AWS Whitepaper

Genomics Data Transfer, Analytics, and Machine Learning using AWS Services

Genomics Data Transfer, Analytics, and Machine Learning using AWS Services: AWS Whitepaper

Table of Contents

Genomics Data Transfer, Analytics, and Machine Learning using AWS Services

Publication date: November 23, 2020 (Document Revisions)

Abstract

Precision medicine is “an emerging approach for disease treatment and prevention that takes into account individual variability in genes, environment, and lifestyle for each person,” according to the Precision Medicine Initiative. This approach allows doctors and researchers to identify and tailor treatments for groups of patients to improve patient outcomes. Precision medicine is powered by studying genomics data from hundreds of thousands of people refining the understanding of normal and disease diversity. The challenge is to turn the genomics data from many large-scale efforts like biobanks, research studies, and biopharma, into useful insights and patient-centric treatments in a rapid, reproducible, and cost-effective manner. The key to enabling scientific discovery is to combine different data streams, ensure global accessibility and availability, and allow high-performance data processing while keeping this sensitive data secure. “The responsible and secure sharing of genomic and health data is key to accelerating research and improving human health,” is a stated objective for the Global Alliance for Genomics and Health (GA4GH). This approach requires technical knowledge and ever-growing compute and storage resources. One of the ways that AWS is enabling this objective is to host many genomics datasets in the Registry of Open Data on AWS.

Raw genomics data is typically processed through a series of steps as part of a pipeline to transform into a form that is ready for analysis. Each step of the secondary analysis workflow could have different compute and memory requirements; some of the steps could be as simple as adding a set of annotations, or as computationally intensive as aligning raw reads to a reference genome. The requirements at this stage are to process the data in a cost effective, scalable, efficient, consistent, and reproducible manner across large datasets.

Once the data is processed, the next step is to query and mine genomic data for useful insights including discovering new biomarkers or drug targets. At this tertiary analysis stage, the goal is to prepare these large datasets so they can be queried easily and in an interactive manner to answer relevant scientific questions, or to use them to build complex machine learning models that can be utilized to analyze population or disease-specific datasets. The aim is accelerating the impact of genomics in the multi-scale and multi-modal data of precision medicine.

The genomics market is highly competitive, so having a development lifecycle that allows for fast adoption of new methods and technologies is critical. This paper answers some of the critical questions that many organizations that work with genomics data have, by showing how to build a next-generation sequencing (NGS) platform from instrument to interpretation using AWS services. We provide recommendations and reference architectures for developing the platform including: 1) transferring genomics data to the AWS Cloud and establishing data access patterns, 2) running secondary analysis workflows, 3) performing tertiary analysis with data lakes, and 4) performing tertiary analysis using machine learning. Solutions for three of the reference architectures in this paper are provided in AWS Solutions Implementations. These solutions leverage continuous delivery (CD), allowing you to develop the solution to fit your organizational need.

Are you Well-Architected?

The AWS Well-Architected Framework helps you understand the pros and cons of the decisions you make when building systems on AWS. Using the Framework allows you to learn architectural best practices for designing and operating reliable, secure, efficient, and cost-effective systems in the cloud.

In the Machine Learning Lens, we focus on how to design, deploy, and architect your machine learning workloads in the AWS Cloud. This lens adds to the best practices described in the Well-Architected Framework.

Introduction

When running genomics workloads in the Amazon Web Services (AWS) Cloud, how does an organization manage cost, optimize workload performance, and move fast with control? How does an organization secure sensitive information? What resources are available to help meet a team's compliance needs? How does an organization perform analytics using machine learning?

This paper answers these questions by showing how to build a next-generation sequencing (NGS) platform from instrument to interpretation using AWS services. We'll provide recommendations and reference architectures for developing the platform including: 1) transferring genomics data to the AWS Cloud and establishing data access patterns, 2) running secondary analysis workflows, 3) performing tertiary analysis with data lakes, and 4) performing tertiary analysis using machine learning.

