AWS HealthOmics Storage S3 URI Access Private Beta User Guide

Thank you for participating in the AWS HealthOmics Storage S3 URI Access private beta. This user guide will provide:

- Overview of AWS HealthOmics and HealthOmics Storage
- Review of the new functionality included in the beta
- Guide for setting up prerequisites for the beta
- Getting started guide on using S3 URIs
- Appendix covering common tool integrations and FAQs

This private beta will run from **Monday, February 19, 2024** to **Monday, April 15, 2024.** During this time, there will be no cost associated with activities done through S3 APIs. There will be charges for storage and HealthOmics API usage as outlined in the documentation. After April 15, usage of the S3 APIs will result in data transfer and API charges.

The Storage S3 URI Access private beta is available in all regions that AWS HealthOmics is available.

Table of Contents

Introduction

AWS HealthOmics is a fully managed service that helps you store, query, analyze, and generate insights from genomics and other biological data. AWS HealthOmics consist of 3 key components:

- ***HealthOmics Storage*** - store and share petabytes of genomics data efficiently and at low cost per gigabase.
- ***HealthOmics Analytics*** - simplifies how you prepare genomics data for multiomics and multimodal analyses.
- ***HealthOmics Workflows*** - automatically provisions and scales the underlying infrastructure for your bioinformatics workflows.

HealthOmics storage is a purpose-built service that helps healthcare and life science organizations build at-scale to store, query, and analyze genomic, transcriptomic, and other omics data. The storage service has 2 components

- A reference store for storing reference FASTA files
- A sequence store to store sequencing "read sets". A read set is the sequencing file(s) along with the metadata and, when applicable, the reference linkage. Currently FASTQ, BAM, uBAM, and CRAM file types are supported.

This beta expands the metadata of a read set to include an S3 URI for all active read sets. This URI can be used to read and list the files with tools built to leverage S3 APIs for file access.

Beta Feature Overview

The AWS HealthOmics Storage S3 URI Access private beta features will only be available for sequence stores created after the user's account has been enabled for the private beta.

This beta will create a S3 Access Point based URI and prefix for each sequence store. Under the sequence store there will be a prefix for each active read set and a S3 URI for every file in the read set. Each file will also have tags for subject id (omics:subjectId) and sample id (omics:sampleId). The prefix and file names are structured as follows:

- The sequence store prefix is `s3://<sequence_store_access_point_alias>/<account_id>/sequenceStore/<seq_store_id>/`
- The read set prefix extends this with `s3://<sequence_store_access_point_alias>/<account_id>/sequenceStore/<seq_store_id>/`**`readSet/<read_set_id>/`**

Under the read set ID prefix will be the files for a read set. For the file name we follow the following cascading logic:

- For imports from S3 (using the Console or startReadSetImportJob), we attempt to maintain the original source name.
    - If for a FASTQ read sets, both file names are the same, we insert in `_source1` or `_source2` before the `.fastq.gz` or `.fq.gz`.
- For Direct Uploads we use the read set name as follows:
    - For FASTQ: `<read_set_name>_<sourcex>.fastq.gz`
    - For uBAM/BAM/CRAM: `<read_set_name>.<file extension>` with extensions of `.bam|.cram`
- For index files generated, we append the index extension at the end of the source file name file name to create `<name of the Source the index is on>.<file index extension>`
    - For BAMs `.bai` is added
    - For CRAMs `.crai` is added

Along with adding the URI, the user's root account is given permissions to list the files at the sequence store (`s3:ListBucket`) and get files for active read sets (`s3:GetObject`). Users or roles that are attempting to access the objects through S3 APIs will need a policy attached to grant them permission to perform S3 actions and to use the sequence store's KMS key for decryption. Additionally, if the KMS key used for the sequence store is an AWS owned key, the user's root account is given permission to use the key for decryption (`kms:Decrypt`).

The `getSequenceStore` and `getReadSetMetadata` APIs have been updated to enable returning the S3 URI information. Please refer to the beta sequence store documentation (attached) for more details on the APIs.

Prerequisite

AWS CLI Setup and Configuration

In this Beta, the S3 URIs for a sequence store and read set can only be retried through the AWS CLI and APIs. Because this feature adds new response elements to the CLI, the CLI will need to be configured after it is installed.

