최대 입력 파일 크기(GiB)	예상 런타임 (HH:MM)
AlphaFold - 601-1200개 구체화	Google DeepMind	아니요	1	11:15
최대 600개의 시크릿에 대한 AlphaFold	Google DeepMind	아니요	1	7:30
2x150용 Bases2Fastq	Element Biosciences	아니요	1000	1:45
2x300용 Bases2Fastq	Element Biosciences	아니요	1000	1:30
2x75용 Bases2Fastq	Element Biosciences	아니요	500	0:45
최대 800개의 시크릿에 대한 ESMFold	메타 연구	아니요	1	0:15
GATK-BP fq2bam	브로드 인스티튜트	아니요	64	10:10
30x 유전체용 GATK-	브로드 인스티튜트	아니요	39	2:45

워크플로 이름	게시자	구독이 필요합니까?	최대 입력 파일 크기(GiB)	예상 런타임 (HH:MM)
BP Germline bam2vcf				
30x 유전체용 GATK-BP Germline fq2vcf	브로드 인스티튜트	아니요	64	12:30
GATK-BP 신체 WES bam2vcf	브로드 인스티튜트	아니요	86	1:30
최대 30X의 NVIDIA Parabricks BAM2FQ2BAM WGS	NVIDIA Corporation	아니요	80	1:39
최대 50X의 NVIDIA Parabricks BAM2FQ2BAM WGS	NVIDIA Corporation	아니요	120	2:45
최대 5X의 NVIDIA Parabricks BAM2FQ2BAM WGS	NVIDIA Corporation	아니요	20	0:18
최대 30X의 NVIDIA Parabricks FQ2BAM WGS	NVIDIA Corporation	아니요	71	1:00

워크플로 이름	게시자	구독이 필요합니까?	최대 입력 파일 크기(GiB)	예상 런타임 (HH:MM)
최대 50X의 NVIDIA Parabricks FQ2BAM WGS	NVIDIA Corporation	아니요	137	1:45
최대 5X의 NVIDIA Parabricks FQ2BAM WGS	NVIDIA Corporation	아니요	13	0:15
최대 30X의 NVIDIA Parabricks Germline DeepVariant WGS	NVIDIA Corporation	아니요	71	2:00
최대 50X의 NVIDIA Parabricks Germline DeepVariant WGS	NVIDIA Corporation	아니요	137	3:30
최대 5X의 NVIDIA Parabricks Germline DeepVariant WGS	NVIDIA Corporation	아니요	12	0:30

워크플로 이름	게시자	구독이 필요합니까?	최대 입력 파일 크기(GiB)	예상 런타임 (HH:MM)
최대 30X의 NVIDIA Parabricks Germline HaplotypeCaller WGS	NVIDIA Corporation	아니요	71	1:15
최대 50X의 NVIDIA Parabricks Germline HaplotypeCaller WGS	NVIDIA Corporation	아니요	137	2:00
최대 5X의 NVIDIA Parabricks Germline HaplotypeCaller WGS	NVIDIA Corporation	아니요	13	0:15
최대 50X의 NVIDIA Parabricks Somatic Mutect2 WGS	NVIDIA Corporation	아니요	196	0:45
KallistoBUStools를 사용하는 scRNAseq	NF-Core	아니요	119	1:30
Salmon Alevin-fry를 사용한 scRNAseq	NF-Core	아니요	119	2:30

워크플로 이름	게시자	구독이 필요합니까?	최대 입력 파일 크기(GiB)	예상 런타임 (HH:MM)
STARsolo를 사용하는 scRNAseq	NF-Core	아니요	119	2:30
최대 300배의 Sentieon Germline BAM WES	Sentieon, Inc.	예	9	1:00
최대 32배의 Sentieon Germline BAM WGS	Sentieon, Inc.	예	18	1:30
최대 100배의 Sentieon Germline FASTQ WES	Sentieon, Inc.	예	5	0:45
최대 300배의 Sentieon Germline FASTQ WES	Sentieon, Inc.	예	26	2:00
최대 32배의 Sentieon Germline FASTQ WGS	Sentieon, Inc.	예	51	3:30
ONT용 Sentieon LongRead	Sentieon, Inc.	예	25	1:30
PacBio HiFi용 Sentieon LongRead	Sentieon, Inc.	예	58	4:00

워크플로 이름	게시자	구독이 필요합니까?	최대 입력 파일 크기(GiB)	예상 런타임 (HH:MM)
Sentieon 신체 WES	Sentieon, Inc.	예	50	2:30
Sentieon Somatic WGS	Sentieon, Inc.	예	113	4:30
최대 40배의 Ultima Genomics DeepVariant	Ultima Genomics	아니요	91	1:55

Ready2Run 워크플로를 사용하면 워크플로가 미리 구성되어 편집할 수 없습니다. 프라이빗 워크플로와 달리 Ready2Run 워크플로는 다음을 지원하지 않습니다.

- 최대 입력 파일 크기 늘리기
- 컴퓨팅 리소스 변경 또는 스토리지 실행
- 워크플로 정의 또는 컨테이너 변경
- 실행 그룹에 실행 추가
- 워크플로 공유

게시자가 GitHub에서 Ready2Run 워크플로를 공유한 경우 Ready2Run 워크플로를 기반으로 자체 프라이빗 워크플로를 만들 수 있습니다. 다음 표에는 각 게시자의 GitHub 워크플로에 대한 링크가 나와 있습니다.

게시자	GitHub의 워크플로
Google DeepMind, Meta Research	단백질 폴딩 워크플로
Element Biosciences	자세한 내용은 Element Biosciences에 문의하세요.
브로드 인스티튜트	GATK 워크플로
NVIDIA Corporation	Parabricks 워크플로

게시자	GitHub의 워크플로
nf-core	NF-Core 워크플로
센티온	Sentieon 워크플로
Ultima Genomics	Ultima Genomics 워크플로

Sentieon Ready2Run 워크플로 구독

Sentieon Ready2Run 워크플로는 구독 기반입니다. 계정에서 처음으로 Sentieon Ready2Run 워크플로를 실행하면 Sentieon은에 대해 2주 평가 라이선스를 자동으로 생성합니다 AWS 계정. 라이선스는 모든 Sentieon Ready2Run 워크플로에 유효합니다. 평가 기간이 종료된 후 영구 라이선스를 요청하거나 평가 라이선스에 대한 확장을 요청할 수 있습니다.

다음 단계에 따라 Sentieon Ready2Run 워크플로를 구독합니다.

- 다음 지침에 따라 AWS 정식 사용자 ID를 찾습니다.
- Sentieon 지원 그룹(support@sentieon.com)에 이메일을 보내 소프트웨어 라이선스를 요청합니다. 이메일에 AWS 정식 사용자 ID를 입력합니다.

콘솔을 사용하여 HealthOmics Ready2Run 워크플로 시작

콘솔에서 Ready2Run 워크플로를 사용하는 것은 프라이빗 워크플로를 사용하는 것과 유사합니다. 한 가지 주요 차이점은 워크플로 게시자가 샘플 데이터를 제공하므로 자체 데이터를 생성하지 않고도 워크플로를 사용해 볼 수 있다는 것입니다.

콘솔에서 Ready2Run 워크플로를 사용하려면

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. Ready2Run 워크플로를 선택합니다.
3. Ready2Run 워크플로 페이지에서 사용할 워크플로를 선택합니다. 콘솔에서 해당 워크플로의 세부 정보 페이지가 열립니다.
4. 세부 정보 탭에는 이름, 실행당 정가, 설명, 워크플로 언어 유형, 실행 스토리지 용량, 상태, 생성 날짜, 설명이 포함된 파라미터와 같은 정보가 나열됩니다. 세부 정보 탭에는 워크플로에 구독이 필요한지 여부도 표시됩니다.

5. 워크플로를 사용하려면 실행 생성을 선택합니다.
6. 실행 세부 정보 지정 페이지에서 실행 이름을 입력합니다. 선택적으로 워크플로 버전을 지정할 수 있습니다. 실행에 실행 우선 순위를 추가할 수도 있습니다.
7. 실행 출력의 Amazon S3 위치를 입력하거나 선택합니다.
8. 메타데이터 보존 모드 실행에서 실행 메타데이터를 보존할지 아니면 제거할지를 선택합니다.
9. 서비스 역할 패널에서 기존 서비스 역할을 사용할지 아니면 새 서비스 역할을 생성할지 선택합니다.
10. (선택 사항) 실행을 식별하고 관리하는 데 도움이 되는 태그를 추가합니다.
11. 다음을 선택합니다.
12. 파라미터 추가 페이지에서 옵션 중 하나를 선택하여 실행 파라미터 값을 추가합니다.
    - Amazon S3 위치에서 파라미터 파일(JSON 형식)을 선택합니다.
    - 로컬 드라이브에서 파라미터 파일(JSON 형식)을 선택합니다.
    - 파라미터 값을 수동으로 입력합니다.
    - 워크플로 게시자가 제공한 Ready2Run 샘플 데이터로 워크플로를 실행합니다.
13. JSON 파일을 업로드하면 콘솔이 파일을 구문 분석하고 인라인 검증을 수행합니다. 그런 다음 필요에 따라 파라미터 값을 수동으로 업데이트할 수 있습니다.
14. 다음을 선택합니다.
15. 입력을 검토한 다음 실행 시작을 선택합니다.

API를 사용하여 HealthOmics Ready2Run 워크플로 시작

대부분의 API 작업은 Ready2Run 워크플로와 프라이빗 워크플로에서 유사한 방식으로 작동합니다.

사용 가능한 Ready2Run 워크플로 목록을 반환하려면 `type` 파라미터가 READY2RUN으로 설정된 list-workflows를 사용합니다.

```
aws omics list-workflows --type READY2RUN
```

list-workflows 응답에서 실행할 워크플로를 식별한 후 `--id` 파라미터와 함께 get-workflow를 사용하여 자세한 내용을 가져올 수 있습니다.

```
aws omics get-workflow --type READY2RUN --id workflow id
```

Ready2Run 워크플로를 실행하려면 다음 예제와 READY2RUN같이 워크플로 유형 파라미터가 로 설정된 상태에서 시작-실행 API 작업을 사용할 수 있습니다.

```
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
```

워크플로 버전을 지정하려면이 예제와 같이 워크플로 버전 파라미터를 사용합니다.

```
aws-omics start-run \
  --workflow-type READY2RUN \
  ...
  --version-name '3.0.0'
```

다음과 같이 실행을 모니터링하기 위해 get-run API 작업을 사용할 수 있습니다.

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

HealthOmics 스토리지

HealthOmics 스토리지를 사용하여 저렴한 비용으로 유전체학 데이터를 효율적으로 저장, 검색, 구성 및 공유할 수 있습니다. HealthOmics 스토리지는 서로 다른 데이터 객체 간의 관계를 이해하므로 동일한 소스 데이터에서 시작된 읽기 세트를 정의할 수 있습니다. 이를 통해 데이터 출처를 알 수 있습니다.

`ACTIVE` 상태로 저장된 데이터는 즉시 검색할 수 있습니다. 30일 이상 액세스하지 않은 데이터는 `ARCHIVE` 상태로 저장됩니다. 보관된 데이터에 액세스하려면 API 작업 또는 콘솔을 통해 다시 활성화하면 됩니다.

HealthOmics 시퀀스 스토어는 파일의 콘텐츠 무결성을 유지하도록 설계되었습니다. 그러나 가져온 데이터 파일과 내보낸 파일의 비트 단위 동등성은 활성 및 아카이브 계층화 중 압축으로 인해 보존되지 않습니다.

수집 중에 HealthOmics는 데이터 파일의 콘텐츠 무결성을 검증할 수 있도록 개체 태그 또는 HealthOmics ETag를 생성합니다. 시퀀싱 부분은 읽기 세트의 소스 수준에서 ETag로 식별되고 캡처됩니다. ETag 계산은 실제 파일 또는 유전체 데이터를 변경하지 않습니다. 읽기 세트가 생성된 후에는 읽기 세트 소스의 수명 주기 동안 ETag가 변경되어서는 안 됩니다. 즉, 동일한 파일을 다시 가져오면 동일한 ETag 값이 계산됩니다.

주제

HealthOmics ETags 및 데이터 출처

HealthOmics ETag(엔터티 태그)는 시퀀스 스토어에서 수집된 콘텐츠의 해시입니다. 이렇게 하면 수집된 데이터 파일의 콘텐츠 무결성을 유지하면서 데이터 검색 및 처리가 간소화됩니다. ETag는 메타데

이터가 아닌 객체의 의미론적 콘텐츠에 대한 변경 사항을 반영합니다. 지정된 읽기 세트 유형 및 알고리즘에 따라 ETag 계산 방법이 결정됩니다. ETag 계산은 실제 파일 또는 유전체 데이터를 변경하지 않습니다. 읽기 세트의 파일 유형 스키마가 허용하는 경우 시퀀스 스토어는 데이터 출처에 연결된 필드를 업데이트합니다.

파일에는 비트 단위 자격 증명과 의미 체계 자격 증명이 있습니다. 비트 단위 자격 증명은 파일의 비트가 동일하다는 의미이고 의미 체계 자격 증명은 파일의 내용이 동일하다는 의미입니다. 의미 체계 자격 증명은 파일의 콘텐츠 무결성을 캡처하므로 메타데이터 변경 및 압축 변경에 대한 복원력이 뛰어납니다.

HealthOmics 시퀀스 스토어의 읽기 세트는 객체의 수명 주기 동안 압축/압축 해제 주기 및 데이터 출처 추적을 거칩니다. 이 처리 중에 수집된 파일의 비트 단위 자격 증명이 변경될 수 있으며 파일이 활성화될 때마다 변경될 것으로 예상되지만 파일의 의미 체계 자격 증명은 유지됩니다. 의미 체계 자격 증명은 HealthOmics 개체 태그 또는 시퀀스 스토어 수집 중에 계산되어 읽기 세트 메타데이터로 사용할 수 있는 ETag로 캡처됩니다.

읽기 세트의 파일 유형 스키마가 허용하는 경우 시퀀스 스토어 업데이트 필드는 데이터 출처에 연결됩니다. uBAM, BAM 및 CRAM 파일의 경우 헤더에 새 `@CO` 또는 `Comment` 태그가 추가됩니다. 주석에는 시퀀스 스토어 ID와 수집 타임스탬프가 포함됩니다.

Amazon S3 ETags

Amazon S3 URI를 사용하여 파일에 액세스할 때 Amazon S3 API 작업은 Amazon S3 ETag 및 체크섬 값도 반환할 수 있습니다. Amazon S3 ETag 및 체크섬 값은 파일의 비트 ID를 나타내기 때문에 HealthOmics ETags와 다릅니다. 설명 메타데이터 및 객체에 대한 자세한 내용은 Amazon S3 객체 API 설명서를 참조하세요. Amazon S3 ETag 값은 읽기 세트의 각 활성화 주기에 따라 변경될 수 있으며 이를 사용하여 파일 읽기를 검증할 수 있습니다. 그러나 파일 수명 주기 동안 파일 자격 증명 검증에 사용할 Amazon S3 ETag 값은 일관되게 유지되지 않으므로 캐시하지 마십시오. 반대로 HealthOmics ETag는 읽기 세트의 수명 주기 동안 일관되게 유지됩니다.

HealthOmics ETags 계산하는 방법

ETag는 수집된 파일 콘텐츠의 해시에서 생성됩니다. ETag 알고리즘 패밀리는 기본적으로 MD5up으로 설정되지만 시퀀스 스토어 생성 중에 다르게 구성할 수 있습니다. ETag가 계산되면 알고리즘과 계산된 해시가 읽기 세트에 추가됩니다. 파일 유형에 지원되는 MD5 알고리즘은 다음과 같습니다.

- FASTQ_MD5up - 압축되지 않은 전체 FASTQ 읽기 세트 소스의 MD5 해시를 계산합니다.
- BAM_MD5up - 사용 가능한 경우 연결된 참조를 기반으로 SAM에 표시된 대로 압축되지 않은 BAM 또는 uBAM 읽기 세트 소스의 정렬 섹션의 MD5 해시를 계산합니다.

- CRAM_MD5up - 연결된 참조를 기반으로 SAM에 표시된 대로 압축되지 않은 CRAM 읽기 세트 소스의 정렬 섹션의 MD5 해시를 계산합니다.

> **Note**
>
> MD5 해싱은 충돌에 취약한 것으로 알려져 있습니다. 따라서 알려진 충돌을 악용하도록 제작된 두 개의 서로 다른 파일은 동일한 ETag를 가질 수 있습니다.

SHA256 패밀리에는 다음 알고리즘이 지원됩니다. 알고리즘은 다음과 같이 계산됩니다.

- FASTQ_SHA256up - 압축되지 않은 전체 FASTQ 읽기 세트 소스의 SHA-256 해시를 계산합니다.
- BAM_SHA256up - 사용 가능한 경우 연결된 참조를 기반으로 SAM에 표시된 대로 압축되지 않은 BAM 또는 uBAM 읽기 세트 소스의 정렬 섹션의 SHA-256 해시를 계산합니다.
- CRAM_SHA256up - 연결된 참조를 기반으로 SAM에 표시된 대로 압축되지 않은 CRAM 읽기 세트 소스의 정렬 섹션의 SHA-256 해시를 계산합니다.

SHA512 패밀리에는 다음 알고리즘이 지원됩니다. 알고리즘은 다음과 같이 계산됩니다.

- FASTQ_SHA512up - 압축되지 않은 전체 FASTQ 읽기 세트 소스의 SHA-512 해시를 계산합니다.
- BAM_SHA512up - 사용 가능한 경우 연결된 참조를 기반으로 SAM에 표시된 대로 압축되지 않은 BAM 또는 uBAM 읽기 세트 소스의 정렬 섹션의 SHA-512 해시를 계산합니다.

- CRAM_SHA512up - 연결된 참조를 기반으로 SAM에 표시된 대로 압축되지 않은 CRAM 읽기 세트 소스의 정렬 섹션의 SHA-512 해시를 계산합니다.

HealthOmics 참조 스토어 생성

HealthOmics의 참조 스토어는 참조 유전체를 저장하기 위한 데이터 스토어입니다. 각 AWS 계정 및 리전에 단일 참조 저장소를 가질 수 있습니다. 콘솔 또는 CLI를 사용하여 참조 저장소를 생성할 수 있습니다.

주제

콘솔을 사용하여 참조 저장소 생성

참조 저장소 생성

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 참조 저장소를 선택합니다.
3. Genomics 데이터 스토리지 옵션에서 참조 유전체를 선택합니다.
4. 이전에 가져온 참조 유전체를 선택하거나 새 참조 유전체를 가져올 수 있습니다. 참조 유전체를 가져오지 않은 경우 오른쪽 상단에서 참조 유전체 가져오기를 선택합니다.
5. 참조 유전체 가져오기 작업 생성 페이지에서 빠른 생성 또는 수동 생성 옵션을 선택하여 참조 저장소를 생성한 다음 다음 다음 정보를 제공합니다.

   - 참조 유전체 이름 -이 저장소의 고유한 이름입니다.
   - 설명(선택 사항) -이 참조 저장소에 대한 설명입니다.
   - IAM 역할 - 참조 유전체에 액세스할 수 있는 역할을 선택합니다.
   - Amazon S3의 참조 - Amazon S3 버킷에서 참조 시퀀스 파일을 선택합니다.
   - 태그(선택 사항) -이 참조 저장소에 최대 50개의 태그를 제공합니다.

CLI를 사용하여 참조 저장소 생성

다음 예제에서는를 사용하여 참조 저장소를 생성하는 방법을 보여줍니다 AWS CLI. AWS 리전당 하나의 참조 저장소를 가질 수 있습니다.

참조 스토어는 확장명이 `.fasta`, , , `.fa`, , , `.faa`, `.fas` `.fsa.ffn`, `.fna.frn`, `.mpfa`, `.seq`인 FASTA 파일의 스토리지를 지원합니다`.txt`. 이러한 확장의 `bgzip` 버전도 지원됩니다.

다음 예제에서 *`reference store name`*를 참조 저장소에 대해 선택한 이름으로 바꿉니다.

```
aws omics create-reference-store --name "reference store name"
```

참조 스토어 ID 및 이름, ARN, 참조 스토어가 생성된 시점의 타임스탬프가 포함된 JSON 응답을 받습니다.

```
{
    "id": "3242349265",
    "arn": "arn:aws:omics:us-west-2:555555555555:referenceStore/3242349265",
```

```
    "name": "MyReferenceStore",
    "creationTime": "2022-07-01T20:58:42.878Z"
}
```

추가 AWS CLI 명령에서 참조 스토어 ID를 사용할 수 있습니다. 다음 예제와 같이 list-reference-stores 명령을 사용하여 계정에 연결된 참조 스토어 IDs 목록을 검색할 수 있습니다.

```
aws omics list-reference-stores
```

이에 대한 응답으로 새로 생성된 참조 저장소의 이름을 받게 됩니다.

```
{
    "referenceStores": [
        {
              "id": "3242349265",
              "arn": "arn:aws:omics:us-west-2:555555555555:referenceStore/3242349265",
              "name": "MyReferenceStore",
             "creationTime": "2022-07-01T20:58:42.878Z"
         }
     ]
}
```

참조 저장소를 생성한 후 가져오기 작업을 생성하여 유전체 참조 파일을 해당 저장소에 로드할 수 있습니다. 이렇게 하려면 IAM 역할을 사용하거나 생성하여 데이터에 액세스해야 합니다. 다음은 예제 정책입니다.

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/*"
```

```
            ]
        }
    ]
}
```

또한 다음 예제와 유사한 신뢰 정책이 있어야 합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                   "omics.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

이제 참조 유전체를 가져올 수 있습니다. 이 예제에서는 오픈 액세스가 가능하고 오픈 데이터 레지스트리 AWS에서 사용할 수 있는 Genome Reference Consortium Human Build 38(hg38)을 사용합니다. 이 데이터를 호스팅하는 버킷은 미국 동부(오하이오)에 기반을 두고 있습니다. 다른 AWS 리전에서 버킷을 사용하려면 해당 리전에서 호스팅되는 Amazon S3 버킷에 데이터를 복사할 수 있습니다. 다음 AWS CLI 명령을 사용하여 유전체를 Amazon S3 버킷에 복사합니다.

```
aws s3 cp s3://broad-references/hg38/v0/Homo_sapiens_assembly38.fasta s3://amzn-s3-
demo-bucket
```

그런 다음 가져오기 작업을 시작할 수 있습니다. *reference store ID*, *role ARN*및를 자체 입력*source file path*으로 바꿉니다.

```
aws omics start-reference-import-job --reference-store-id reference store ID --role-
arn role ARN --sources source file path
```

데이터를 가져온 후 JSON으로 다음 응답을 받게 됩니다.

```
{
        "id": "7252016478",
        "referenceStoreId": "3242349265",
        "roleArn": "arn:aws:iam::111122223333:role/OmicsReferenceImport",
        "status": "CREATED",
        "creationTime": "2022-07-01T21:15:13.727Z"
}
```

다음 명령을 사용하여 작업 상태를 모니터링할 수 있습니다. 다음 예제에서 *reference store ID* 및를 참조 스토어 ID와 자세히 알아볼 작업 ID*job ID*로 바꿉니다.

```
aws omics get-reference-import-job --reference-store-id reference store ID --id job ID
```

이에 대한 응답으로 해당 참조 저장소 및 해당 상태에 대한 세부 정보가 포함된 응답을 받게 됩니다.

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

참조를 나열하고 참조 이름을 기준으로 필터링하여 가져온 참조를 찾을 수도 있습니다. 를 참조 스토어 ID*reference store ID*로 바꾸고 선택적 필터를 추가하여 목록의 범위를 좁힙니다.

```
aws omics list-references --reference-store-id reference store ID --filter
 name=MyReference
```

이에 대한 응답으로 다음 정보를 받게 됩니다.

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

참조 메타데이터에 대해 자세히 알아보려면 get-reference-metadata API 작업을 사용합니다. 다음 예제에서를 참조 스토어 ID*reference store ID*로 바꾸고를 자세히 알아볼 참조 ID*reference ID*로 바꿉니다.

```
aws omics get-reference-metadata --reference-store-id reference store ID --id reference
 ID
```

응답으로 다음 정보를 받게 됩니다.

```
{
    "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-reference를 사용하여 참조 파일의 일부를 다운로드할 수도 있습니다. 다음 예제에서를 참조 스토어 ID*reference store ID*로 바꾸고를 다운로드하려는 참조 ID*reference ID*로 바꿉니다.

```
aws omics get-reference --reference-store-id reference store ID --id reference ID --
part-number 1 outfile.fa
```

HealthOmics 시퀀스 스토어 생성

HealthOmics 시퀀스 스토어는 (FASTQgzip 전용) 및의 정렬되지 않은 형식의 게놈 파일 저장을 지원합니다uBAM. 또한 BAM 및의 정렬된 형식도 지원합니다CRAM.

가져온 파일은 읽기 세트로 저장됩니다. 읽기 세트에 태그를 추가하고 IAM 정책을 사용하여 읽기 세트에 대한 액세스를 제어할 수 있습니다. 정렬된 읽기 세트에는 유전체 시퀀스를 정렬하기 위한 참조 유전체가 필요하지만 정렬되지 않은 읽기 세트의 경우 선택 사항입니다.

읽기 세트를 저장하려면 먼저 시퀀스 저장소를 생성합니다. 시퀀스 스토어를 생성할 때 선택적 Amazon S3 버킷을 대체 위치 및 S3 액세스 로그가 저장되는 위치로 지정할 수 있습니다. 대체 위치는 직접 업로드 중에 읽기 세트를 생성하지 못하는 파일을 저장하는 데 사용됩니다. 폴백 위치는 2023년 5월 15일 이후에 생성된 시퀀스 스토어에 사용할 수 있습니다. 시퀀스 스토어를 생성할 때 대체 위치를 지정합니다.

최대 5개의 읽기 세트 태그 키를 지정할 수 있습니다. 이러한 키 중 하나와 일치하는 태그 키로 읽기 세트를 생성하거나 업데이트하면 읽기 세트 태그가 해당 Amazon S3 객체로 전파됩니다. HealthOmics에서 생성한 시스템 태그는 기본적으로 전파됩니다.

주제

- 콘솔을 사용하여 시퀀스 스토어 생성
- CLI를 사용하여 시퀀스 스토어 생성
- 시퀀스 스토어 업데이트
- 시퀀스 스토어의 읽기 세트 태그 업데이트
- 게놈 파일 가져오기

콘솔을 사용하여 시퀀스 스토어 생성

시퀀스 저장소 생성

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 시퀀스 저장소를 선택합니다.
3. 시퀀스 스토어 생성 페이지에서 다음 정보를 제공합니다.
   - 시퀀스 저장소 이름 -이 저장소의 고유한 이름입니다.
   - 설명(선택 사항) -이 시퀀스 저장소에 대한 설명입니다.
4. S3의 폴백 위치에서 Amazon S3 위치를 지정합니다. HealthOmics는 직접 업로드 중에 읽기 세트를 생성하지 못하는 파일을 저장하기 위해 대체 위치를 사용합니다. HealthOmics 서비스에 Amazon S3 폴백 위치에 대한 쓰기 액세스 권한을 부여해야 합니다. 정책 예제는 대체 위치 구성을 참조하세요.

   폴백 위치는 2023년 5월 16일 이전에 생성된 시퀀스 스토어에는 사용할 수 없습니다.
5. (선택 사항) S3 전파를 위한 읽기 세트 태그 키의 경우 최대 5개의 읽기 세트 키를 입력하여 읽기 세트에서 기본 S3 객체로 전파할 수 있습니다. 읽기 세트의 태그를 S3 객체로 전파하면 Amazon S3 getObjectTagging API 작업을 통해 전파된 태그를 볼 수 있는 태그 및/또는 최종 사용자를 기반으로 Amazon S3 액세스 권한을 부여할 수 있습니다.
   a. 텍스트 상자에 키 값 하나를 입력합니다. 콘솔은 새 텍스트 상자를 생성하여 다음 키를 추가합니다.
   b. (선택 사항) 제거를 선택하여 모든 키를 제거합니다.
6. 데이터 암호화에서 데이터 암호화를에서 소유 및 관리할지 AWS 아니면 고객 관리형 CMK를 사용할지 선택합니다.
7. (선택 사항) S3 데이터 액세스에서 Amazon S3를 통해 시퀀스 스토어에 액세스하기 위한 새 역할 및 정책을 생성할지 여부를 선택합니다.
8. (선택 사항) S3 액세스 로깅에서 Amazon S3가 액세스 로그 레코드를 수집할`Enabled`지 여부를 선택합니다.

   S3의 액세스 로깅 위치에서 로그를 저장할 Amazon S3 위치를 지정합니다. 이 필드는 S3 액세스 로깅을 활성화한 경우에만 표시됩니다.
9. 태그(선택 사항) -이 시퀀스 스토어에 최대 50개의 태그를 제공합니다. 이러한 태그는 읽기 세트 가져오기/태그 업데이트 중에 설정된 읽기 세트 태그와 별개입니다.