The genomics market is highly competitive so having a development lifecycle that allows you to move fast with control is critical. Solutions for three of the reference architectures in this paper are provided in AWS Solutions Implementations. These solutions leverage continuous delivery (CD), allowing you to develop the solution to fit your organizational need.

Note

To access guidance providing an AWS CloudFormation template to automate the deployment of the secondary analysis solution in the AWS Cloud, see Genomics Secondary Analysis Using AWS Step Functions and AWS Batch.
To access an AWS Solution Implementation providing an AWS CloudFormation template to automate the deployment of the tertiary analysis and data lakes solution in the AWS Cloud, see the Guidance for Multi-Omics and Multi-Modal Data Integration and Analysis on AWS Implementation Guide.
To access guidance providing an AWS CloudFormation template to automate the deployment of the tertiary analysis and machine learning solution in the AWS Cloud, see Genomics Tertiary Analysis and Machine Learning using Amazon SageMaker.

A summary of the services used in this platform is shown in Table 1. You can learn about the compliance resources available to you in Compliance resources.

Table 1 – AWS services for data transfer, secondary analysis, and tertiary analyses

Data Transfer	Secondary Analysis	Tertiary Analysis
Data Access Patterns	**Secondary Analysis**	**Data Lakes**
AWS DataSync	AWS Step Functions	Amazon Athena
AWS Storage Gateway for files	AWS Batch	AWS Glue
Cost Optimization	**Monitor & Alert**	**Machine Learning**
AWS DataSync	Amazon CloudWatch	Amazon SageMaker
Amazon S3		
	DevOps	**DevOps**
	AWS CodeCommit	AWS CodeCommit
	AWS CodeBuild	AWS CodeBuild
	AWS CodePipeline	AWS CodePipeline

Transferring genomics data to the Cloud and establishing data access patterns using AWS DataSync and AWS Storage Gateway for files

Transferring genomics data to the AWS Cloud requires preparation in order to manage data transfer and storage cost, optimize data transfer performance, and manage the data lifecycle.

Recommendations

When you are ready to transfer your organization's genomics data to the AWS Cloud, consider the following recommendations to help optimize the data transfer process.

Use Amazon Simple Storage Service (Amazon S3) for genomics data storage—Genomics data is persisted in files by sequencers while genomics analysis tools take files as inputs and write files as outputs. This makes Amazon S3 a natural fit for storing genomics data, data lake analytics, and managing the data lifecycle.

Use the Amazon S3 Standard-Infrequent Access storage tier for transferring genomics data to Amazon S3—Typically, genomics Binary Base Call (BCL) files and Binary Alignment Map (BAM) files are accessed infrequently, perhaps a few times in a month. These files can be stored using the Amazon S3 Standard-Infrequent Access (S3 Standard-IA) tier to lower monthly cost when compared to storing the data using the S3 Standard access tier. Storing these file types using Amazon S3 Standard for 30 days before moving the data to S3 Standard-IA is more expensive than moving the data to infrequent access immediately. However, if you're organization accesses BCL and BAM files frequently, store those files in S3 Standard.

Manage genomics data lifecycle by archiving to a low-cost storage option such as Amazon S3 Glacier Deep Archive—Typically, genomics data is written to Amazon S3 using the Standard-Infrequent Access storage class before transitioning for long term storage to a lower cost storage option such as Amazon S3 Glacier Deep Archive. Even if you restore a sizeable amount of the data from archival storage to infrequent-access in a month, there is still a significant cost savings in archiving the data. Perform the storage class analysis and compute your expected cost before making any changes to your lifecycle policies in Amazon S3. You can learn more about archiving genomics data in Optimizing storage cost and data lifecycle management.