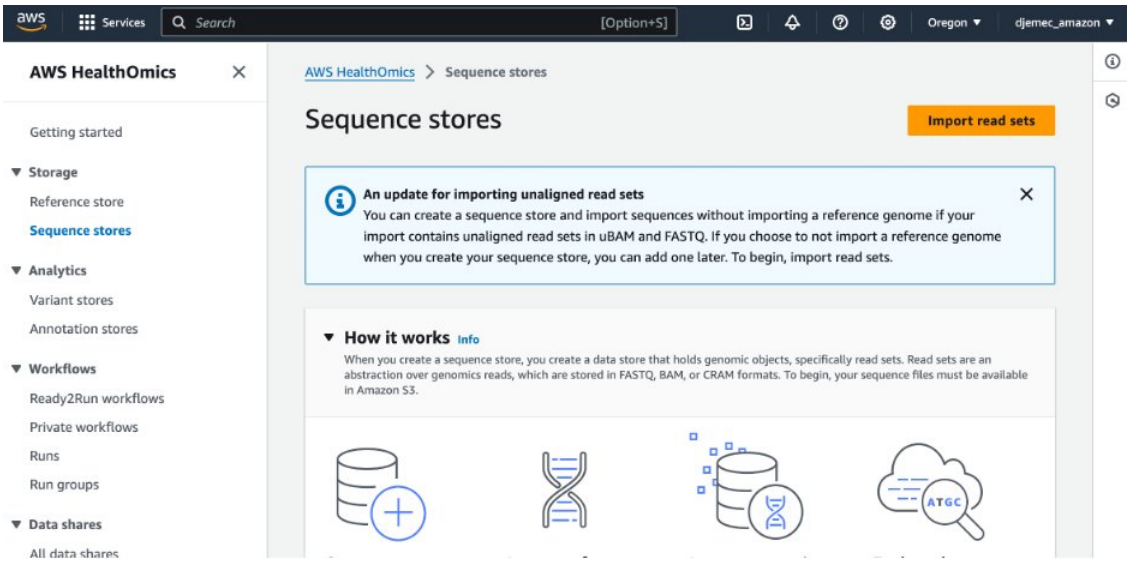
1. Ensure that AWS CLI v2.15.20 or newer is installed. Installation instructions can be found in the documentation. The version can be determined using `aws --version`
2. For the beta you will need 3 custom files to update the HealthOmics portion of the CLI. Download the files from here: `s3://omics-beta-assets/omics-2024-02-19.zip`
    a. `unzip omics-2024-02-19.zip`
    b. note that the role you use will need "s3:GetObject" permissions on the bucket and be part of the allow-listed account.
3. Unzip the downloaded file. It should contain 3 files:
    a. `omics-2022-11-28.json`
    b. `omics-2022-11-28.paginators.json`
    c. `omics-2022-11-28.waiters2.json`
    d. `omics-s3-uri-beta-commands.txt`
4. From a command line or terminal, navigate to the uncompressed folder and execute the following (also in `omics-s3-uri-beta-commands.txt`)

```
a. aws configure add-model --service-name omics --
   service-model file://omics-2022-11-28.json
   cp omics-2022-11-28.paginators.json
   ~/.aws/models/omics/2022-11-28/paginators-2.json
   cp omics-2022-11-28.waiters2.json
   ~/.aws/models/omics/2022-11-28/waiters-2.json
```

The AWS CLI is now ready to be used in the beta and will return the S3 URIs.

Creating a sequence store and importing read sets.

To get the S3 URI in beta, a new sequence store will need to be created and new read sets imported. Sequence stores and imports can be completed through the console or the APIs and this process remains unchanged. The documentation guides users through a CLI based setup. For a console based import, navigate to the HealthOmics sequence store page in the desired region and select "Import read sets". This will launch a guide that walks you through creating the necessary resources and importing read sets.



For large imports from S3, the manifest generator can be used to create import manifests from CSVs. For direct uploads, the HealthOmics transfer manager can be used to simplify the multipart upload process.

Getting Started

Once the sequence store and read sets have been successfully created, S3 URIs are now available through the CLI. For the beta, this information is only available through the CLI.