스토어를 생성하면에 대한 준비가 된 것입니다게놈 파일 가져오기.

CLI를 사용하여 시퀀스 스토어 생성

다음 예제에서 *sequence store name*를 시퀀스 스토어에 대해 선택한 이름으로 바꿉니다.

```
aws omics create-sequence-store --name sequence store name --fallback-location "s3://
amzn-s3-demo-bucket"
```

새로 생성된 시퀀스 스토어의 ID 번호가 포함된 JSON으로 다음 응답을 받습니다.

```
{
    "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"
}
```

다음과 같이 list-sequence-stores 명령을 사용하여 계정과 연결된 모든 시퀀스 스토어를 볼 수도 있습니다.

```
aws omics list-sequence-stores
```

다음과 같은 응답을 받게 됩니다.

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

다음 예제와 같이 get-sequence-store를 사용하여 ID를 사용하여 시퀀스 저장소에 대해 자세히 알아볼 수 있습니다.

```
aws omics get-sequence-store --id sequence store ID
```

다음과 같은 응답을 받게 됩니다.

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

생성 후 여러 저장소 파라미터를 업데이트할 수도 있습니다. 이는 콘솔 또는 API updateSequenceStore 작업을 통해 수행할 수 있습니다.

시퀀스 스토어 업데이트

시퀀스 저장소를 업데이트하려면 다음 단계를 따릅니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 시퀀스 저장소를 선택합니다.

3. 업데이트할 시퀀스 스토어를 선택합니다.
4. 세부 정보 패널에서 편집을 선택합니다.
5. 세부 정보 편집 페이지에서 다음 필드를 업데이트할 수 있습니다.

   - 시퀀스 저장소 이름 -이 저장소의 고유한 이름입니다.
   - 설명 -이 시퀀스 저장소에 대한 설명입니다.
   - S3의 대체 위치, Amazon S3 위치를 지정합니다. HealthOmics는 직접 업로드 중에 읽기 세트를 생성하지 못하는 파일을 저장하기 위해 대체 위치를 사용합니다.
   - S3 전파를 위한 읽기 세트 태그 키 최대 5개의 읽기 세트 키를 입력하여 Amazon S3에 전파할 수 있습니다.
   - (선택 사항) S3 액세스 로깅에서 Amazon S3가 액세스 로그 레코드를 수집할`Enabled`지 여부를 선택합니다.

     S3의 액세스 로깅 위치에서 로그를 저장할 Amazon S3 위치를 지정합니다. 이 필드는 S3 액세스 로깅을 활성화한 경우에만 표시됩니다.
   - 태그(선택 사항) -이 시퀀스 스토어에 최대 50개의 태그를 제공합니다.

시퀀스 스토어의 읽기 세트 태그 업데이트

시퀀스 스토어의 읽기 세트 태그 또는 기타 필드를 업데이트하려면 다음 단계를 따르세요.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 시퀀스 저장소를 선택합니다.
3. 업데이트할 시퀀스 스토어를 선택합니다.
4. 세부 정보 탭을 선택하십시오.
5. 편집을 선택합니다.
6. 필요에 따라 새 읽기 세트 태그를 추가하거나 기존 태그를 삭제합니다.
7. 필요에 따라 이름, 설명, 대체 위치 또는 S3 데이터 액세스를 업데이트합니다.
8. 변경 사항 저장을 선택합니다.

게놈 파일 가져오기

게놈 파일을 시퀀스 저장소로 가져오려면 다음 단계를 따릅니다.

유전체학 파일을 가져오려면

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(☰)을 엽니다. 시퀀스 저장소를 선택합니다.
3. 시퀀스 저장소 페이지에서 파일을 가져올 시퀀스 저장소를 선택합니다.
4. 개별 시퀀스 스토어 페이지에서 유전체 파일 가져오기를 선택합니다.
5. 가져오기 세부 정보 지정 페이지에서 다음 정보를 제공합니다.
   - IAM 역할 - Amazon S3의 게놈 파일에 액세스할 수 있는 IAM 역할입니다.
   - 참조 유전체 -이 유전체 데이터의 참조 유전체입니다.
6. 가져오기 매니페스트 지정 페이지에서 다음 정보 매니페스트 파일을 지정합니다. 매니페스트 파일은 게놈 데이터의 필수 정보를 설명하는 JSON 또는 YAML 파일입니다. 매니페스트 파일에 대한 자세한 내용은 섹션을 참조하세요HealthOmics 시퀀스 스토어로 읽기 세트 가져오기.
7. 가져오기 작업 생성을 클릭합니다.

HealthOmics 참조 및 시퀀스 저장소 삭제

참조 및 시퀀스 저장소를 모두 삭제할 수 있습니다. 시퀀스 저장소는 읽기 세트를 포함하지 않는 경우에만 삭제할 수 있으며, 참조 저장소는 참조를 포함하지 않는 경우에만 삭제할 수 있습니다. 시퀀스 또는 참조 저장소를 삭제하면 해당 저장소와 연결된 태그도 삭제됩니다.

다음 예제에서는를 사용하여 참조 저장소를 삭제하는 방법을 보여줍니다 AWS CLI. 작업이 성공하면 응답을 받지 못합니다. 다음 예제에서는를 참조 스토어 ID*reference store ID*로 바꿉니다.

```
aws omics delete-reference-store --id reference store ID
```

다음 예제에서는 시퀀스 저장소를 삭제하는 방법을 보여줍니다. 작업이 성공하면 응답을 받지 못합니다. 다음 예제에서는를 시퀀스 스토어 ID*sequence store ID*로 바꿉니다.

```
aws omics delete-sequence-store --id sequence store ID
```

다음 예제와 같이 참조 스토어에서 참조를 삭제할 수도 있습니다. 참조는 읽기 세트, 변형 저장소 또는 주석 저장소에서 사용되지 않는 경우에만 삭제할 수 있습니다. 다음 예제에서를 참조 스토어 ID*reference store ID*로 바꾸고를 삭제하려는 참조의 ID*reference ID*로 바꿉니다.

```
aws omics delete-reference  --id reference ID --reference-store-id reference store ID
```

HealthOmics 시퀀스 스토어로 읽기 세트 가져오기

시퀀스 스토어를 생성한 후 데이터 스토어에 읽기 세트를 업로드하는 가져오기 작업을 생성합니다. Amazon S3 버킷에서 파일을 업로드하거나 동기 API 작업을 사용하여 직접 업로드할 수 있습니다. Amazon S3 버킷은 시퀀스 스토어와 동일한 리전에 있어야 합니다.

정렬된 읽기 세트와 정렬되지 않은 읽기 세트의 조합을 시퀀스 저장소에 업로드할 수 있지만 가져오기의 읽기 세트 중 하나라도 정렬된 경우 참조 유전체를 포함해야 합니다.

참조 저장소를 생성하는 데 사용한 IAM 액세스 정책을 재사용할 수 있습니다.

다음 주제에서는 시퀀스 스토어로 읽기 세트를 가져온 다음 가져온 데이터에 대한 정보를 가져오는 데 따르는 주요 단계를 설명합니다.

주제

- Amazon S3에 파일 업로드
- 매니페스트 파일 생성
- 가져오기 작업 시작
- 가져오기 작업 모니터링
- 가져온 시퀀스 파일 찾기
- 읽기 세트에 대한 세부 정보 가져오기
- 읽기 세트 데이터 파일 다운로드

Amazon S3에 파일 업로드

다음 예제에서는 파일을 Amazon S3 버킷으로 이동하는 방법을 보여줍니다.

```
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
```

이 예제에서 CRAM 사용되는 샘플 BAM 및 에는 서로 다른 유전체 참조인 Hg19 및가 필요합니다Hg38. 자세한 내용을 알아보거나 이러한 참조에 액세스하려면 오픈 데이터 레지스트리의 The Broad Genome References를 참조하세요 AWS.

매니페스트 파일 생성

또한 가져오기 작업을 모델링하려면 JSON으로 매니페스트 파일을 생성해야 합니다import.json(다음 예제 참조). 콘솔에서 시퀀스 저장소를 생성하는 경우 sequenceStoreId 또는를 지정할 필요가 roleARN없으므로 매니페스트 파일이 sources 입력으로 시작됩니다.

API manifest

다음 예제에서는 API를 사용하여 , FASTQ, BAM의 세 가지 읽기 세트를 가져옵니다CRAM.

```
{
  "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

이 예제 코드는 콘솔을 사용하여 단일 읽기 세트를 가져오는 데 사용됩니다.

```
[
```

```
  {
      "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"
  }
]
```

또는 매니페스트 파일을 YAML 형식으로 업로드할 수 있습니다.

가져오기 작업 시작

가져오기 작업을 시작하려면 다음 AWS CLI 명령을 사용합니다.

```
aws omics start-read-set-import-job --cli-input-json file://import.json
```

작업 생성이 성공했음을 나타내는 다음과 같은 응답을 받게 됩니다.

```
{
  "id": "3660451514",
  "sequenceStoreId": "3936421177",
  "roleArn": "arn:aws:iam::111122223333:role/OmicsImport",
  "status": "CREATED",
  "creationTime": "2022-07-13T22:14:59.309Z"
}
```

가져오기 작업 모니터링

가져오기 작업이 시작된 후 다음 명령을 사용하여 진행 상황을 모니터링할 수 있습니다. 다음 예제에서를 시퀀스 스토어 ID*sequence store id*로 바꾸고를 가져오기 ID*job import ID*로 바꿉니다.

```
aws omics get-read-set-import-job --sequence-store-id sequence store id --id job import
 ID
```

다음은 지정된 시퀀스 스토어 ID와 연결된 모든 가져오기 작업의 상태를 보여줍니다.

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

가져온 시퀀스 파일 찾기

작업이 완료되면 list-read-sets API 작업을 사용하여 가져온 시퀀스 파일을 찾을 수 있습니다. 다음 예제에서는를 시퀀스 스토어 ID*sequence store id*로 바꿉니다.

```
aws omics list-read-sets --sequence-store-id sequence store id
```

다음과 같은 응답을 받게 됩니다.

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

읽기 세트에 대한 세부 정보 가져오기

읽기 세트에 대한 자세한 내용을 보려면 GetReadSetMetadata API 작업을 사용합니다. 다음 예제에서를 시퀀스 스토어 ID*sequence store id*로 바꾸고를 읽기 세트 ID*read set id*로 바꿉니다.

```
aws omics get-read-set-metadata --sequence-store-id sequence store id --id read set id
```

다음과 같은 응답을 받게 됩니다.

```
{
"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"
}
```


읽기 세트 데이터 파일 다운로드

Amazon S3 GetObject API 작업을 사용하여 활성 읽기 세트의 객체에 액세스할 수 있습니다. 객체의 URI는 GetReadSetMetadata API 응답에 반환됩니다. 자세한 내용은 Amazon S3 URIs를 사용하여 HealthOmics 읽기 세트에 액세스 단원을 참조하십시오.

또는 HealthOmics GetReadSet API 작업을 사용합니다. GetReadSet를 사용하여 개별 부분을 다운로드하여 병렬로 다운로드할 수 있습니다. 이러한 부분은 Amazon S3 부분과 유사합니다. 다음은 읽

기 세트에서 파트 1을 다운로드하는 방법의 예입니다. 다음 예제에서를 시퀀스 스토어 ID*sequence store id*로 바꾸고를 읽기 세트 ID*read set id*로 바꿉니다.

```
aws omics get-read-set --sequence-store-id sequence store id --id read set id  --part-
number 1 outfile.bam
```

HealthOmics Transfer Manager를 사용하여 HealthOmics 참조 또는 읽기 세트에 대한 파일을 다운로드할 수도 있습니다. 여기에서 HealthOmics Transfer Manager를 다운로드할 수 있습니다. Transfer Manager 사용 및 설정에 대한 자세한 내용은이 GitHub 리포지토리를 참조하세요.

HealthOmics 시퀀스 스토어에 직접 업로드

HealthOmics Transfer Manager를 사용하여 시퀀스 스토어에 파일을 추가하는 것이 좋습니다. Transfer Manager 사용에 대한 자세한 내용은이 GitHub 리포지토리를 참조하세요. 직접 업로드 API 작업을 통해 시퀀스 스토어에 직접 읽기 세트를 업로드할 수도 있습니다.

직접 업로드 읽기 세트는 먼저 PROCESSING_UPLOAD 상태로 존재합니다. 즉, 파일 부분이 현재 업로드 중이며 읽기 세트 메타데이터에 액세스할 수 있습니다. 파트가 업로드되고 체크섬이 검증되면 읽기 세트는 가져온 읽기 세트ACTIVE와 동일하게 되고 동작합니다.

직접 업로드에 실패하면 읽기 세트 상태가 로 표시됩니다UPLOAD_FAILED. Amazon S3 버킷을 업로드에 실패한 파일의 대체 위치로 구성할 수 있습니다. 폴백 위치는 2023년 5월 15일 이후에 생성된 시퀀스 스토어에 사용할 수 있습니다.

주제

- 를 사용하여 시퀀스 스토어에 직접 업로드 AWS CLI
- 대체 위치 구성

를 사용하여 시퀀스 스토어에 직접 업로드 AWS CLI

시작하려면 멀티파트 업로드를 시작합니다. 다음 예제 AWS CLI와 같이를 사용하여이 작업을 수행할 수 있습니다.

AWS CLI 명령을 사용하여 직접 업로드하려면

1. 다음 예제와 같이 데이터를 분리하여 부분을 생성합니다.

```
split -b 100MiB SRR233106_1.filt.fastq.gz source1_part_
```

2. 소스 파일이 여러 부분으로 나누어진 후 다음 예제와 같이 멀티파트 읽기 세트 업로드를 생성합니다. *sequence store ID* 및 기타 파라미터를 시퀀스 스토어 ID 및 기타 값으로 바꿉니다.

```
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"
```

응답에서 uploadID 및 기타 메타데이터를 가져옵니다. 업로드 프로세스의 다음 단계에 uploadID를 사용합니다.

```
{
"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. 업로드에 읽기 세트를 추가합니다. 파일이 충분히 작으면이 단계를 한 번만 수행하면 됩니다. 더 큰 파일의 경우 파일의 각 부분에 대해이 단계를 수행합니다. 이전에 사용한 파트 번호를 사용하여 새 파트를 업로드하면 이전에 업로드한 파트를 덮어씁니다.

   다음 예제에서는 , *sequence store ID upload ID*및 기타 파라미터를 값으로 바꿉니다.

```
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
```

응답은 업로드된 파일이 의도한 파일과 일치하는지 확인하는 데 사용할 수 있는 ID입니다.

```
{
"checksum": "984979b9928ae8d8622286c4a9cd8e99d964a22d59ed0f5722e1733eb280e635"
}
```

4. 필요한 경우 파일의 부분을 계속 업로드합니다. 읽기 세트가 업로드되었는지 확인하려면 다음과 같이 list-read-set-upload-parts API 작업을 사용합니다. 다음 예제에서 , *upload ID*및 *sequence store ID* 를 자체 입력*part source*으로 바꿉니다.

```
aws omics list-read-set-upload-parts \
 --sequence-store-id sequence store ID \
 --upload-id upload ID \
 --part-source SOURCE1
```

응답은 가장 최근에 업데이트된 시점의 읽기 세트 수, 크기 및 타임스탬프를 반환합니다.

```
{
"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. 모든 활성 멀티파트 읽기 세트 업로드를 보려면 다음과 같이 list-multipart-read-set-uploads를 사용합니다. 를 자체 시퀀스 스토어의 ID*sequence store ID*로 바꿉니다.

```
aws omics list-multipart-read-set-uploads --sequence-store-id
             sequence store ID
```

이 API는 진행 중인 멀티파트 읽기 세트 업로드만 반환합니다. 수집된 읽기 세트가 이거나 업로드에 실패한 ACTIVE경우 list-multipart-read-set-uploads API에 대한 응답으로 업로드가 반환되지 않습니다. 활성 읽기 세트를 보려면 list-read-sets API를 사용합니다. list-multipart-read-set-uploads에 대한 예제 응답은 다음과 같습니다.

```
{
"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. 파일의 모든 부분을 업로드한 후 다음 예제와 같이 complete-multipart-read-set-upload를 사용하여 업로드 프로세스를 완료합니다. 파트의 *sequence store IDupload ID*, 및 파라미터를 자체 값으로 바꿉니다.

```
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-upload에 대한 응답은 가져온 읽기 세트의 읽기 세트 IDs입니다.

```
{
"readSetId": "0000000001"
}
```

7. 업로드를 중지하려면 업로드 ID와 함께 abort-multipart-read-set-upload를 사용하여 업로드 프로세스를 종료합니다. *sequence store ID* 및 *upload ID*를 고유한 파라미터 값으로 바꿉니다.

```
aws omics abort-multipart-read-set-upload \
--sequence-store-id sequence store ID \
--upload-id upload ID
```

8. 업로드가 완료되면 다음과 같이 get-read-set를 사용하여 읽기 세트에서 데이터를 검색합니다. 업로드가 아직 처리 중인 경우 get-read-set는 제한된 메타데이터를 반환하고 생성된 인덱스 파일은 사용할 수 없습니다. *sequence store ID* 및 기타 파라미터를 자체 입력으로 바꿉니다.

```
aws omics get-read-set
 --sequence-store-id sequence store ID \
 --id read set ID \
 --file SOURCE1 \
 --part-number 1 myfile.fastq.gz
```

9. 업로드 상태를 포함한 메타데이터를 확인하려면 get-read-set-metadata API 작업을 사용합니다.

```
aws omics get-read-set-metadata --sequence-store-id sequence store ID --id read set
 ID
```

응답에는 파일 유형, 참조 ARN, 파일 수, 시퀀스 길이와 같은 메타데이터 세부 정보가 포함됩니다. 또한 상태가 포함됩니다. 가능한 상태는 PROCESSING_UPLOAD, ACTIVE및 입니다UPLOAD_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"
```

```
}
```

대체 위치 구성

시퀀스 스토어를 생성하거나 업데이트할 때 Amazon S3 버킷을 업로드에 실패한 파일의 대체 위치로 구성할 수 있습니다. 이러한 읽기 세트의 파일 부분은 대체 위치로 전송됩니다. 폴백 위치는 2023년 5월 15일 이후에 생성된 시퀀스 스토어에 사용할 수 있습니다.

다음 예제와 같이 Amazon S3 버킷 정책을 생성하여 HealthOmics에 Amazon S3 폴백 위치에 대한 쓰기 액세스 권한을 부여합니다.

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "omics.amazonaws.com"
    },
    "Action": "s3:PutObject",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
}
```

폴백 또는 액세스 로그용 Amazon S3 버킷이 고객 관리형 키를 사용하는 경우 키 정책에 다음 권한을 추가합니다.

```
 {
    "Sid": "Allow use of key",
    "Effect": "Allow",
    "Principal": {
        "Service": "omics.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

Amazon S3 버킷으로 HealthOmics 읽기 세트 내보내기

Amazon S3 버킷으로 배치 내보내기 작업으로 읽기 세트를 내보낼 수 있습니다. 이렇게 하려면 먼저 다음 IAM 정책 예제와 마찬가지로 버킷에 대한 쓰기 액세스 권한이 있는 IAM 정책을 생성합니다.

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 정책이 적용되면 읽기 세트 내보내기 작업을 시작합니다. 다음 예제에서는 start-read-set-export-job API 작업을 사용하여이 작업을 수행하는 방법을 보여줍니다. 다음 예제에서는 , , *sequence store ID* *destination* 및 *role ARN*와 같은 모든 파라미터를 입력*sources*으로 바꿉니다.

```
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
```

오리진 시퀀스 스토어 및 대상 Amazon S3 버킷에 대한 정보와 함께 다음 응답을 받습니다.

```
{
"id": <job-id>,
"sequenceStoreId": <sequence-store-id>,
"destination": <destination-s3-uri>,
"status": "SUBMITTED",
"creationTime": "2022-10-22T01:33:38.079000+00:00"
}
```

작업이 시작된 후 다음과 같이 get-read-set-export-job API 작업을 사용하여 상태를 확인할 수 있습니다. *sequence store ID* 및를 각각 시퀀스 스토어 ID 및 작업 ID*job ID*로 바꿉니다.

```
aws omics get-read-set-export-job --id job-id --sequence-store-id sequence store ID
```

다음과 같이 list-read-set-export-jobs API 작업을 사용하여 시퀀스 스토어에 대해 초기화된 모든 내보내기 작업을 볼 수 있습니다. 를 시퀀스 스토어 ID*sequence store ID*로 바꿉니다.

```
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"
  }
]
}
```

읽기 세트를 내보내는 것 외에도 Amazon S3 액세스 URIs. 자세한 내용은 Amazon S3 URIs를 사용하여 HealthOmics 읽기 세트에 액세스를 참조하세요.

Amazon S3 URIs를 사용하여 HealthOmics 읽기 세트에 액세스

Amazon S3 URI 경로를 사용하여 활성 시퀀스 스토어 읽기 세트에 액세스할 수 있습니다.

Amazon S3 URI 경로를 사용하면 Amazon S3 작업을 사용하여 읽기 세트를 나열, 공유 및 다운로드할 수 있습니다. S3 APIs 가속화합니다. S3 또한 S3 APIs에 대한 액세스를 다른 계정과 공유하고 데이터에 대한 리전 간 읽기 액세스를 제공할 수 있습니다.

HealthOmics는 아카이브된 읽기 세트에 대한 Amazon S3 URI 액세스를 지원하지 않습니다. 읽기 세트를 활성화하면 매번 동일한 URI 경로로 복원됩니다.

데이터를 HealthOmics 스토어에 로드하면 Amazon S3 URI가 Amazon S3 액세스 포인트를 기반으로 하므로 다음과 같이 Amazon S3 URIs를 읽는 업계 표준 도구와 직접 통합할 수 있습니다.

- Integrative Genomics Viewer(IGV) 또는 UCSC Genome Browser와 같은 시각적 분석 애플리케이션.
- CWL, WDL, Nextflow와 같은 Amazon S3 확장을 사용하는 일반적인 워크플로입니다.
- 액세스 포인트 Amazon S3 URIs 또는 미리 서명된 Amazon S3 URIs.
- Mountpoint 또는 CloudFront와 같은 Amazon S3 유틸리티.

Amazon S3 Mountpoint를 사용하면 Amazon S3 버킷을 로컬 파일 시스템으로 사용할 수 있습니다. Mountpoint에 대해 자세히 알아보고 사용하기 위해 설치하려면 Mountpoint for Amazon S3를 참조하세요.

Amazon CloudFront는 고성능, 보안 및 개발자 편의를 위해 구축된 콘텐츠 전송 네트워크(CDN) 서비스입니다. Amazon CloudFront 사용에 대한 자세한 내용은Amazon CloudFront 설명서를 참조하세요. 시퀀스 스토어로 CloudFront를 설정하려면 AWS HealthOmics 팀에 문의하세요.

데이터 소유자 루트 계정은 시퀀스 스토어 접두사의 S3:GetObject, S3:GetObjectTagging 및 S3:List 버킷 작업에 대해 활성화됩니다. 계정의 사용자가 데이터에 액세스할 수 있도록 IAM 정책을 생성하여 사용자 또는 역할에 연결합니다. 정책 예제는 Amazon S3 URIs를 사용한 데이터 액세스 권한을 참조하세요.

활성 읽기 세트에서 다음 Amazon S3 API 작업을 사용하여 데이터를 나열하고 검색할 수 있습니다. Amazon S3 URIs 활성화된 후 이를 통해 아카이브된 읽기 세트에 액세스할 수 있습니다.

- GetObject - Amazon S3에서 객체를 검색합니다.

- HeadObject - HEAD 작업은 객체 자체를 반환하지 않고 객체에서 메타데이터를 검색합니다. 이 작업은 객체의 메타데이터만 원하는 경우에 유용합니다.
- ListObjects 및 ListObject v2 - 버킷에 있는 객체의 일부 또는 전부(최대 1,000개)를 반환합니다.
- CopyObject - Amazon S3에 이미 저장된 객체의 복사본을 생성합니다. HealthOmics는 Amazon S3 액세스 포인트로의 복사를 지원하지만 액세스 포인트에 쓸 수는 없습니다.

HealthOmics 시퀀스 스토어는 ETags를 통해 파일의 의미 체계 ID를 유지합니다. 파일의 수명 주기 동안 비트 ID를 기반으로 하는 Amazon S3 ETag는 변경될 수 있지만 HealthOmics ETag는 동일하게 유지됩니다. 자세한 내용은 HealthOmics ETags 및 데이터 출처를 참조하세요.

주제

- HealthOmics 스토리지의 Amazon S3 URI 구조
- 호스팅 또는 로컬 IGV를 사용하여 읽기 세트 액세스
- HealthOmics에서 Samtools 또는 HTSlib 사용
- Mountpoint HealthOmics 사용
- HealthOmics에서 CloudFront 사용

HealthOmics 스토리지의 Amazon S3 URI 구조

Amazon S3 URIs 있는 모든 파일에는 `omics:subjectId` 및 `omics:sampleId` 리소스 태그가 있습니다. 이러한 태그를 사용하여와 같은 패턴을 통해 IAM 정책을 사용하여 액세스를 공유할 수 있습니다`"s3:ExistingObjectTag/omics:subjectId": "pattern desired"`.

파일 구조는 다음과 같습니다.

`.../`*`account_id`*`/sequenceStore/`*`seq_store_id`*`/readSet/`*`read_set_id`*`/`*`files`*`.`

Amazon S3에서 시퀀스 스토어로 가져온 파일의 경우 시퀀스 스토어는 원래 소스 이름을 유지하려고 시도합니다. 이름이 충돌하면 시스템은 읽기 세트 정보를 추가하여 파일 이름이 고유한지 확인합니다. 예를 들어 fastq 읽기 세트의 경우 두 파일 이름이 동일한 경우 이름을 고유하게 만들기 위해가 .fastq.gz 또는 .fq.gz 앞에 삽입`sourceX`됩니다. 직접 업로드의 경우 파일 이름은 다음 패턴을 따릅니다.

- FASTQ의 경우 - *`read_set_name_sourcex`*.fastq.gz
- uBAM/BAM/CRAM의 경우 - 확장자가 `.bam` 또는 인 *`read_set_name.file`* 확장입니다`.cram`. 예를 들면, `NA193948.bam`입니다.

BAM 또는 CRAM인 읽기 세트의 경우 인덱스 파일은 수집 프로세스 중에 자동으로 생성됩니다. 생성된 인덱스 파일의 경우 파일 이름 끝에 적절한 인덱스 확장명이 적용됩니다. *#### ## ## ### <name>.<file index extension> ### ####.* 인덱스 확장자는 .bai 또는 입니다.crai.

호스팅 또는 로컬 IGV를 사용하여 읽기 세트 액세스

IGV는 BAM 및 CRAM 파일을 분석하는 데 사용되는 유전체 브라우저입니다. 한 번에 유전체의 일부만 표시하기 때문에 파일과 인덱스가 모두 필요합니다. IGV는 로컬에서 다운로드하여 사용할 수 있으며 AWS 호스팅 IGV를 생성하는 방법에 대한 가이드가 있습니다. 퍼블릭 웹 버전은 CORS가 필요하므로 지원되지 않습니다.

로컬 IGV는 로컬 AWS 구성을 사용하여 파일에 액세스합니다. 해당 구성에 사용되는 역할에 액세스 중인 읽기 세트의 s3 URI에 대한 kms:Decrypt 및 s3:GetObject 권한을 활성화하는 정책이 연결되어 있는지 확인합니다. 그런 다음 IGV에서 “파일 > URL에서 로드”를 사용하고 소스 및 인덱스의 URI에 붙여넣을 수 있습니다. 또는 미리 서명된 URLs 동일한 방식으로 생성하고 사용할 수 있으며, 이는 AWS 구성을 우회합니다. CORS는 Amazon S3 URI 액세스에서 지원되지 않으므로 CORS에 의존하는 요청은 지원되지 않습니다.

AWS 호스팅 IGV 예제는 AWS Cognito를 사용하여 환경 내에서 올바른 구성과 권한을 생성합니다. 액세스 중인 읽기 세트의 Amazon S3 URI에 대한kms:Decrypt 및 s3:GetObject 권한을 활성화하는 정책이 생성되었는지 확인하고이 정책을 Cognito 사용자 풀에 할당된 역할에 추가합니다. 그런 다음 IGV에서 “파일 > URL에서 로드”를 사용하고 소스 및 인덱스의 URI에를 입력할 수 있습니다. 또는 AWS 구성을 우회하는 동일한 방식으로 미리 서명된 URLs을 생성하고 사용할 수 있습니다.

는 AWS 프로파일이 구성된 리전에서 사용자가 소유한 버킷만 표시하므로 시퀀스 스토어는 “Amazon” 탭 아래에 표시되지 않습니다.

HealthOmics에서 Samtools 또는 HTSlib 사용

HTSlib는 Samtools, rSamtools, PySam 등과 같은 여러 도구에서 공유하는 코어 라이브러리입니다. HTSlib 버전 1.20 이상을 사용하여 Amazon S3 액세스 포인트를 원활하게 지원합니다. 이전 버전의 HTSlib 라이브러리의 경우 다음 해결 방법을 사용할 수 있습니다.