Stage genomics data on-premises first before uploading the data to Amazon S3—To keep genomics sequencers running 24/7, sequencer output files such as Binary Base Call (BCL) files are

written to on-premises storage first before uploading those files to the cloud. If there is a network outage, the sequencers can continue to run for as long as you have local storage available. Verify that you have enough local storage available to meet your organization's disaster recovery plans including Recovery Point Objective (RPO) and Recovery Time Objective (RTO). Data can always be written to external storage on-premises before being uploaded to the cloud.

Filter genomics sequencing data before uploading the data to the cloud—Consider eliminating log and thumbnail files that are not used for cloud-based analytics to minimize transfer cost, transfer time, and storage cost for a specified instrument run.

Use AWS DataSync to transfer data to Amazon S3—AWS DataSync makes it simple to transfer large amounts of data to Amazon S3 with minimal IT operational burden and optimal data transfer performance. DataSync eliminates or handles common tasks including scripting copy jobs, scheduling and monitoring transfers, validating data, and optimizing network utilization.

If file-system based access to data in Amazon S3 is required, use Amazon FSx or AWS Storage Gateway—Many research organizations use third-party tools, open-source tools, or their own tools to work with their research data. These tools often use file system-based access to data. Consider creating an Amazon Elastic Compute Cloud (Amazon EC2) instance to perform analytics on data in Amazon S3. If your applications require file-based access to Amazon S3, use Amazon FSx to provide a file-system that can be mounted on your Amazon EC2 instance. If your applications must run on-premises and require file-based access to Amazon S3, use File Gateway.

Reference architecture

Transferring your organization's genomics data to Amazon S3 using AWS DataSync starts with setting up your sequencing instruments to write data to a common folder on your on-premises storage system. Writing first to on-premises storage enables you to take advantage of the high availability (HA) built into your storage system and stage your data for processing before transferring to the cloud.

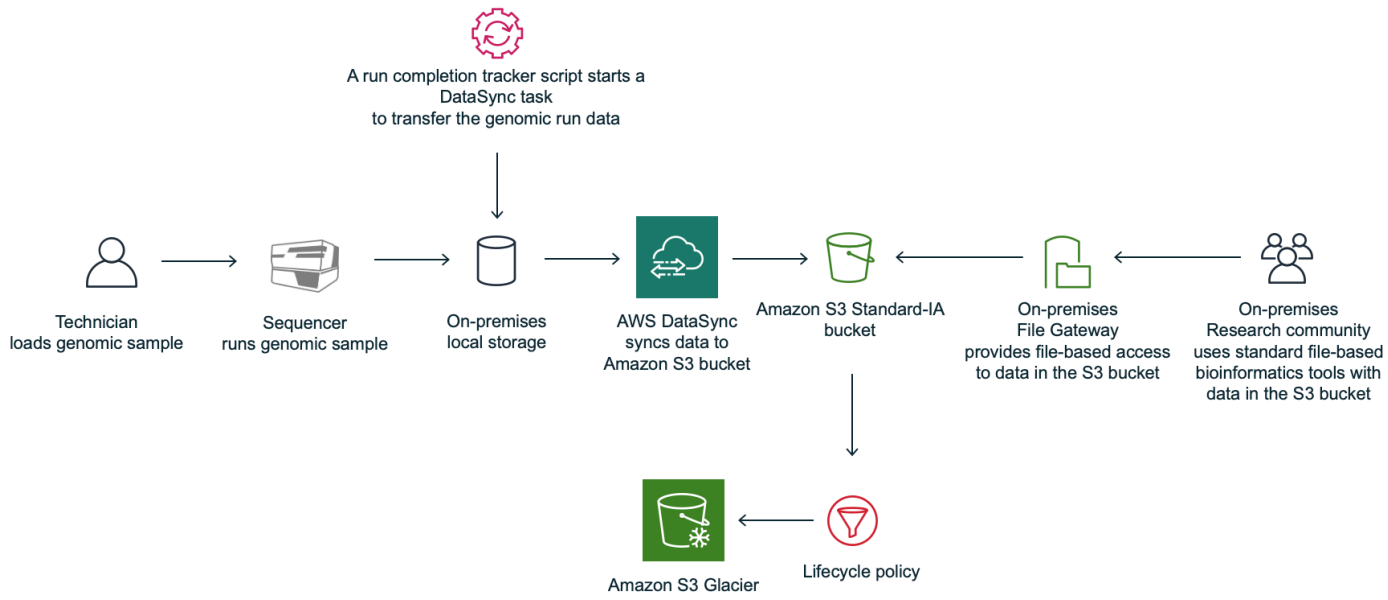

Figure 1: Process workflow using a run completion tracker script with AWS DataSync