Retrieving S3 URIs

1. To retrieve the sequence store S3 prefix, access point ARN, and KMS key, use the `getSequenceStore` API. Through the CLI the command and response are as follows:
   a.
```
$ aws omics get-sequence-store --id 1234567890
{
    "id": "1234567890",
    "arn": "arn:aws:omics:us-west-
2:555555555555:sequenceStore/1234567890",
    "name": "S3URI Test",
    "sseConfig": {
        "type": "KMS",
        "keyArn": "arn:aws:kms:us-west-
2:777777777777:key/a3a33a33-3a33-33aa-aaaa-
333a3a33333"
    },
    "creationTime": "2024-02-
02T22:10:51.865000+00:00",
    "fallbackLocation": "s3://fallback_s3_bucket/",
    "s3Access": {
        "s3Uri": "s3://555555555555-1234567-
```

```
ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
s3alias/555555555555/sequenceStore/1234567890/",
        "s3AccessPointArn": "arn:aws:s3:us-west-
2:777777777777:accesspoint/555555555555-1234567890"
    }
}
```

b. To quickly get just the URI and access point ARN for the sequence store in bash using `jq` a 1 liner can be written as

i.
```
$ aws omics get-sequence-store --id 1234567890 |
jq -r .s3Access.s3Uri
s3://555555555555-1234567-
ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
s3alias/555555555555/sequenceStore/1234567890/

$ aws omics get-sequence-store --id 1234567890 |
jq -r .s3Access.s3AccessPointArn
arn:aws:s3:us-west-
2:777777777777:accesspoint/555555555555-
1234567890
```

2. To retrieve the S3 URI for an active read set, use the `getReadSetMetadata` API. Through the CLI the command and response are as follows:

a.
```
$ aws omics get-read-set-metadata --id 2379756261 --
sequence-store-id 1234567890
{
    "id": "2379756261",
    "arn": "arn:aws:omics:us-west-
2:555555555555:sequenceStore/1234567890/readSet/237975
6261",
    "sequenceStoreId": "1234567890",
    "subjectId": "HG00553",
    "sampleId": "HG00553 chr 21 22",
    "status": "ACTIVE",
    "name": "HG00553 Filtered BAM",
    "description": "filtered version of 1000 genomes
extract",
    "fileType": "BAM",
    "creationTime": "2024-02-
06T00:48:44.646000+00:00",
    "sequenceInformation": {
        "totalReadCount": 23317205,
        "totalBaseCount": 3497580750,
        "alignment": "ALIGNED"
    },
    "referenceArn": "arn:aws:omics:us-west-
2:555555555555:referenceStore/9012345678/reference/143
3017845",
    "files": {
```

```
        "source1": {
            "totalParts": 12,
            "partSize": 104857600,
            "contentLength": 1198232155,
            "s3Access": {
                "s3Uri": "s3://555555555555-1234567-
ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
s3alias/555555555555/sequenceStore/1234567890/readSet/
2379756261/HG00553_2122.bam"
            }
        },
        "index": {
            "totalParts": 1,
            "partSize": 104857600,
            "contentLength": 266664,
            "s3Access": {
                "s3Uri": "s3://555555555555-1234567-
ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
s3alias/555555555555/sequenceStore/1234567890/readSet/
2379756261/HG00553_2122.bam.bai"
            }
        }
    },
    "creationType": "IMPORT",
    "etag": {
        "algorithm": "BAM_MD5up",
        "source1": "a8b81530ee1488cb1269483c7f8db74f"
    }
}
```

i. to quickly get just the URI for source1 and index in bash using `jq` a 1 liner can be written as

1. ```
   $ aws omics get-read-set-metadata —id
   2379756261 —sequence-store-id 1234567890 |
   jq -r .files.source1.s3Access.s3Uri
   s3://555555555555-1234567-
   ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
   s3alias/555555555555/sequenceStore/123456789
   0/readSet/2379756261/HG00553_2122.bam

   $ aws omics get-read-set-metadata —id
   2379756261 —sequence-store-id 1234567890 |
   jq -r .files.index.s3Access.s3Uri
   s3://555555555555-1234567-
   ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
   s3alias/555555555555/sequenceStore/123456789
   0/readSet/2379756261/HG00553_2122.bam.bai
   ```
   ```

Granting Users and Roles S3 Based Access

The root of the account that created the sequence store is only given `s3:ListBucket` and `s3:GetObject` permission at the sequence store level. For a user or role to access the data, they must have a policy attached that grants them explicit access to the resource. In addition to S3 permissions, the user will also require decrypt access to the KMS key that was configured during sequence store creation.

Note that IAM policies do have set size limits. For the beta, only users and roles in the same account can be granted access.

