Genomics Tertiary Analysis and Data Lakes Using AWS Glue

Implementation Guide

Genomics Tertiary Analysis and Data Lakes Using AWS Glue: Implementation Guide

Table of Contents

Create a scalable environment in AWS to prepare genomic data for large-scale analysis and perform interactive queries against a genomics data lake

Publication date: *July 2020 (last update (p. 36): September 2020)*

This implementation guide discusses architectural considerations and configuration steps for deploying the Genomics Tertiary Analysis and Data Lakes Using AWS Glue and Amazon Athena solution in the Amazon Web Services (AWS) Cloud. It includes a link to an AWS CloudFormation template that launches and configures the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT infrastructure architects, administrators, data scientists, software engineers, and DevOps professionals who have practical experience architecting in the AWS Cloud.

Overview

The Genomics Tertiary Analysis and Data Lakes Using AWS Glue and Amazon Athena solution creates a scalable environment in AWS to prepare genomic data for large-scale analysis and perform interactive queries against a genomics data lake. This solution demonstrates how to 1) build, package, and deploy libraries used for genomics data conversion, 2) provision data ingestion pipelines for genomics data preparation and cataloging, 3) run interactive queries against a genomics data lake.

Build, package, and deploy libraries used for genomics data conversion

Note:
Hail (p. 17) is an open-source library from the Broad Institute for scalable genomic data exploration.

The solution uses AWS CodeBuild and AWS CodePipeline to build, package and deploy Hail as a jar file to an Amazon Simple Storage Service (Amazon S3) bucket. Hail is used to load Variant Call Files (VCFs) into Apache Spark data frames for data processing and format conversion. You can learn more about adding or modifying solution CodeBuild jobs to build, package, and deploy library dependencies in the *Genomics Tertiary Analysis and Data Lakes using AWS Glue and Amazon Athena Developer Guide*.

Provision data ingestion pipelines for genomics data preparation and cataloging

During the solution setup, a clinvar.tsv.gz file, an example VCF file, and a subset of the 1000 genomes dataset (in Parquet format and partitioned by sample ID) are copied into the solution data lake bucket. Parquet versions of each dataset are also included to make running the following Extract, Transform, and Load (ETL) jobs optional.

Note:
Apache Parquet and Apache ORC are popular columnar data stores that are optimized for best performance and cost-savings when querying data in Amazon S3. They provide features that store data efficiently by employing column-wise compression, different encoding, compression based on data type, and predicate pushdown. They can also be split. Better compression ratios or skipping blocks of data means reading fewer bytes from Amazon S3, leading to better query performance.

AWS Glue jobs are provided to prepare data. The vcf-to-parquet (p. 29) job converts variant call data in the Variant Call File (VCF) format into Apache Parquet format. The clinvar-to-parquet (p. 31) job converts ClinVar data in a Tab Separated Value (TSV) format into Apache Parquet format.

AWS Glue crawlers are provided to catalog data. Crawlers crawl data files to infer their data schema and use that schema to create or update tables in a data catalog. The variants (p. 30) crawler crawls the 1000 genomes Parquet data files and adds or updates a variants table to the data catalog. The clinvar (p. 32) crawler crawls the clinvar Parquet data files and adds or updates a clinvar table to the data catalog. The sample (p. 33) crawler crawls the sample Parquet data files and adds or updates a sample table to the data catalog. You can learn more about adding or modifying solution jobs and crawlers to prepare and catalog datasets, in the *Genomics Tertiary Analysis and Data Lakes using AWS Glue and Amazon Athena Developer Guide*.

Run big data queries against a genomics data lake

Note:
PyAthena is a Python DB API 2.0 (PEP 249) compliant client for Amazon Athena.

An Amazon SageMaker notebook instance is provisioned with an example Jupyter notebook that demonstrates how to work with data in a genomics data lake. The solution notebook uses Amazon Athena to identify genomic variants related to drug response for a given cohort of individuals. The below query is run against data in the data lake using the PyAthena driver to 1) filter by samples in a subpopulation, 2) aggregate variant frequencies for the subpopulation-of-interest, 3) join on the ClinVar dataset, 4) filter by variants that have been implicated in drug-response, 5) order by highest frequency variants. The query can also be run in the Amazon Athena console.

```
SELECT
        count(*)/cast(numsamples AS DOUBLE) AS genotypefrequency
        ,cv.rsid
        ,cv.phenotypelist
        ,sv.chromosome
        ,sv.startposition
        ,sv.endposition
        ,sv.referenceallele
        ,sv.alternateallele
        ,sv.genotype0
        ,sv.genotype1
        FROM genomicsanalysis.onekg_chr22_by_sample sv
        CROSS JOIN
        (SELECT count(1) AS numsamples
        FROM
        (SELECT DISTINCT sampleid
        FROM genomicsanalysis.onekg_chr22_by_sample
        WHERE sampleid LIKE 'NA12%'))
        JOIN genomicsanalysis.clinvar cv
        ON sv.chromosome = cv.chromosome
        AND sv.startposition = cv.start - 1
        AND sv.endposition = cv.stop
        AND sv.referenceallele = cv.referenceallele
        AND sv.alternateallele = cv.alternateallele
        WHERE assembly='GRCh37'
        AND cv.clinicalsignificance LIKE '%response%'
        AND sampleid LIKE 'NA12%'
        GROUP BY  sv.chromosome
        ,sv.startposition
        ,sv.endposition
        ,sv.referenceallele
        ,sv.alternateallele
        ,sv.genotype0
        ,sv.genotype1
        ,cv.clinicalsignificance
        ,cv.phenotypelist
        ,cv.rsid
        ,numsamples
        ORDER BY  genotypefrequency DESC LIMIT 50
```

The solution includes continuous integration and continuous delivery (CI/CD) using AWS CodeCommit source code repositories and AWS CodePipeline for building and deploying updates to the data preparation jobs, crawlers, data analysis notebooks, and the data lake infrastructure. This solution fully leverages infrastructure as code principles and best practices that enable you to rapidly evolve the solution. After deployment, you can modify the solution to fit your particular needs, for example, by adding new data preparation jobs and crawlers. Each change is tracked by the CI/CD pipeline, facilitating change control management, rollbacks, and auditing.