Figure 1 shows the process workflow using a run completion tracker script with AWS DataSync:

1. A technician loads a genomic sample on a sequencer.
2. The genomic sample is sequenced and written to a landing folder that is stored in a local on-premises storage system.
3. An AWS DataSync sync task is preconfigured to sync the data from the parent directory of the landing folder on on-premises storage, to an Amazon S3 bucket.
4. A run completion tracker script starts a DataSync task run to transfer the run data to an Amazon S3 bucket. An inclusion filter can be used when running a DataSync task run, to only include a given run folder. Exclusion filters can be used to exclude files from data transfer. In addition, consider incorporating a zero-byte file as a flag when uploading the data. Technicians can then indicate when a run has passed a manual Quality Control (QC) check by placing an empty file in the data folder. Then, the run completion tracker script will only trigger a sync task if the success file is present.
5. On-premises researchers use existing bioinformatics tools with data in Amazon S3 via NFS or SMB using the File Gateway solution from AWS Storage Gateway.

In this scenario, a run completion tracker is used to monitor the staging folder and start DataSync task runs to transfer the data to Amazon S3. See Optimizing data transfer cost and performance for information about the run completion tracker pattern.

For information about getting started with DataSync, see Getting started with AWS DataSync. To learn more about optimizing storage cost, see Optimizing storage cost and data lifecycle management.

File-based access to Amazon S3

Amazon EC2 file-based access to Amazon S3 starts with setting up an FSx file system that can be mounted on your EC2 instance. Amazon FSx provides two file systems to choose from: Amazon FSx for Windows File Server for business applications and Amazon FSx for Lustre for compute-intensive workloads.

For information about setting up Amazon FSx for Windows File Server, see Create Your File System in the *Amazon FSx for Windows File Server User Guide*. For information about setting up Amazon FSx for Lustre *User Guide*, see Create Your Amazon FSx for Lustre File System in the *Amazon FSx for Lustre Users Guide*.

On-premises file-based access to Amazon S3 starts with setting up a file gateway on-premises to present objects stored in Amazon S3 as NFS or SMB on-premises.

AWS Storage Gateway connects an on-premises software appliance with cloud-based storage to provide seamless integration with data security features between your on-premises IT environment and the AWS storage infrastructure. You can use the service to store data in the AWS Cloud for scalable and cost-effective storage that helps maintain data security. A file gateway supports a file interface into Amazon S3 and combines a service and a virtual software appliance. By using this combination, you can store and retrieve objects in Amazon S3 using industry-standard file protocols such as NFS and SMB. For information about setting up a file gateway, see Creating a file gateway in the *AWS Storage Gateway User Guide*.

> **Note**
>
> You will incur egress charges when transferring data to on-premises from Amazon S3.

Running secondary analysis workflows using AWS Step Functions and AWS Batch

Running secondary analysis workflows to perform sequence alignment, variant calling, quality control (QC), annotation, and custom processing can be done in AWS using native AWS services. We'll provide recommendations and a reference architecture for running secondary analysis workflows in AWS using AWS Step Functions and AWS Batch.

Recommendations

When running secondary analysis workloads in the AWS Cloud, consider the following recommendations to optimally run secondary analysis.

Use AWS Batch to run tasks in your genomics workflows—Most secondary analysis tasks are perfectly parallel, meaning that those tasks can be run independently and often in parallel. Provisioning resources on-demand for tasks in AWS Batch is more cost-effective and optimizes performance better than using a traditional High-Performance Computing (HPC) environment.