- 를 사용하여 HTS Amazon S3 호스트의 환경 변수를 설정합니다`export HTS_S3_HOST="s3.`*region*`.amazonaws.com"`.
- 사용하려는 파일에 대해 미리 서명된 URL을 생성합니다. BAM 또는 CRAM을 사용하는 경우 파일과 인덱스 모두에 대해 미리 서명된 URL이 생성되었는지 확인합니다. 그런 다음 두 파일을 라이브러리와 함께 사용할 수 있습니다.

- Mountpoint를 사용하여 HTSlib 라이브러리를 사용하는 동일한 환경에 시퀀스 스토어 또는 읽기 세트 접두사를 탑재합니다. 여기에서 로컬 파일 경로를 사용하여 파일에 액세스할 수 있습니다.

Mountpoint HealthOmics 사용

Mountpoint for Amazon S3는 Amazon S3 버킷을 로컬 파일 시스템으로 탑재하기 위한 간단한 대용량 파일 클라이언트입니다. Mountpoint for Amazon S3를 사용하면 애플리케이션이 열기 및 읽기와 같은 파일 작업을 통해 Amazon S3에 저장된 객체에 액세스할 수 있습니다. Mountpoint for Amazon S3는 이러한 작업을 Amazon S3 객체 API 호출로 자동 변환하여 애플리케이션이 파일 인터페이스를 통해 Amazon S3의 탄력적 스토리지 및 처리량에 액세스할 수 있도록 합니다.

Mountpoint 설치 지침을 사용하여 Mountpoint를 설치할 수 있습니다. Mountpoint는 설치에 로컬이고 Amazon S3 접두사 수준에서 작동하는 AWS 프로파일을 사용합니다. 사용 중인 프로필에 액세스 중인 읽기 세트 또는 시퀀스 스토어의 Amazon S3 URI 접두사에 대한 s3:GetObject, s3:ListBucket 및 kms:Decrypt 권한을 활성화하는 정책이 있는지 확인합니다. 그런 다음 다음 경로를 사용하여 버킷을 탑재할 수 있습니다.

```
mount-s3 access point arn local path to mount --prefix prefix to sequence store or read
 set --region region
```

HealthOmics에서 CloudFront 사용

Amazon CloudFront는 고성능, 보안 및 개발자 편의를 위해 구축된 콘텐츠 전송 네트워크(CDN) 서비스입니다. CloudFront를 사용하려는 고객은 서비스 팀과 협력하여 CloudFront 배포를 활성화해야 합니다. 계정 팀과 협력하여 HealthOmics 서비스 팀을 참여시킵니다.

HealthOmics에서 읽기 세트 활성화

다음 예제와 AWS CLI같이 start-read-set-activation-job API 작업 또는를 통해 아카이브된 읽기 세트를 활성화할 수 있습니다. *sequence store ID* 및를 시퀀스 스토어 ID 및 읽기 세트 ID*read set id*로 바꿉니다. 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
```

다음과 같이 활성화 작업 정보가 포함된 응답을 받게 됩니다.

```
{
    "id": "12345678",
    "sequenceStoreId": "1234567890",
    "status": "SUBMITTED",
    "creationTime": "2022-10-22T00:50:54.670000+00:00"
}
```

활성화 작업이 시작된 후 get-read-set-activation-job API 작업으로 진행 상황을 모니터링할 수 있습니다. 다음은를 사용하여 활성화 작업 상태를 AWS CLI 확인하는 방법의 예입니다. *job ID* 및를 각각 시퀀스 스토어 ID 및 작업 ID*sequence store ID*로 바꿉니다. IDs

```
aws omics get-read-set-activation-job --id job ID --sequence-store-id sequence store ID
```

응답은 다음과 같이 활성화 작업을 요약합니다.

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

get-read-set-metadata API 작업을 사용하여 활성화 작업의 상태를 확인할 수 있습니다. 가능한 상태는 ACTIVE, ACTIVATING및 입니다ARCHIVED. 다음 예제에서를 시퀀스 스토어 ID*sequence store ID*로 바꾸고를 읽기 세트 ID*read set ID*로 바꿉니다.

```
aws omics get-read-set-metadata --sequence-store-id sequence store ID --id read set ID
```

다음 응답은 읽기 세트가 활성 상태임을 보여줍니다.


```
{
    "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"
    }
}
```

다음 예제와 같이 list-read-set-activation-jobs를 사용하여 모든 읽기 세트 활성화 작업을 볼 수 있습니다. 다음 예제에서는를 시퀀스 스토어 ID*sequence store ID*로 바꿉니다.

```
aws omics list-read-set-activation-jobs --sequence-store-id sequence store ID
```

다음과 같은 응답을 받게 됩니다.

```
{
    "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 분석

> **Important**
>
> AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경 단원을 참조하십시오.

HealthOmics 분석은 게놈 변형 및 주석의 저장 및 분석을 지원합니다. Analytics는 변형 저장소와 주석 저장소라는 두 가지 유형의 스토리지 리소스를 제공합니다. 이러한 리소스를 사용하여 유전체 변형 데이터 및 주석 데이터를 저장, 변환 및 쿼리합니다. 데이터를 데이터 스토어로 가져온 후 Athena를 사용하여 데이터에 대한 고급 분석을 수행할 수 있습니다.

HealthOmics 콘솔 또는 API를 사용하여 저장소를 생성 및 관리하고, 데이터를 가져오고, 분석 저장소 데이터를 공동 작업자와 공유할 수 있습니다.

변형 저장소는 VCF 형식의 데이터를 지원하고 주석 저장소는 TSV/CSV 및 GFF3 형식을 지원합니다. 유전체 좌표는 0 기반 반닫힌 반개방 간격으로 표시됩니다. 데이터가 HealthOmics 분석 데이터 스토어에 있는 경우 VCF 파일에 대한 액세스는를 통해 관리됩니다 AWS Lake Formation. 그런 다음 Amazon Athena를 사용하여 VCF 파일을 쿼리할 수 있습니다. 쿼리는 Athena 쿼리 엔진 버전 3을 사용해야 합니다. Athena 쿼리 엔진 버전에 대한 자세한 내용은 Amazon Athena 설명서를 참조하세요.

주제

- HealthOmics 변형 저장소 생성
- HealthOmics 변형 저장소 가져오기 작업 생성
- HealthOmics 주석 저장소 생성
- HealthOmics 주석 저장소에 대한 가져오기 작업 생성
- HealthOmics 주석 저장소 버전 생성
- HealthOmics 분석 저장소 삭제
- HealthOmics 분석 데이터 쿼리
- HealthOmics 분석 스토어 공유

HealthOmics 변형 저장소 생성

> **Important**
>
> AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경 단원을 참조하십시오.

다음 주제에서는 콘솔 및 API를 사용하여 HealthOmics 변형 저장소를 생성하는 방법을 설명합니다.

주제

콘솔을 사용하여 변형 저장소 생성

HealthOmics 콘솔을 사용하여 변형 저장소를 생성할 수 있습니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(™)을 엽니다. 변형 저장소를 선택합니다.
3. 변형 저장소 생성 페이지에서 다음 정보를 제공합니다.
   - 변형 저장소 이름 -이 저장소의 고유한 이름입니다.
   - 설명(선택 사항) -이 변형 저장소에 대한 설명입니다.
   - 참조 유전체 -이 변형 저장소의 참조 유전체입니다.
   - 데이터 암호화 - 데이터 암호화를 직접 소유 및 관리할지 여부를 선택합니다 AWS .
   - 태그(선택 사항) -이 변형 저장소에 최대 50개의 태그를 제공합니다.
4. 변형 저장소 생성을 선택합니다.

API를 사용하여 변형 저장소 생성

HealthOmics `CreateVariantStore` API 작업을 사용하여 변형 저장소를 생성합니다. 를 사용하여이 작업을 수행할 수도 있습니다 AWS CLI.

변형 저장소를 생성하려면 저장소의 이름과 참조 저장소의 ARN을 제공합니다. 변형 저장소는 상태가 READY로 변경되면 데이터를 수집할 준비가 된 것입니다.

다음 예제에서는 AWS CLI 를 사용하여 변형 저장소를 생성합니다.

```
aws omics create-variant-store --name myvariantstore \
    --reference referenceArn="arn:aws:omics:us-
west-2:555555555555:referenceStore/123456789/reference/5987565360"
```

변형 저장소 생성을 확인하려면 다음 응답을 받습니다.

```
{
    "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"
}
```

변형 저장소에 대해 자세히 알아보려면 get-variant-store API를 사용합니다.

```
aws omics get-variant-store --name myvariantstore
```

다음과 같은 응답을 받게 됩니다.

```
{
    "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
```

```
}
```

계정과 연결된 모든 변형 저장소를 보려면 list-variant-stores API를 사용합니다.

```
aws omics list-variant-stores
```

다음 예제 응답과 같이 ID, IDs 상태 및 기타 세부 정보와 함께 모든 변형 저장소를 나열하는 응답을 받습니다.

```
{
    "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
        }
    ]
}
```

상태 또는 기타 기준에 따라 list-variant-stores API에 대한 응답을 필터링할 수도 있습니다.

2023년 5월 15일 이후에 생성된 분석 저장소로 가져온 VCF 파일은 변형 효과 예측기(VEP) 주석에 대한 스키마를 정의했습니다. 이렇게 하면 가져온 VCF 데이터를 더 쉽게 쿼리하고 구문 분석할 수 있습니다. `annotation fields` 파라미터가 API 또는 CLI 호출에 포함된 경우를 제외하고 변경 사항은 2023년 5월 15일 이전에 생성된 스토어에는 영향을 주지 않습니다. 이러한 스토어의 경우 `annotation fields` 파라미터를 사용하면 요청이 실패합니다.

HealthOmics 변형 저장소 가져오기 작업 생성

> **Important**
>
> AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경 단원을 참조하십시오.

다음 예제에서는를 사용하여 변형 저장소 AWS CLI 에 대한 가져오기 작업을 생성하는 방법을 보여줍니다.

```
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"}
    ]
}
```

2023년 5월 15일 이후에 생성된 스토어의 경우 다음 예제에서는 `--annotation-fields` 파라미터를 추가하는 방법을 보여줍니다. 주석 필드는 가져오기로 정의됩니다.

```
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-job을 사용하여 상태를 확인합니다.

```
aws omics get-variant-import-job --job-id 08279950-a9e3-4cc3-9a3c-a574f9c9e229
```

가져오기 작업의 상태를 보여주는 JSON 응답을 받게 됩니다. VCF의 VEP 주석은 INFO 열에 ID/값 페어로 저장된 정보에 대해 구문 분석됩니다. 변형 효과 예측기 주석 앙상블 INFO 열의 기본 ID는 CSQ이지만 `--annotation-fields` 파라미터를 사용하여 INFO 열에 사용되는 사용자 지정 값을 나타낼 수 있습니다. 구문 분석은 현재 VEP 주석에 대해 지원됩니다.

2023년 5월 15일 이전에 생성된 저장소 또는 VEP 주석이 포함되지 않은 VCF 파일의 경우 응답에 주석 필드가 포함되지 않습니다.

```
{
    "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 파일의 일부인 VEP 주석은 다음 구조의 사전 정의된 스키마로 저장됩니다. 추가 항목 필드는 기본 스키마에 포함되지 않은 추가 VEP 필드를 저장하는 데 사용할 수 있습니다.

```
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>
    >>
>
```

구문 분석은 최선의 방법으로 수행됩니다. VEP 항목이 VEP 표준 사양을 따르지 않으면 구문 분석되지 않고 배열의 행이 비어 있습니다.

새 변형 저장소의 경우 get-variant-import-job에 대한 응답에는 다음과 같이 주석 필드가 포함됩니다.

```
aws omics get-variant-import-job --job-id 08279950-a9e3-4cc3-9a3c-a574f9c9e229
```

가져오기 작업의 상태를 보여주는 JSON 응답을 받습니다.

```
{
    "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-jobs를 사용하여 모든 가져오기 작업과 해당 상태를 볼 수 있습니다.

```
aws omics list-variant-import-jobs --ids 7a1c67e3-b7f9-434d-817b-9c571fd63bea
```

응답에는 다음과 같은 정보가 포함됩니다.

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

필요한 경우 다음 명령을 사용하여 가져오기 작업을 취소할 수 있습니다.

```
aws omics cancel-variant-import-job
     --job-id edd7b8ce-xmpl-47e2-bc99-258cac95a508
```

HealthOmics 주석 저장소 생성

> **Important**
>
> AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경 단원을 참조하십시오.

주석 저장소는 TSV, VCF 또는 GFF 파일의 것과 같은 주석 데이터베이스를 나타내는 데이터 저장소입니다. 동일한 참조 유전체가 지정되면 주석 저장소는 가져오기 중에 변형 저장소와 동일한 좌표계에 매핑됩니다. 다음 주제에서는 HealthOmics 콘솔 및를 사용하여 주석 저장소 AWS CLI 를 생성하고 관리하는 방법을 보여줍니다.

주제

콘솔을 사용하여 주석 저장소 생성

다음 절차에 따라 HealthOmics 콘솔을 사용하여 주석 저장소를 생성합니다.

주석 저장소를 생성하려면

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(☰)을 엽니다. 주석 저장소를 선택합니다.
3. 주석 저장소 페이지에서 주석 저장소 생성을 선택합니다.
4. 주석 저장소 생성 페이지에서 다음 정보를 제공합니다.
   - 주석 저장소 이름 -이 저장소의 고유한 이름입니다.
   - 설명(선택 사항) -이 참조 유전체에 대한 설명입니다.
   - 데이터 형식 및 스키마 세부 정보 - 데이터 파일 형식을 선택하고이 저장소에 대한 스키마 정의를 업로드합니다.
   - 참조 유전체 -이 주석의 참조 유전체입니다.
   - 데이터 암호화 - 데이터 암호화를 직접 소유 및 관리할지 여부를 선택합니다 AWS .

- 태그(선택 사항) -이 주석 저장소에 최대 50개의 태그를 제공합니다.

5. 주석 저장소 생성을 선택합니다.

API를 사용하여 주석 저장소 생성

다음 예제에서는를 사용하여 주석 저장소를 생성하는 방법을 보여줍니다 AWS CLI. 모든 AWS CLI 및 API 작업에 대해 데이터 형식을 지정해야 합니다.

```
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
```

주석 저장소 생성을 확인하는 다음 응답을 받게 됩니다.

```
{
           "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"
       }
```

주석 저장소에 대해 자세히 알아보려면 get-annotation-store API를 사용합니다.

```
aws omics get-annotation-store --name my_annotation_store
```

다음과 같은 응답을 받게 됩니다.

```
{
          "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
    }
```

계정과 연결된 모든 주석 저장소를 보려면 list-annotation-stores API 작업을 사용합니다.

```
aws omics list-annotation-stores
```

다음 예제 응답과 같이 ID, IDs 상태 및 기타 세부 정보와 함께 모든 주석 저장소를 나열하는 응답을 받습니다.

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

상태 또는 기타 기준에 따라 응답을 필터링할 수도 있습니다.

HealthOmics 주석 저장소에 대한 가져오기 작업 생성

> **Important**
>
> AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경 단원을 참조하십시오.

주제

API를 사용하여 주석 가져오기 작업 생성

다음 예제에서는를 사용하여 주석 가져오기 작업을 AWS CLI 시작하는 방법을 보여줍니다.

```
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"}'
```

2023년 5월 15일 이전에 생성된 주석 저장소는 주석 필드가 포함된 경우 오류 메시지를 반환합니다. 주석 저장소 가져오기 작업과 관련된 API 작업에 대한 출력은 반환되지 않습니다.

그런 다음 get-annotation-import-job API 작업과 `job ID` 파라미터를 사용하여 주석 가져오기 작업에 대한 자세한 내용을 알아볼 수 있습니다.

```
aws omics get-annotation-import-job --job-id 9e4198fb-fa85-446c-9301-9b823a1a8ba8
```

주석 필드를 포함하여 다음과 같은 응답을 받게 됩니다.

```
{
          "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"}
       }
```

모든 주석 저장소 가져오기 작업을 보려면 list-annotation-import-jobs를 사용합니다.

```
aws omics list-annotation-import-jobs --ids 9e4198fb-fa85-446c-9301-9b823a1a8ba8
```

응답에는 주석 저장소 가져오기 작업의 세부 정보와 상태가 포함됩니다.

```
{
          "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 및 VCF 형식에 대한 추가 파라미터

TSV 및 VCF 형식의 경우 입력 구문 분석 방법을 API에 알려주는 추가 파라미터가 있습니다.

> **Important**
>
> 쿼리 엔진으로 내보낸 CSV 주석 데이터는 데이터 세트 가져오기의 정보를 직접 반환합니다. 가져온 데이터에 수식 또는 명령이 포함된 경우 파일에 CSV 삽입이 적용될 수 있습니다. 따라서 쿼리 엔진으로 내보낸 파일은 보안 경고를 표시할 수 있습니다. 악의적인 활동을 방지하려면 내보내기 파일을 읽을 때 링크와 매크로를 끄세요.

또한 TSV 구문 분석기는 다음 표에 나열된 유전체 좌표의 왼쪽 정규화 및 표준화와 같은 기본 생물 정보학 작업을 수행합니다.

형식 유형	설명
Generic	일반 텍스트 파일. 게놈 정보가 없습니다.
CHR_POS	시작 위치 - 1,와 동일한 종료 위치 추가POS.
CHR_POS_REF_ALT	contig, 1-base 위치, ref 및 alt 대립형질 정보를 포함합니다.
CHR_START_END_REF_ALT_ONE_BASE	contig, start, end, ref 및 alt 대립형질 정보를 포함합니다. 좌표는 1 기반입니다.
CHR_START_END_ZERO_BASE	contig, start 및 end 위치를 포함합니다. 좌표는 0 기반입니다.
CHR_START_END_ONE_BASE	contig, start 및 end 위치를 포함합니다. 좌표는 1 기반입니다.
CHR_START_END_REF_ALT_ZERO_BASE	contig, start, end, ref 및 alt 대립형질 정보를 포함합니다. 좌표는 0 기반입니다.

TSV 가져오기 주석 저장소 요청은 다음 예제와 같습니다.

```
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 형식의 주석 저장소 생성

다음 예제에서는 헤더, 행 및 설명이 포함된 탭 제한 파일을 사용하여 주석 저장소를 생성합니다. 좌표는 이며CHR_START_END_ONE_BASED, OMIM의 인적 Gene Map 개요에 있는 HG19 Gene Map이 포함되어 있습니다. 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}]}'
```

헤더를 사용하거나 사용하지 않고 파일을 가져올 수 있습니다. CLI 요청에서 이를 나타내려면 다음 가져오기 작업 예제와 header=false같이를 사용합니다.

```
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="#"}}'
```

다음 예시에서는 침대 파일에 대한 주석 저장소를 생성합니다. 침대 파일은 탭으로 구분된 간단한 파일입니다. 이 예제에서 열은 염색체, 시작, 끝 및 리전 이름입니다. 좌표는 0을 기반으로 하며 데이터에 헤더가 없습니다.

```
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}]}'
```

그런 다음 CLI 명령을 사용하여 Bed 파일을 주석 저장소로 가져올 수 있습니다.

```
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="#"}}'
```

다음 예제에서는 VCF 파일의 처음 몇 개의 열과 주석 정보가 있는 열이 포함된 탭으로 구분된 파일에 대한 주석 저장소를 생성합니다. 여기에는 염색체, 시작, 참조 및 대체 대립형질에 대한 정보가 포함된 유전체 위치가 포함되며 헤더가 포함됩니다.

```
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}]}'
```

그런 다음 CLI 명령을 사용하여 파일을 주석 저장소로 가져옵니다.

```
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="#"}}'
```

다음 예제에서는 고객이 mim2gene 파일에 대한 주석 저장소를 생성하는 방법을 보여줍니다. mim2gene 파일은 OMIM의 유전자와 다른 유전자 식별자 간의 링크를 제공합니다. 탭이 구분되어 있으며 주석이 포함되어 있습니다.

```
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}]}'
```

그런 다음 다음과 같이 스토어로 데이터를 가져올 수 있습니다.

```
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 형식의 가져오기 작업 시작

VCF 파일의 경우 표시된 대로 해당 파라미터를 무시하거나 포함하는 `ignoreQualField` 및 `ignoreFilterField`라는 두 개의 추가 입력이 있습니다.

```
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
    }
   }'
```

그림과 같이 주석 저장소 가져오기를 취소할 수도 있습니다. 취소에 성공하면이 AWS CLI 호출에 대한 응답을 받지 못합니다. 그러나 가져오기 작업 ID를 찾을 수 없거나 가져오기 작업이 완료되면 오류 메시지가 표시됩니다.

```
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 및 list-variant-import-jobs에 대한 메타데이터 가져오기 작업 기록은 2년 후에 자동으로 삭제됩니다. 가져온 변형 및 주석 데이터는 자동으로 삭제되지 않으며 데이터 스토어에 남아 있습니다.

HealthOmics 주석 저장소 버전 생성

> **Important**
>
> AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경 단원을 참조하십시오.

새 버전의 주석 저장소를 생성하여 다양한 버전의 주석 데이터베이스를 수집할 수 있습니다. 이렇게 하면 정기적으로 업데이트되는 주석 데이터를 구성하는 데 도움이 됩니다.

기존 주석 저장소의 새 버전을 생성하려면 다음 예제와 같이 create-annotation-store-version API를 사용합니다.

```
aws omics create-annotation-store-version \
     --name my_annotation_store \
     --version-name my_version
```

주석 저장소 버전 ID로 다음 응답을 받아 주석의 새 버전이 생성되었음을 확인합니다.

```
{
     "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"
}
```

주석 저장소 버전의 설명을 업데이트하려면 update-annotation-store-version을 사용하여 주석 저장소 버전에 업데이트를 추가할 수 있습니다.

```
aws omics update-annotation-store-version \
    --name my_annotation_store \
    --version-name my_version \
```

```
    --description "New Description"
```

주석 저장소 버전이 업데이트되었음을 확인하는 다음 응답을 받게 됩니다.

```
{
     "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"
}
```

주석 저장소 버전의 세부 정보를 보려면 get-annotation-store-version을 사용합니다.

```
aws omics get-annotation-store-version --name my_annotation_store --version-name
 my_version
```

버전 이름, 상태 및 기타 세부 정보가 포함된 응답을 받게 됩니다.

```
{
     "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
    }
```

다음 예제와 같이 list-annotation-store-versions를 사용하여 주석 저장소의 모든 버전을 볼 수 있습니다.

```
aws omics list-annotation-store-versions --name my_annotation_store
```

다음 정보가 포함된 응답을 받게 됩니다.

```
{
  "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
    }
}
```

주석 저장소 버전이 더 이상 필요하지 않은 경우 다음 예제와 같이 delete-annotation-store-versions를 사용하여 주석 저장소 버전을 삭제할 수 있습니다.

```
aws omics delete-annotation-store-versions --name my_annotation_store --versions
 my_version
```

저장소 버전이 오류 없이 삭제되면 다음과 같은 응답을 받게 됩니다.

```
{
  "errors": []
}
```

오류가 있는 경우 다음과 같이 오류 세부 정보가 포함된 응답을 받게 됩니다.

```
{
  "errors": [
    {
      "versionName": "my_version",
      "message": "Version with versionName: my_version was not found."
    }
  ]
}
```

활성 가져오기 작업이 있는 주석 저장소 버전을 삭제하려고 하면 다음과 같이 오류가 포함된 응답을 받게 됩니다.

```
{
  "errors": [
    {
      "versionName": "my_version",
      "message": "version has an inflight import running"
    }
  ]
}
```

이 경우 다음 예제와 같이 주석 저장소 버전을 강제로 삭제할 수 있습니다.

```
aws omics delete-annotation-store-versions --name my_annotation_store --versions
 my_version --force
```

HealthOmics 분석 저장소 삭제

> **Important**
>
> AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경 단원을 참조하십시오.

변형 또는 주석 저장소를 삭제하면 시스템은 해당 저장소에서 가져온 모든 데이터와 연결된 태그도 삭제합니다.

다음 예제에서는를 사용하여 변형 저장소를 삭제하는 방법을 보여줍니다 AWS CLI. 작업이 성공하면 변형 저장소 상태가 로 전환됩니다DELETING.

```
aws omics delete-variant-store --id <variant-store-id>
```

다음 예제에서는 주석 저장소를 삭제하는 방법을 보여줍니다. 작업이 성공하면 주석 저장소 상태가 로 전환됩니다DELETING. 둘 이상의 버전이 있는 경우 주석 저장소를 삭제할 수 없습니다.

```
aws omics delete-annotation-store --id <annotation-store-id>
```

HealthOmics 분석 데이터 쿼리

> **Important**
>
> AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경 단원을 참조하십시오.

AWS Lake Formation 및 Amazon Athena 또는 Amazon EMR을 사용하여 변형 스토어에 대한 쿼리를 수행할 수 있습니다. 쿼리를 실행하기 전에 Lake Formation 및 Amazon Athena에 대한 설정 절차(다음 섹션에 설명됨)를 완료합니다.

Amazon EMR에 대한 자세한 내용은 자습서: Amazon EMR 시작하기를 참조하세요.

2024년 9월 26일 이후에 생성된 변형 저장소의 경우 HealthOmics는 샘플 ID로 저장소를 분할합니다. 이 파티셔닝은 HealthOmics가 샘플 ID를 사용하여 변형 정보의 저장을 최적화함을 의미합니다. 샘플 정보를 필터로 사용하는 쿼리는 쿼리가 데이터를 더 적게 스캔하므로 결과를 더 빨리 반환합니다.

HealthOmics는 샘플 IDs 파티션 파일 이름으로 사용합니다. 데이터를 수집하기 전에 샘플 ID에 PHI 데이터가 포함되어 있는지 확인합니다. 그럴 경우 데이터를 수집하기 전에 샘플 ID를 변경합니다. 샘플 IDs에 포함하거나 포함하지 않을 콘텐츠에 대한 자세한 내용은 AWS HIPAA 규정 준수 웹 페이지의 지침을 참조하세요.

주제

- HealthOmics를 사용하도록 Lake Formation 구성
- 쿼리에 대한 Athena 구성
- HealthOmics 변형 저장소에서 쿼리 실행

HealthOmics를 사용하도록 Lake Formation 구성

> **Important**
>
> AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경 단원을 참조하십시오.

Lake Formation을 사용하여 HealthOmics 데이터 스토어를 관리하기 전에 다음 Lake Formation 구성 절차를 수행합니다.

주제

- Lake Formation 관리자 생성 또는 확인
- Lake Formation 콘솔을 사용하여 리소스 링크 생성
- AWS RAM 리소스 공유에 대한 권한 구성

Lake Formation 관리자 생성 또는 확인

Lake Formation에서 데이터 레이크를 생성하기 전에 하나 이상의 관리자를 정의합니다.

관리자는 리소스 링크를 생성할 권한이 있는 사용자 및 역할입니다. 리전별로 계정당 데이터 레이크 관리자를 설정합니다.

Lake Formation 콘솔에서 관리자 생성

1. AWS Lake Formation 콘솔 열기: Lake Formation 콘솔
2. 콘솔에 Lake Formation 시작 패널이 표시되면 시작하기를 선택합니다.

   Lake Formation에서 데이터 레이크 관리자 테이블에 사용자를 추가합니다.
3. 그렇지 않으면 왼쪽 메뉴에서 관리 역할 및 작업을 선택합니다.
4. 필요에 따라 관리자를 추가합니다.

Lake Formation 콘솔을 사용하여 리소스 링크 생성

사용자가 쿼리할 수 있는 공유 리소스를 만들려면 기본 액세스 제어를 비활성화해야 합니다. 기본 액세스 제어 비활성화에 대한 자세한 내용은 Lake Formation 설명서의 데이터 레이크의 기본 보안 설정 변

경을 참조하세요. 리소스 링크를 개별적으로 또는 그룹으로 생성하여 Amazon Athena 또는 기타 AWS 서비스(예: Amazon EMR)의 데이터에 액세스할 수 있습니다.

AWS Lake Formation 콘솔에서 리소스 링크 생성 및 HealthOmics Analytics 사용자와 공유

1. AWS Lake Formation 콘솔 열기: Lake Formation 콘솔
2. 기본 탐색 모음에서 데이터베이스를 선택합니다.
3. 데이터베이스 테이블에서 원하는 데이터베이스를 선택합니다.
4. 생성 메뉴에서 리소스 링크를 선택합니다.
5. 리소스 링크 이름을 입력합니다. Athena에서 데이터베이스에 액세스하려면 소문자(최대 256자)만 사용하여 이름을 입력합니다.
6. 생성(Create)을 선택합니다.
7. 이제 새 리소스 링크가 데이터베이스 아래에 나열됩니다.

Lake Formation 콘솔을 사용하여 공유 리소스에 대한 액세스 권한 부여

Lake Formation 데이터베이스 관리자는 다음 절차를 사용하여 공유 리소스에 대한 액세스 권한을 부여할 수 있습니다.