Cost

You are responsible for the cost of the AWS services used while running this reference deployment. As of the date of publication, the cost for running this solution *with default settings* in the US East (N. Virginia) Region is approximately **$0.45** during setup to run the three crawlers, **$0.05 per month** to run the Amazon SageMaker Notebook Instance, and **$0.00025 each drug response query execution** for interpreting the data with Amazon Athena. Prices are subject to change. For full details, see the pricing webpage for each AWS service used in this solution.

Note:
AWS Glue job execution: 2 DPUs * 1/6 hour at $0.44 per DPU-Hour or $0.15
AWS Glue crawler execution: 2 DPUs * 1/6 hour at $0.44 per DPU-Hour or $0.15
Drug response query execution: Less than 0.0005 TB scanned * 0.0005 * $5/TB = $0.0025

If you customize the solution to analyze your genomics dataset, the cost factors include the storage size of the data being analyzed, the number of Extract Transform and Load (ETL) jobs and crawlers being used, compute resources required for each job, number of notebook instances provisioned and volume of data scanned when using Athena. For a more accurate estimate of cost, we recommend working with a sample dataset of your choosing as a benchmark. Prices are subject to change. For full details, see the pricing webpage for each AWS service used in this solution.

Architecture

The following diagram describes the overall data lake architecture; how data is ingested, curated, cataloged, and queried.

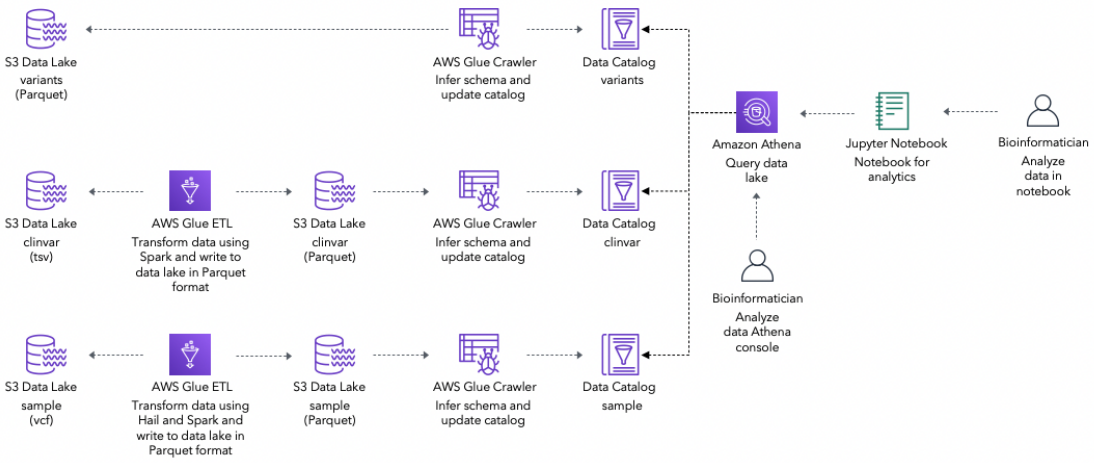

Figure 1: Genomics Tertiary Analysis and Data Lakes Using AWS Glue and Amazon Athena data lake architecture

This solution demonstrates how to ingest common genomics data sets into a centralized data lake and work with that data using Amazon Athena and Jupyter notebooks. There is an example ingestion pipeline for genomic variant calls data (1000 Genomes), annotation data (ClinVar) and an individual Variant Call File (VCF) data.

A portion of the 1000 Genomes dataset (Chromosome 22) is copied into the solution data lake bucket during solution setup. The dataset is in Apache Parquet format and partitioned by sample ID. The variants (p. 30) crawler is provided to crawl the dataset, infer the data schema, and add a table to the solution AWS Glue data catalog.

A ClinVar file, in Tab Separated Values (TSV) format, is copied into the solution data lake bucket during solution setup. The clinvar-to-parquet (p. 31) job is provided to convert the dataset to Parquet format. The clinvar (p. 32) crawler is provided to crawl the dataset, infer the data schema, and add a table to the solution AWS Glue data catalog.

An example Variant Call File (VCF) is copied into the solution data lake bucket during solution setup. The vcf-to-parquet (p. 29) job is provided to convert the dataset to Parquet format. The sample (p. 33) crawler is provided to crawl the dataset, infer the data schema, and add a table to the solution AWS Glue data catalog.

> **Note:**
> A version of each dataset in Parquet format are copied into the solution data lake bucket during setup. Also, the variants, clinvar and sample crawlers are run during solution setup. This allows users immediately work with the data once the setup is complete.

Deploying this solution with the **default parameters** builds the following environment in the AWS Cloud.

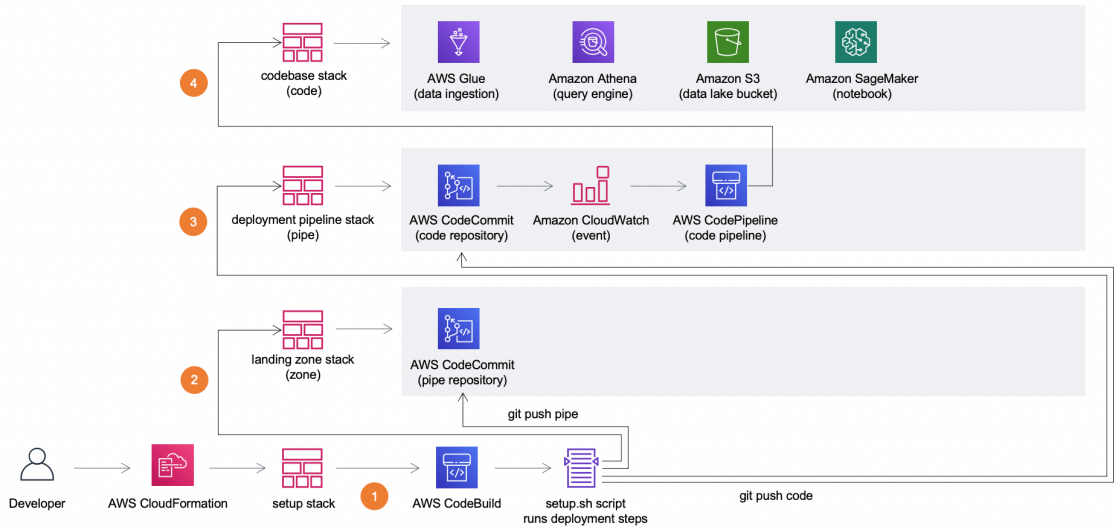

Figure 2: Genomics Tertiary Analysis and Data Lakes Using AWS Glue and Amazon Athena deployment architecture

The AWS CloudFormation template creates four CloudFormation stacks in your AWS account including a `setup` stack to install the solution. The other stacks include a landing zone (`zone`) stack containing the common solution resources and artifacts, a deployment pipeline (`pipe`) stack defining the solution's CI/CD pipeline, and a codebase (`code`) stack providing the ETL scripts, jobs, crawlers, a data catalog, and notebook resources.