A policy with full S3 based sequence store access

To setup a policy that gives a user or role access to read and list all read sets in a sequence store:

1. Run `getSequenceStore` API to get the `keyArn` and the. `s3AccessPointArn`
2. Create a policy with the following permissions:
    a.
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3DirectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:DataAccessPointArn":
"<s3AccessPointArn>"
                }
            }
        },
        {
            "Sid": "DefaultSequenceStoreKMSDecrypt",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "<keyArn>"
        }
    ]
}
```

A policy with only S3 get permissions on selected read sets

To setup a policy that restricts a user or role only the ability to get specific read sets through S3 APIs, we would modify the A policy with full S3 based sequence store access policy to further restrict the action and resource.

1. Run `getSequenceStore` API to get the `keyArn` and the `s3AccessPointArn`
2. Identify the Read Set IDs that access will be granted to.
3. Create a policy with the following permissions:
    a.
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "OnlyS3GetOnReadSetsSpecified",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "<s3AccessPointArn>/<account_id>/sequenceStore/<seq_store_id>/readSet/<read_set_id_1>/*",
                "<s3AccessPointArn>/<account_id>/sequenceStore/<seq_store_id>/readSet/<read_set_id_2>/*",
                "<s3AccessPointArn>/<account_id>/sequenceStore/<seq_store_id>/readSet/<read_set_id_3>/*",
                ...
            ]
            }
        },
        {
            "Sid": "DefaultSequenceStoreKMSDecrypt",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "<keyArn>"
        }
    ]
}
```

A policy with S3 permissions on selected read sets based on Subject ID

To setup a policy that gives a user or role s3 list and get access to specific read sets based on Sample ID through S3 APIs, we would modify the A policy with full S3 based sequence store access policy to further restrict the condition based on the tag. Restricting below the sequence store level does remove the ability to list all files available at the sequence store prefix.

1. Run `getSequenceStore` API to get the `keyArn` and the. `s3AccessPointArn`
2. Identify the Sample IDs that access will be granted to.

a. The Omics ListReadSets API can be used with a `sampleID` filter to validate that the Sample IDs to be restricted are returning the desired set of read sets.

3. Create a policy with the following permissions:

a.
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3DirectAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:DataAccessPointArn":
"<s3AccessPointArn>"
                }
                "StringEquals": {
                    "s3:ExistingObjectTag/omics:sample
Id": "<sample_id>"
                }
            }
        },
        {
            "Sid": "DefaultSequenceStoreKMSDecrypt",
            "Effect": "Allow",
            "Action": "kms:Decrypt",
            "Resource": "<keyArn>"
        }
    ]
}
```

Testing access and permissions

Once a policy is attached to the role or user the fastest way to verify that access has been configured correct is using S3 `get` or `list`.

1. To test S3 list permissions at the sequence store level with the AWS CLI, you can run s3 with or without the recursion option. If the sequence store has many read sets, using recursion may return a lengthy list.

a.
```
$ aws s3 ls s3://555555555555-1234567-
ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
s3alias/555555555555/sequenceStore/1234567890/
                           PRE readSet/