1. AWS Lake Formation 콘솔 열기: https://console.aws.amazon.com/lakeformation/
2. 기본 탐색 모음에서 데이터베이스를 선택합니다.
3. 데이터베이스 페이지에서 이전에 생성한 리소스 링크를 선택합니다.
4. 작업 메뉴에서 대상에 부여를 선택합니다.
5. 보안 주체 아래의 데이터 권한 부여 페이지에서 IAM 사용자 또는 역할을 선택합니다.
6. IAM 사용자 또는 역할 드롭다운 메뉴에서 액세스 권한을 부여할 사용자를 찾습니다.
7. 그런 다음 LF 태그 또는 카탈로그 리소스 카드에서 명명된 데이터 카탈로그 리소스 옵션을 선택합니다.
8. 테이블-선택 사항 드롭다운 메뉴에서 모든 테이블 또는 이전에 생성한 테이블을 선택합니다.
9. 테이블 권한 카드의 테이블 권한에서 설명 및 선택을 선택합니다.
10. 그런 다음 부여를 선택합니다.

Lake Formation 권한을 보려면 기본 탐색 창에서 데이터 레이크 권한을 선택합니다. 이 표에는 사용 가능한 데이터베이스 및 리소스 링크가 나와 있습니다.

AWS RAM 리소스 공유에 대한 권한 구성

AWS Lake Formation 콘솔의 기본 탐색 모음에서 데이터 레이크 권한을 선택하여 권한을 확인합니다. 데이터 권한 페이지에서 리소스 유형, 데이터베이스 및 공유 리소스와 **ARN** 관련된 테이블을 RAM 리소스 공유에서 볼 수 있습니다. AWS Resource Access Manager (AWS RAM) 리소스 공유를 수락해야 하는 경우 콘솔에서에 AWS Lake Formation 알립니다.

HealthOmics는 저장소 생성 중에 AWS RAM 리소스 공유를 암시적으로 수락할 수 있습니다. AWS RAM 리소스 공유를 수락하려면 또는 `CreateAnnotationStore` API 작업을 호출하는 IAM 사용자 `CreateVariantStore` 또는 역할이 다음 작업을 허용해야 합니다.

- `ram:GetResourceShareInvitations` -이 작업을 통해 HealthOmics는 초대를 찾을 수 있습니다.
- `ram:AcceptResourceShareInvitation` -이 작업을 통해 HealthOmics는 FAS 토큰을 사용하여 초대를 수락할 수 있습니다.

이러한 권한이 없으면 스토어 생성 중에 권한 부여 오류가 표시됩니다.

다음은 이러한 작업을 포함하는 샘플 정책입니다. AWS RAM 리소스 공유를 수락하는 IAM 사용자 또는 역할에이 정책을 추가합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "omics:*",
        "ram:AcceptResourceShareInvitation",
        "ram:GetResourceShareInvitations"
      ],
      "Resource": "*"
    }
  ]
}
```

쿼리에 대한 Athena 구성

> **Important**
>
> AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경 단원을 참조하십시오.

Athena를 사용하여 변형 및 주석을 쿼리할 수 있습니다. 쿼리를 실행하기 전에 다음 설정 작업을 수행합니다.

주제

Athena 콘솔을 사용하여 쿼리 결과 위치 구성

쿼리 결과 위치를 구성하려면 다음 단계를 따릅니다.

1. Athena 콘솔 열기: Athena 콘솔
2. 기본 탐색 모음에서 쿼리 편집기를 선택합니다.
3. 쿼리 편집기에서 설정 탭을 선택한 다음 관리를 선택합니다.
4. 위치의 S3 접두사를 입력하여 쿼리 결과를 저장합니다.

Athena 엔진 v3를 사용하여 작업 그룹 구성

작업 그룹을 구성하려면 다음 단계를 따릅니다.

1. Athena 콘솔 열기: Athena 콘솔
2. 기본 탐색 모음에서 작업 그룹을 선택한 다음 작업 그룹 생성을 선택합니다.
3. 작업 그룹의 이름을 입력합니다.
4. 엔진 유형으로 Athena SQL을 선택합니다.
5. 쿼리 엔진 업그레이드에서 수동을 선택합니다.
6. 쿼리 버전 엔진에서 Athena 버전 3을 선택합니다.

7. 작업 그룹 생성을 선택합니다.

HealthOmics 변형 저장소에서 쿼리 실행

> **Important**
>
> AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경 단원을 참조하십시오.

Amazon Athena를 사용하여 변형 저장소에서 쿼리를 수행할 수 있습니다. 변형 및 주석 저장소의 게놈 좌표는 제로 기반 반닫힌 반개방 간격으로 표시됩니다.

Athena 콘솔을 사용하여 간단한 쿼리 실행

다음 예제에서는 단순 쿼리를 실행하는 방법을 보여줍니다.

1. Athena 쿼리 편집기 열기: Athena 쿼리 편집기
2. 작업 그룹에서 설정 중에 생성한 작업 그룹을 선택합니다.
3. 데이터 소스가 AwsDataCatalog인지 확인합니다.
4. 데이터베이스에서 Lake Formation 설정 중에 생성한 데이터베이스 리소스 링크를 선택합니다.
5. 쿼리 1 탭 아래의 쿼리 편집기에 다음 쿼리를 복사합니다.

```
SELECT * from omicsvariants limit 10
```

6. 그런 다음 Run(실행)을 선택하여 쿼리를 실행합니다. 콘솔은 결과 테이블을 omicsvariants 테이블의 처음 10개 행으로 채웁니다.

Athena 콘솔을 사용하여 복잡한 쿼리 실행

다음 예제에서는 복잡한 쿼리를 실행하는 방법을 보여줍니다. 이 쿼리를 실행하려면 주석 저장소`ClinVar`로 가져옵니다.

복잡한 쿼리 실행

1. Athena 쿼리 편집기 열기: Athena 쿼리 편집기

2. 작업 그룹에서 설정 중에 생성한 작업 그룹을 선택합니다.
3. 데이터 소스가 AwsDataCatalog인지 확인합니다.
4. 데이터베이스에서 Lake Formation 설정 중에 생성한 데이터베이스 리소스 링크를 선택합니다.
5. 오른쪽 + 상단의를 선택하여 쿼리 2라는 새 쿼리 탭을 생성합니다.
6. 쿼리 2 탭 아래의 쿼리 편집기에 다음 쿼리를 복사합니다.

```
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. 실행을 선택하여 쿼리 실행을 시작합니다.

HealthOmics 분석 스토어 공유

> **Important**
>
> AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경 단원을 참조하십시오.

변형 저장소 또는 주석 저장소의 소유자는 저장소를 다른 AWS 계정과 공유할 수 있습니다. 소유자는 공유를 삭제하여 공유 리소스에 대한 액세스를 취소할 수 있습니다.

공유 스토어의 구독자는 먼저 공유를 수락합니다. 그런 다음 공유 저장소를 사용하는 워크플로를 정의할 수 있습니다. 데이터는 AWS Glue 및 Lake Formation 모두에서 테이블로 표시됩니다.

스토어에 더 이상 액세스할 필요가 없으면 공유를 삭제합니다.

리소스 공유에 대한 자세한 내용은 섹션을 참조의 교차 계정 리소스 공유 AWS HealthOmics하세요.

스토어 공유 생성

스토어 공유를 생성하려면 공유 생성 API 작업을 사용합니다. 보안 주체 구독자는 공유 AWS 계정 를 구독할 사용자의 입니다. 다음 예제에서는 변형 저장소에 대한 공유를 생성합니다. 한 스토어를 두 개 이상의 계정과 공유하려면 동일한 스토어의 여러 공유를 생성합니다.

```
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"
```

생성에 성공하면 공유 ID 및 상태가 포함된 응답을 받게 됩니다.

```
{
       "shareId": "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a",
       "name": "my_Share-123",
       "status": "PENDING"
  }
```

구독자가 공유 수락 API 작업을 사용하여 수락할 때까지 공유는 보류 상태로 유지됩니다.

의 교차 계정 리소스 공유 AWS HealthOmics

교차 계정 공유를 사용하면 사본을 생성하거나 IAM 리소스 정책을 수정하지 않고도 공동 작업자와 리소스를 공유할 수 있습니다. 다음 리소스는 교차 계정 공유를 지원합니다.

- HealthOmics 변형 저장소
- HealthOmics 주석 저장소
- 프라이빗 워크플로

리소스 공유에는 다음 단계가 포함됩니다.

1. 리소스 소유자는 공유를 생성하고 리소스의 ARN과 의도한 구독 AWS 계정 자의를 지정합니다. 구독자가 공유를 수락할 때까지 리소스 공유는 보류 상태로 유지됩니다.
2. 구독자는 리소스 공유를 수락하여 리소스에 액세스합니다. 리소스 공유가 활성화 상태로 전환됩니다.
3. HealthOmics 서비스는 구독자 계정에 리소스에 대한 액세스 권한을 제공합니다.
4. 리소스 소유자는 공유를 삭제하거나 구독자는 공유에 대한 액세스를 취소할 수 있습니다. 구독자는 공유 또는 연결된 리소스를 삭제할 수 없습니다.

주제

공유 생성

공유 생성 API 작업을 사용하여 공유를 생성할 수 있습니다. 보안 주체 구독자는 공유 리소스를 구독할 AWS 계정 사용자의 입니다. 다음 예제에서는 변형 저장소에 대한 공유를 생성합니다.

```
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"
```

생성에 성공하면 공유 ID 및 상태가 포함된 응답을 받게 됩니다.

```
{
"shareId": "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a",
     "name": "my_Share-123",
     "status": "PENDING"
}
```

구독자가 accept-share API 작업을 사용하여 수락할 때까지 공유는 보류 상태로 유지됩니다.

```
 aws omics accept-share \
    --share-id "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a"
```

구독자가 공유를 수락하면 공유가 활성 상태로 전환됩니다.

```
{
"status": "ACTIVATING"
}
```

공유에 대한 정보 검색

공유 가져오기 API 작업을 사용하여 공유에 대한 정보를 검색합니다.

```
aws omics get-share --share-id "495c21bedc889d07d0ab69d710a6841e-
dd75ab7a1a9c384fa848b5bd8e5a7e0a"
```

API 응답에는 공유에 대한 메타데이터 정보가 포함됩니다.

```
{
  "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"
    }
}
```

소유한 공유 보기

list-shares API를 사용하여 소유한 각 공유에 대한 정보를 검색합니다.

```
aws omics list-shares  --resource-owner SELF
```

API 응답에는 소유한 각 공유의 메타데이터가 포함됩니다.

다른 계정에서 수락된 공유 보기

list-shares API를 사용하여 다른 계정에서 수락한 모든 공유를 볼 수 있습니다.

```
aws omics list-shares  --resource-owner OTHER
```

API 응답에는 수락한 각 공유의 메타데이터가 포함됩니다.

공유 삭제

더 이상 필요하지 않은 공유를 삭제하려면 delete-share API를 사용합니다.

```
aws omics delete-share \
    --share-id "495c21bedc889d07d0ab69d710a6841e-dd75ab7a1a9c384fa848b5bd8e5a7e0a"
```

HealthOmics에서 리소스 태그 지정

주제

중요 공지 사항

HealthOmics는 AWS 공동 책임 모델 정책에 따라 고객 데이터를 보호합니다. 즉, 모든 고객 데이터는 전환 시와 유휴 시 모두 암호화됩니다. 그러나 데이터 스토어 또는 작업 기반 작업과 같은 리소스의 모든 고객 입력 이름이 암호화되는 것은 아닙니다. 개인 식별 정보 또는 보호 대상 건강 정보를 포함해서는 안 됩니다. 자세한 내용은 AWS HealthOmics의 보안 단원을 참조하십시오.

HealthOmics 리소스에 태그 지정

태그를 사용하여 AWS 리소스에 메타데이터를 할당할 수 있습니다. 각 태그는 사용자 정의 키와 값으로 구성된 레이블입니다. 태그를 사용하면 리소스를 손쉽게 관리, 식별, 정리, 검색, 필터링할 수 있습니다.

이 주제에서는 일관되고 효과적인 태깅 전략을 구현하는 데 도움이 되는 일반적인 태깅 범주 및 전략에 대해 설명합니다. 다음 섹션에서는 AWS 리소스, 태그 지정, 세부 결제 및에 대한 기본 지식을 가정합니다 AWS Identity and Access Management.

각 태그는 두 부분으로 구성됩니다.

- 태그 키(예: CostCenter, Environment 또는 Project) 태그 키는 대소문자를 구별합니다.
- 태그 값(예: 111122223333 또는 프로덕션). 태그 키처럼 태그 값은 대/소문자를 구별합니다.

태그를 사용하여 용도, 소유자, 환경 또는 기타 기준으로 리소스를 분류할 수 있습니다. 자세한 내용은 AWS 태그 지정 전략을 참조하십시오.

리소스의 서비스 콘솔, 서비스 API 또는에서 리소스에 대한 태그를 추가, 변경 또는 제거할 수 있습니다 AWS CLI.

태그 지정을 활성화하려면 TagResources에 권한이 있는지 확인합니다. 다음 예제와 같이 IAM 정책을 연결하여 TagResources에 권한을 부여할 수 있습니다.

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": "*"
        }
    ]
}
```

모범 사례

AWS 리소스에 대한 태그 지정 전략을 생성할 때 모범 사례를 따르세요.

- 개인 식별 정보(PII), 보호 대상 건강 정보(PHI) 또는 기타 민감한 정보를 태그에 저장하지 마십시오.
- 대/소문자를 구분하는 표준화된 태그 형식을 사용하고 모든 리소스 유형에 일관되게 적용합니다.
- 리소스 액세스 제어, 비용 추적, 자동화 및 조직 관리와 같은 다양한 용도를 지원하는 태그 지침을 고려합니다.
- 리소스 태그를 관리하는 데 도움이 되는 자동화 도구를 사용합니다. AWS Resource Groups 및 Resource Groups Tagging API를 사용하면 태그를 프로그래밍 방식으로 제어할 수 있으므로 태그와 리소스를 자동으로 관리, 검색 및 필터링할 수 있습니다.
- 태그를 더 많이 사용하면 태그 지정이 더 효과적입니다.
- 사용자의 필요에 따라 태그를 편집하거나 수정할 수 있습니다. 그러나 액세스 제어 태그를 업데이트하려면 리소스에 대한 액세스를 제어하기 위해 해당 태그를 참조하는 정책도 업데이트해야 합니다.

태그 지정 요구 사항

태그를 지정할 때 요건은 다음과 같습니다.

- 키에는 aws: 접두사를 붙일 수 없습니다.
- 키는 태그 집합에 대해 고유해야 합니다.
- 키는 1~128자 사이의 허용된 문자이어야 합니다.
- 값은 0~256자 사이의 허용된 문자이어야 합니다.
- 값은 태그 세트별로 고유할 필요는 없습니다.
- 키와 값의 문자로는 Unicode 문자, 숫자, 공백 그리고 다음 기호가 허용됩니다. _ . : / = + - @.
- 키와 값은 대/소문자를 구분합니다.

시퀀스 스토어 읽기 세트 태그

시퀀스 스토어의 경우 읽기 세트에 생성된 태그는 읽기 세트 리소스 수준에 있습니다. 읽기 세트에는 S3 APIs를 사용하여 액세스, 검색 및 제한할 수 있는 객체도 포함됩니다. 기본적으로 샘플 ID(omics:sampleId) 및 주제 ID(omics:subjectId)가 객체에 추가됩니다.

또한 읽기 세트와 그 아래의 객체 간에 최대 5개의 태그를 동기화할 수 있습니다. 태그를 동기화할 구성은 propogatedSetLevelTags 파라미터를 사용하여 저장소를 생성하거나 업데이트하는 동안 설정된 저장소 수준 구성입니다.

스토어에 이미 데이터가 있는 경우 키를 업데이트하는 데 시간이 걸릴 수 있습니다. 이 업데이트 중에 HealthOmics는 스토어 상태를 로 변경합니다Updating. 완료되면 HealthOmics는 스토어 상태를 로 설정합니다Active. 태그가 전파되는 동안 태그에 의존하는 권한이 적용되지 않을 수 있습니다. 태그 전파가 완료된 후 권한이 적용됩니다.

읽기 세트에 태그가 설정되거나 업데이트되면 시스템은 스토어 구성에 따라 해당 읽기 세트의 객체를 업데이트할지 여부를 결정합니다.

HealthOmics 리소스에 태그 추가

리소스에 태그를 추가하면 AWS 리소스를 식별 및 구성하고 리소스에 대한 액세스를 관리하는 데 도움이 될 수 있습니다. 먼저 리소스에 하나 이상의 태그(키-값 페어)를 추가합니다. 리소스당 최대 50개의 태그를 사용할 수 있습니다. 키 및 값 필드에 사용할 수 있는 문자에도 제한이 있습니다.

태그를 추가한 후 이러한 태그를 기반으로 리소스에 AWS 대한 액세스를 관리하는 IAM 정책을 생성할 수 있습니다. HealthOmics 콘솔 또는를 사용하여 리소스 AWS CLI 에 태그를 추가할 수 있습니다. 리포지토리에 태그를 추가하면 해당 리포지토리에 대한 액세스에 영향을 줄 수 있습니다. 데이터 스토어에 태그를 추가하기 전에 태그를 사용하여 데이터 스토어와 같은 리소스에 대한 액세스를 제어할 수 있는 IAM 정책을 검토합니다.

서비스 태그는 제목과 시퀀스 저장소의 샘플 ID 모두에 대해 자동으로 생성됩니다.

다음 단계에 따라 AWS CLI 를 사용하여 HealthOmics 리소스에 태그를 추가합니다. 예를 들어 시퀀스 스토어가 생성되는 동안 시퀀스 스토어에 태그를 추가하려면에서 다음 명령을 사용합니다 AWS CLI. 시퀀스 스토어의 이름은 MySequenceStore이고 키가 있는 추가된 두 태그는 각각 값이 value1 및 value2인 key1 및 key2입니다.
:

```
aws omics create-sequence-store --name "MySequenceStore" --tags key1=value1,key2=value2
```

출력에는 태그가 나열되지 않습니다. 다음 응답을 반환합니다.

```
{
    "id": "6860403586",
```

```
    "referenceStoreId": "4889894479",
    "roleArn": "arn:aws:iam::555555555555:role/ImportTest",
    "status": "CREATED",
    "creationTime": "2022-07-21T01:19:07.194Z"
}
```

기존 리소스에 태그를 추가하려면 다음 예제 명령을 실행합니다.

```
aws omics tag-resource --resource-arn arn:aws:omics:us-
west-2:555555555555:sequenceStore/2275234794 --tags key1=value1,key2=value2
```

성공하면 이 명령에서 빈 응답을 반환합니다.

리소스에 대한 태그 나열

다음 단계에 따라 AWS CLI 를 사용하여 HealthOmics 리소스의 AWS 태그 목록을 봅니다. 태그가 추가되지 않은 경우 반환되는 목록은 비어 있습니다.

터미널 또는 명령줄에서 다음 예제와 같이 list-tags-for-resource 명령을 실행합니다.

```
aws omics list-tags-for-resource --resource-arn arn:aws:omics:us-
west-2:555555555555:sequenceStore/2275234794
```

응답으로 JSON 형식의 태그 목록을 받게 됩니다.

```
 {
    "tags": {
        "key1": "value1",
        "key2": "value2"
    }
}
```

데이터 스토어에서 태그 제거

리소스와 연결된 하나 이상의 태그를 제거할 수 있습니다. 태그를 제거해도 해당 태그와 연결된 다른 AWS 리소스에서 태그는 삭제되지 않습니다.

터미널 또는 명령줄에서 untag-resource 명령을 실행하여 태그를 제거할 리소스의 Amazon 리소스 이름(ARN)과 제거할 태그의 태그 키를 지정합니다.

```
aws omics untag-resource --resource-arn arn:aws:omics:us-
west-2:555555555555:sequenceStore/2275234794 --tag-keys key1,key2
```

성공하면이 명령은 응답을 반환하지 않습니다. 리소스와 연결된 태그를 확인하려면 list-tags-for-resource 명령을 실행합니다.

HealthOmics에 대한 IAM 권한

AWS Identity and Access Management (IAM)를 사용하여 HealthOmics API 및 스토어 및 워크플로와 같은 리소스에 대한 액세스를 관리할 수 있습니다. HealthOmics를 사용하는 계정의 사용자 및 애플리케이션의 경우 IAM 사용자, 그룹 또는 역할에 적용할 수 있는 권한 정책에서 권한을 관리합니다.

계정의 사용자 및 애플리케이션에 대한 권한을 관리하려면 HealthOmics에서 제공하는 정책을 사용하거나 직접 작성합니다. HealthOmics 콘솔은 여러 서비스를 사용하여 함수의 구성 및 트리거에 대한 정보를 가져옵니다. 제공된 정책을 있는 그대로 사용하거나 보다 제한적인 정책의 시작점으로 사용할 수 있습니다.

HealthOmics는 IAM 서비스 역할을 사용하여 사용자를 대신하여 다른 서비스에 액세스합니다. 예를 들어 Amazon S3에서 데이터를 읽는 워크플로를 실행할 때 서비스 역할을 생성하거나 선택할 수 있습니다. 일부 기능의 경우 다른 서비스의 리소스에 대한 권한도 구성해야 합니다. HealthOmics 작업을 시작하기 전에 이러한 요구 사항 검토

IAM에 대한 자세한 내용은 IAM 사용 설명서의 IAM이란 무엇입니까? 단원을 참조하세요.

주제

- HealthOmics에 대한 자격 증명 기반 IAM 정책
- 에 대한 서비스 역할 AWS HealthOmics
- Amazon ECR 권한
- HealthOmics 리소스 권한
- Amazon S3 URIs를 사용한 데이터 액세스 권한

HealthOmics에 대한 자격 증명 기반 IAM 정책

계정의 사용자에게 HealthOmics에 대한 액세스 권한을 부여하려면 AWS Identity and Access Management (IAM)의 자격 증명 기반 정책을 사용합니다. 자격 증명 기반 정책은 IAM 사용자 또는 사용자와 연결된 IAM 그룹 및 역할에 직접 적용할 수 있습니다. 다른 계정의 사용자에게 계정의 역할을 수임하고 HealthOmics 리소스에 액세스할 수 있는 권한을 부여할 수도 있습니다.

사용자가 워크플로 버전에 대한 작업을 수행할 수 있는 권한을 부여하려면 리소스 목록에 워크플로와 특정 워크플로 버전을 추가해야 합니다.

다음 IAM 정책은 사용자가 모든 HealthOmics API 작업에 액세스하고 서비스 역할을 HealthOmics에 전달할 수 있도록 허용합니다.

Example사용자 정책

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "omics:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": "omics.amazonaws.com"
        }
      }
    }
  ]
}
```

HealthOmics를 사용하면 다른 AWS 서비스와도 상호 작용합니다. 이러한 서비스에 액세스하려면 각 서비스에서 제공하는 관리형 정책을 사용합니다. 리소스 하위 집합에 대한 액세스를 제한하려면 관리형 정책을 시작점으로 사용하여 보다 제한적인 자체 정책을 생성할 수 있습니다.

- AmazonS3FullAccess - 작업에서 사용하는 Amazon S3 버킷 및 객체에 대한 액세스입니다.
- AmazonEC2ContainerRegistryFullAccess - 워크플로 컨테이너 이미지의 Amazon ECR 레지스트리 및 리포지토리에 액세스할 수 있습니다.
- AWSLakeFormationDataAdmin - 분석 스토어에서 생성한 Lake Formation 데이터베이스 및 테이블에 대한 액세스입니다.

- ResourceGroupsandTagEditorFullAccess – HealthOmics 태깅 API 작업으로 HealthOmics 리소스에 태깅합니다.

위의 정책은 사용자가 IAM 역할을 생성하도록 허용하지 않습니다. 이러한 권한이 있는 사용자가 작업을 실행하려면 관리자가 HealthOmics에 데이터 소스에 액세스할 수 있는 권한을 부여하는 서비스 역할을 생성해야 합니다. 자세한 내용은 에 대한 서비스 역할 AWS HealthOmics 단원을 참조하십시오.

실행에 대한 사용자 지정 IAM 권한 정의

권한 부여 요청에 `StartRun` 요청에서 참조하는 워크플로, 실행 또는 실행 그룹을 포함할 수 있습니다. 이렇게 하려면 IAM 정책에서 원하는 워크플로, 실행 또는 실행 그룹 조합을 나열합니다. 예를 들어 워크플로 사용을 특정 실행 또는 실행 그룹으로 제한할 수 있습니다. 워크플로를 실행 그룹에만 사용하도록 지정할 수도 있습니다.

다음은 단일 실행 그룹으로 단일 워크플로를 허용하는 IAM 정책의 예입니다.

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/*"
            ]
        }
    ]
}
```

에 대한 서비스 역할 AWS HealthOmics

서비스 역할은 AWS 서비스가 계정의 리소스에 액세스할 수 있는 권한을 부여하는 AWS Identity and Access Management (IAM) 역할입니다. 가져오기 작업을 시작하거나 실행을 시작할 AWS HealthOmics 때에 서비스 역할을 제공합니다.

HealthOmics 콘솔에서 필요한 역할을 생성할 수 있습니다. HealthOmics API를 사용하여 리소스를 관리하는 경우 IAM 콘솔을 사용하여 서비스 역할을 생성합니다. 자세한 내용은 에 권한을 위임할 역할 생성을 참조하세요 AWS 서비스.

서비스 역할에는 다음과 같은 신뢰 정책이 있어야 합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "omics.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole"
    }
  ]
}
```

신뢰 정책은 HealthOmics 서비스가 역할을 수임하도록 허용합니다.

주제

- IAM 서비스 정책 예
- 예제 CloudFormation 템플릿

IAM 서비스 정책 예

이 예제에서 리소스 이름과 계정 IDs는 실제 값으로 바꿀 수 있는 자리 표시자입니다.

다음 예제에서는 실행을 시작하는 데 사용할 수 있는 서비스 역할에 대한 정책을 보여줍니다. 이 정책은 실행을 위해 Amazon S3 출력 위치, 워크플로 로그 그룹 및 Amazon ECR 컨테이너에 액세스할 수 있는 권한을 부여합니다.

> **Note**
>
> 실행에 통화 캐싱을 사용하는 경우 실행 캐시 Amazon S3 위치를 s3 권한의 리소스로 추가합니다.

Example실행을 시작하기 위한 서비스 역할 정책

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/*"
            ]
```

```
      }
    ]
}
```

다음 예제에서는 스토어 가져오기 작업에 사용할 수 있는 서비스 역할에 대한 정책을 보여줍니다. 이 정책은 Amazon S3 입력 위치에 액세스할 수 있는 권한을 부여합니다.

Example참조 스토어 작업에 대한 서비스 역할

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"
            ]
        }

    ]
}
```

예제 CloudFormation 템플릿

다음 샘플 CloudFormation 템플릿은 이름이 접두사로 붙은 Amazon S3 버킷에 액세스omics-하고 워크플로 로그를 업로드할 수 있는 권한을 HealthOmics에 부여하는 서비스 역할을 생성합니다.

Example참조 스토어, Amazon S3 및 CloudWatch Logs 권한

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

Amazon ECR 권한

HealthOmics 서비스가 프라이빗 Amazon ECR 리포지토리의 컨테이너에서 워크플로를 실행하기 전에 리포지토리에 대한 리소스 정책을 생성합니다. 이 정책은 HealthOmics 서비스가 컨테이너를 사용할 수 있는 권한을 부여합니다. 워크플로에서 참조하는 각 프라이빗 리포지토리에이 리소스 정책을 추가합니다.

> **Note**
>
> 프라이빗 리포지토리와 워크플로는 동일한 리전에 있어야 합니다.

다른 AWS 계정이 워크플로와 리포지토리를 소유한 경우 교차 계정 권한을 구성해야 합니다.

공유 워크플로에 대한 추가 리포지토리 액세스 권한을 부여할 필요가 없습니다. 그러나 컨테이너 이미지에 대한 특정 워크플로 액세스를 허용하거나 거부하는 정책을 생성할 수 있습니다.

Amazon ECR 풀스루 캐시 기능을 사용하려면 레지스트리 권한 정책을 생성해야 합니다.

다음 섹션에서는 이러한 시나리오에 대해 Amazon ECR 리소스 권한을 구성하는 방법을 설명합니다. Amazon ECR의 권한에 대한 자세한 내용은 Amazon ECR의 프라이빗 레지스트리 권한을 참조하세요.

주제

- Amazon ECR 리포지토리에 대한 리소스 정책 생성
- 교차 계정 컨테이너를 사용하여 워크플로 실행
- 공유 워크플로에 대한 Amazon ECR 정책
- Amazon ECR 풀스루 캐시에 대한 정책

Amazon ECR 리포지토리에 대한 리소스 정책 생성

HealthOmics 서비스가 리포지토리의 컨테이너를 사용하여 워크플로를 실행하도록 허용하는 리소스 정책을 생성합니다. 이 정책은 HealthOmics 서비스 보안 주체가 필요한 Amazon ECR 작업에 액세스할 수 있는 권한을 부여합니다.