1. The `setup` stack creates an AWS CodeBuild project containing the `setup.sh` script. This script creates the remaining CloudFormation stacks and provides the source code for both the AWS CodeCommit `pipe` repository and the code repository.
2. The landing zone (`zone`) stack creates the CodeCommit `pipe` repository. After the landing zone (`zone`) stack completes its setup, the `setup.sh` script pushes source code to the CodeCommit `pipe` repository.
3. The deployment pipeline (`pipe`) stack creates the CodeCommit `code` repository, an Amazon CloudWatch event, and the CodePipeline `code` pipeline. After the deployment pipeline (`pipe`) stack completes its setup, the `setup.sh` script pushes source code to the CodeCommit `code` repository.
4. The CodePipeline (`code`) pipeline deploys the codebase (`code`) CloudFormation stack. After the AWS CodePipeline pipelines complete their setup, the resources deployed in your account include Amazon Simple Storage Service (Amazon S3) buckets for storing object access logs, build artifacts, and data in

your data lake; CodeCommit repositories for source code; an AWS CodeBuild project for building code artifacts (for example, third-party libraries used for data processing); an AWS CodePipeline pipeline for automating builds and deployment of resources; example AWS Glue jobs, crawlers, and a data catalog; and an Amazon SageMaker Jupyter notebook instance.

The example code includes the resources needed to prepare genomic data for large-scale analysis and perform interactive queries against a genomics data lake.

Solution components

CI/CD pipeline

A complete continuous integration and continuous deployment (CI/CD) pipeline is created when you launch the solution. This pipeline is built using AWS CodeCommit as the source code, AWS CodeBuild projects to build the solution's artifacts (for example, hail.jar), and an AWS CodePipeline pipeline to run a build project and automate deployment (using AWS CloudFormation) after the updated source code is published.

The AWS resources that compose the CI/CD pipeline are defined in the `Pipe/template_cfn.yml` file. Changes to the solution requiring new artifacts to be built require an update to the CI/CD pipeline definition.

Solution demonstration datasets

This solution copies the following datasets into your solution data lake bucket.

- **1000 Genomes, Chromosome 22 (cohort)** – A portion of the 1000 genomes public dataset of human genomic variant data, partitioned by sample ID and in Apache Parquet format. This dataset is used as our cohort dataset for creating the drug response report.
- **ClinVar (annotation)** – The public dataset that aggregates information about genomic variation and its relationship to human health. Two copies of this dataset are copied into your data lake bucket, one in Tab Separated Values (TSV) format and another in Apache Parquet format. The TSV file is used as an input to the clinvar-to-parquet job which produces the dataset in Apache Parquet format. We copy the parquet version of the dataset into the data lake bucket after the solution is set up so that running the AWS Glue job crawler is optional.
- **Individual Sample Variants (sample)** – An individual sample Variant Call File (VCF) dataset used to demonstrate VCF to Apache Parquet conversion. Two copies (one in VCF and format and one in Parquet format) of this dataset are copied into your data lake bucket. The VCF copy is used as an input to the vcf-to-parquet AWS Glue job, which produces the dataset in Apache Parquet format. The Parquet version of the dataset is copied into the data lake bucket so that running the AWS Glue Job crawler is optional.

AWS Glue jobs

This solution creates the following AWS Glue jobs.

- `vcf-to-parquet` – Transforms variant calls in a Variant Call File (VCF) format into Apache Parquet format using Hail from the Broad Institute and writes the resulting files to the solution data lake.
- `clinvar-to-parquet` – Transforms Clinical Variant (ClinVar) data in a Tab Separated Values (TSV) format into Apache Parquet format and writes the resulting files to the solution data lake.

  Hail is built from open source code and is publicly available under open source license. For more information, see Appendix A (p. 17).

AWS Glue crawlers

This solution creates the following AWS Glue crawlers.

- `variants-crawler` – Creates/Updates the *variants* table in the solution's AWS Glue data catalog to reflect the data schema of the 1000 Genomes example cohort variant data in the solution data lake.
- `clinvar-crawler` – Creates/Updates the *clinvar* table in the solution's AWS Glue data catalog to reflect the data schema of the ClinVar dataset in the solution data lake.

AWS Glue data catalog

This solution creates an AWS Glue data catalog with a *genomicsanalysis* database that contains *variants* and *clinvar* tables. AWS Glue is configured to encrypt the metadata stored in the data catalog, data files stored in Amazon S3 buckets, and all logs stored in Amazon CloudWatch.

SageMaker notebook instance

This solution creates an Amazon SageMaker notebook instance that demonstrates how to use AWS Glue and Amazon Athena to identify variants related to drug response.

Amazon Simple Storage Service buckets

This solution creates the following Amazon Simple Storage Service (Amazon S3) buckets. Each bucket has encryption and logging enabled.

- **Data Lake Bucket** – Stores genomic variant and ClinVar variant annotation data.
- **Resources Bucket** – Stores notebooks and shell scripts.
- **Build Bucket** – Stores build artifacts deployed through the pipeline.

Considerations

The Genomics Tertiary Analysis and Data Lakes Using AWS Glue and Amazon Athena solution fully leverages infrastructure as code principles and best practices that enable you to rapidly evolve the solution. Storing your Extract Transform and Load (ETL) job, crawler, and data lake definitions as code makes them easier to share, inspect for compliance, and reproduce. Additionally, each change you make is tracked by the CI/CD pipeline, facilitating change control management, rollbacks, and auditing.

Regional deployment

This solution uses the AWS CodePipeline service, which is currently available in specific AWS Regions only. Therefore, you must launch this solution in an AWS Region where this service is available. For the most current service availability by AWS Region, see AWS service offerings by region. The solution was tested in all regions.