Use AWS Step Functions to orchestrate tasks in your secondary analysis workflows— Keep task execution separate from task orchestration so that purpose-built solutions are used for each activity and tools and workflows can be deployed independently. This approach limits the impact of change which minimizes risk. AWS Step Functions is serverless which minimizes operational burden.

Package tools in Docker containers and use standard Amazon Machine Images (AMIs)—Package tools in Docker containers so they are portable and you can take advantage of serverless container solutions to orchestrate your container execution. Using standard AMIs removes the operational burden of maintaining machine images.

Package tools independent of workflows in their own Docker container—Package tools independently to right-size the compute for each tool, which can optimize performance and minimize cost when running each tool container. Multiple workflows can also use the same tool containers.

Treat configuration as code for secondary analysis tools and workflows—Fully automate the build and deployment of secondary analysis tools and workflows. Automation empowers your teams to move quickly while maintaining control, providing a repeatable development process for advancing your genomics solution into production.

Use Amazon Elastic Compute Cloud (Amazon EC2) Spot Instances to optimize for cost—Amazon EC2 Spot Instances offers significant savings when you run jobs. You can fall back to on-demand instances if spot instances are not available.

Reference architecture

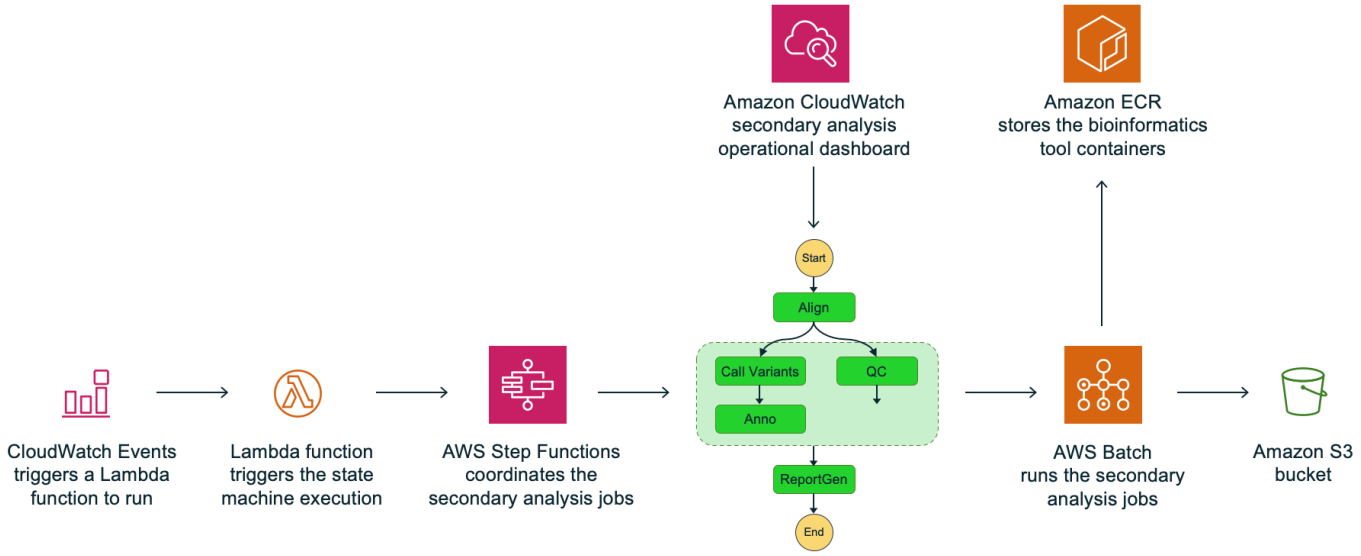

Figure 2: Secondary analysis reference architecture using AWS Step Functions and AWS Batch

Figure 2 shows the reference architecture for the secondary analysis workflow using AWS Step Functions and AWS Batch.