다음 단계에 따라 정책을 생성합니다.

1. Amazon ECR 콘솔에서 프라이빗 리포지토리 페이지를 열고 액세스 권한을 부여할 리포지토리를 선택합니다.
2. 사이드바 탐색에서 권한을 선택합니다.
3. 편집을 선택합니다.
4. 정책 JSON 편집을 선택합니다.
5. 다음 정책 설명을 추가한 다음 저장을 선택합니다.

   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": "*"
        }
    ]
}
```

교차 계정 컨테이너를 사용하여 워크플로 실행

다른 AWS 계정이 워크플로와 컨테이너를 소유한 경우 다음과 같은 교차 계정 권한을 구성해야 합니다.

1. 워크플로를 소유한 계정에 명시적으로 권한을 부여하도록 리포지토리에 대한 Amazon ECR 정책을 업데이트합니다.
2. 워크플로를 소유한 계정의 서비스 역할을 업데이트하여 컨테이너 이미지에 대한 액세스 권한을 부여합니다.

다음 예제에서는 워크플로를 소유한 계정에 액세스 권한을 부여하는 Amazon ECR 리소스 정책을 보여줍니다.

이 예시는 다음과 같이 설정되어 있습니다.

- 워크플로 계정 ID: 111122223333
- 컨테이너 리포지토리 계정 ID: 444455556666
- 컨테이너 이름: samtools

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "omics.amazonaws.com"
            },
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowAccessToTheServiceRoleOfTheAccountThatOwnsTheWorkflow",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/DemoCustomer"
            },
            "Action": [
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        }
    ]
}
```

설정을 완료하려면 워크플로를 소유한 계정의 서비스 역할에 다음 정책 설명을 추가합니다. 이 정책은 서비스 역할이 “samtools” 컨테이너 이미지에 액세스할 수 있는 권한을 부여합니다. 계정 번호, 컨테이너 이름 및 리전을 고유한 값으로 바꿔야 합니다.

```
{
    "Sid": "CrossAccountEcrRepoPolicy",
    "Effect": "Allow",
    "Action": ["ecr:BatchCheckLayerAvailability", "ecr:BatchGetImage",
 "ecr:GetDownloadUrlForLayer"],
    "Resource": "arn:aws:ecr:us-west-2:444455556666:repository/samtools"
}
```

공유 워크플로에 대한 Amazon ECR 정책

> **Note**
>
> HealthOmics는 워크플로가 구독자 계정에서 실행되는 동안 공유 워크플로가 워크플로 소유자 계정의 Amazon ECR 리포지토리에 자동으로 액세스할 수 있도록 허용합니다. 공유 워크플로에 대한 추가 리포지토리 액세스 권한을 부여할 필요가 없습니다. 자세한 내용은 HealthOmics 워크플로 공유를 참조하세요.

기본적으로 구독자는 기본 컨테이너를 사용하기 위해 Amazon ECR 리포지토리에 액세스할 수 없습니다. 선택적으로 리포지토리의 리소스 정책에 조건 키를 추가하여 Amazon ECR 리포지토리에 대한 액세스를 사용자 지정할 수 있습니다. 다음 섹션에서는 예제 정책을 제공합니다.

특정 워크플로에 대한 액세스 제한

조건문에 개별 워크플로를 나열할 수 있으므로 이러한 워크플로만 리포지토리의 컨테이너를 사용할 수 있습니다. SourceArn 조건 키는 공유 워크플로의 ARN을 지정합니다. 다음 예제에서는 지정된 워크플로가이 리포지토리를 사용할 수 있는 권한을 부여합니다.

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"
          }
        }
      }
  ]
}
```

특정 계정에 대한 액세스 제한

조건문에 구독자 계정을 나열할 수 있으므로 이러한 계정만 리포지토리의 컨테이너를 사용할 권한이 있습니다. SourceAccount 조건 키는 구독 AWS 계정 자의를 지정합니다. 다음 예제에서는 지정된 계정이이 리포지토리를 사용할 수 있는 권한을 부여합니다.

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"
        }
      }
    }
  ]
}
```

다음 예제 정책과 같이 특정 구독자에 대한 Amazon ECR 권한을 거부할 수도 있습니다.

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"
                }
              }
          }
      ]
  }
```

Amazon ECR 풀스루 캐시에 대한 정책

Amazon ECR 풀스루 캐시를 사용하려면 레지스트리 권한 정책을 생성합니다. 또한 Amazon ECR 풀스루 캐시에서 생성된 리포지토리에 대한 권한을 정의하는 리포지토리 생성 템플릿을 생성합니다.

다음 섹션에는 이러한 정책의 예가 포함되어 있습니다. 풀스루 캐시에 대한 자세한 내용은 Amazon Elastic Container Registry 사용 설명서의 Amazon ECR 프라이빗 레지스트리와 업스트림 레지스트리 동기화를 참조하세요.

레지스트리 권한 정책

Amazon ECR 풀스루 캐시를 사용하려면 레지스트리 권한 정책을 생성합니다. 레지스트리 권한 정책은 복제 및 풀스루 캐시 권한을 제어합니다.

교차 계정 복제의 경우 리포지토리를 레지스트리에 복제할 수 AWS 계정 있는 각를 명시적으로 허용해야 합니다.

기본적으로 풀스루 캐시 규칙을 생성할 때 프라이빗 레지스트리에서 이미지를 가져올 권한이 있는 모든 IAM 보안 주체도 풀스루 캐시 규칙을 사용할 수 있습니다. 레지스트리 권한을 사용하여 이러한 권한을 특정 리포지토리로 더 좁힐 수 있습니다.

컨테이너 이미지를 소유한 계정에 레지스트리 권한 정책을 추가합니다.

다음 예제에서 정책은 HealthOmics 서비스가 각 업스트림 레지스트리에 대한 리포지토리를 생성하고 생성된 리포지토리에서 업스트림 풀 요청을 시작하도록 허용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPTCinRegPermissions",
            "Effect": "Allow",
            "Principal": {
                "Service": "omics.amazonaws.com"
            },
            "Action": [
                "ecr:CreateRepository",
                "ecr:BatchImportUpstreamImage"
            ],
            "Resource": [
                "arn:aws:ecr:us-east-1:123456789012:repository/ecr-public/*",
                "arn:aws:ecr:us-east-1:123456789012:repository/docker-hub/*"
            ]
        }
    ]
}
```

리포지토리 생성 템플릿

HealthOmics에서 풀스루 캐시를 사용하려면 Amazon ECR 리포지토리에 리포지토리 생성 템플릿이 있어야 합니다. 템플릿은 업스트림 레지스트리에 대해 생성된 프라이빗 리포지토리의 구성 설정을 정의합니다.

각 템플릿에는 Amazon ECR이 새 리포지토리를 특정 템플릿과 일치시키는 데 사용하는 리포지토리 네임스페이스 접두사가 포함되어 있습니다. 템플릿은 리소스 기반 액세스 정책, 태그 불변성, 암호화, 수명 주기 정책을 비롯한 모든 리포지토리 설정의 구성을 지정할 수 있습니다. 자세한 내용은 Amazon Elastic Container Registry 사용 설명서의 리포지토리 생성 템플릿을 참조하세요.

다음 예제에서 정책은 HealthOmics 서비스가 업스트림 리포지토리에서 업스트림 풀 요청을 시작하도록 허용합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PTCRepoCreationTemplate",
            "Effect": "Allow",
            "Principal": {
                "Service": "omics.amazonaws.com"
            },
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*"
        }
    ]
}
```

교차 계정 Amazon ECR 액세스에 대한 정책

교차 계정 액세스의 경우 프라이빗 리포지토리의 소유자는 레지스트리 권한 정책과 리포지토리 생성 템플릿을 업데이트하여 다른 계정과 해당 계정의 실행 역할에 대한 액세스를 허용합니다.

레지스트리 권한 정책에서 다른 계정의 실행 역할이 Amazon ECR 작업에 액세스할 수 있도록 허용하는 정책 설명을 추가합니다.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCrossAccountPTCinRegPermissions",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/RUN_ROLE"},
            "Action": [
                "ecr:CreateRepository",
                "ecr:BatchGetImage",
                "ecr:BatchImportUpstreamImage"
            ],
            "Resource": "arn:aws:ecr:us-east-1:123456789012:repository/path/*"
        }
    ]
}
```

리포지토리 생성 템플릿에서 다른 계정의 실행 역할이 새 컨테이너 이미지에 액세스할 수 있도록 허용하는 정책 설명을 추가합니다. 선택적으로 조건문을 추가하여 특정 워크플로에 대한 액세스를 제한할 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCrossAccountPTCinRepoCreationTemplate",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/RUN_ROLE"},
            "Action": [
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceArn": "arn:aws:omics:us-
east-1:444455556666:workflow/WORKFLOW_ID",
                    "aws:SourceAccount": "111122223333"
```

```
                }
            }
        }
    ]
}
```

실행 역할에 두 개의 추가 작업(CreateRepository 및 BatchImportUpstreamImage)에 대한 권한을 추가하고 실행 역할이 액세스할 수 있는 리소스를 지정합니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccountPTCRunRolePolicy",
            "Effect": "Allow",
            "Action": [
                "ecr:CreateRepository",
                "ecr:BatchImportUpstreamImage",
                "ecr:BatchCheckLayerAvailability",
                "ecr:BatchGetImage",
                "ecr:GetDownloadUrlForLayer",
                "ecr:BatchGetImage"
            ],
            "Resource": "arn:aws:ecr:us-east-1:123456789012::repository/{path}/*"
        }
    ]
}
```

HealthOmics 리소스 권한

AWS HealthOmics 는 작업을 실행하거나 스토어를 생성할 때 사용자를 대신하여 다른 서비스의 리소스를 생성하고 액세스합니다. 경우에 따라 리소스에 액세스하거나 HealthOmics가 리소스에 액세스할 수 있도록 다른 서비스의 권한을 구성해야 합니다.

Amazon ECR과 관련된 리소스 권한은 섹션을 참조하세요Amazon ECR 권한.

Lake Formation 권한

HealthOmics에서 분석 기능을 사용하기 전에 Lake Formation에서 기본 데이터베이스 설정을 구성합니다.

Lake Formation에서 리소스 권한을 구성하려면

1. Lake Formation 콘솔에서 데이터 카탈로그 설정 페이지를 엽니다.
2. 새로 생성된 데이터베이스 및 테이블에 대한 기본 권한에서 데이터베이스 및 테이블에 대한 IAM 액세스 제어 요구 사항을 선택 취소합니다.
3. 저장을 선택합니다.

HealthOmics Analytics는 서비스 정책에 다음 예제와 같이 올바른 RAM 권한이 있는 경우 데이터를 자동으로 수락합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "omics:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ram:AcceptResourceShareInvitation",
        "ram:GetResourceShareInvitations"
      ],
      "Resource": "*"
    }
  ]
}
```

Amazon S3 URIs를 사용한 데이터 액세스 권한

HealthOmics API 작업 또는 Amazon S3 API 작업을 사용하여 시퀀스 스토어 데이터에 액세스할 수 있습니다.

HealthOmics API 액세스의 경우 HealthOmics 권한은 IAM 정책을 통해 관리됩니다. 그러나 S3 액세스에는 스토어의 S3 액세스 정책 및 IAM 정책의 명시적 허용이라는 두 가지 수준의 구성이 필요합니다. HealthOmics에서 IAM 정책을 사용하는 방법에 대한 자세한 내용은 HealthOmics.

Amazon S3 APIs.

1. 정책 기반 공유 -이 공유를 사용하려면 S3 액세스 정책에서 IAM 보안 주체를 활성화하고 IAM 정책을 작성하여 IAM 보안 주체에 연결해야 합니다. 자세한 내용은 다음 주제를 참조하세요.
2. 미리 서명된 URLs- 시퀀스 스토어의 파일에 대해 공유 가능한 미리 서명된 URL을 생성할 수도 있습니다. Amazon S3를 사용하여 미리 서명된 URLs 생성하는 방법에 대한 자세한 내용은 Amazon S3 설명서의 미리 서명된 URLs 사용을 참조하세요. 시퀀스 스토어 S3 액세스 정책은 미리 서명된 URL 기능을 제한하기 위한 문을 지원합니다.
3. 수임된 역할 - 데이터 소유자의 계정 내에 사용자가 해당 역할을 수임할 수 있도록 허용하는 액세스 정책이 있는 역할을 생성합니다.

주제

정책 기반 공유

직접 S3 URI를 사용하여 시퀀스 스토어 데이터에 액세스하는 경우 HealthOmics는 연결된 S3 버킷 액세스 정책에 대한 향상된 보안 조치를 제공합니다.

새 S3 액세스 정책에는 다음 규칙이 적용됩니다. 기존 정책의 경우 다음에 정책을 업데이트할 때 규칙이 적용됩니다.

- S3 액세스 정책은 다음 정책 요소를 지원합니다.
  - 버전, Id, 문, Sid, 효과, 보안 주체, 작업, 리소스, 조건
- S3 액세스 정책은 다음 조건 키를 지원합니다.
  - s3:ExistingObjectTag/<key>, s3:prefix, s3:signatureversion, s3:TlsVersion

- 또한 정책은 ArnEquals 및 ArnLike 조건 연산자를 사용하여 aws:PrincipalArn을 지원합니다.

지원되지 않는 요소 또는 조건을 포함하도록 정책을 추가하거나 업데이트하려고 하면 시스템에서 요청을 거부합니다.

주제

- 기본 S3 액세스 정책
- 액세스 정책 사용자 지정
- IAM 정책
- 태그 기반 액세스 제어

기본 S3 액세스 정책

시퀀스 저장소를 생성할 때 HealthOmics는 데이터 저장소 소유자의 루트 계정에 시퀀스 저장소의 액세스 가능한 모든 객체에 대해 SS3:GetObject, S3GetObjectTagging 및 S3:ListBucket 권한을 부여하는 기본 S3 액세스 정책을 생성합니다. 기본 생성 정책은 다음과 같습니다.

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/*"
        }
    ]
}
```

액세스 정책 사용자 지정

S3 액세스 정책이 비어 있으면 S3 액세스가 허용되지 않습니다. 기존 정책이 있고 s3 액세스를 제거해야 하는 경우 `deleteS3AccessPolicy`를 사용하여 모든 액세스를 제거합니다.

공유에 대한 제한을 추가하거나 다른 계정에 액세스 권한을 부여하려면 `PutS3AccessPolicy` API를 사용하여 정책을 업데이트할 수 있습니다. 정책 업데이트는 시퀀스 스토어의 접두사 또는 지정된 작업을 초과할 수 없습니다.

IAM 정책

사용자 또는 IAM 보안 주체가 Amazon S3 APIs를 사용하여 액세스하도록 허용하려면 S3 액세스 정책의 권한 외에도 IAM 정책을 생성하여 보안 주체에 연결하여 액세스 권한을 부여해야 합니다. Amazon S3 API 액세스를 허용하는 정책은 시퀀스 저장소 수준 또는 읽기 세트 수준에서 적용할 수 있습니다. 읽기 세트 수준에서는 접두사를 통해 또는 샘플 또는 주제 ID 패턴에 대한 리소스 태그 필터를 사용하여 권한을 제한할 수 있습니다.

시퀀스 스토어가 고객 관리형 키(CMK)를 사용하는 경우 보안 주체는 복호화에 KMS 키를 사용할 권한도 있어야 합니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 교차 계정 KMS 액세스를 참조하세요.

다음 예제에서는 사용자에게 시퀀스 저장소에 대한 액세스 권한을 부여합니다. 추가 조건 또는 리소스 기반 필터를 사용하여 액세스를 미세 조정할 수 있습니다.

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/*"
        }
      }
    }
  ]
}
```

태그 기반 액세스 제어

태그 기반 액세스 제어를 사용하려면 먼저 시퀀스 스토어를 업데이트하여 사용할 태그 키를 전파해야 합니다. 이 구성은 시퀀스 스토어 생성 또는 업데이트 중에 설정됩니다. 태그가 전파되면 태그 조건을

사용하여 제한을 더 추가할 수 있습니다. 제한은 S3 액세스 정책 또는 IAM 정책에 배치할 수 있습니다. 다음은 설정할 탭 기반 S3 액세스 정책의 예입니다.

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

제한 예제

시나리오: 데이터 소유자가 사용자의 '철회됨' 데이터 다운로드 기능을 제한할 수 있는 공유 생성.

이 시나리오에서는 데이터 소유자(계정 #111111111111)가 데이터 스토어를 관리했습니다. 이 데이터 소유자는 연구원(계정 #999999999999)을 포함하여 광범위한 타사 사용자와 데이터를 공유합니다. 데이터 관리의 일환으로 데이터 소유자는 주기적으로 참가자 데이터 철회 요청을 받습니다. 이 취소를 관리하기 위해 데이터 소유자는 먼저 요청을 수신할 때 직접 다운로드 액세스를 제한하고 결국 요구 사항에 따라 데이터를 삭제합니다.

이러한 요구 사항을 충족하기 위해 데이터 소유자는 시퀀스 저장소를 설정하고 각 읽기 세트는 취소 요청이 전달될 경우 “withdrawn”으로 설정되는 “status”에 대한 태그를 수신합니다. 태그가이 값으로 설정된 데이터의 경우이 파일에서 'getObject'를 실행할 수 있는 사용자가 없는지 확인합니다. 이 설정을 수행하려면 데이터 소유자가 두 단계를 수행해야 합니다.

1단계. 시퀀스 스토어의 경우 상태 태그가 전파되도록 업데이트되었는지 확인합니다. 이는 createSequenceStore 또는를 호출할 propogatedSetLevelTags 때에 “상태” 키를 추가하여 수행됩니다. updateSequenceStore.

2단계. 스토어의 s3 액세스 정책을 업데이트하여 상태 태그가 취소됨으로 설정된 객체에 대한 getObject를 제한합니다. 이는 PutS3AccesPolicy API를 사용하여 스토어 액세스 정책을 업데이트하여 수행됩니다. 다음 정책을 통해 고객은 객체를 나열할 때 철회된 파일을 계속 볼 수 있지만 액세스하지 못하도록 할 수 있습니다.

- 문 1(restrictedGetWithdrawal): 계정 999999999999은 철회된 객체를 검색할 수 없습니다.
- 문 2(ownerGetAll): 데이터 소유자인 계정 111111111111은 철회된 객체를 포함하여 모든 객체를 검색할 수 있습니다.
- 문 3(everyoneListAll): 모든 공유 계정 111111111111 및 999999999999은 전체 접두사에서 ListBucket 작업을 실행할 수 있습니다.

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 HealthOmics의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 는에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다 AWS 클라우드. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 AWS 규정 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. AWS HealthOmics에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스규정 준수 프로그램 .
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 AWS HealthOmics를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표에 맞게 AWS HealthOmics를 구성하는 방법을 보여줍니다. 또한 AWS HealthOmics 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법을 알아봅니다.

주제

- 의 데이터 보호 AWS HealthOmics
- HealthOmics의 ID 및 액세스 관리
- 에 대한 규정 준수 검증 AWS HealthOmics
- HealthOmics의 복원력
- AWS HealthOmics 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)

의 데이터 보호 AWS HealthOmics

AWS 공동 책임 모델 AWS HealthOmics의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 관한 자세한 내용은 데이터 프라

이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 AWS HealthOmics 또는 기타 AWS 서비스 에서 콘솔 AWS CLI, API 또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

저장된 데이터 암호화

주제

- AWS 소유 키
- 고객 관리형 키
- 고객 관리형 키 생성
- 고객 관리형 키를 사용하는 데 필요한 IAM 권한
- 자세히 알아보기

저장된 민감한 고객 데이터를 보호하기 위해는 기본적으로 서비스 소유 AWS Key Management Service(AWS KMS) 키를 사용하여 암호화를 AWS HealthOmics 제공합니다. 고객 관리형 키도 지원됩니다. 고객 관리형 키에 대한 자세한 내용은 Amazon Key Management Service를 참조하세요.

모든 HealthOmics 데이터 스토어(스토리지 및 분석)는 고객 관리형 키 사용을 지원합니다. 데이터 스토어가 생성된 후에는 암호화 구성을 변경할 수 없습니다. 데이터 스토어가를 사용하는 경우 AWS_OWNED_KMS_KEY로 AWS 소유 키표시되며 저장 시 암호화에 사용되는 특정 키가 표시되지 않습니다.

HealthOmics 워크플로의 경우 고객 관리형 키는 임시 파일 시스템에서 지원되지 않지만 모든 데이터는 XTS-AES-256 블록 암호 암호화 알고리즘을 사용하여 저장 시 자동으로 암호화되어 파일 시스템을 암호화합니다. 워크플로 실행을 시작하는 데 사용되는 IAM 사용자 및 역할에는 워크플로 입력 및 출력 버킷에 사용되는 AWS KMS 키에 대한 액세스 권한도 있어야 합니다. 워크플로는 권한 부여를 사용하지 않으며 AWS KMS 암호화는 입력 및 출력 Amazon S3 버킷으로 제한됩니다. 워크플로 APIs에 사용되는 IAM 역할에는 사용된 AWS KMS 키와 입력 및 출력 Amazon S3 버킷에 대한 액세스 권한도 있어야 합니다. IAM 역할 및 권한을 사용하여 액세스 또는 AWS KMS 정책을 제어할 수 있습니다. 자세한 내용은에 대한 인증 및 액세스 제어를 AWS KMS 참조하세요.

HealthOmics Analytics와 AWS Lake Formation 함께를 사용하면 Lake Formation과 연결된 모든 복호화 권한도 입력 및 출력 Amazon S3 버킷에 부여됩니다. 가 권한을 AWS Lake Formation 관리하는 방법에 대한 자세한 내용은 AWS Lake Formation 설명서에서 확인할 수 있습니다.

HealthOmics Analytics는 Lake Formation kms:Decrypt에 Amazon S3 버킷에서 암호화된 데이터를 읽을 수 있는 권한을 부여합니다. Lake Formation을 통해 데이터를 쿼리할 수 있는 권한이 있는 한 암호화된 데이터를 읽을 수 있습니다. 데이터에 대한 액세스는 KMS 키 정책이 아닌 Lake Formation의 데이터 액세스 제어를 통해 제어됩니다. 자세한 내용은 Lake Formation 설명서의 AWS 통합 AWS 서비스 요청을 참조하세요.

AWS 소유 키

기본적으로 HealthOmics는 AWS 소유 키 를 사용하여 저장된 데이터를 자동으로 암호화합니다.이 데이터에는 개인 식별 정보(PII) 또는 보호 대상 건강 정보(PHI)와 같은 민감한 정보가 포함될 수 있기 때문입니다. AWS 소유 키 aren은 계정에 저장되지 않습니다. 여러 AWS 계정에서 사용하기 위해 AWS가 소유하고 관리하는 KMS 키 모음의 일부입니다.

AWS 서비스는 AWS 소유 키 를 사용하여 데이터를 보호할 수 있습니다. 를 확인, 관리 또는 액세스 AWS 소유 키하거나 사용을 감사할 수 없습니다. 하지만 사용자는 데이터를 암호화하는 키를 보호하기 위해 어떤 작업을 수행하거나 어떤 프로그램을 변경할 필요가 없습니다.

사용에 대한 월별 요금 또는 사용 요금이 부과되지 AWS 소유 키않으며 계정의 AWS KMS 할당량에 포함되지 않습니다. 자세한 내용은 AWS 관리형 키 단원을 참조하십시오.

고객 관리형 키

HealthOmics는 사용자가 생성, 소유 및 관리하는 대칭 고객 관리형 키를 사용하여 기존 AWS 소유 암호화에 두 번째 암호화 계층을 추가할 수 있도록 지원합니다. 사용자가 이 암호화 계층을 완전히 제어할 수 있으므로 다음과 같은 작업을 수행할 수 있습니다.

- 키 정책, IAM 정책 및 권한 부여 설정 및 유지 관리
- 키 암호화 자료 교체
- 키 정책 활성화 및 비활성화
- 태그 추가
- 키 별칭 만들기
- 삭제를 위한 스케줄 키

CloudTrail을 사용하여 HealthOmics가 사용자를 대신하여에 보내는 요청을 추적할 수도 AWS KMS 있습니다. 추가 AWS KMS 요금이 발생합니다. 자세한 내용은 고객 관리형 키를 참조하세요.

고객 관리형 키 생성

AWS Management Console 또는 AWS KMS APIs.

AWS Key Management Service 개발자 안내서의 대칭 고객 관리형 키 생성 단계를 따릅니다.

키 정책에서는 고객 관리형 키에 대한 액세스를 제어합니다. 모든 고객 관리형 키에는 키를 사용할 수 있는 사람과 키를 사용하는 방법을 결정하는 문장이 포함된 정확히 하나의 키 정책이 있어야 합니다. 고객 관리형 키를 생성할 때 키 정책을 지정할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 고객 관리형 키에 대한 액세스 관리를 참조하세요.

HealthOmics Analytics 리소스와 함께 고객 관리형 키를 사용하려면 호출 보안 주체가 키 정책에서 kms:CreateGrant 작업을 수행해야 합니다. 이를 통해 시스템은 FAS 토큰을 사용하여 지정된 KMS 키에 대한 액세스를 제어하는 고객 관리형 키에 대한 권한 부여를 생성할 수 있습니다. 이 키는 사용자에게 HealthOmics에 필요한 kms:grant 작업에 대한 액세스 권한을 부여합니다. 자세한 내용은 권한 부여 사용을 참조하세요.

HealthOmics 분석의 경우 호출 보안 주체에 대해 다음 API 작업을 허용해야 합니다.

- kms:CreateGrant는 특정 고객 관리형 키에 권한 부여를 추가하여 HealthOmics Analytics에서 권한 부여 작업에 대한 액세스를 허용합니다.
- kms:DescribeKey는 키를 검증하는 데 필요한 고객 관리형 키 세부 정보를 제공합니다. 이것은 모든 작업에 필요합니다.
- kms:GenerateDataKey는 모든 쓰기 작업에 대해 저장 리소스를 암호화할 수 있는 액세스를 제공합니다. 또한이 작업은 서비스가 호출자가 키를 사용할 수 있는 액세스 권한이 있는지 검증하는 데 사용할 수 있는 고객 관리형 키 세부 정보를 제공합니다.
- kms:Decrypt는 암호화된 리소스에 대한 읽기 또는 검색 작업에 대한 액세스를 제공합니다.

HealthOmics 스토리지 리소스에서 고객 관리형 키를 사용하려면 키 정책에서 HealthOmics 서비스 보안 주체와 호출 보안 주체를 허용해야 합니다. 이를 통해 서비스는 호출자가 키에 액세스할 수 있는지 확인하고 서비스 보안 주체를 사용하여 고객 관리형 키를 사용하여 스토어 관리를 실행할 수 있습니다. HealthOmics 스토리지의 경우 서비스 보안 주체의 키 정책은 다음 API 작업을 허용해야 합니다.

- kms:DescribeKey는 키를 검증하는 데 필요한 고객 관리형 키 세부 정보를 제공합니다. 이것은 모든 작업에 필요합니다.
- kms:GenerateDataKey는 모든 쓰기 작업에 대해 저장 리소스를 암호화할 수 있는 액세스를 제공합니다. 또한이 작업은 서비스가 호출자가 키를 사용할 수 있는 액세스 권한이 있는지 검증하는 데 사용할 수 있는 고객 관리형 키 세부 정보를 제공합니다.
- kms:Decrypt는 암호화된 리소스에 대한 읽기 또는 검색 작업에 대한 액세스를 제공합니다.

다음 예제는 서비스 보안 주체가 고객 관리형 키를 사용하여 암호화된 HealthOmics 시퀀스 또는 참조 저장소를 생성하고 상호 작용할 수 있도록 허용하는 정책 설명을 보여줍니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
{
    "Effect": "Allow",
    "Principal": {
        "Service": "omics.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
```

```
            "kms:DescribeKey",
            "kms:Encrypt",
            "kms:GenerateDataKey*"
        ],
        "Resource": "*"
            }
        ]
}
```

다음 예제는 데이터 스토어가 Amazon S3 버킷에서 데이터를 복호화할 수 있는 권한을 생성하는 정책을 보여줍니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "omics:GetReference",
                "omics:GetReferenceMetadata"
            ],
            "Resource": [
                "arn:aws:omics:us-east-1:123456789012:referenceStore/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::[[s3path]]/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
```