AWS CloudFormation template

This solution uses AWS CloudFormation to automate the deployment of the Genomics Tertiary Analysis and Data Lakes Using AWS Glue and Amazon Athena solution in the AWS Cloud. It includes the following AWS CloudFormation template, which you can download before deployment.

genomics-tertiary-analysis-and-data-lakes-using-aws-glue-and-amazon-athena.template: Use this template to launch this solution and all associated components. The default configuration deploys an AWS CodePipeline deployment pipeline, an AWS CodeCommit repository for the pipeline code, an AWS CodeCommit repository for the solution code, AWS Glue jobs, crawlers, and a data catalog, AWS Identity and Access Management (IAM) roles and policies, and an Amazon SageMaker notebook instance. You can also customize the template based on your specific needs.

Automated deployment

Before you launch the automated deployment, review the architecture, configuration, network security, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 30 minutes.

Prerequisites

Before deploying this solution, verify that you have an administrator role in AWS Identity and Access Management (IAM) in your AWS account. See Appendix B (p. 18) for details on permissions used by this solution. For information about setting up an administrator user, see Creating Your First IAM Admin User and Group in the *AWS Identity and Access Management User Guide*.

What we'll cover

The procedure for deploying this architecture on AWS consists of the following steps. For detailed instructions, follow the links for each step.

Step 1. Launch the stack (p. 11)

- Launch the AWS CloudFormation template into your AWS account.

Step 2. Confirm crawler completion (p. 12)

Step 3. Query data in the genomics data lake (p. 12)

Step 1. Launch the stack

This automated AWS CloudFormation template deploys the solution in the AWS Cloud. You must have the appropriate IAM permissions before launching the stack.

> **Note:**
> You are responsible for the cost of the AWS services used while running this solution. Refer to the Cost (p. 4) section for more details. For full details, see the pricing webpage for each AWS service used in this solution.

1. Sign in to the AWS Management Console and click the button below to launch the `genomics-tertiary-analysis-and-data-lakes-using-aws-glue-and-amazon-athena.template` AWS CloudFormation template.

You can also download the template as a starting point for your own implementation.

2. The template launches in the US East (N. Virginia) Region by default. To launch this solution in a different AWS Region, use the Region selector in the console navigation bar.

   Note:
   This solution uses the AWS CodePipeline service, which is currently available in specific AWS Regions only. Therefore, you must launch this solution in an AWS Region where this service is available. For the most current service availability by AWS Region, see AWS service offerings by Region.

3. On the **Create stack** page, verify that the correct template URL shows in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack and provide a project name for the solution installation.
5. Choose **Next.**
6. On the **Configure stack options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
8. Choose **Create stack** to deploy the stack.

   You can view the status of the stack in the AWS CloudFormation Console in the **Status** column. You should see a status of CREATE_COMPLETE in approximately 30 minutes.

Step 2. Confirm crawler completion

Note:
The Parquet version of the ClinVar(annotation) and 1000 Genomes Chr22 (variants) and VCF sample datasets is copied into the data lake bucket after the solution is set up so that running the AWS Glue jobs are optional. We run the AWS Glue crawlers on these datasets automatically during setup of the solution to create the annotation, variants and sample tables in the data catalog. This allows the user to work with the data as soon as solution setup is complete. The vcf-to-parquet and clinvar-to-parquet jobs are provided for modification and reference.

The annotation (p. 30), variants (p. 30) and sample (p. 33) AWS Glue crawlers are launched after the solution deployment completes. You must confirm that the crawlers are complete before creating the demonstration drug response report using the solution's Jupyter notebook.

Use the following steps to verify the crawler's status:

1. Sign in to the AWS Glue console.
2. Select **Crawlers** on the left navigation pane.
3. Wait for the crawler's status to display **READY**, and the **tables added** status to each be **1**.

Step 3. Query data in the genomics data lake

Note:
An Amazon SageMaker notebook instance is provisioned with an example Jupyter notebook that demonstrates how to work with data in a genomics data lake. The solution notebook uses Amazon Athena to identify genomic variants related to drug response for a given cohort of individuals. The below query is run against data in the data lake using the PyAthena driver to 1) filter by samples in a subpopulation, 2) aggregate variant frequencies for the subpopulation-of-interest, 3) join on the ClinVar dataset, 4) filter by variants that have been implicated in drug-response, 5) order by highest frequency variants.

An example drug response report can be generated by executing each step in a provided demonstration Jupyter notebook.

Option 1: Use the provided Jupyter notebook

Use the following steps to execute the notebook:

1. Sign in to the Amazon SageMaker console.
2. Select **Notebook Instances** under the Notebook category on the left pane.
3. Select your notebook instance and choose **Open Jupyter**.
4. Select **runbook.ipynb**.
5. Run each step in the notebook.

Option 2: Run the query in the Amazon Athena console

Use the following setups to execute the query in the Amazon Athena console:

1. Sign in to the Amazon Athena console.
2. Under **Data Source** select the **AwsDataCatalog** data source on the left pane.
3. Under **Database** select the **genomicsanalysis** database on the left pane.
4. In the query window paste the query below.
5. Select **Run Query**.

```
  SELECT
    count(*)/cast(numsamples AS DOUBLE) AS genotypefrequency
    ,cv.rsid
    ,cv.phenotypelist
    ,sv.chromosome
    ,sv.startposition
    ,sv.endposition
    ,sv.referenceallele
    ,sv.alternateallele
    ,sv.genotype0
    ,sv.genotype1
FROM genomicsanalysis.onekg_chr22_by_sample sv
CROSS JOIN
    (SELECT count(1) AS numsamples
    FROM
        (SELECT DISTINCT sampleid
        FROM genomicsanalysis.onekg_chr22_by_sample
        WHERE sampleid LIKE 'NA12%'))
JOIN genomicsanalysis.clinvar cv
ON sv.chromosome = cv.chromosome
    AND sv.startposition = cv.start - 1
    AND sv.endposition = cv.stop
    AND sv.referenceallele = cv.referenceallele
    AND sv.alternateallele = cv.alternateallele
WHERE assembly='GRCh37'
    AND cv.clinicalsignificance LIKE '%response%'
    AND sampleid LIKE 'NA12%'
GROUP BY  sv.chromosome
          ,sv.startposition
          ,sv.endposition
```

```
          ,sv.referenceallele
          ,sv.alternateallele
          ,sv.genotype0
          ,sv.genotype1
          ,cv.clinicalsignificance
          ,cv.phenotypelist
          ,cv.rsid
          ,numsamples
ORDER BY  genotypefrequency DESC LIMIT 50
```

Security

This solution is preconfigured with all of the IAM policies and roles necessary to run the solution with least privileges.