1. A CloudWatch event triggers an AWS Lambda function to start an AWS Step Functions secondary analysis state machine.
2. Step Functions submits jobs to AWS Batch to run in a Spot Instance compute environment. The Docker images for the tools are stored in private Amazon Elastic Container Registry (Amazon ECR) repositories in the customer's account. The job output files are written to an Amazon S3 bucket, including the Variant Call File (VCF) and Binary Alignment Map (BAM) file. AWS Step Functions manages the execution of AWS Batch jobs for alignment, variant calling, QC, annotation, and custom processing steps.

Running secondary analysis using AWS Step Functions and AWS Batch starts by triggering a Step Functions state machine to run. For example, you can use either the AWS Command Line Interface (AWS CLI) or an AWS API client. To fully automate the process, you can set up an Amazon CloudWatch rule to run the state machine when genomic sample FASTQ files are put in an Amazon

S3 bucket. The rule could start the state machine execution directly or the event could be put in an Amazon Simple Queue Service (Amazon SQS) queue that is polled by an AWS Lambda function that starts the state machine execution. Using a Lambda function to start a state machine makes it easy to add quality checks or routing logic, that is, to check for missing files or route to a particular secondary analysis workflow.

You can pass parameters when running a Step Functions state machine and use them for each job in your secondary analysis workflow. Use a convention to read and write objects to Amazon S3 so that you can compute, up front, the input and output paths for each tool when you execute the state machine.

Once Step Functions submits a job to an AWS Batch queue, the job is scheduled by AWS Batch to run on an instance in a compute environment associated with the queue. Compute environments are used in the order they are listed in the compute environments list in the queue. The job will be scheduled to run on an instance that matches the resource requirements specified in the job definition, such as compute and memory required. Using EC2 Spot Instances and allowing AWS Batch to choose the optimal instance type can help optimize compute costs by increasing the pool of instances types to choose from in EC2 Spot Instances.

To learn more about optimizing secondary analysis compute cost, see Optimizing secondary analysis compute cost.

From time to time a secondary analysis job will fail. AWS Step Functions has support for job retries and try catch logic to handle job failures with custom code. To learn more about handling secondary analysis task errors or failures, see Handling secondary analysis task errors and workflow failures.

Create an operational dashboard to monitor the status of your secondary analysis workflow. AWS Step Functions and AWS Batch are both integrated with Amazon CloudWatch so creating a secondary analysis dashboard in Amazon CloudWatch is easy to do. To learn more about creating a secondary analysis operational dashboard in Amazon CloudWatch, see Monitoring secondary analysis workflow status, cost, and performance.

Note

To access an AWS Solutions Implementation providing an AWS CloudFormation template to automate the deployment of the solution in the AWS Cloud, see the Genomics Secondary Analysis Using AWS Step Functions and AWS Batch Implementation Guide.

Performing tertiary analysis with data lakes using AWS Glue and Amazon Athena

Genomic tertiary analysis can be performed on data in an Amazon S3 data lake using AWS Glue, Amazon Athena, and Amazon SageMaker Jupyter notebooks.

Recommendations

When building and operating a genomics data lake in AWS, consider the following recommendations to optimize data lake operations, performance, and cost.

Use AWS Glue extract, transform, and load (ETL) jobs, crawlers, and triggers to build your data workflows—AWS Glue is a fully managed ETL service that makes it easy for you to prepare and load data for analytics. You can create Spark jobs to transform data, Python jobs to perform PHI and virus scans, crawlers to catalog the data, and workflows to orchestrate data ingestion, all within the same service.

Use AWS Glue Python jobs to integrate with external services—Use Python shells in AWS Glue to execute tasks in workflows that require callouts to external services such as running a virus scan or a personal health information (PHI) scan.

Use AWS Glue Spark ETL to transform data—AWS Glue Spark jobs make it easy to run a complex data processing job across a cluster of instances using Apache Spark.