```
            "Resource": [
                "arn:aws:kms:us-east-1:123456789012:key/key_id"
            ],
            "Condition": {
              "StringEquals": {
                  "kms:ViaService": [
                    "s3.us-east-1.amazonaws.com"
                  ]
              }
            }
        }
    ]
}
```

고객 관리형 키를 사용하는 데 필요한 IAM 권한

고객 관리형 키를 사용하여 AWS KMS 암호화가 포함된 데이터 스토어와 같은 리소스를 생성할 때 IAM 사용자 또는 역할에 대한 키 정책 및 IAM 정책 모두에 필요한 권한이 있습니다.

kms:ViaService 조건 키를 사용하여 KMS 키 사용을 HealthOmics.

키 정책에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 IAM 정책 활성화를 참조하세요.

주제

- 분석 API 권한
- 스토리지 API 권한
- HealthOmics가 AWS KMS에서 권한 부여를 사용하는 방법
- AWS HealthOmics에 대한 암호화 키 모니터링

분석 API 권한

HealthOmics 분석의 경우 스토어를 생성하는 IAM 사용자 또는 역할에 kms:CreateGrant, kms:GenerateDataKey, kms:Decrypt 및 kms:DescribeKey 권한과 필요한 HealthOmics 권한이 있어야 합니다.

스토리지 API 권한

HealthOmics 스토리지 APIs의 경우 다음 API 작업을 호출하는 IAM 사용자 또는 역할에는 나열된 권한이 필요합니다.

CreateReferenceStore, CreateSequenceStore

스토어를 생성하려면 IAM 호출자에게 `kms:DescribeKey` 권한과 필요한 HealthOmics 권한이 있어야 합니다. HealthOmics 서비스 보안 주체는를 호출`kms:GenerateDataKeyWithoutPlaintext`하여 데이터 로드 및 액세스에 대한 액세스 검증 검사를 수행합니다.

StartReadSetImportJob, StartReferenceImportJob

데이터 가져오기 작업을 시작하려면 IAM 호출자에게 가져오기를 위한 스토어의 KMS 키에 대한 `kms:Decrypt` 및 `kms:GenerateDataKey` 권한과 가져올 객체가 포함된 Amazon S3 버킷에 대한 `kms:Decrypt` 권한이 있어야 합니다. 또한 호출에 전달된 역할에는 가져올 객체가 포함된 Amazon S3 버킷에 대한 `kms:Decrypt` 권한이 있어야 합니다. IAM 호출자에게는 해당 역할을 작업에 전달할 수 있는 권한도 있어야 합니다.

CreateMultipartReadSetUpload, UploadReadSetPart, CompleteMultipartReadSetUpload

멀티파트 업로드를 완료하려면 IAM 호출자에게 멀티파트 업로드를 생성, 업로드 및 완료하기 `kms:GenerateDataKey` 위한 `kms:Decrypt` 및가 있어야 합니다.

StartReadSetExportJob

데이터 내보내기 작업을 시작하려면 IAM 호출자에게 스토어의 KMS 키가 및에서 내보낼 수 있는 `kms:Decrypt` 권한`kms:GenerateDataKey`과 객체를 수신하는 Amazon S3 버킷에 대한 `kms:Decrypt` 권한이 있어야 합니다. 또한 호출에 전달된 역할에는 객체를 수신하는 Amazon S3 버킷에 대한 `kms:Decrypt` 권한이 있어야 합니다. IAM 호출자에게는 해당 역할을 작업에 전달할 수 있는 권한도 있어야 합니다.

StartReadsetActivationJob

읽기 세트 활성화 작업을 시작하려면 IAM 호출자에게 객체에 대한 `kms:Decrypt` 및 `kms:GenerateDataKey` 권한이 있어야 합니다.

GetReference, GetReadSet

스토어에서 객체를 읽으려면 IAM 호출자에게 객체에 대한 `kms:Decrypt` 권한이 있어야 합니다.

읽기 세트 S3 GetObject

Amazon S3 `GetObject` API를 사용하여 스토어에서 객체를 읽으려면 IAM 호출자에게 객체에 대한 `kms:Decrypt` 권한이 있어야 합니다. 고객 관리형 키와 AWS 소유 키 구성 모두에 대해이 권한을 설정합니다.

HealthOmics가 AWS KMS에서 권한 부여를 사용하는 방법

HealthOmics Analytics에서 고객 관리형 KMS 키를 사용하려면 권한 부여가 필요합니다. HealthOmics 워크플로에는 권한 부여가 필요하지 않거나 사용되지 않습니다. HealthOmics Storage는 서비스 보안 주체로부터 직접 고객 관리형 키를 사용하므로 권한 부여를 사용하지 마십시오. 고객 관리형 키로 암호화된 분석 스토어를 생성하면 HealthOmics 분석은 CreateGrant 요청을 AWS KMS에 전송하여 사용자를 대신하여 권한 부여를 생성합니다. AWS KMS의 권한 부여는 HealthOmics에 고객 계정의 KMS 키에 대한 액세스 권한을 부여하는 데 사용됩니다.

HealthOmics 분석이 사용자를 대신하여 생성하는 권한 부여를 취소하거나 사용 중지하는 것은 권장되지 않습니다. 계정에서 AWS KMS 키를 사용할 수 있는 권한을 HealthOmics에 부여하는 권한 부여를 취소하거나 사용 중지하면 HealthOmics는이 데이터에 액세스하거나, 데이터 스토어에 푸시된 새 리소스를 암호화하거나, 데이터 스토어를 가져올 때 복호화할 수 없습니다.

HealthOmics에 대한 권한 부여를 취소하거나 사용 중지하면 변경 사항이 즉시 적용됩니다. 액세스 권한을 취소하려면 권한 부여를 취소하는 대신 데이터 스토어를 삭제하는 것이 좋습니다. 데이터 스토어를 삭제하면 HealthOmics는 사용자를 대신하여 권한 부여를 사용 중지합니다.

AWS HealthOmics에 대한 암호화 키 모니터링

CloudTrail을 사용하여 고객 관리형 키를 사용할 때가 사용자를 대신하여 AWS KMS 에 AWS HealthOmics 보내는 요청을 추적할 수 있습니다. CloudTrail 로그의 로그 항목은 userAgent 필드에 HealthOmics.amazonAWS.com을 표시하여 HealthOmics의 요청을 명확하게 구분합니다. userAgent

다음 예제는 고객 관리형 키로 암호화된 데이터에 액세스하기 위해 HealthOmics에서 호출한 AWS KMS 작업을 모니터링하기 위한 CreateGrant, GenerateDataKey, Decrypt 및 DescribeKey에 대한 CloudTrail 이벤트입니다.

다음은 CreateGrant를 사용하여 HealthOmics 분석이 고객 제공 KMS 키에 액세스하도록 허용하여 HealthOmics가 해당 KMS 키를 사용하여 저장된 모든 고객 데이터를 암호화할 수 있도록 하는 방법도 보여줍니다.

자체 권한 부여를 생성할 필요는 없습니다. HealthOmics는 AWS KMS에 CreateGrant 요청을 전송하여 사용자를 대신하여 권한 부여를 생성합니다. 의 권한 부여 AWS KMS 는 HealthOmics에 고객 계정의 AWS KMS 키에 대한 액세스 권한을 부여하는 데 사용됩니다.

```
{
    "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"
}
```

다음 예제에서는 GenerateDataKey를 사용하여 사용자가 데이터를 저장하기 전에 암호화하는 데 필요한 권한을 갖도록 하는 방법을 보여줍니다.

```
{
    "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"
}
```

자세히 알아보기

다음 리소스는 저장 데이터 암호화에 대한 자세한 정보를 제공합니다.

AWS Key Management Service 기본 개념에 대한 자세한 내용은 설명서를 참조하세요 AWS KMS .

AWS KMS 설명서의 보안 모범 사례에 대한 자세한 내용을 참조하세요.

전송 중 암호화

AWS HealthOmics 는 TLS 1.2 이상을 사용하여 퍼블릭 엔드포인트와 백엔드 서비스를 통해 전송 중인 데이터를 암호화합니다.

HealthOmics의 ID 및 액세스 관리

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어하는 데 도움이 AWS 서비스 되는 입니다. IAM 관리자는 AWS HealthOmics 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는 사용자를 제어합니다. HealthOmics IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

주제

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 역할에 따라 다릅니다.

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한 요청(참조AWS HealthOmics 자격 증명 및 액세스 문제 해결)
- 서비스 관리자 - 사용자 액세스 결정 및 권한 요청 제출(AWS HealthOmics 에서 IAM을 사용하는 방법 참조)
- IAM 관리자 - 액세스를 관리하기 위한 정책 작성(에 대한 자격 증명 기반 정책 예제 AWS HealthOmics 참조)

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증되어야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다.

로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 API 요청용AWS Signature Version 4 섹션을 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 작업 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자가 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 IAM 사용자 사용 사례 섹션을 참조하세요.

IAM 역할

IAM 역할은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. 사용자에서 IAM 역할(콘솔)로 전환하거나 또는 API 작업을 호출하여 역할을 수임할 수 있습니다. AWS CLI AWS 자세한 내용은 IAM 사용 설명서의 역할 수임 방법을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 교차 계정 리소스 액세스를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 작업에 대한 권한을 정의합니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책 및 인라인 정책 중에서 선택 섹션을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

기타 정책 유형

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - ID 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 IAM 엔터티의 권한 범위를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

AWS HealthOmics 에서 IAM을 사용하는 방법

IAM을 사용하여 AWS HealthOmics에 대한 액세스를 관리하기 전에 AWS HealthOmics에서 사용할 수 있는 IAM 기능에 대해 알아봅니다.

에서 사용할 수 있는 IAM 기능 AWS HealthOmics

IAM 특성	HealthOmics 지원
자격 증명 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예

IAM 특성	HealthOmics 지원
정책 조건 키	아니요
ACL	아니요
ABAC(정책의 태그)	예
임시 보안 인증	예
엔터티 권한	예
서비스 역할	예
서비스 연결 역할	아니요

HealthOmics 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방식을 전체적으로 알아보려면 IAM 사용 설명서의 AWS IAM으로 작업하는 서비스를 참조하세요.

교차 서비스 혼동된 대리자 방지

혼동된 대리자 문제는 작업을 수행할 권한이 없는 엔터티가 권한이 더 많은 엔터티에게 작업을 수행하도록 강요할 수 있는 보안 문제입니다. 에서 AWS교차 서비스 가장은 혼동된 대리자 문제를 초래할 수 있습니다. 교차 서비스 가장은 한 서비스(직접적으로 호출하는 서비스)가 다른 서비스(직접적으로 호출되는 서비스)를 직접적으로 호출할 때 발생할 수 있습니다. 호출하는 서비스는 다른 고객의 리소스에 대해 액세스 권한이 없는 방식으로 작동하게 권한을 사용하도록 조작될 수 있습니다. 이를 방지하기 위해 AWS 에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 위탁자를 사용하여 모든 서비스에 대한 데이터를 보호하는 데 도움이 되는 도구를 제공합니다.

리소스 정책에서 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하여 AWS HealthOmics가 리소스에 다른 서비스를 제공하는 권한을 제한하는 것이 좋습니다.

HealthOmics가 수임하는 역할에서 혼동된 대리자 문제를 방지하려면 역할의 신뢰 정책`arn:aws:omics:region:accountNumber:*`에서의 값을 `aws:SourceArn`로 설정합니다. 와일드카드(*)는 모든 HealthOmics 리소스에 조건을 적용합니다.

다음 신뢰 관계 정책은 HealthOmics에 리소스에 대한 액세스 권한을 부여하고 `aws:SourceArn` 및 `aws:SourceAccount` 전역 조건 컨텍스트 키를 사용하여 혼동된 대리자 문제를 방지합니다. HealthOmics에 대한 역할을 생성할 때이 정책을 사용합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "omics.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "123456789012"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:omics:us-east-1:123456789012:*"
        }
      }
    }
  ]
}
```

HealthOmics에 대한 자격 증명 기반 정책

ID 기반 정책 지원: 예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 고객 관리형 정책으로 사용자 지정 IAM 권한 정의를 참조하세요.

IAM ID 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

HealthOmics의 자격 증명 기반 정책 예제

AWS HealthOmics 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요에 대한 자격 증명 기반 정책 예제 AWS HealthOmics.

HealthOmics 내의 리소스 기반 정책

리소스 기반 정책 지원: 아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예제는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려는 경우, 전체 계정이나 다른 계정의 IAM 개체를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM에서 교차 계정 리소스 액세스를 참조하세요.

HealthOmics에 대한 정책 작업

정책 작업 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

HealthOmics 작업 목록을 보려면 서비스 승인 참조의 AWS HealthOmics에서 정의한 작업을 참조하세요.

HealthOmics의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
omics
```

단일 문에서 여러 작업을 지정하려면 쉼표로 구분합니다.

```
"Action": [
```

```
      "omics:action1",
      "omics:action2"
         ]
```

AWS HealthOmics 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요에 대한 자격 증명 기반 정책 예제 AWS HealthOmics.

HealthOmics에 대한 정책 리소스

정책 리소스 지원: 예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

HealthOmics 리소스 유형 및 해당 ARNs 목록을 보려면 서비스 승인 참조의 AWS HealthOmics에서 정의한 리소스를 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 AWS HealthOmics에서 정의한 작업을 참조하세요.

AWS HealthOmics 자격 증명 기반 정책의 예를 보려면 섹션을 참조하세요에 대한 자격 증명 기반 정책 예제 AWS HealthOmics.

HealthOmics에 사용되는 정책 조건 키

정책 조건 키는 HealthOmics에서 지원되지 않습니다.

HealthOmics의 액세스 제어 목록(ACLs)

ACL 지원: 아니요

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

HealthOmics를 사용한 ABAC(속성 기반 액세스 제어)

ABAC 지원(정책의 태그): 예

속성 기반 액세스 제어(ABAC)는 태그라고 불리는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. IAM 엔터티 및 AWS 리소스에 태그를 연결한 다음 보안 주체의 태그가 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계할 수 있습니다.

태그에 근거하여 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`또는 `aws:TagKeys`조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여를 통한 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

HealthOmics 리소스 태그 지정에 대한 자세한 내용은 섹션을 참조하세요HealthOmics에서 리소스 태그 지정.

다음 예제에서는 특정 태그 없이 리소스에 대한 액세스를 거부하는 IAM 정책을 작성하는 방법을 보여줍니다.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "omics:*"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "Null": {
                  "aws:RequestTag/MyCustomTag": "true"
```

```
                }
            }
        }
    ]
}
```

HealthOmics에서 임시 자격 증명 사용

임시 자격 증명 지원: 예

임시 자격 증명은 AWS 리소스에 대한 단기 액세스를 제공하며 페더레이션을 사용하거나 역할을 전환할 때 자동으로 생성됩니다. 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 것이 AWS 좋습니다. 자세한 내용은 IAM 사용 설명서의 IAM의 임시 보안 자격 증명 및 IAM으로 작업하는AWS 서비스 섹션을 참조하세요.

HealthOmics에 대한 교차 서비스 보안 주체 권한

전달 액세스 세션(FAS) 지원: 예

전달 액세스 세션(FAS)은를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

HealthOmics의 서비스 역할

서비스 역할 지원: 예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스 AWS에 권한을 위임할 역할 생성을 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 HealthOmics 기능이 중단될 수 있습니다. HealthOmics가 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

HealthOmics의 서비스 연결 역할

서비스 연결 역할 지원: 아니요

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는AWS 서비스를 참조하세요. 서비스 연결 역할 열에서 Yes가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

에 대한 자격 증명 기반 정책 예제 AWS HealthOmics

기본적으로 사용자 및 역할에는 AWS HealthOmics 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성(콘솔)을 참조하세요.

각 리소스 유형에 대한 ARNs 형식을 포함하여 AWS HealthOmics에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 승인 참조의 AWS HealthOmics에 사용되는 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- HealthOmics 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 AWS HealthOmics 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.

- 최소 권한 적용 – IAM 정책을 사용하여 권한을 설정하는 경우, 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- IAM 정책의 조건을 사용하여 액세스 추가 제한 – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 확인하여 안전하고 기능적인 권한 보장 - IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 새로운 및 기존 정책을 확인합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer에서 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 통한 보안 API 액세스를 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

HealthOmics 콘솔 사용

AWS HealthOmics 콘솔에 액세스하려면 최소 권한 집합이 있어야 합니다. 이러한 권한은에서 AWS HealthOmics 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다. 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "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 에 대한 관리형 정책 AWS HealthOmics

AWS 관리형 정책은에서 생성하고 관리하는 독립 실행형 정책입니다 AWS. AWS 관리형 정책은 사용자, 그룹 및 역할에 권한 할당을 시작할 수 있도록 많은 일반적인 사용 사례에 대한 권한을 제공하도록 설계되었습니다.

AWS 관리형 정책은 모든 AWS 고객이 사용할 수 있으므로 특정 사용 사례에 대해 최소 권한을 부여하지 않을 수 있습니다. 사용 사례에 고유한 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다.

AWS 관리형 정책에 정의된 권한은 변경할 수 없습니다. 가 관리형 정책에 정의된 권한을 AWS 업데이트하는 AWS 경우 업데이트는 정책이 연결된 모든 보안 주체 자격 증명(사용자, 그룹 및 역할)에 영향을 줍니다. AWS AWS 서비스 는 새가 시작되거나 기존 서비스에 새 API 작업을 사용할 수 있게 될 때 AWS 관리형 정책을 업데이트할 가능성이 높습니다.

자세한 내용은 IAM 사용자 가이드의 AWS 관리형 정책을 참조하세요.

AWS 관리형 정책: AmazonOmicsFullAccess

`AmazonOmicsFullAccess` 정책을 IAM 자격 증명에 연결하여 HealthOmics에 대한 전체 액세스 권한을 부여할 수 있습니다.

이 정책은 모든 HealthOmics 작업에 대한 전체 액세스 권한을 부여합니다. 주석 또는 변형 저장소를 생성하면 Omics는 Resource Access Manager(RAM) 콘솔의 리소스 공유 초대를 통해 해당 저장소에 대한 액세스 권한도 부여합니다. Lake Formation을 통한 리소스 공유 초대에 대한 자세한 내용은 Lake Formation의 교차 계정 데이터 공유를 참조하세요. Omics 관리자 정책의 경우 Amazon S3 버킷에 액세스하려면 다음 권한도 필요합니다.

- 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 관리형 정책: AmazonOmicsReadOnlyAccess

해당 자격 증명에 대한 권한을 읽기 전용 액세스로 제한하려는 경우 IAM 자격 증명에 `AWSOmicsReadOnlyAccess` 정책을 연결할 수 있습니다.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "omics:Get*",
                "omics:List*"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 관리형 정책에 대한 HealthOmics 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후부터 HealthOmics의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 HealthOmics 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경	설명	Date
AmazonOmicsFullAccess - 새 정책 추가	HealthOmics는 사용자에게 모든 작업 및 리소스에 대한 전체 액세스 권한을 부여하는 새 정책을 추가했습니다. 자세한 내용은 AmazonOmicsFullAccess를 참조하세요.	2023년 2월 23일
HealthOmics에서 변경 사항 추적 시작	HealthOmics는 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2022년 11월 29일
AmazonOmicsReadOnlyAccess - 새 정책 추가	HealthOmics는 읽기 전용으로 액세스를 제한하는 새 정책을 추가했습니다. 자세한 내용은 AmazonOmicsReadOnlyAccess를 참조하세요.	2022년 11월 29일

AWS HealthOmics 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 AWS HealthOmics 및 IAM 작업 시 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- HealthOmics에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내 외부의 사람이 내 HealthOmics 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

HealthOmics에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 표시되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 omics:*GetWidget* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 omics:GetWidget on resource: my-example-widget
```

이 경우, omics:*GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

iam:PassRole 작업을 수행할 권한이 없다는 오류가 수신되면 AWS HealthOmics에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 예제 오류는 라는 IAM 사용자가 콘솔을 사용하여 AWS HealthOmics에서 작업을 수행하려고 marymajor 할 때 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우, Mary가 `iam:PassRole`작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 HealthOmics 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- AWS HealthOmics가 이러한 기능을 지원하는지 여부를 알아보려면 섹션을 참조하세요AWS HealthOmics 에서 IAM을 사용하는 방법.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 타사에 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사가 AWS 계정 소유한에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM의 크로스 계정 리소스 액세스를 참조하세요.

에 대한 규정 준수 검증 AWS HealthOmics

타사 감사자는 여러 규정 준수 프로그램의 AWS HealthOmics 일환으로의 보안 및 AWS 규정 준수를 평가합니다. 여기에는 HIPAA, FedRAMP 등이 포함됩니다. 다음 표에는 HealthOmics 서비스에 대한 규정 준수 인증이 나와 있습니다.

인증	링크
HIPAA	HIPAA 적격 서비스 참조
HiTrust-CSF	Health Information Trust Alliance 공통 보안 프레임워크

인증	링크
FedRAMP Moderate(동부/서부)	연방 위험 및 권한 부여 관리 프로그램
ISO/CSA STAR	ISO 및 CSA STAR 인증
C5	클라우드 컴퓨팅 규정 준수 제어 카탈로그
DoD CC SRG IL2	국방부 클라우드 컴퓨팅 보안 요구 사항 가이드
ENS High	Esquema Nacional de Seguridad
FINMA	스위스 금융 시장 감독 기관
ISMAP	정보 시스템 보안 관리 및 평가 프로그램
OSPAR	아웃소싱된 서비스 공급자의 감사 보고서
PCI	결제 카드 산업 데이터 보안 표준
피네이크	은행 연결 CCI - 타사 검증
PiTuKri	클라우드 서비스의 정보 보안 평가 기준
SOC 1,2,3	시스템 및 조직 제어

특정 규정 준수 프로그램의 범위에 속하는 모든 AWS 서비스 목록은 규정 준수 프로그램 제공 범위 내 AWS 서비스규정 준수 프로그램 참조하세요. 일반적인 정보는 AWS 규정 준수 프로그램을 참조하세요.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 에서 보고서 다운로드 AWS Artifact에서 .

HealthOmics 데이터 스토어는 내부 파일 이름 지정 및 리소스 태그 지정에 샘플 ID를 사용합니다. 데이터를 수집하기 전에 샘플 ID에 PHI 데이터가 포함되어 있는지 확인합니다. 그럴 경우 데이터를 수집하기 전에 샘플 ID를 변경합니다. 자세한 내용은 AWS HIPAA 규정 준수 웹 페이지의 지침을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS HealthOmics 는 규정 준수를 지원할 다음과 같은 리소스를 AWS 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서: 이 배포 안내서에서는 아키텍처 고려 사항에 관해 설명하고 AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- HIPAA 보안 및 규정 준수를 위한 설계 백서 -이 백서에서는 기업이 AWS 를 사용하여 HIPAA 준수 애플리케이션을 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 -이 워크북 및 가이드 모음은 산업 및 위치에 적용될 수 있습니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 - 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 AWS Config평가합니다.
- AWS Security Hub CSPM -이 AWS 서비스는 보안 업계 표준 및 모범 사례 준수 여부를 확인하는 데 도움이 AWS 되는 내 보안 상태에 대한 포괄적인 보기를 제공합니다.

HealthOmics의 복원력

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.는 지연 시간이 짧고 처리량이 많으며 중복성이 높은 네트워킹과 연결된 물리적으로 분리되고 격리된 여러 가용 영역을 AWS 리전 제공합니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

AWS 글로벌 인프라 외에도 AWS HealthOmics는 데이터 복원력 및 백업 요구 사항을 지원하는 몇 가지 기능을 제공합니다.

AWS HealthOmics 및 인터페이스 VPC 엔드포인트(AWS PrivateLink)

인터페이스 VPC 엔드포인트를 생성 AWS HealthOmics 하여 VPC와 간에 프라이빗 연결을 설정할 수 있습니다. 인터페이스 엔드포인트는 인터넷 게이트웨이AWS PrivateLink, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 HealthOmics API 작업에 비공개로 액세스하는 데 사용할 수 있는 기술로 구동됩니다. VPC의 인스턴스는 HealthOmics API 작업과 통신하는 데 퍼블릭 IP 주소가 필요하지 않습니다. VPC와 HealthOmics 간의 트래픽은 Amazon 네트워크를 벗어나지 않습니다.

각 인터페이스 엔드포인트는 서브넷에서 하나 이상의 Elastic Network Interfaces로 표현됩니다.

자세한 내용은 Amazon VPC 사용 설명서에서 인터페이스 VPC 종단점(AWS PrivateLink)을 참조하세요.

VPC 엔드포인트 정책은 이스라엘(텔아비브)을 제외한 모든 리전의 HealthOmics에서 지원됩니다. 기본적으로 엔드포인트를 통해 HealthOmics에 대한 전체 액세스가 허용됩니다.

HealthOmics VPC 엔드포인트에 대한 고려 사항

HealthOmics에 대한 인터페이스 VPC 엔드포인트를 설정하기 전에 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 속성 및 제한 사항을 검토해야 합니다.

HealthOmics는 VPC에서 모든 HealthOmics 스토리지 API 작업을 호출할 수 있도록 지원합니다.

VPC 엔드포인트 정책은 기본적으로 HealthOmics에서 지원되지 않지만 HealthOmics 스토리지 작업에 대한 전체 HealthOmics 액세스를 위한 VPC 엔드포인트HealthOmics 생성할 수 있습니다. 자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

HealthOmics용 인터페이스 VPC 엔드포인트 생성

Amazon VPC 콘솔 또는 AWS Command Line Interface ()를 사용하여 HealthOmics 서비스에 대한 VPC 엔드포인트를 생성할 수 있습니다AWS CLI. 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

다음 서비스 이름을 사용하여 HealthOmics용 VPC 엔드포인트를 생성합니다.

- 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

미국 동부(버지니아 북부) 및 미국 서부(오레곤) 리전은 AWS PrivateLink FIPS 엔드포인트를 지원합니다. 이러한 리전의 경우 다음 서비스 이름을 사용할 수도 있습니다.

- 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

엔드포인트에 대해 프라이빗 DNS를 켜면 리전의 기본 DNS 이름인를 사용하여 HealthOmics에 API 요청을 할 수 있습니다`omics.us-east-1.amazonaws.com`.

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트를 통해 서비스 액세스를 참조하세요.

HealthOmics에 대한 VPC 엔드포인트 정책 생성

HealthOmics에 대한 액세스를 제어하는 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업
- 작업을 수행할 수 있는 리소스

자세한 내용은 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

예: HealthOmics 작업에 대한 VPC 엔드포인트 정책.

다음은 HealthOmics에 대한 엔드포인트 정책의 예입니다. 엔드포인트에 연결되면이 정책은 모든 리소스의 모든 보안 주체에 대해 HealthOmics 작업에 대한 액세스 권한을 부여합니다.

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 URIs를 사용하여 읽기 세트에 액세스하기 위한 특별 고려 사항

프라이빗 연결을 사용할 때 Amazon S3 URIs를 통해 읽기 세트에 액세스하려면 시퀀스 스토어에서 PrivateLink 인터페이스 엔드포인트를 설정합니다. 설정 후 엔드포인트의 형식은 다음과 같습니다.