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared model can reduce your operational burden as AWS operates, manages, and controls the components from the host operating system and virtualization layer down to the physical security of the facilities in which the services operate. For more information about security on AWS, visit the AWS Security Center.

IAM roles

AWS Identity and Access Management (IAM) roles enable you to secure jobs and crawlers running in AWS Glue, and restrict access to the data catalog, the data lake bucket, and the notebook instance. All of the IAM roles in this solution have been defined with least privileges. Refer to Appendix B (p. 18) for details about roles and permissions used in this solution.

Additional resources

- AWS CloudFormation
- AWS CodeBuild
- AWS CodeCommit
- Amazon CloudWatch
- Amazon SageMaker
- AWS CodePipeline
- AWS Glue
- Amazon Athena
- Amazon Simple Storage Service

Appendix A: Open source licenses

This solution builds the following open source bioinformatics tools as container images and pushes the images to an Amazon Elastic Compute Cloud (Amazon EC2) Amazon Elastic Container Registry (Amazon ECR) in your account. These tools are built from open source when their respective container images are created. They are publicly available under open source licenses as provided in the following table. Ensure that these license specifications meet your organizational requirements.

Tool Name	Description	License
Hail	Hail is an open-source, general-purpose, Python-based data analysis library with additional data types and methods for working with genomic data. For more information, refer to the Hail documentation.	MIT
PyAthena	PyAthena is a Python DB API 2.0 (PEP 249) compliant client for Amazon Athena.	

Appendix B: Roles and permissions

This solution uses AWS CodeBuild, AWS CloudFormation, and AWS CodePipeline for Continuous Delivery (CD) and AWS Glue and Amazon Athena for scientific analysis using a genomics data lake. Review the following CodeBuild, AWS CloudFormation, CodePipeline, AWS Glue, and Amazon Athena permissions to ensure that you have the appropriate permissions enabled.

Code deployment pipeline permissions

Use IAM to manage access to AWS CodeBuild jobs, AWS CloudFormation stacks, and the AWS CodePipeline `code` pipeline. CodeBuild jobs and the CodePipeline code pipeline have their own IAM roles and IAM policies.

The following code examples demonstrate the IAM roles and supporting IAM policies defined in the `GenomicsAnalysisPipe/pipe_cfn.yml` file; including `CodeBuildRole`, `CodePipelineRole`, `CloudFormationRole`, and `SourceEventRole`.

CloudFormation Role

`CloudFormationRole` defines the permissions needed for AWS CloudFormation to provision IAM roles, S3 buckets, an Amazon SageMaker notebook instance, and AWS Glue resources. AWS CloudFormation uses the CloudFormation action type in the CodePipeline.

```
CloudFormationRole:
  Type: AWS::IAM::Role
  Properties:
    Path: /
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Action:
            - sts:AssumeRole
          Principal:
            Service:
              - cloudformation.amazonaws.com
    Policies:
      - PolicyName: CloudFormationRolePolicy
        PolicyDocument:
          Version: 2012-10-17
          Statement:
            - Effect: Allow
              Action:
                - iam:CreateRole
                - iam:DeleteRole
                - iam:PutRolePolicy
                - iam:GetRolePolicy
                - iam:DeleteRolePolicy
                - iam:AttachRolePolicy
                - iam:DetachRolePolicy
                - iam:UpdateAssumeRolePolicy
                - iam:PassRole
                - iam:GetRole
              Resource:
```

```
                - !Sub arn:aws:iam::${AWS::AccountId}:role/${ResourcePrefix}*
              - Effect: Allow
                Action:
                  - glue:CreateJob
                  - glue:UpdateJob
                  - glue:DeleteJob
                  - glue:GetJob
                Resource: '*'
              - Effect: Allow
                Action:
                  - glue:CreateSecurityConfiguration
                  - glue:GetSecurityConfiguration
                  - glue:DeleteSecurityConfiguration
                Resource: '*'
              - Effect: Allow
                Action:
                  - glue:CreateWorkflow
                  - glue:DeleteWorkflow
                  - glue:UpdateWorkflow
                Resource: '*'
              - Effect: Allow
                Action:
                  - glue:GetDataCatalogEncryptionSettings
                  - glue:PutDataCatalogEncryptionSettings
                  - glue:DeleteDataCatalogEncryptionSettings
                Resource:
                  - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:catalog
              - Effect: Allow
                Action:
                  - glue:CreateDatabase
                  - glue:UpdateDatabase
                  - glue:DeleteDatabase
                  - glue:GetCrawler
                  - glue:CreateCrawler
                  - glue:UpdateCrawler
                  - glue:DeleteCrawler
                  - glue:StopCrawler
                  - glue:StopTrigger
                  - glue:GetTrigger
                  - glue:CreateTrigger
                  - glue:DeleteTrigger
                  - glue:UpdateTrigger
                Resource:
                  - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:catalog
                  - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:database/
${ResourcePrefixLowercase}
                  - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:table/
${ResourcePrefixLowercase}/*
                  - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:userDefinedFunction/
${ResourcePrefixLowercase}/*
                  - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:crawler/
${ResourcePrefixLowercase}*
                  - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:trigger/
${ResourcePrefixLowercase}*
              - Effect: Allow
                Action:
                  - glue:CreateTable
                  - glue:UpdateTable
                  - glue:DeleteTable
                Resource:
                  - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:catalog
                  - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:database/
${ResourcePrefixLowercase}
                  - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:table/
${ResourcePrefixLowercase}/*
              - Effect: Allow
```

```
                Action:
                  - kms:CreateKey
                  - kms:GenerateDataKey
                Resource: '*'
              - Effect: Allow
                Action:
                  - s3:CreateBucket
                  - s3:DeleteBucket
                  - s3:GetObject
                Resource:
                  - !Sub ${BuildBucket.Arn}
                  - !Sub ${BuildBucket.Arn}/*
              - Effect: Allow
                Action:
                  - sagemaker:CreateNotebookInstanceLifecycleConfig
                  - sagemaker:DescribeNotebookInstanceLifecycleConfig
                  - sagemaker:UpdateNotebookInstanceLifecycleConfig
                  - sagemaker:DeleteNotebookInstanceLifecycleConfig
                  - sagemaker:CreateNotebookInstance
                  - sagemaker:UpdateNotebookInstance
                  - sagemaker:StartNotebookInstance
                  - sagemaker:DescribeNotebookInstance
                  - sagemaker:DeleteNotebookInstance
                  - sagemaker:StopNotebookInstance
                Resource:
                  - !Sub arn:aws:sagemaker:${AWS::Region}:${AWS::AccountId}:notebook-
instance-lifecycle-config/${ResourcePrefixLowercase}*
                  - !Sub arn:aws:sagemaker:${AWS::Region}:${AWS::AccountId}:notebook-
instance/${ResourcePrefixLowercase}*
    Metadata:
      cfn_nag:
        rules_to_suppress:
          - id: W11
            reason: Glue does not support resource-level permissions for these actions.
```