$ aws s3 ls s3://555555555555-1234567-
```

```
ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
s3alias/555555555555/sequenceStore/1234567890/ --
recursive
2024-02-02 14:27:24   28808998
555555555555/sequenceStore/1234567890/readSet/23561933
29/HG00548_u_filtered.bam
2024-02-05 16:46:49 1198232155
555555555555/sequenceStore/1234567890/readSet/23797562
61/HG00553_2122.bam
2024-02-05 16:46:49     266664
555555555555/sequenceStore/1234567890/readSet/23797562
61/HG00553_2122.bam.bai
...
```

b. To do this in one line with `jq` you can do the following:

i.
```
$ aws s3 ls $(aws omics get-sequence-store --id
1234567890 | jq -r .s3Access.s3Uri)
                           PRE readSet/
```

2. To test S3 list permissions at the read set level, add `readset/<readSetID>/` to the list command

a.
```
$ aws s3 ls s3://555555555555-1234567-
ajdpi90jdas90a79fh9a8ja98jdfa9jf98-
s3alias/555555555555/sequenceStore/1234567890/readSet/
2379756261/
2024-02-05 16:46:49 1198232155 HG00553_2122.bam
2024-02-05 16:46:49     266664 HG00553_2122.bam.bai
```

3. To test S3 get permissions on a specific path, you will need to deconstruct the URI of the object. The files URI is structured as `s3://<bucket>/<key>` where bucket ends in `-s3alias` and the key will end in the filename. Be careful to not prefix the key with `/` as that will result in an error.

a.
```
$ aws s3api get-object --bucket 555555555555-1234567-
ajdpi90jdas90a79fh9a8ja98jdfa9jf98-s3alias --key
555555555555/sequenceStore/1234567890/readSet/23797562
61/HG00553_2122.bam.bai Output_File_Name.bam.bai
{
    "AcceptRanges": "bytes",
    "LastModified": "2024-02-06T00:46:49+00:00",
    "ContentLength": 266664,
    "ETag": "\"ab40794183c804791b677c09f2cbdbba-1\"",
    "VersionId": "VFAau8qrMAU3Kukw4Vemg1Qj0A7J0G1H",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "aws:kms",
    "Metadata": {},
    "SSEKMSKeyId": "arn:aws:kms:us-west-
2:777777777777:key/a3a33a33-3a33-33aa-aaaa-
333a3a33333",
    "BucketKeyEnabled": true
}
```

i. Note that the "ETag" in this response is the S3 ETag for the file returned. This differs from the ETag that Omics calculates on the read set to represent the file's semantic content.

Sharing beyond your account

For this beta, direct access is restricted to users and roles within the same account as the sequence store data owner. To share with users beyond your account, there are two methods supported: presigned URLs and assumed roles.

Presigned URLs

You can use S3's presigned URLs to grant time-limited access to objects in the sequence store throught the S3 URI. A presigned URL can be entered in a browser or used by a program to download an object. The credentials used by the presigned URL are those of the AWS user who generated the URL. To generate a presigned URL, the user has to have `s3:getObject` access on the object. More details can be found on the S3 presign documentation.

1. For a specific read set file, to generate a presigned URL using the S3 CLI `presign` capabilities, you will need the file's URI which can be retrieved using `getReadSetMetadata`. With the URI you can then generate a presigned URI as follows.
    a. `$ aws s3 presign --expires-in 1440 s3://555555555555-1234567-ajdpi90jdas90a79fh9a8ja98jdfa9jf98-s3alias/555555555555/sequenceStore/1234567890/HG00553_2122.bam.bai`
    `https://....`

Assumed Roles

Assumed roles allow you as the data owner to create a role and tie specific policies to it. Once the role has been properly restricted, a policy will need to be added to grant the external user or role access to assume that role. The S3 guide for cross account sharing can be followed to setup a role and grant assume rights to external users.

S3 API capabilities

By enabling `s3:GetObject` and `s3:ListBucket` permissions, users can access the following S3 APIs. Additionally any tools that require those APIs can be used.

- GetObject - Retrieves an object from Amazon S3.
- HeadObject - The `HEAD` operation retrieves metadata from an object without returning the object itself. This operation is useful if you're interested only in an object's metadata.
- ListObjects - Returns some or all (up to 1,000) of the objects in a bucket. You can use the request parameters as selection criteria to return a subset of the objects in a bucket.

- ListObjectsV2 - Returns some or all (up to 1,000) of the objects in a bucket with each request. You can use the request parameters as selection criteria to return a subset of the objects in a bucket.
- CopyObject - Creates a copy of an object that is already stored in Amazon S3.