Promote data across S3 buckets, multiple accounts are not necessary—Use different Amazon S3 buckets to implement different access controls and auditing mechanisms as data is promoted through your data ingestion pipeline, such as, quarantine, pre-curated, and curated. Segregating data across accounts is not necessary.

For interactive queries, use Amazon Athena or Amazon Redshift— Query data residing in Amazon S3 using either Amazon Redshift or Athena. Amazon Redshift efficiently queries and retrieves structured and semi-structured data from files in Amazon S3 by leveraging Redshift Spectrum Request Accelerator to improve performance. Athena is a serverless engine for querying data directly in Amazon S3. Users who already have Amazon Redshift can extend their analytical queries to Amazon S3 by pointing to their AWS AWS Glue Data Catalog. Users who are looking for a fast, serverless analytics query engine for data on Amazon S3 can use Athena. Many customers use both services to meet diverse use cases.

For queries that require low latency such as dashboards, use Amazon Redshift—Amazon Redshift is a large-scale data warehouse solution ideal for big data, and low latency queries, such as dashboard queries.

Use partitions and the AWS Glue Data Catalog for data changes instead of creating new databases—Use table partitions in an AWS Glue Data Catalog to accommodate multiple versions of a dataset with minimal overhead and operational burden.

Design data access around least privileges and provide data governance using AWS Lake Formation—Limit data lake users to select permissions only. Service accounts used for ETL may have create/update permissions for tables.

Use a tenant/year/month/day partitioning scheme in Amazon S3 to support multiple data providers—Data producers provide recurring delivery of datasets that need to be ingested, processed, and made available for query. Partitioning the incoming datasets by tenant/year/month/day allows you to maintain versions of datasets, lifecycle the data over time, and re-ingest older datasets, if necessary.

Use data lifecycle management in Amazon S3 to lifecycle data and restore, if needed—Manage your data lake objects for cost effective storage throughout their lifecycle. Archive data when it is no longer being used and consider Amazon S3 Intelligent-Tiering if the access patterns are unpredictable.

Convert your datasets to parquet format to optimize query performance—Parquet is a compressed, columnar data format optimized for big data queries. Analytics services that support columnar format only need to read accessed columns which greatly reduces I/O and speed of data processing.

Treat configuration as code for jobs and workflows—Fully automate the building and deployment of ETL jobs and workflows to more easily move your genomics data into production. Automation provides control and a repeatable development process for handling your genomics data.