CodeBuild Role

`CodeBuildRole` defines the permissions needed for CodeBuild to run a code build job that builds Hail and copies the resources needed to Amazon S3 buckets. The CodeBuild job is run using the CodeBuild action type in the CodePipeline.

```
  CodeBuildRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - sts:AssumeRole
            Effect: Allow
            Principal:
              Service:
                - codebuild.amazonaws.com
      Path: /
      Policies:
        - PolicyName: CodeBuildAccess
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Resource:
```

```
                  - !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-group:/aws/
codebuild/${ResourcePrefix}*
                Action:
                  - logs:CreateLogGroup
                  - logs:CreateLogStream
                  - logs:PutLogEvents
              - Effect: Allow
                Action:
                  - s3:GetObject
                  - s3:GetObjectVersion
                  - s3:PutObject
                Resource: !Sub ${BuildBucket.Arn}/*
              - Effect: Allow
                Action:
                  - s3:ListBucket
                Resource:
                  - !Sub ${ResourcesBucket.Arn}
                  - !Sub ${DataLakeBucket.Arn}
              - Effect: Allow
                Action:
                  - s3:PutObject
                  - s3:PutObjectAcl
                Resource:
                  - !Sub ${ResourcesBucket.Arn}
                  - !Sub ${ResourcesBucket.Arn}/*
                  - !Sub ${DataLakeBucket.Arn}
                  - !Sub ${DataLakeBucket.Arn}/*
```

CodePipeline Role

`CodePipelineRole` defines the permissions needed for CodePipeline to run a deployment pipeline that builds Hail, copies resources to S3 buckets, and provisions the AWS Glue resources needed to process the solution data.

```
  CodePipelineRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Action:
              - sts:AssumeRole
            Effect: Allow
            Principal:
              Service:
                - codepipeline.amazonaws.com
      Path: /
      Policies:
        - PolicyName: CloudFormationAccess
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Action:
                  - cloudformation:CreateStack
                  - cloudformation:UpdateStack
                  - cloudformation:DescribeStacks
                Effect: Allow
                Resource: !Sub arn:aws:cloudformation:${AWS::Region}:
${AWS::AccountId}:stack/${ResourcePrefix}/*
        - PolicyName: IamAccess
          PolicyDocument:
            Version: 2012-10-17
```

```
          Statement:
            - Action:
                - iam:PassRole
              Effect: Allow
              Resource: !GetAtt CodeBuildRole.Arn
      - PolicyName: IamAccessCF
        PolicyDocument:
          Version: 2012-10-17
          Statement:
            - Action:
                - iam:PassRole
              Effect: Allow
              Resource: !Sub ${CloudFormationRole.Arn}
      - PolicyName: S3Access
        PolicyDocument:
          Version: 2012-10-17
          Statement:
            - Effect: Allow
              Action:
                - s3:GetObject
                - s3:GetObjectVersion
                - s3:GetBucketVersioning
                - s3:DeleteObject
                - s3:PutObject
              Resource:
                - !Sub ${BuildBucket.Arn}
                - !Sub ${BuildBucket.Arn}/*
      - PolicyName: CodeBuildAccess
        PolicyDocument:
          Version: 2012-10-17
          Statement:
            - Action:
                - codebuild:StartBuild
                - codebuild:BatchGetBuilds
              Effect: Allow
              Resource:
                - !GetAtt CodeBuildCopyResourcesProject.Arn
                - !GetAtt CodeBuildBuildHailProject.Arn
      - PolicyName: CodeCommitAccess
        PolicyDocument:
          Version: 2012-10-17
          Statement:
            - Effect: Allow
              Action:
                - codecommit:UploadArchive
                - codecommit:GetBranch
                - codecommit:GetCommit
                - codecommit:GetUploadArchiveStatus
              Resource: !GetAtt Repo.Arn
```

Source Event Role

`SourceEventRole` defines the permissions needed for an Amazon CloudWatch event to trigger the deployment pipeline.

```
SourceEventRole:
  Type: AWS::IAM::Role
  DependsOn: CodePipeline
  Description: IAM role to allow Amazon CloudWatch Events to trigger AWS CodePipeline
    execution
  Properties:
    AssumeRolePolicyDocument:
```

```
        Statement:
          - Action: sts:AssumeRole
            Effect: Allow
            Principal:
              Service:
                - events.amazonaws.com
            Sid: 1
      Policies:
        - PolicyName: CloudWatchEventPolicy
          PolicyDocument:
            Statement:
              - Action:
                  - codepipeline:StartPipelineExecution
                Effect: Allow
                Resource:
                  - !Sub arn:aws:codepipeline:${AWS::Region}:${AWS::AccountId}:
${CodePipeline}*
```

AWS Glue and Amazon SageMaker notebook permissions

Use IAM to manage access to the datasets and scripts in Amazon S3 using AWS Glue, and to define the permissions for your Amazon SageMaker Jupyter notebook instance. Adding new AWS Glue jobs and crawlers does not require any changes to the following roles or policies, as long as you add those resources with the `${Project}` prefix.

The following code examples demonstrate the IAM roles and supporting IAM policies defined in the `GenomicsAnalysisCode/code_cfn.yml` file, including `JobRole`, `CrawlerRole`, and `RunbookRole`.

Job Role

`JobRole` defines the permissions needed for AWS Glue to run Extract, Transform, and Load (ETL).