    - Note that copy object can only be done if the user or role has permission to write to the destination bucket.
    - When using this API, ensure that x-amz-tagging-directive is configured to "REPLACE". Using the default "COPY" will result in an API failure. This is important when using functions like `aws s3 sync` that leverage the CopyObject API.

Appendix

Appendix 1: Common Tools

AWS HealthOmics Workflows

When launching workflow in HealthOmics, a role is provided. To use S3 URIs from data in the sequence store, ensure that the role used by the run has a policy that gives it `kms:Decrypt` and `s3:GetObject` permissions and to the s3 URI of the read sets being accessed. For runs that access a large number of files, during this beta, the s3 URI based access is recommended over the Omics URI.

Boto3

The S3 URI provided is recognized by Boto3 as an S3 URI. Ensure that the profile that Boto3 is configured with has the proper policy attached based on the commands that will be run.

Workflow engines beyond HealthOmics

Many workflow engines like CWL, WDL, Nextflow, Snakemake are run on infrastructure outside of HealthOmics. These workflow engines have the ability to access S3 based files using a AWS profile or environmental variable configuration. When using S3 URIs from the sequence store, ensure that the user or role configured in the AWS profile being used by the engines has a policy attached that gives it `kms:Decrypt` and `s3:GetObject` permissions to the s3 URI of the read sets being accessed.

Hosted or Local IGV

IGV is a genome browser used to analyze BAM and CRAM files. It requires both the file and the index since it only displays a portion of the genome at a time. IGV can be downloaded and used locally and there are guides to creating an AWS hosted IGV. The public web version is not supported as it requires CORS.

For local IGV, it relies on the local AWS configuration to access files. Ensure that the role used in that configuration has a policy attached that enables `kms:Decrypt` and `s3:GetObject` permissions to the s3 URI of the read sets being accessed. After that, in IGV you can use "File > load from URL" and paste in the URI for the source and index. Alternatively, presigned URLs can be generated and used in the same manner which will bypass the AWS configuration.

The example AWS Hosted IGV relies on AWS Cognito to create the correct configurations and permissions inside environment. Ensure that a policy is created that enables `kms:Decrypt` and `s3:GetObject` permissions to the s3 URI of the read sets being accessed, and add this policy to the role that is assigned to the Cognito user pool. After that, in IGV you can use "File > load from URL" and enter in the URI for the source and index. Alternatively, presigned URLs can be generated and used in the same manner which will bypass the AWS configuration.

Note that the sequence store will not show up under the "Amazon" tab as that only displays buckets owned by you in the region the AWS profile is configured.

Samtools/HTSLib

HTSLib is the core library that is shared by several commonly used tools such as Samtools, rSamtools, PySam and others. Currently the HTSlib library has a bug when working with AWS S3 Access Point based URIs. At this time there are 2 workarounds when using the library.

1. Generate a presigned URL for the files you want to use. If a BAM or CRAM is being used, ensure that a presigned URL is generated for both the file and the index. After that, both files can be used with the libraries.
2. Use Mountpoint to mount the sequence store or read set prefix in the same environment where you're using HTSlib libraries. From here, the files can be accessed using local file paths.

Mountpoint for S3

Mountpoint for Amazon S3 is a simple, high-throughput file client for mounting an Amazon S3 bucket as a local file system. With Mountpoint for Amazon S3, your applications can access objects stored in Amazon S3 through file operations like `open` and `read`. Mountpoint for Amazon S3 automatically translates these operations into S3 object API calls, giving your applications access to the elastic storage and throughput of Amazon S3 through a file interface.

Mountpoint can be installed using the installation instructions. Mountpoint uses the AWS Profile local to the installation and works at a S3 prefix level. Ensure that the profile being used has a policy that enables `s3:GetObject, s3:ListBucket` and `kms:Decrypt` permissions to the s3 URI prefix of the read set(s) or sequence store being accessed. After that the bucket can be mounted using the following:

```
mount-s3 <access point arn> <local/path/to/mount> --prefix
<prefix to sequence store or read set> --region <region>
```

CloudFront

Amazon CloudFront is a content delivery network (CDN) service built for high performance, security, and developer convenience. Customers wishing to use CloudFront will need to work with the Service team to get the CloudFront distribution enabled. Please work with your account team to engage the HealthOmics service team.

VPC Based Access

Accessing read set files using S3 URIs through a private connection requires setting up PrivateLink interface endpoints on the Sequence Store. This guide outlines the setup of PrivateLink on a sequence store. After setup, the storage endpoints will be `com.amazonaws.<region>.storage-omics` and `com.amazonaws.region.control-storage-omics.`

After PrivateLink is setup, follow the guide for configuring gateway endpoints for S3. Since the S3 bucket is owned by HealthOmics, the steps for adjusting the bucket policy can be skipped. Gateway endpoints will rely on the policy attached to the user or role accessing the data but endpoints can also be configured with further restrictive policies if desired. The policies can include restricting the access based on the S3 Access Point ARN and S3 actions desired.

Appendix 2: FAQ

> *If there are still issues after these steps are taken, please reach out to your AWS Account Team and/or the AWS HealthOmics team for further help troubleshooting.*

1. I do not see the S3 URI in my `getSequenceStore` or `getReadSetMetadata` response. Where is it?
    a. Ensure that sequence store and the read set were created after the account was allow listed. The sequence store creation date can be found using `getSequenceStore`
    b. If the sequence store was created after, ensure that the steps were done to update the AWS CLI AWS CLI Setup and Configuration
    c. Validate that the read set you are trying to access is Active. Currently S3 URIs are only available for active read sets.
2. I get permission denied when trying to do `s3 list` or `s3 get` on the S3 URI. What's wrong?
    a. Validate that the URI is correct.
    b. If the URI is correct, validate that your role has a policy attached with explicit access to the S3 URI you are trying to access. In the CLI you can run `aws sts get-caller-identity` to determine which role is being used. Using that role, validate that your policy allows the action you have selected. See Granting

Users and Roles S3 Based Access for more examples. Ensure that both S3 actions and KMS actions are available.

   c. If you are using a user managed KMS key, ensure that the role or user is also allowed on the KMS key resource policy.

3. I haven't used a read set in 30 days but I still see it. Why is that?
   a. In the Beta the sequence stores created are all frozen with just active read sets. This feature will be removed for GA where read sets that are archived are removed from S3 access. Please work with your account team and the HealthOmics team for options around mitigating the storage cost impacts.
4. How can I get reference store S3 URIs?
   a. The Beta is only available for sequence stores. To use your reference files through S3 URI we recommend uploading a copy of the reference in an S3 bucket in the same region.
5. How can I get a S3 URIs for a sequence store that I created before the beta?
   a. The Beta only supports S3 URIs for sequence stores created after the account is enabled for the functionality. Enablement of previously created sequence stores or stores without access to the Beta will be done as part of the GA release.
6. How can i remove S3 access to my sequence store?
   a. The fastest way is to remove the policy from the user and/or role you wish to restrict. For fully removing all access please contact the HealthOmics team.
7. I cannot write to the S3 URI provided?
   a. Accounts are only given permission for the actions `s3:GetObject` and `s3:ListBucket`. Writing to the S3 URI is not available. If this is necessary we recommend creating a staging S3 bucket that can be written to and then using Read Set Import Jobs to import sequencing files from an S3 bucket. Alternatively, the HealthOmics direct upload API can be used to skip the S3 Bucket and directly write to a sequencing store.
8. Why is the ETag on my S3 object different than the ETag on the Read Set?
   a. The ETag on a read set that is returned by HealthOmics APIs reflects changes to the semantic content of the object, not its metadata (docs). The Etag returned by the S3 APIs is is a hash of the file object. (S3 docs). HealthOmics Sequence Stores maintain semantic identity of files but not bitwise identity so throughout a files lifecycle the S3 ETag may change.
9. What happens if i allow `s3:*` permission on the policy I've attached to the user?
   a. The user's account is restricted by the access point to only support the actions `s3:GetObject` and `s3:ListBucket`. Because the access point sits within the HealthOmics account, this policy is not editable. If a user writes a policy with `s3:*` access and attaches it to a role, the access point will provide the access restrictions and prevent access beyond the enabled permissions.
10. Will the S3 URI change for a file?
    a. No, once the URI is generated for a read set and files, the URI will remain the same throughout the lifecycle of a read set. Since the URI is based on the sequence store ID, the read set ID, and the file name, the prefix pattern will remain the same as long as these values remain consistent.

Appendix 3: Known issues and resolution timelines

1. Using HTSlib, Samtools, RSamtools, PySam.
    a. Issue: HTSlib and it's derivatives currently do not work with Access points: https://github.com/samtools/samtools/issues/1984
    b. Fix: We are working with to deliver a fix (ETA by April 2024) and then will will work with the community to get this incorporated into a version release
    c. Workaround: Presigned URLs can be used with the commands. Alternatively, the sequence store can be used with mountpoint and then the file paths can be used.
2. CopyObject based actions like S3 Sync.
    a. Issue: CopyObject APIs attempt to copy the tags when copying an object across buckets. This requires `s3:GetObjectTagging` permission, a permission that's currently blocked.
    b. Fix: We are evaluating the impact of this issue and considering adding this permission for GA.
    c. Workaround: When using this API, ensure that x-amz-tagging-directive is configured to "REPLACE". Using the default "COPY" will result in an API failure.
3. FSx for Lustre and File Cache are not supported
    a. Issue: These tools are commonly used in conjunction with S3 to support HPC workflows where file system like integrations are required. These tools require PutObject permissions, along with several others, that are not enabled as part of this beta.
    b. Fix: We are evaluating options, customer appetite, and paths forward where "read only" versions of those tools are supported. No fix or timeline is current planned.
    c. Workaround: Customers have seen success using Mountpoint for S3 with different cache size configurations for supporting file system like integrations. Mountpoint for S3