```
com.amazonaws.region.storage-omics
com.amazonaws.region.control-storage-omics
```

게이트웨이 엔드포인트를 사용하려면 Amazon S3용 게이트웨이 엔드포인트 가이드에 따라 게이트웨이 엔드포인트를 구성합니다. HealthOmics는 Amazon S3 버킷을 소유하므로 버킷 정책을 생성하거나 조정할 필요가 없습니다. 게이트웨이 엔드포인트는 데이터에 액세스하는 사용자 또는 역할에 연결된 정책에 의존하지만 더 제한적인 정책으로 엔드포인트를 구성할 수도 있습니다. 이러한 정책에는 Amazon S3 액세스 포인트 ARN 및 Amazon S3 작업에 따른 액세스 제한이 포함될 수 있습니다.

AWS HealthOmics 모니터링

모니터링은 AWS HealthOmics 및 기타 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는 데 중요한 부분입니다.는 AWS HealthOmics를 모니터링하고, 이상이 있을 때 이를 보고하고, 적절한 경우 자동 조치를 취할 수 있도록 다음과 같은 모니터링 도구를 AWS 제공합니다.

- Amazon CloudWatch는 AWS 리소스와 AWS 실행 중인 애플리케이션을 실시간으로 모니터링합니다. 지표를 수집 및 추적하고, 사용자 지정 대시보드를 생성할 수 있으며, 지정된 지표가 지정한 임곗값에 도달하면 사용자에게 알리거나 조치를 취하도록 경보를 설정할 수 있습니다. 예를 들어 CloudWatch에서 Amazon EC2 인스턴스의 CPU 사용량 또는 기타 지표를 추적하고 필요할 때 자동으로 새 인스턴스를 시작할 수 있습니다. 자세한 내용은 CloudWatch 사용 설명서를 참조하세요.
- Amazon CloudWatch Logs로 Amazon EC2 인스턴스, CloudTrail, 기타 소스의 로그 파일을 모니터링, 저장 및 액세스할 수 있습니다. CloudWatch Logs는 로그 파일의 정보를 모니터링하고 특정 임곗값에 도달하면 사용자에게 알릴 수 있습니다. 또한 매우 내구력 있는 스토리지에 로그 데이터를 저장할 수 있습니다. 자세한 내용은 Amazon CloudWatch Logs 사용 설명서를 참조하십시오.
- AWS CloudTrail은 직접 수행하거나 AWS 계정 를 대신하여 수행한 API 호출 및 관련 이벤트를 캡처하고 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 어떤 사용자 및 계정이 AWS를 호출했는지 어떤 소스 IP 주소에 호출이 이루어졌는지 언제 호출이 발생했는지 확인할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.
- EventBridge: 애플리케이션을 다양한 소스의 데이터와 쉽게 연결할 수 있는 서버리스 이벤트 버스 서비스입니다. EventBridge는 자체 애플리케이션, Software-as-a-Service(SaaS) 애플리케이션 및 AWS 서비스의 실시간 데이터 스트림을 제공하고 해당 데이터를 Lambda와 같은 대상으로 라우팅합니다. 이를 통해 서비스에서 발생하는 이벤트를 모니터링하고 이벤트 기반 아키텍처를 구축할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용 설명서를 참조하세요.

> **Note**
>
> 서비스 업데이트의 경우 Personal Health Dashboard를 구성하고 모니터링합니다. 대시보드 관리 방법에 대한 자세한 내용은 AWS Health Dashboard 시작하기를 참조하세요.

주제

- 를 사용하여 AWS HealthOmics API 호출 로깅 AWS CloudTrail
- 에서 EventBridge 사용 AWS HealthOmics

S3 액세스 로깅

저장 생성 액세스 로그를 사용하여 HealthOmics 시퀀스 스토어 데이터에 대한 Amazon S3 API 액세스를 모니터링할 수 있습니다. CloudWatch를 사용하여 HealthOmics API 작업의 S3 액세스를 모니터링할 수 있습니다. CloudWatch는 자체 계정에서 발생하는 Amazon S3 액세스에 대한 가시성을 제공합니다. 데이터 소유자가 타사 계정에 대한 액세스를 공유하는 경우 CloudWatch에서 액세스 로깅을 사용할 수 없습니다. 대신 구성된 Amazon S3 버킷의 데이터에 대한 모든 S3 액세스를 로깅하는 스토어의 S3 액세스 로그를 사용합니다. Amazon S3

`CreateSequenceStore` 또는 `UpdateSequenceStore` API 작업을 사용하여 S3 액세스 로그를 구성합니다. 또한 HealthOmics 서비스 보안 주체(`omics.amazonaws.com`)에 구성된 S3 접두사에 대한 `s3:PutObject` 권한이 있는지 확인합니다.

> **Note**
>
> 로그는 대상 버킷의 기본 암호화 구성을 사용합니다. 버킷이 고객 관리형 키를 사용하는 경우 서비스 보안 주체는 쓰기에 키를 사용할 수 있는 액세스 권한이 있어야 합니다.

액세스 로깅을 끄려면 `UpdateSequenceStore`를 사용하고 액세스 로그 구성을 공백으로 설정합니다.

CloudWatch 지표를 사용하여 HealthOmics 모니터링

원시 데이터를 수집하여 읽기 가능하며 실시간에 가까운 지표로 처리하는 CloudWatch를 사용하여 HealthOmics를 모니터링할 수 있습니다. 이러한 통계는 15개월간 보관되므로 기록 정보에 액세스하고 웹 애플리케이션 또는 서비스가 어떻게 실행되고 있는지 전체적으로 더 잘 파악할 수 있습니다. 특정 임곗값을 주시하다가 해당 임곗값이 충족될 때 알림을 전송하거나 조치를 취하도록 경보를 설정할 수도 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하세요.

AWS HealthOmics 서비스는 `AWS/Omics` 네임스페이스에 다음 지표를 보고합니다.

API 호출 수 지표는 다음 AWS HealthOmics APIs. API 작업 차원만 보고됩니다.

- 참조 및 참조 스토어 APIs- CreateReferenceStore, DeleteReferenceStore, StartReferenceImportJob

- 시퀀스 스토어 및 읽기 세트 APIs - CreateSequenceStore, DeleteSequenceStore, StartReadSetImportJob, StartReadSetActivationJob, StartReadSetExportJob
- 변형 저장소 APIs- CreateVariantStore, DeleteVariantStore, StartVariantImportJob, CancelVariantImportJob
- 주석 저장소 APIs- CreateAnnotationStore, DeleteAnotationStore, StartAnnotationImportJob, CancelAnnotationImportJob
- 워크플로, 실행 및 실행 그룹 APIs- CreateWorkflow, DeleteWorkflow, StartRun, CancelRun, DeleteRun, CreateRunGroup, DeleteRunGroup

AWS HealthOmics 지표 보기

에 대한 CloudWatch 지표 AWS HealthOmics 는 CloudWatch 콘솔에서 볼 수 있습니다.

지표 보기(CloudWatch 콘솔)

1. AWS Management Console에 로그인하고 CloudWatch 콘솔을 엽니다.
2. 지표를 선택하고 모든 지표를 선택한 다음 AWS/사용을 선택합니다.
3. 에 대한 서비스를 필터링합니다AWS HealthOmics.
4. 차원과 지표 이름을 선택한 다음 그래프에 추가를 선택합니다.
5. 날짜 범위 값을 선택합니다. 선택한 날짜 범위에 대한 지표 개수가 그래프에 표시됩니다.

CloudWatch를 사용하여 경보 생성

CloudWatch 경보는 지정한 기간 동안 단일 지표를 감시하고, Amazon Simple Notification Service (Amazon SNS) 주제 또는 Auto Scaling 정책에 알림 보내기와 같은 하나 이상의 작업을 수행합니다. 이러한 작업은 지정한 여러 기간 동안 지정된 임계값에 따른 지표의 값을 기반으로 합니다. 또한 CloudWatch는 경보로 인해 상태가 변경되면 Amazon SNS 메시지를 전송할 수 있습니다.

CloudWatch 경보는 상태가 변경되어 지정한 기간 동안 지속되는 경우에만 작업을 호출합니다.

지표 보기(CloudWatch 콘솔)

1. AWS Management Console에 로그인하고 CloudWatch 콘솔을 엽니다.
2. 알람를 선택한 다음 알람 생성을 선택합니다.
3. AWS/사용을 선택한 다음 서비스 AWS HealthOmics 차원을 사용하여 지표를 선택합니다.
4. 시간 범위에서 모니터링할 시간 범위를 선택한 후, 다음를 선택합니다.

5. 이름 및 설명을 입력합니다.
6. 항상에서 >=를 선택하고 최대값을 입력합니다.
7. 경보 상태에 도달할 때 CloudWatch에서 이메일을 보내도록 하려면 작업 섹션의이 경보가 발생할 때마다 상태에서 경보를 선택합니다. 알림 전송 대상에서 메일링 목록을 선택하거나 새 목록을 선택하고 새 메일링 목록을 생성합니다.
8. 알람 미리보기 섹션에서 경보를 미리 볼 수 있습니다. 경보가 만족스러우면 경보 생성을 선택합니다.

CloudWatch Logs를 사용하여 HealthOmics 모니터링

HealthOmics는 실행을 이해하고 문제를 해결하는 데 도움이 되는 다양한 로그를 생성합니다. 로그는 CloudWatch와 Amazon S3의 두 곳에서 사용할 수 있습니다.

기본적으로 실행에는 로깅이 켜져 있습니다. 요청`LogLevel = OFF`에서를 설정하여 실행에 대한 로깅을 끌 수도 있습니다startrun.

> **Note**
>
> 서비스 업데이트의 경우 Personal Health Dashboard를 구성하고 모니터링합니다. 대시보드 관리 방법에 대한 자세한 내용은 AWS Health Dashboard 시작하기를 참조하세요.

주제

- HealthOmics 워크플로의 로그 유형
- CloudWatch의 로그그
- Amazon S3의 로그
- CLI의 대화형 CloudWatch Logs
- 콘솔에서 CloudWatch Logs에 액세스

HealthOmics 워크플로의 로그 유형

HealthOmics는 워크플로에 대해 다음과 같은 유형의 로그를 제공합니다.

- 엔진 로그 - 기본 워크플로 엔진(Nextflow, WDL 및 CWL)은 실행을 위한 엔진 로그를 생성합니다. 이러한 로그는 워크플로 정의 문제를 해결하는 데 도움이 될 수 있습니다.

- 매니페스트 로그 실행 - 이러한 로그는 작업 상태, 시작 시간, 중지 시간 및 실패 이유(작업이 실패한 경우)와 같은 각 실행 작업에 대한 상위 수준의 정보를 제공합니다.

  매니페스트 로그 실행은 리소스 최적화 기회를 이해하는 데 도움이 될 수 있는 리소스 사용률 통계도 보고합니다. 이러한 통계에는 다음이 포함됩니다.

  - cpusAverage
  - cpusMaximum
  - cpusReserved
  - gpusReserved
  - memoryAverageGiB
  - memoryMaximumGiB
  - memoryReservedGiB
  - runningSeconds
- 실행 로그 - 실행 로그는 전체 실행 상태와 개별 작업이 시작, 실행, 중지 및 완료된 시간을 제공합니다. 또한 실행 로그를 통해 파일 가져오기 및 내보내기 단계를 파악할 수 있습니다.
- 작업 로그 - 작업 로그는 실행 중인 개별 작업에 대한 자세한 로깅 정보를 제공합니다. 작업 로그의 출력은 작업 정의와 코드에서 로그 문을 사용하는 위치에 따라 달라집니다. 작업 로그가 필요한 수준의 인사이트를 제공하지 않는 경우 작업 정의에 추가 로그 문을 추가하여 더 통찰력 있는 작업 로그를 생성하는 것이 좋습니다.
- 캐시 로그 실행 - 캐시 로그 실행은 실행 캐시의 전체 상태와 작업 출력의 캐싱을 제공합니다. 캐시 실행 로그는 캐싱을 사용하는 각 실행의 캐시 적중 및 누락에 대한 가시성을 제공합니다.
- Outputs.json – WDL 및 CWL 워크플로의 경우 HealthOmics는 실행 완료 후 라는 엔진 생성 파일을 Amazon S3 버킷`outputs.json`에 전달합니다. 이 파일에는 실행을 위한 모든 출력의 목록과 맵이 포함되어 있습니다.

CloudWatch의 로그그

CloudWatch는 실패한 실행 및 성공적인 실행에 대한 워크플로 로그를 생성합니다. 엔진 로그는 실패한 실행에만 사용할 수 있다는 점을 제외하고 모든 로그는 실패한 실행 및 성공적인 실행에 사용할 수 있습니다.

CloudWatch 워크플로 로그는 로그 그룹에서 찾을 수 있습니다`/aws/omics/WorkflowLog`. 또한 get-run API 작업의 출력은 엔진 로그 및 실행 로그에 대한 CloudWatch 로그 스트림 ARNs을 제공합니다.

기본적으로는 CloudWatch Logs를 무기한 AWS 유지합니다. 로그 그룹의 보존 정책을 조정하여 보존 기간을 10년에서 1일로 설정할 수 있습니다.

다음 표에는 HealthOmics의 CloudWatch Logs에 대한 요약이 나와 있습니다. HealthOmics 모든 워크플로 로그는 성공한 실행과 실패한 실행에 사용할 수 있습니다. 단, 엔진 로그는 실패한 실행에만 사용할 수 있습니다.

로그 이름	CloudWatch Logs에서 사용 가능	에서 로그를 사용할 수 있는 경우	로그 스트림 형식
엔진 로그	예, 실패한 실행의 경우	실행이 완료된 후	run/*runID*/engine
매니페스트 로그 실행	예	실행이 완료된 후	매니페스트/실행/*runID*/*runUUID*
로그 실행	예	실시간	run/*runID*
작업 로그	예	실시간	run/*runID*/ task/*taskID*
캐시 로그 실행	예	실시간	runCache/ *runCacheId* /*runCacheUUID*
Outputs.json(WDL 및 CWL)	아니요	해당 사항 없음	해당 사항 없음

Amazon S3의 로그

엔진 로그와 `outputs.json` 파일만 Amazon S3로 전송됩니다.

실행이 완료되면 엔진 로그가 S3 버킷으로 전송되고 삭제할 때까지 무기한 사용할 수 있습니다. 이러한 로그는 워크플로에 지정한 S3 출력 URI의 로그 디렉터리에 있습니다.

로그 디렉터리의 경로 형식은 입니다`s3://{user_provided_path}/logs/`.

다음 표에는 Amazon S3 버킷에서 사용할 수 있는 HealthOmics 로그의 요약이 나와 있습니다.

로그 이름	Amazon S3에서 사용 가능	에서 로그를 사용할 수 있는 경우	로그 스트림 경로
엔진 로그	예	실행이 완료된 후	s3://*user_provided_path* /logs/engine.log
Outputs.json(WDL 및 CWL)	예	실행이 완료된 후	s3://*user_provided_path* /*runID*/*runUUID*/logs/outputs.json
매니페스트 로그, 실행 로그 및 작업 로그 실행	아니요	해당 사항 없음	해당 사항 없음

CLI의 대화형 CloudWatch Logs

대화형 모드에서 Live Tail 명령을 사용하여 CloudWatch Logs를 대화형으로 볼 수 있습니다. 실행 진행 상황을 실시간으로 추적하고 로그에서 강조 표시할 키워드를 최대 5개까지 정의할 수 있습니다.

```
aws logs start-live-tail  \
  --mode interactive  \
  --log-group-identifiers arn:aws:logs:region:account-ID:log-group:/aws/omics/
WorkflowLog
```

자세한 내용은 AWS CLI 명령 참조의 라이브 테일 시작을 참조하세요.

콘솔에서 CloudWatch Logs에 액세스

실행에 대한 로그에 액세스하려면 HealthOmics 콘솔의 실행 세부 정보 페이지에서 이러한 로그에 직접 연결할 수 있습니다.

1. HealthOmics 콘솔을 엽니다.
2. 필요한 경우 왼쪽 탐색 창(☰)을 엽니다. 실행을 선택합니다.
3. 실행 테이블에서 실행을 선택합니다.
4. 실행 세부 정보 페이지에서 다음 작업 중 하나를 선택할 수 있습니다.

a. 실행 요약에서 실행 로그 보기를 선택합니다. 콘솔은 CloudWatch 콘솔에서 실행 로그를 엽니다.
b. 요약 실행에서 Amazon S3에서 로그 보기를 선택합니다. 콘솔은 Amazon S3 콘솔에서 로그 폴더를 엽니다.
c. 작업 실행에서 작업에 대한 로그 보기, 실행 로그 보기 또는 실행 매니페스트 로그 보기를 선택합니다. 콘솔은 CloudWatch 콘솔에서 로그를 엽니다.

CloudWatch 콘솔에서 로그로 이동할 수도 있습니다.

1. CloudWatch 콘솔 https://console.aws.amazon.com/cloudwatch/ 엽니다.
2. 왼쪽 메뉴에서 로그 그룹을 선택합니다.
3. `/aws/omics/WorkflowLog` 그룹을 선택합니다.

   로그 그룹 목록이 긴 경우 검색 텍스트 상자에 omics를 입력하여 목록의 범위를 좁힐 수 있습니다.
4. 로그 그룹 세부 정보 페이지가 열리면 보려는 로그 스트림을 선택합니다. 콘솔에이 로그 스트림에 대한 이벤트가 표시됩니다.

를 사용하여 AWS HealthOmics API 호출 로깅 AWS CloudTrail

AWS HealthOmics 는 HealthOmics에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. HealthOmics CloudTrail은 HealthOmics에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 HealthOmics 콘솔의 호출과 HealthOmics API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 HealthOmics 이벤트를 포함한 CloudTrail 이벤트를 지속적으로 Amazon S3 버킷에 배포할 수 있습니다. 트레일을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 HealthOmics에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 HealthOmics 정보

CloudTrail은 계정을 생성할 AWS 계정 때에서 활성화됩니다. HealthOmics에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. 에서 최근 이벤트를 보고 검색하고 다운로드할 수 있습니다 AWS 계정. 자세한 내용은 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

HealthOmics 이벤트를 AWS 계정포함하여에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 HealthOmics 작업은 CloudTrail에서 로깅되며 AWS HealthOmics API 참조에 문서화됩니다. 예를 들어 `CreateReferenceStore`, `StartVariantImportJob`, `CreateWorkflow` 작업을 직접 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 대한 정보가 포함됩니다. 자격 증명을 이용하면 다음을 쉽게 판단할 수 있습니다.

- IAM 사용자 자격 증명으로 요청이 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에서 이루어졌는지 여부입니다.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

HealthOmics 로그 파일 항목 이해

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음 예제에서는 CreateWorkflow 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다.

```
{
    "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"
}
```

에서 EventBridge 사용 AWS HealthOmics

HealthOmics는 리소스 상태가 변경될 때 Amazon EventBridge로 이벤트를 전송합니다. 리소스에는 가져오기 작업, 내보내기 작업, 리소스 공유, 워크플로, 작업 및 실행이 포함됩니다. 각 리소스 유형에는 이벤트를 생성하는 상태 변경 목록이 있습니다.

이벤트 버스는 이벤트를 수신하여 대상으로 전송하는 라우터입니다. 계정에는 AWS 서비스에서 이벤트를 자동으로 수신하는 기본 이벤트 버스가 포함되어 있습니다. 추가 사용자 지정 이벤트 버스를 생성할 수 있습니다.

이벤트 버스가 이벤트를 수신할 때 수행할 작업을 지정하는 EventBridge 규칙을 생성합니다. 예를 들어 리소스의 상태 변경에 대해 알리는 규칙을 생성할 수 있습니다.

이벤트 사용에 대한 일반적인 시나리오는 다음과 같습니다.

- 사용자가 리소스를 공유하거나 공유를 취소하는 시기를 모니터링합니다.
- 실행이 실패하는지 또는 성공적으로 완료되었는지 모니터링합니다.

EventBridge 사용에 대한 자세한 내용은 Amazon EventBridge란 무엇입니까?를 참조하세요.

주제

HealthOmics용 EventBridge 설정

EventBridge 이벤트를 모니터링하려면 먼저 EventBridge 버스를 생성하고 관심 이벤트에 대한 규칙을 생성합니다.

EventBridge 버스 구성

에 기본 이벤트 버스를 사용하거나 사용자 지정 이벤트 버스를 AWS 계정 구성할 수 있습니다. 사용자 지정 이벤트 버스를 구성하려면 다음 단계를 따릅니다.

1. EventBridge 콘솔: https://console.aws.amazon.com/events/ 엽니다.
2. 왼쪽 탐색 창에서 이벤트 버스를 선택합니다.
3. Create event bus(이벤트 버스 생성)를 선택하세요.
4. 이벤트 버스 생성 양식에 버스의 이름을 입력합니다.
5. 생성을 선택하여 버스를 생성합니다.

EventBridge 규칙 생성

다음 절차에서는 간단한 규칙을 생성하는 방법을 보여줍니다. 규칙에 대한 자세한 내용은 EventBridge의 규칙을 참조하세요.

1. EventBridge 콘솔: https://console.aws.amazon.com/events/ 엽니다.
2. 왼쪽 탐색 창에서 규칙을 선택합니다.
3. 규칙 생성을 선택합니다. 콘솔에서 규칙 생성 양식이 열립니다.
4. 규칙 세부 정보 정의에서 규칙의 이름을 입력합니다.
   - 이름에 버스의 이름을 입력합니다.
   - 이벤트 버스에서이 규칙의 버스를 선택합니다.
   - 다음을 선택합니다.
5. 빌드 이벤트 패턴의 이벤트 소스에서 AWS 이벤트 또는 EventBridge 파트너 이벤트를 선택합니다.
6. 이벤트 패턴까지 아래로 스크롤합니다.
   a. 이벤트 소스에서 AWS 서비스를 선택합니다.
   b. AWS 서비스의 경우 텍스트 필터에 omics를 입력하고 서비스로 AWS HealthOmics를 선택합니다.
   c. 이벤트 유형에서 관심 있는 이벤트(또는 모든 이벤트)를 선택합니다.
   d. 다음을 선택합니다.
7. 대상 선택(Select target)에서 이벤트의 대상을 선택합니다. 예를 들어 AWS 서비스, 선택한 CloudWatch 로그 그룹을 선택하고 로그 그룹을 구성합니다.

여러 대상 유형에 대해 EventBridge에서는 대상에 이벤트를 보낼 권한이 필요합니다. 콘솔에서 이러한 권한을 생성합니다.

8. (선택 사항) 태그 구성에서 태그를 규칙과 연결합니다.
9. 검토 및 업데이트에서 구성을 검토하고 규칙 생성을 선택합니다.

HealthOmics의 EventBridge 이벤트

다음 표에는 HealthOmics가 EventBridge로 보내는 이벤트와 이벤트의 가능한 상태 값 목록이 나와 있습니다.

이벤트 이름	가능한 상태 값
주석 가져오기 작업 상태 변경	제출됨, 진행 중, 취소됨, 완료됨, 실패함 또는 실패와 함께 완료됨
주석 저장소 공유 상태 변경	보류 중, 활성화, 활성, 삭제, 삭제, 실패
주석 저장소 상태 변경	생성, 생성, 업데이트, 업데이트, 삭제, 삭제 또는 생성 실패
읽기 세트 활성화 작업 상태 변경	제출됨, 진행 중, 완료됨, 실패함 또는 실패와 함께 완료됨
읽기 세트 내보내기 작업 상태 변경	제출됨, 진행 중, 완료됨, 실패함 또는 실패와 함께 완료됨
읽기 세트 가져오기 작업 상태 변경	제출됨, 진행 중, 완료됨, 실패함 또는 실패와 함께 완료됨
읽기 세트 상태 변경	업로드, 업로드 실패, 활성, 아카이브, 활성화 또는 삭제 처리
참조 가져오기 작업 상태 변경	제출됨, 진행 중, 완료됨, 실패함 또는 실패와 함께 완료됨
참조 상태 변경	활성 또는 삭제됨
참조 스토어 상태 변경	생성, 업데이트, 활성 또는 삭제됨

이벤트 이름	가능한 상태 값
실행 상태 변경	보류 중, 시작 중, 실행 중, 중지 중, 완료됨, 삭제됨, 실패 또는 취소됨
시퀀스 스토어 상태 변경	생성, 업데이트, 활성 또는 삭제됨
작업 상태 변경	보류 중, 시작 중, 실행 중, 중지 중, 완료됨, 삭제됨, 실패 또는 취소됨
변형 가져오기 작업 상태 변경	제출됨, 진행 중, 취소됨, 완료됨, 실패함 또는 실패와 함께 완료됨
변형 스토어 공유 상태 변경	보류 중, 활성화, 활성, 삭제, 삭제, 실패
변형 저장소 상태 변경	생성, 생성, 업데이트, 업데이트, 삭제, 삭제 또는 생성 실패
워크플로 공유 상태 변경	보류 중, 활성화, 활성, 삭제, 삭제, 실패
워크플로 상태 변경	생성 성공, 생성 실패, 삭제 성공 또는 삭제 실패

이벤트 메시지 구조

HealthOmics는 상태 변경 이벤트 메시지를 EventBridge로 전송하기 위한 최선의 노력을 제공합니다. 이벤트는 메타데이터 세부 정보도 포함하는 JSON 구조의 객체입니다. 메타데이터를 입력으로 사용하여 이벤트를 다시 생성하거나 자세한 내용을 알아볼 수 있습니다. 이벤트에는 다음 필드가 포함됩니다.

- `version` - 현재 모든 이벤트에 대해 0(0)입니다.
- `id` - 모든 이벤트에 대해 생성된 버전 4 UUID입니다.
- `detail-type` - 전송 중인 이벤트의 유형입니다.
- `account` - 버킷 소유자의 12자리 AWS 계정 ID입니다.
- `source` - 이벤트를 생성한 서비스를 식별합니다.
- `time` - 이벤트가 발생한 시간입니다.
- `region` - 버킷 AWS 리전 의를 식별합니다.
- `resources` - 버킷의 Amazon 리소스 이름(ARN)을 포함하는 JSON 배열입니다.
- `detail` - 이벤트에 대한 정보가 포함된 JSON 객체입니다.

실행 이벤트에는 다음 필드가 포함됩니다.

- uuid - 실행의 범용 고유 식별자입니다.
- workflowId -이 실행과 연결된 워크플로의 워크플로 식별자입니다.
- workflowName -이 실행과 연결된 워크플로의 이름입니다.
- runId - 실행 식별자입니다.
- runName - 실행 이름입니다.
- runOutputUri - 실행이 출력 데이터를 쓸 URI입니다.

이벤트 메시지 예제

다음 예제는 추가 필드를 보여주는 실행 상태 변경 이벤트입니다.

```
{
    "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"
    }
}
```

다음 예제는 작업 상태 변경에 대한 이벤트입니다.

```
{
```

```
    "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"
    }
}
```


다음은 읽기 세트 상태 변경 이벤트의 예입니다.


```
{
  "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"
  }
}
```

변형 저장소 가져오기 작업에 대해 유사한 이벤트가 생성됩니다.

```
{
    "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"
    }
}
```

다음은 가져오기 작업 상태 변경에 대한 이벤트입니다.