This role must be updated to `get` or `put` objects in additional S3 buckets in your AWS account or S3 buckets in other accounts.

```
  JobRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - glue.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/service-role/AWSGlueServiceRole
      Policies:
        - PolicyName: s3_access
          PolicyDocument:
            Version: 2012-10-17
            Statement:
```

```
            - Effect: Allow
              Action:
                - athena:StartQueryExecution
                - athena:GetQueryExecution
                - athena:GetQueryResults
              Resource:
                - !Sub arn:aws:athena:${AWS::Region}:${AWS::AccountId}*
            - Effect: Allow
              Action:
                - s3:GetObject
                - s3:ListBucket
              Resource:
                - !Sub arn:aws:s3:::${ResourcesBucket}
                - !Sub arn:aws:s3:::${ResourcesBucket}/*
            - Effect: Allow
              Action:
                - s3:PutObject
                - s3:GetObject
                - s3:ListBucket
                - s3:DeleteObject
              Resource:
                - !Sub arn:aws:s3:::${DataLakeBucket}
                - !Sub arn:aws:s3:::${DataLakeBucket}/*
      - PolicyName: kms_access
        PolicyDocument:
          Version: 2012-10-17
          Statement:
            - Effect: Allow
              Action:
                - kms:GenerateDataKey
                - kms:Decrypt
                - kms:Encrypt
              Resource:
                - !GetAtt DataCatalogEncryptionKey.Arn
```

Crawler Role

`CrawlerRole` defines the permissions needed to run an AWS Glue crawler on a dataset in an Amazon S3 bucket, infer the schema, and add or update a table in the AWS Glue data catalog.

```
CrawlerRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - glue.amazonaws.com
          Action:
            - sts:AssumeRole
    Path: /
    ManagedPolicyArns:
      - arn:aws:iam::aws:policy/service-role/AWSGlueServiceRole
    Policies:
      - PolicyName: s3_access
        PolicyDocument:
          Version: 2012-10-17
          Statement:
            - Effect: Allow
              Action:
```

```
                - s3:GetObject
                - s3:ListBucket
              Resource:
                - !Sub arn:aws:s3:::${DataLakeBucket}/*
        - PolicyName: kms_access
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                - kms:GenerateDataKey
                - kms:Decrypt
                - kms:Encrypt
                Resource:
                - !GetAtt DataCatalogEncryptionKey.Arn
```

Runbook Role

`RunbookRole` provides the permissions needed for the Amazon SageMaker Jupyter notebook instance to access the AWS Glue data catalog and use Amazon Athena to run queries against the solution's data lake.

```
  RunbookRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - sagemaker.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      Policies:
        - PolicyName: logs_access
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                - logs:CreateLogStream
                - logs:DescribeLogStreams
                - logs:CreateLogGroup
                - logs:PutLogEvents
                Resource:
                - !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-group:/aws/
sagemaker/*
                - !Sub arn:aws:logs:${AWS::Region}:${AWS::AccountId}:log-group:/aws/
sagemaker/*:log-stream:aws-glue-*
        - PolicyName: s3_access
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                - s3:ListBucket
                - s3:GetBucketLocation
                Resource:
```

```
                - !Sub arn:aws:s3:::${DataLakeBucket}
                - !Sub arn:aws:s3:::${ResourcesBucket}
              - Effect: Allow
                Action:
                - s3:GetObject
                - s3:GetObjectAcl
                - s3:PutObject
                - s3:DeleteObject
                Resource:
                - !Sub arn:aws:s3:::${DataLakeBucket}/*
              - Effect: Allow
                Action:
                - s3:GetObject
                Resource:
                - !Sub arn:aws:s3:::${ResourcesBucket}/*
        - PolicyName: glue_access
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                - glue:StartCrawler
                - glue:StartJobRun
                - glue:StartTrigger
                Resource:
                - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:crawler/
${ResourcePrefixLowercase}*
                - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:job/
${ResourcePrefixLowercase}*
                - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:trigger/
${ResourcePrefixLowercase}*
              - Effect: Allow
                Action:
                - kms:GenerateDataKey
                - kms:Decrypt
                - kms:Encrypt
                Resource:
                - !GetAtt DataCatalogEncryptionKey.Arn
        - PolicyName: glue_table_access
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                - glue:GetTable
                - glue:UpdateTable
                - glue:GetDatabase
                - glue:GetPartition
                - glue:GetPartitions
                - glue:GetDevEndpoint
                - glue:GetDevEndpoints
                - glue:UpdateDevEndpoint
                Resource:
                - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:catalog
                - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:database/
${ResourcePrefixLowercase}
                - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:table/
${ResourcePrefixLowercase}/*
                - !Sub arn:aws:glue:${AWS::Region}:${AWS::AccountId}:devEndpoint/*
        - PolicyName: athena_access
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                - athena:StartQueryExecution
```

```
                  - athena:GetQueryExecution
                  - athena:GetQueryResults
                Resource:
                  - !Sub arn:aws:athena:${AWS::Region}:${AWS::AccountId}:workgroup/primary
        - PolicyName: cfn_access
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - cloudformation:DescribeStacks
                Resource:
                  - !Sub arn:aws:cloudformation:${AWS::Region}:${AWS::AccountId}:stack/
${ResourcePrefix}*
        - PolicyName: kms_access
          PolicyDocument:
            Version: 2012-10-17
            Statement:
              - Effect: Allow
                Action:
                  - kms:GenerateDataKey
                  - kms:Decrypt
                  - kms:Encrypt
                Resource:
                  - !GetAtt DataCatalogEncryptionKey.Arn
```

Appendix C: Uninstall this solution

You can uninstall the Genomics Tertiary Analysis and Data Lakes Using AWS Glue and Amazon Athena solution using the AWS Management Console, the AWS Command Line Interface (AWS CLI), or manually.

> **Note:**
> Uninstalling this solution deletes the Amazon Simple Storage Service (Amazon S3) solution buckets and the data in those buckets; AWS CodeCommit repositories and the code in them; the AWS Glue jobs, crawlers, triggers, and data; and the Amazon SageMaker notebook instance.

Using the AWS Management console

1. Sign in to the AWS CloudFormation console.
2. Select the solution's installation stack that has a name ending in `-Setup`. All other solutions stacks will be deleted automatically.
3. Choose **Delete**.

Using AWS CLI

Determine whether AWS CLI is available in your environment. For installation instructions, see What Is the AWS Command Line Interface in the *AWS CLI User Guide*. After confirming the AWS CLI is available, run the following command.

```
$ aws cloudformation delete-stack --stack-name installation-stack-name
```

Uninstalling manually

To manually uninstall this solution, you must delete the related AWS CloudFormation stacks using the following procedure, in the specified order.

1. Delete all *<project-name>-*bucket* Amazon S3 bucket contents.
2. Delete the *<ProjectName>Code* stack.

   > **Important:**
   > Wait for deletion to complete successfully before proceeding.
3. Delete the *<ProjectName>Pipe* stack.

   > **Important:**
   > Wait for deletion to complete successfully before proceeding.
4. Delete the *<ProjectName>Zone* stack.

   > **Important:**
   > Wait for deletion to complete successfully before proceeding.
5. Delete the solution's installation stack.
6. Delete the *<ProjectName>Code* AWS CodeCommit repository.
7. Delete the *<ProjectName>Pipe* AWS CodeCommit repository.
8. Delete all *<project-name>-** image repositories in Amazon ECR.

Appendix D: Run the vcf-to-parquet AWS Glue job

This solution includes an example VCF to Apache Parquet Extract, Transform, and Load (ETL) AWS Glue job. You can run the job using either the AWS Command Line Interface (AWS CLI) or the AWS Glue console.

To start the job using the AWS CLI, run the following command:

```
aws glue start-job-run \
--name vcf-to-parquet -arguments \
BucketName=<data-lake-bucket>,KeyPrefix=variants/example.vcf
```

Note:
The data lake bucket name can be found as the *DataLakeBucket* output value in the *GenomicsAnalysisPipe* CloudFormation stack.

Use the following process to run the job in the AWS Glue console:

1. Sign in to the AWS Glue console.
2. Choose **Jobs** from the left navigation menu. On the **Jobs** page, select the name of the example jobs – `vcf-to-parquet`.
3. Choose **Actions** and select **Run Job**.
4. Expand **Security configuration, script libraries, and job parameters**.
5. Under **Job Parameters** observe the values for the `–input-path` and `–output-path` keys.

   Note:
   Do not change these values.
6. Select **Run Job**.

Appendix E: Run the variant AWS Glue crawler

This solution includes an example AWS Glue crawler to infer the schema of the variant call dataset in the solution data lake and add or update the *variants* table in the solution data catalog. You can run the crawler using either the AWS Command Line Interface (AWS CLI) or the AWS Glue console.

To start the crawler using the AWS CLI, run the following command:

```
aws glue start-crawler --name variants
```

Use the following process to run the crawler in the AWS Glue console.

1. Sign in to the AWS Glue console.
2. Choose **Crawlers** from the left navigation menu. On the **Crawlers** page, select the name of the example crawler —`variants`.
3. Choose **Actions** and select **Run Crawler**.

Appendix F: Run the clinvar-to-parquet AWS Glue job

This solution includes an example Clinvar to Parquet Extract, Transform, and Load (ETL) AWS Glue job. You can run the job using either the AWS Command Line Interface (AWS CLI) or the AWS Glue console.

To start the job using the AWS CLI, run the following command:

```
aws glue start-job-run \
--name clinvar-to-parquet -arguments \
BucketName=<data-lake-bucket>,KeyPrefix=annotation/clinvar.tsv'
```

Note:
The data lake bucket name can be found as the *DataLakeBucket* output value in the *GenomicsAnalysisPipe* CloudFormation stack.

Use the following process to run the job in the AWS Glue console:

1. Sign in to the AWS Glue console.
2. Choose **Jobs** from the left navigation menu. On the **Jobs** page, select the name of the example jobs – `clinvar-to-parquet`.
3. Choose **Actions** and select **Run Job**.
4. Expand **Security configuration, script libraries, and job parameters**.
5. Under **Job Parameters** observe the values for the `-input-path` and `-output-path` keys.

   Note:
   Do not change these values.
6. Select **Run Job**.

Appendix G: Run the clinvar AWS Glue crawler

This solution includes an example AWS ClinVar crawler to infer the schema of the ClinVar genomic annotation dataset in the solution data lake and add or update a *clinvar* table in solution data catalog. You can run the crawler using either the AWS Command Line Interface (AWS CLI) or the AWS Glue console.

To start the crawler using the AWS CLI, run the following command:

```
aws glue start-crawler --name clinvar
```

Use the following process to run the crawler in the AWS Glue console:

1. Sign in to the AWS Glue console.
2. Choose **Crawlers** from the left navigation menu. On the **Crawlers** page, select the name of the example crawler —`clinvar`.
3. Choose **Actions** and select **Run Crawler**.

Appendix H: Run the sample Glue crawler

This solution includes an example AWS Glue crawler to infer the schema of the VCF sample dataset in the solution data lake and add or update the *samples* table in the solution data catalog. You can run the crawler using either the AWS Command Line Interface (AWS CLI) or the AWS Glue console.

To start the crawler using the AWS CLI, run the following command:

```
aws glue start-crawler --name sample
```

Use the following process to run the crawler in the AWS Glue console.

1. Sign in to the AWS Glue console.
2. Choose **Crawlers** from the left navigation menu. On the **Crawlers** page, select the name of the example crawler —`sample`.
3. Choose **Actions** and select **Run Crawler**.

Appendix I: Collection of operational metrics

This solution includes an option to send anonymous operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When enabled, the following information is collected and sent to AWS:

- **Solution ID:** The AWS solution identifier.
- **Unique ID (UUID):** Randomly generated, unique identifier for each solution deployment.
- **Timestamp:** Data-collection timestamp.
- **Instance Data:** Count of the state and type of instances that are managed by the EC2 Scheduler in each AWS Region.

Example data:

```
Running: {t2.micro: 2}, {m3.large:2}

Stopped: {t2.large: 1}, {m3.xlarge:3}
```

Note that AWS owns the data gathered via this survey. Data collection will be subject to the AWS Privacy Policy. To opt out of this feature, complete one of the following tasks:

a) Modify the AWS CloudFormation template mapping section as follows:

```
  "Send" : {
"AnonymousUsage" : { "Data" : "Yes" }
},
```

to

```
    "Send" : {
"AnonymousUsage" : { "Data" : "No" }
},
```

OR

b) After the solution launches, find the serverless-image-handler function in the Lambda console and set the **SEND_ANONYMOUS_DATA** environment variable to **No**.

Source Code

You can visit our GitHub repository to download the templates and scripts for this solution, and to share your customizations with others.

Document Revisions

Date	Change
July 2020	*Initial release*
September 2020	*Removed the need for an administrator role from the encryption key; for more information, refer to the CHANGELOG.md file in the GitHub repository*