```
{
    "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"
    }
}
```

문제 해결

다음 주제는 HealthOmics 워크플로 및 데이터 스토어를 사용할 때 발생하는 문제를 해결하는 데 도움이 될 수 있습니다.

주제

워크플로 문제 해결

주제

실패한 실행 문제를 해결하려면 어떻게 해야 합니까?

GetRun API 작업을 사용하여 실패 이유를 검색합니다. 자세한 내용은 실행 실패 이유 단원을 참조하십시오.

실패한 작업의 문제를 해결하려면 어떻게 해야 합니까?

작업 실패 메시지의 오류 코드를 검토하여 실패를 이해합니다. CloudWatch의 작업 로그를 검토하여 작업에 대한 자세한 로깅 메시지를 확인합니다. 자세한 로그 메시지가 표시되지 않는 경우 워크플로를 수정하여 추가 로그 문을 출력할 수 있습니다. 자세한 내용은 CloudWatch Logs를 사용하여 HealthOmics 모니터링 단원을 참조하십시오.

성공적으로 완료된 실행에 대한 엔진 로그는 어디에서 찾을 수 있습니까?

HealthOmics는 실패한 실행에 대해서만 CloudWatch에 로그를 게시합니다. 실행이 성공적으로 완료되면 HealthOmics는 엔진 로그를 Amazon S3 버킷에 전송합니다. 자세한 내용은 Amazon S3의 로그 단원을 참조하십시오.

워크플로의 입력 파라미터 크기를 줄이려면 어떻게 해야 하나요?

워크플로에 대해 최대 50KB의 입력 파라미터를 지정할 수 있습니다. 디렉터리 가져오기 또는 샘플 시트를 사용하여이 크기 제약 조건을 유지할 수 있습니다. 자세한 내용은 실행 파라미터 크기 관리 단원을 참조하십시오.

실행이 완료되지 않는 이유는 무엇인가요?

코드에 문제가 있고 프로세스가 제대로 종료되지 않은 경우 실행이 응답하지 않거나 “멈출” 수 있습니다. 응답하지 않는 실행을 방지하고 포착하는 방법에 대한 자세한 내용은 섹션을 참조하세요응답하지 않는 실행에 대한 지침.

통화 캐싱 문제 해결

다음 주제는 통화 캐싱에서 발생하는 문제를 해결하는 데 도움이 될 수 있습니다.

주제

- 실행이 캐시에 저장되지 않는 이유는 무엇인가요?
- 작업이 캐시 항목을 사용하지 않는 이유는 무엇입니까?
- 작업에 대한 호출 캐싱이 비활성화된 이유는 무엇입니까?

실행이 캐시에 저장되지 않는 이유는 무엇인가요?

1. GetRun API 작업 응답에서 cacheId 필드를 확인하여 실행이 캐시를 사용하도록 구성되어 있는지 확인합니다. CLI를 사용하여 명령을 실행합니다`aws omics get-run –id <run_id>`.
2. 실행에 성공한 경우 GetRun 응답에 반환된 캐시 동작이 CACHE_ALWAYS인지 확인합니다. 캐시 동작이 CACHE_ON_FAILURE로 설정된 경우 실행은 실패할 때만 캐시에 저장됩니다.

작업이 캐시 항목을 사용하지 않는 이유는 무엇입니까?

/aws/omics/WorkflowLog CloudWatch 로그 그룹에서 실행 캐시의 로그 스트림을 엽니다. runCache/<cache_id>/<cache_uuid>.

1. 이전 실행이 캐시될 것으로 예상한 작업에 대한 캐시 항목을 생성했는지 확인합니다. 캐시에 저장된 실행은 CACHE_ENTRY_CREATED 로그 메시지와 함께 기록됩니다.
2. 작업에 대한 CACHE_MISS 로그를 찾아 완료된 로그를 실행합니다. 로그 항목이 없는 경우 실행이 캐시를 사용하도록 구성되었는지 확인합니다.
3. 캐시 항목이 생성된 경우 CPU, 메모리 CPUs, GPUs 및 컨테이너 다이제스트가 두 태스크에 대해 동일한지 확인합니다. 캐시 항목을 생성한 태스크의 태스크 ARN은 로그 메시지에 있습니다.
4. 두 작업의 컴퓨팅 요구 사항이 일치하는 경우 작업 간에 입력이 변경되지 않았는지 확인합니다. 이렇게 하려면 엔진 로그를 엽니다. 실행 상태가 FAILED인 경우 로그는 Cloudwatch Log Group /aws/omics/WorkflowLog에 있습니다. 그렇지 않으면 실행의 출력 디렉터리에서 엔진 로그를 찾을 수 있습니다.

작업에 대한 호출 캐싱이 비활성화된 이유는 무엇입니까?

워크플로 엔진 기능을 사용하여 캐싱을 옵트아웃하도록 작업이 구성되어 있는지 확인합니다.

- WDL 워크플로의 경우: 메타 섹션에서 작업에 휘발성이 로 설정되어 있는지 확인합니다`true`.
- Nextflow 워크플로의 경우: 태스크에 캐시 명령이 로 설정되어 있는지 확인합니다. `false`
- CWL 워크플로의 경우: WorkReuse 기능에 대해 작업의 enableReuse가 로 설정되어 있는지 확인합니다. `false`

데이터 스토어 문제 해결

주제

- 읽기 세트에서 S3 GetObject가 실패하는 이유는 무엇인가요?
- Athena에서 주석 저장소 또는 변형 저장소를 볼 수 없는 이유는 무엇인가요?
- Athena의 데이터 스토어에 액세스할 수 없는 이유는 무엇인가요?

읽기 세트에서 S3 GetObject가 실패하는 이유는 무엇인가요?

가장 일반적으로 실패는 권한 누락으로 인한 것입니다. 시퀀스 스토어 S3 읽기 권한은 시퀀스 스토어 S3 액세스 정책이 액세스를 허용하고 IAM 보안 주체가 액세스를 허용하는 정책을 연결해야 하는 양방향 구성입니다. 정책 요구 사항에 대한 자세한 내용은 섹션을 참조하세요Amazon S3 URIs를 사용한 데이터 액세스 권한. 다음 구성이 있는지 확인합니다.

- 시퀀스 스토어 S3 액세스 정책은 IAM 보안 주체 또는 보안 주체 계정의 루트에 대한 액세스를 명시적으로 허용했습니다.
- IAM 보안 주체에 액세스 중인 리소스에 대한 권한을 명시적으로 제공하는 정책이 있는지 확인합니다. IAM 보안 주체 정책은 권한을 정의할 때 액세스 포인트 별칭 기반 경로가 아닌 액세스 포인트 ARN을 사용해야 하며 ARN은 조건에 있고 리소스를 지정하는 데 사용되지 않습니다.
- 스토어에서 고객 관리형 키(CMK-KMS)를 사용하는 경우 IAM 보안 주체에 키에 대한 kms:decrypt 권한이 있는지 확인합니다. 계정 간 사용을 구성하려면 KMS 교차 계정 액세스 가이드를 참조하세요.

태그 기반 액세스 제어를 사용하는 정책이 있는 경우 다음을 확인하세요.

- 시퀀스 스토어가 태그 동기화를 완료했는지 확인합니다. 이렇게 하려면 스토어의 상태가 active가 아닌 여야 합니다updating.
- 읽기 세트 및 정책의 태그 키 또는 키 값에 오타가 없는지 확인합니다.

Athena에서 주석 저장소 또는 변형 저장소를 볼 수 없는 이유는 무엇인가요?

Lake Formation에서 공유된 스토어를 기반으로 리소스 링크를 생성해야 합니다. 액세스 권한이 있는 리소스 링크를 생성하면 Athena에 스토어가 표시됩니다. 자세한 내용은 HealthOmics를 사용하도록 Lake Formation 구성 단원을 참조하십시오.

Athena의 데이터 스토어에 액세스할 수 없는 이유는 무엇인가요?

주석 또는 변형 저장소가 표시되지만 액세스가 거부되었다는 오류 메시지가 표시되는 경우 사용 중인 쿼리 엔진 버전을 확인합니다. 엔진 버전 3을 사용하여 실행되는 쿼리만 지원됩니다. Athena 쿼리 엔진 버전에 대한 자세한 내용은 Amazon Athena 설명서를 참조하세요.

Amazon Q CLI를 사용한 문제 해결

Amazon Q CLI는 다음을 통해 문제 해결 프로세스를 간소화하는 데 도움이 될 수 있습니다.

- 워크플로 실행 분석 및 작업 실패 디버깅
- 관련 로그 및 오류 메시지 수집
- 필요한 모든 디버깅 로그가 연결된 AWS 지원 사례 생성
- AWS Support에 제출된 정보에서 개인 식별 정보(PII)를 수정합니다.

문제 해결 및 지원 사례 생성을 AWS HealthOmics 위해에서 Amazon Q CLI를 사용하는 방법에 대한 자세한 내용은 GitHub의 HealthOmics Agentic 생성형 AI 자습서를 참조하세요.

> **Warning**
>
> Amazon Q CLI로 작업할 때는 계속하기 전에 생성된 모든 콘텐츠와 제안된 작업을 검토합니다. 응답 품질을 개선하고 워크플로의 요구 사항에 맞는 피드백을 제공합니다. 자세한 내용은 Amazon Q의 보안 고려 사항 및 모범 사례를 참조하세요.

에 대한 할당량 AWS HealthOmics

AWS 는 HealthOmics 할당량의 기본값으로 계정을 채웁니다. 달리 명시되지 않는 한 각 할당량 값은 리전별 최대값입니다.

> **Important**
>
> 대부분의 서비스 할당량 및 API 할당량에 대한 증가를 요청할 수 있습니다. 자세한 정보는 이하의 주제를 참조하세요.

주제

HealthOmics 서비스 할당량

아래 표에는 HealthOmics 서비스 할당량과 기본값이 나와 있습니다. 각 리전의 현재 할당량을 보려면 Service Quotas 콘솔을 엽니다.

> **Important**
>
> Service Quotas 콘솔을 사용하여 조정 가능한 할당량 증가를 요청할 수 있습니다.

서비스 할당량에 대한 자세한 내용은 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요. Service Quotas 콘솔에서 사용할 수 없는 할당량의 경우 할당량 증가 양식을 사용합니다.

이름	기본값	조정 가능	설명
분석 - 최대 주석 저장소	지원되는 각 리전: 10개	예	현재 AWS 리전의 최대 주석 저장소 수

이름	기본값	조정 가능	설명
분석 - 최대 동시 변형 또는 주석 저장소 가져오기 작업	지원되는 각 리전: 5개	예	현재 AWS 리전의 최대 동시 가져오기 작업 수
분석 - 변형 저장소 가져오기 작업당 최대 파일	지원되는 각 리전: 1,000	예	현재 AWS 리전의 변형 가져오기 작업당 최대 파일 수
분석 - 주석 저장소당 최대 공유	지원되는 각 리전: 10개	예	현재 AWS 리전의 주석 저장소당 최대 공유 수
분석 - 변형 저장소당 최대 공유	지원되는 각 리전: 10개	예	현재 AWS 리전의 변형 저장소당 최대 공유 수
분석 - 변형 가져오기 작업에서 각 파일의 최대 크기	지원되는 각 리전: 20기가바이트	예	현재 AWS 리전의 변형 가져오기 작업에 있는 파일 하나의 최대 크기
분석 - 주석 가져오기 작업에서 각 파일의 최대 크기	지원되는 각 리전: 20기가바이트	예	현재 AWS 리전의 주석 가져오기 작업에 있는 파일 하나의 최대 크기
분석 - 최대 변형 저장소	지원되는 각 리전: 10개	예	현재 AWS 리전의 최대 변형 저장소 수
분석 - 주석 저장소당 최대 버전	지원되는 각 리전: 10개	예	현재 AWS 리전의 주석 저장소당 최대 버전 수
구성 - 최대 구성	지원되는 각 리전: 10개	예	현재 AWS 리전의 최대 구성 수입니다.
스토리지 - 최대 동시 읽기 세트 활성화 작업	지원되는 각 지역: 25	예	현재 AWS 리전의 최대 동시 읽기 세트 활성화 작업 수

이름	기본값	조정 가능	설명
스토리지 - 최대 동시 시퀀스 및 참조 저장소 내보내기 작업	지원되는 각 리전: 5개	예	현재 AWS 리전의 시퀀스 또는 참조 스토어에서 동시 내보내기 작업의 최대 수
스토리지 - 최대 동시 시퀀스 또는 참조 저장소 가져오기 작업	지원되는 각 리전: 5개	예	현재 AWS 리전의 시퀀스 또는 참조 스토어에 대한 최대 동시 가져오기 작업 수
스토리지 - 시퀀스 저장소당 최대 읽기 세트	지원되는 각 리전: 1,000,000	예	현재 AWS 리전의 시퀀스 스토어에 있는 최대 읽기 세트 수
스토리지 - 참조 저장소당 최대 참조	지원되는 각 지역: 50	예	현재 AWS 리전의 참조 저장소에 있는 참조의 최대 수
스토리지 - 최대 시퀀스 저장소	지원되는 각 리전: 20개	예	현재 AWS 리전의 최대 시퀀스 저장소 수
워크플로 - 최대 활성 GPU	지원되는 각 리전: 12	예	현재 AWS 리전의 최대 동시 활성 GPUs. us-east-1 및 us-west-2에서는 최대 500까지의 값에 대한 할당량 증가 요청이 자동으로 승인됩니다.

이름	기본값	조정 가능	설명
워크플로 - 동적 실행 스토리지를 사용하는 최대 동시 활성 실행	지원되는 각 지역: 50	예	현재 AWS 리전에서 동적 실행 스토리지를 사용하는 최대 활성 실행 수입니다. 최대 200까지의 값에 대한 할당량 증가 요청이 자동으로 승인됩니다.
워크플로 - 정적 실행 스토리지를 사용하는 최대 동시 활성 실행	지원되는 각 리전: 10개	예	현재 AWS 리전에서 정적 실행 스토리지를 사용하는 최대 활성 실행 수입니다. 최대 50까지의 값에 대한 할당량 증가 요청이 자동으로 승인됩니다.
워크플로 - 실행당 최대 동시 작업	지원되는 각 지역: 25	예	현재 AWS 리전에서 실행되는 각 실행의 최대 동시 작업 수입니다. us-east-1 및 us-west-2에서는 최대 100까지의 값에 대한 할당량 증가 요청이 자동으로 승인됩니다.
워크플로 - 최대 실행 기간	지원되는 각 리전: 604,800초	예	현재 AWS 리전의 최대 워크플로 실행 기간입니다.
워크플로 - 최대 실행(활성 또는 비활성)	지원되는 각 리전: 100,000	예	현재 AWS 리전의 최대 실행 수(활성 또는 비활성)입니다.
워크플로 - 워크플로당 최대 공유	지원되는 각 리전: 100	예	현재 AWS 리전의 워크플로당 최대 공유 수

이름	기본값	조정 가능	설명
워크플로 - 실행당 최대 정적 실행 스토리지 용량	지원되는 각 리전: 9,600	예	현재 AWS 리전의 각 실행에 대한 기비바이트(GiB) 단위의 최대 정적 실행 스토리지 용량입니다. us-east-1 및 us-west-2에서는 최대 50,000까지의 값에 대한 할당량 증가 요청이 자동으로 승인됩니다.
워크플로 - 최대 워크플로	지원되는 각 리전: 1,000	예	현재 AWS 리전의 최대 워크플로 수입니다.
워크플로 - StartRun 작업의 초당 트랜잭션 수(TPS)	지원되는 각 리전: 5개	예	현재 AWS 리전의 StartRun 작업에 대한 초당 최대 트랜잭션 수(TPS)입니다.

HealthOmics 고정 크기 할당량

HealthOmics 서비스 할당량HealthOmics에는 외에도 크기가 고정된 할당량이 포함되어 있습니다. 이러한 값에 대해서는 증가를 요청할 수 없습니다.

달리 명시되지 않는 한, 각 할당량에는 리전당 최대값이 나열됩니다.

주제

- HealthOmics 분석 고정 크기 할당량
- HealthOmics 스토리지 고정 크기 할당량
- HealthOmics 워크플로 고정 크기 할당량
- HealthOmics Ready2Run 워크플로 고정 크기 할당량

HealthOmics 분석 고정 크기 할당량

다음 표에는 분석 할당량에 지원되는 최대 값이 나와 있습니다. 이러한 값은 조정할 수 없습니다.

이름	설명	최대	조정 가능 예/아니요
분석 - 주석 저장소 가져오기 작업당 최대 파일 수	주석 가져오기 작업당 최대 파일 수입니다.	1	아니요

HealthOmics 스토리지 고정 크기 할당량

다음 표에는 스토리지 파일에 지원되는 최대 값이 나와 있습니다. 이러한 값은 조정할 수 없습니다.

이름	설명	최대	조정 가능 예/아니요
스토리지 - 최대 S3 액세스 리소스 정책 크기	S3 액세스 리소스 정책의 최대 크기	15KB	아니요
스토리지 - 전파된 최대 설정 수준 태그	S3 객체에 전파되는 저장소당 설정 수준 태그 키의 최대 수	5	아니요
스토리지 - 활성화 작업당 최대 읽기 세트 수	활성화 작업당 최대 읽기 세트 수입니다.	20	아니요
스토리지 - 내보내기 작업당 최대 읽기 세트 수	내보내기 작업당 최대 읽기 세트 수입니다.	100	아니요
스토리지 - 가져오기 작업당 최대 읽기 세트 수	가져오기 작업당 최대 읽기 세트 수입니다.	100	아니요
스토리지 - 최대 참조 저장소	참조 스토어의 최대 수입니다.	1	아니요

이름	설명	최대	조정 가능 예/아니요
스토리지 - 직접 업로드를 위한 최대 파트 크기	시퀀스 스토어에 직접 업로드할 수 있는 최대 파트 크기입니다.	100MB	아니요
스토리지 - 직접 업로드를 위한 파일의 최대 부분	시퀀스 스토어에 직접 업로드하기 위한 파일의 최대 파트 수입니다.	10,000	아니요
스토리지 - 최대 참조 크기	참조 스토어로 가져올 수 있는 참조 파일의 최대 크기입니다.	15GB	아니요
스토리지 - 최대 읽기 세트 소스 크기	시퀀스 저장소로 가져올 수 있는 읽기 세트에 있는 단일 소스 파일의 최대 크기입니다.	976GB	아니요

HealthOmics 워크플로 고정 크기 할당량

다음 표에는 워크플로 할당량에 지원되는 최대 값이 나와 있습니다. 이러한 값은 조정할 수 없습니다.

이름	설명	최대 크기	조정 가능 예/아니요
워크플로 - 최대 실행 그룹	최대 실행 그룹 수입니다.	1000	아니요
워크플로 - 최대 실행 캐시	한 계정에 대해 생성할 수 있는 실행 캐시의 최대 수입니다. 하나 이상의 실행이 동일한 실행 캐시를 공유할 수 있습니다. HealthOmics가 계정당 캐싱할 수 있는 실	1000	아니요

이름	설명	최대 크기	조정 가능 예/아니요
	행 수에는 할당량이 없습니다.		
워크플로 - 최대 워크플로 버전	워크플로당 최대 워크플로 버전 수입니다.	1000	아니요
워크플로 - CPU 인스턴스 컨테이너 크기	CPU 인스턴스의 최대 컨테이너 이미지 크기입니다.	45GiB	아니요
워크플로 - GPU 인스턴스 컨테이너 크기	GPU 인스턴스의 최대 컨테이너 이미지 크기입니다.	95GiB	아니요
GPU 인스턴스 /dev/shm 공유 메모리	GPU 인스턴스당 최대 공유 메모리 양입니다.	GPU당 8GB	아니요
워크플로 - 파라미터 파일 실행	실행 파라미터 파일의 최대 크기입니다.	50,000바이트	아니요
워크플로 - 워크플로 파라미터 템플릿 파일	워크플로 파라미터 템플릿 파일의 최대 항목 수 및 최대 파일 크기입니다. 이 할당량은 콘솔 또는 API를 사용하여 생성하는 워크플로에 적용됩니다.	항목 1,000개, 400KB	아니요
워크플로 - 워크플로 정의 파일 크기 - API	API 작업 또는 AWS SDK를 사용하여 워크플로를 생성할 때 워크플로 정의 파일의 최대 크기입니다.	100MB	아니요

이름	설명	최대 크기	조정 가능 예/아니요
워크플로 - 워크플로 정의 파일 크기 - 콘솔(직접 업로드)	콘솔을 사용하여 워크플로를 생성할 때 직접 업로드로 제공할 수 있는 워크플로 정의 파일의 최대 크기입니다.	4.4MB	아니요
워크플로 - 워크플로 정의 파일 크기 - 콘솔(Amazon S3에서 업로드)	콘솔을 사용하여 워크플로를 생성할 때 Amazon S3에서 업로드로 제공할 수 있는 워크플로 정의 파일의 최대 크기입니다.	100MB	아니요
워크플로 - 리포지토리 크기	외부 코드 리포지토리의 최대 크기입니다.	1GiB	아니요
워크플로 - 리포지토리 개별 파일 크기	외부 코드 리포지토리의 개별 파일의 최대 크기입니다.	100MiB	아니요
워크플로 - README 파일 크기	README 파일의 최대 크기입니다.	500KiB	아니요

실행 파라미터 파일의 크기를 줄이는 방법에 대한 제안은 섹션을 참조하세요실행 파라미터 크기 관리.

HealthOmics Ready2Run 워크플로 고정 크기 할당량

각 Ready2Run 워크플로에는 최대 입력 파일 크기가 있습니다. 다음 표에서 파일 크기 단위는 기비바이트(GiB)로 나열됩니다. 이러한 최대 파일 크기는 조정할 수 없습니다.

Ready2Run 워크플로 이름	최대 입력 파일 크기(GiB)	조정 가능(예/아니요)
AlphaFold - 601-1200개 구체화	1	아니요

Ready2Run 워크플로 이름	최대 입력 파일 크기(GiB)	조정 가능(예/아니요)
최대 600개의 시크릿에 대한 AlphaFold	1	아니요
2x150용 Bases2Fastq	1000	아니요
2x300용 Bases2Fastq	1000	아니요
2x75용 Bases2Fastq	500	아니요
최대 800개의 시크릿에 대한 ESMFold	1	아니요
GATK-BP fq2bam	64	아니요
30x 유전체용 GATK-BP Germline bam2vcf	39	아니요
30x 유전체용 GATK-BP Germline fq2vcf	64	아니요
GATK-BP 신체 WES bam2vcf	86	아니요
최대 30X의 NVIDIA Parabricks BAM2FQ2BAM WGS	80	아니요
최대 50X의 NVIDIA Parabricks BAM2FQ2BAM WGS	120	아니요
최대 5X의 NVIDIA Parabricks BAM2FQ2BAM WGS	20	아니요
최대 30X의 NVIDIA Parabricks FQ2BAM WGS	71	아니요
최대 50X의 NVIDIA Parabricks FQ2BAM WGS	137	아니요

Ready2Run 워크플로 이름	최대 입력 파일 크기(GiB)	조정 가능(예/아니요)
최대 5X의 NVIDIA Parabricks FQ2BAM WGS	13	아니요
최대 30X의 NVIDIA Parabricks Germline DeepVariant WGS	71	아니요
최대 50X의 NVIDIA Parabricks Germline DeepVariant WGS	137	아니요
최대 5X의 NVIDIA Parabricks Germline DeepVariant WGS	12	아니요
최대 30X의 NVIDIA Parabricks Germline HaplotypeCaller WGS	71	아니요
최대 50X의 NVIDIA Parabricks Germline HaplotypeCaller WGS	137	아니요
최대 5X의 NVIDIA Parabricks Germline HaplotypeCaller WGS	13	아니요
최대 50X의 NVIDIA Parabricks Somatic Mutect2 WGS	196	아니요
KallistoBUStools를 사용하는 scRNAseq	119	아니요
Salmon Alevin-fry를 사용한 scRNAseq	119	아니요
STARsolo를 사용하는 scRNAseq	119	아니요

Ready2Run 워크플로 이름	최대 입력 파일 크기(GiB)	조정 가능(예/아니요)
최대 300배의 Sentieon Germline BAM WES	9	아니요
최대 32배의 Sentieon Germline BAM WGS	18	아니요
최대 100배의 Sentieon Germline FASTQ WES	5	아니요
최대 300x용 Sentieon Germline FASTQ WES	26	아니요
최대 32배의 Sentieon Germline FASTQ WGS	51	아니요
ONT용 Sentieon LongRead	25	아니요
PacBio HiFi용 Sentieon LongRead	58	아니요
Sentieon 신체 WES	50	아니요
Sentieon 신체 WGS	113	아니요
최대 40배의 Ultima Genomics DeepVariant	91	아니요

HealthOmics API 할당량

HealthOmics에는 API 작업과 관련된 다음과 같은 할당량이 있습니다. 표시된 경우 할당량을 조정할 수 있습니다. 증가를 요청하려면 할당량 증가 양식을 사용합니다.

나열된 각 API 작업에 대해 할당량은 각 리전의 해당 API 작업에 대한 초당 최대 트랜잭션 수(TPS)입니다.

주제

- 일반 API 할당량

일반 API 할당량

다음 표에는 둘 이상의 범주(스토리지, 워크플로 및 분석)에 적용되는 일반적인 API 작업이 나열되어 있습니다.

API 작업	기본 최대 TPS	조정 가능(예/아니요)
AcceptShare, CreateShare, DeleteShare, GetShare, ListShares	1TPS	예

스토리지 API 할당량

다음 표에는 스토리지 API 작업이 나열되어 있습니다.

스토리지 API 작업	기본 최대 TPS	조정 가능(예/아니요)
CreateSequenceStore, UpdateSequenceStore, DeleteSequenceStore, CreateReferenceStore, DeleteReferenceStore	1TPS	예
BatchDeleteReadSet, DeleteReference	1TPS	예
CreateMultipartReadSetUpload, CompleteMultipartReadSetUpload, AbortMultipartReadSetUpload	1TPS	아니요

스토리지 API 작업	기본 최대 TPS	조정 가능(예/아니요)
GetS3AccessPolicy, PutS3AccessPolicy, DeleteS3AccessPolicy	1TPS	예
GetReference	10TPS	예
UploadReadSetPart	10TPS	예
GetReadSet	30TPS	예
GetSequenceStore, ListSequenceStores	5TPS	예
GetReadSetMetadata, ListReadSets	5TPS	예
StartReadSetImportJob, GetReadSetImportJob, ListReadSetImportJobs	5TPS	예
StartReadSetExportJob, GetReadSetExportJob, ListReadSetExportJobs	5TPS	예
ListReferenceStores	5TPS	예
StartReferencetImportJob, GetReferenceImportJob, ListReferenceImportJobs	5TPS	예
ListReferences, GetReferenceMetadata	5TPS	예
StartReadsetActivationJob	5TPS	예
ListReadsetActivationJobs, GetReadSetActivationJob	5TPS	예

스토리지 API 작업	기본 최대 TPS	조정 가능(예/아니요)
ListMultipartReadSetUploads, ListReadSetUploadParts	5TPS	예
TagResource, UntagResource, ListTagsForResource	5TPS	예

워크플로 API 할당량

다음 표에는 워크플로 API 작업이 나열되어 있습니다.

워크플로 API 작업	기본 최대 TPS	조정 가능(예/아니요)
StartRun	1TPS	예
CreateWorkflow	5TPS	예
CancelRun, DeleteRun, GetRun, GetRunTask, ListRunTasks, ListRuns	10TPS	예
CreateRunGroup, DeleteRunGroup, GetRunGroup, ListRunGroups, UpdateRunGroup	10TPS	예
CreateRunCache, UpdateRunCache, DeleteRunCache, GetRunCache, ListRunCaches	10TPS	예
DeleteWorkflow, GetWorkflow, ListWorkflows, UpdateWorkflow	10TPS	예

분석 API 할당량

다음 표에는 분석 API 작업이 나열되어 있습니다.

분석 API 작업	기본 최대 TPS	조정 가능(예/아니요)
CreateVariantStore, DeleteVariantStore, GetVariantStore, ListVariantStores, UpdateVariantStore	1TPS	아니요
StartVariantImportJob, CancelVariantImportJob, GetVariantImportJob, ListVariantImportJobs	1TPS	아니요
CreateAnnotationStore, DeleteAnnotationStore, GetAnnotationStore, ListAnnotationStores, UpdateAnnotationStore	1TPS	아니요
StartAnnotationImportJob, ListAnnotationImportJobs, GetAnnotationImportJob, CancelAnnotationImportJob	1TPS	아니요

HealthOmics 사용 설명서의 문서 기록

다음 표에서는 HealthOmics의 설명서 릴리스를 설명합니다.

변경 사항	설명	날짜
AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다.	AWS HealthOmics 변형 저장소 및 주석 저장소는 더 이상 신규 고객에게 공개되지 않습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경을 참조하세요.	2025년 11월 7일
AWS HealthOmics 변형 저장소 및 주석 저장소는 2025년 11월 7일부터 신규 고객에게 더 이상 공개되지 않습니다.	AWS HealthOmics 변형 저장소 및 주석 저장소는 2025년 11월 7일부터 신규 고객에게 더 이상 공개되지 않습니다. 변형 저장소 또는 주석 저장소를 사용하려면 해당 날짜 이전에 가입하세요. 기존 고객은 정상적으로 서비스를 계속 이용할 수 있습니다. 자세한 내용은 AWS HealthOmics 변형 저장소 및 주석 저장소 가용성 변경을 참조하세요.	2025년 10월 7일
새로운 기능	HealthOmics는 프라이빗 Amazon ECR 리포지토리를 업스트림 레지스트리와 동기화하는 워크플로에 대한 지원을 추가했습니다. 자세한 내용은 HealthOmics.	2025년 8월 28일
새로운 README 및 리포지토리 통합 기능	외부 코드 리포지토리 및 README 파일에서 워크플로	2025년 7월 24일

	생성에 대한 지원이 추가되었습니다.	
새로운 기능	HealthOmics는 Nextflow 자동 파라미터 보간에 대한 지원을 추가했습니다. 자세한 내용은 HealthOmics 워크플로용 파라미터 템플릿 파일을 참조하세요.	2025년 6월 27일
새로운 기능	HealthOmics는 WDL 워크플로 정의 파일에서 실행 파라미터를 보간하는 워크플로에 대한 지원을 추가했습니다. 자세한 내용은 HealthOmics 워크플로용 파라미터 템플릿 파일을 참조하세요.	2025년 5월 30일
새로운 기능	HealthOmics는 워크플로 버전 관리에 대한 지원을 추가했습니다. 자세한 내용은 HealthOmics의 워크플로 버전 관리를 참조하세요.	2025년 4월 18일
새로운 기능	HealthOmics는 동적 실행 스토리지에 탄력적 처리량을 추가했습니다. 자세한 내용은 HealthOmics.	2025년 4월 16일
새로운 기능	HealthOmics는 Sequence Store S3 위치에 대한 속성 기반 액세스 제어와 최대 5개의 읽기 세트 태그를 Sequence Store S3 객체에 동기화할 수 있는 기능을 추가했습니다. 자세한 내용은 HealthOmics 시퀀스 스토어 생성을 참조하세요.	2024년 11월 22일

새로운 기능	HealthOmics는 프라이빗 워크플로에 대해 재개라고도 하는 통화 캐싱에 대한 지원을 추가했습니다. 자세한 내용은 캐싱 호출을 참조하세요.	2024년 11월 20일
새로운 기능	HealthOmics는 시퀀스 스토어 입력 작업과 읽기 세트 간에 매핑하는 데 도움이 되는 새 API 필드를 추가했습니다.	2024년 8월 29일
새로운 기능	HealthOmics는 Nextflow 버전 관리에 대한 지원을 추가했습니다. 자세한 내용은 Nextflow 버전을 참조하세요.	2024년 8월 14일
새로운 기능	HealthOmics는 공유 워크플로 및 동적 실행 스토리지에 대한 지원을 추가했습니다.	2024년 4월 30일
새로운 기능	HealthOmics는 참조 및 시퀀스 저장소에 대한 Amazon S3 액세스에 대한 지원과 SHA256 ETags.	2024년 4월 15일
새로운 기능	HealthOmics 시퀀스 스토어에 대한 개체 태그(ETags)를 추가했습니다.	2023년 10월 6일
새로운 기능	HealthOmics는 주석 저장소 버전 관리 및 분석 저장소 공유를 추가했습니다.	2023년 8월 15일
새로운 기능	HealthOmics는 HealthOmics 워크플로에 지원되는 언어로 공통 워크플로 언어(CWL)를 추가했습니다.	2023년 6월 30일

새로운 기능	HealthOmics는 새로운 Ready2Run 워크플로, 워크플로에 대한 GPU 지원, 주석 저장소에 대한 데이터 구문 분석, HealthOmics 스토리지에 직접 업로드, EventBridge와의 통합을 추가했습니다.	2023년 5월 15일
새 관리형 정책	HealthOmics는 전체 액세스를 제공하는 새로운 관리형 정책을 추가했습니다. 자세한 내용은 AWS 관리형 정책을 참조하세요.	2023년 2월 23일
새 관리형 정책	HealthOmics는 읽기 전용으로 액세스를 제한하는 새로운 관리형 정책을 추가했습니다. 자세한 내용은 AWS 관리형 정책을 참조하세요.	2022년 11월 29일
최초 릴리스	HealthOmics 사용 설명서의 최초 릴리스	2022년 11월 29일