Reference architecture

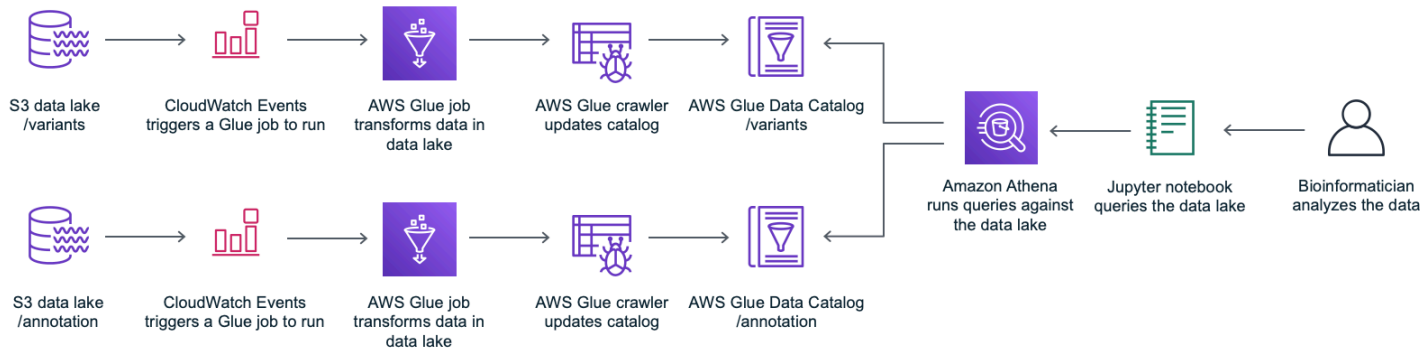

Figure 3: Tertiary analysis with data lakes reference architecture

1. A CloudWatch Events triggers an ingestion workflow for variant or annotation files into the Amazon S3 genomics data lake.
2. A bioinformatician uses a Jupyter notebook to query the data in the data lake using Amazon Athena with the PyAthena python driver. Queries can also be performed using the Amazon Athena console, AWS CLI, or an API.

Processing and ingesting data into your Amazon S3 genomics data lake starts with triggering the data ingestion workflows to run in AWS Glue. Workflow runs are triggered through the AWS Glue console, the AWS CLI, or using an API within a Jupyter notebook. You can use a Glue ETL job to transform annotation datasets like Clinvar from TSV format to Parquet format and write the Parquet files to a data lake bucket.

You can convert VCF to Parquet in an Apache Spark ETL job by using open-source frameworks like Hail to read the VCF into a Spark data frame and then write the data as Parquet to your data lake bucket. Use AWS Glue crawlers to crawl the data lake dataset files, infer their schema, and create or update a table in your AWS Glue data catalog, making the dataset available for query with Amazon Redshift or Amazon Athena.

To run AWS Glue jobs and crawlers in a workflow, use AWS Glue triggers to stitch together workflows, then start the trigger. To run queries on Amazon Athena, use the Amazon Athena console, AWS CLI, or an API. You can also run Athena queries from within Jupyter notebooks using the PyAthena python package that can be installed using pip.

Optimizing data lake query cost and performance are important considerations when working with large amounts of genomics data. To learn more about optimizing data lake query performance

with Amazon Athena, see Optimizing the performance of data lake queries. To learn more about optimizing data lake query cost with Amazon Athena, see Optimizing the cost of data lake queries.

Note

To access an AWS Solutions Implementation providing an AWS CloudFormation template to automate the deployment of the solution in the AWS Cloud, see the Guidance for Multi-Omics and Multi-Modal Data Integration and Analysis on AWS.

Performing tertiary analysis with machine learning using Amazon SageMaker

Creating machine learning (ML) training sets and training models, and generating ML predictions using genomics data can be done using AWS Glue and Amazon SageMaker. We'll provide recommendations and a reference architecture for creating training sets, training models, and generating predictions using AWS Glue, Amazon SageMaker, and Amazon SageMaker Jupyter notebooks.

Recommendations

When creating ML training sets, training ML models, and generating predictions, consider the following recommendations to optimize performance and cost.

Use AWS Glue extract, transform, and load (ETL) service to build your training sets—AWS Glue is a fully managed ETL service that makes it easy for you to extract data from object files in Amazon Simple Storage Service (Amazon S3), transform the dataset adding features needed for training machine learning models, and write the resulting data to an Amazon S3 bucket.

Use Amazon SageMaker Autopilot to quickly generate model generation pipelines—Amazon SageMaker Autopilot automatically trains and tunes the best machine learning models for classification or regression, based on your data while allowing you to maintain full control and visibility. Sagemaker Autopilot analyses the data and produces an execution plan that includes feature engineering, algorithm identification, hyperparameter optimization to select the best model, and deploys the model to an endpoint. The process of generating the models are transparent and the execution plan is made available to the users via automatically generated notebooks that users can review and modify.

Use Amazon SageMaker hosting services to host your machine learning models—Amazon SageMaker provides model hosting services for model deployment and an HTTPS endpoint where your machine learning model is available to provide inferences.

Use Amazon SageMaker notebook instances to create notebooks, generate predictions, and test your deployed machine learning models—Use Jupyter notebooks in your notebook instance to prepare and process data, write code to train models, deploy models to Amazon SageMaker hosting, and test or validate your models.

Reference architecture

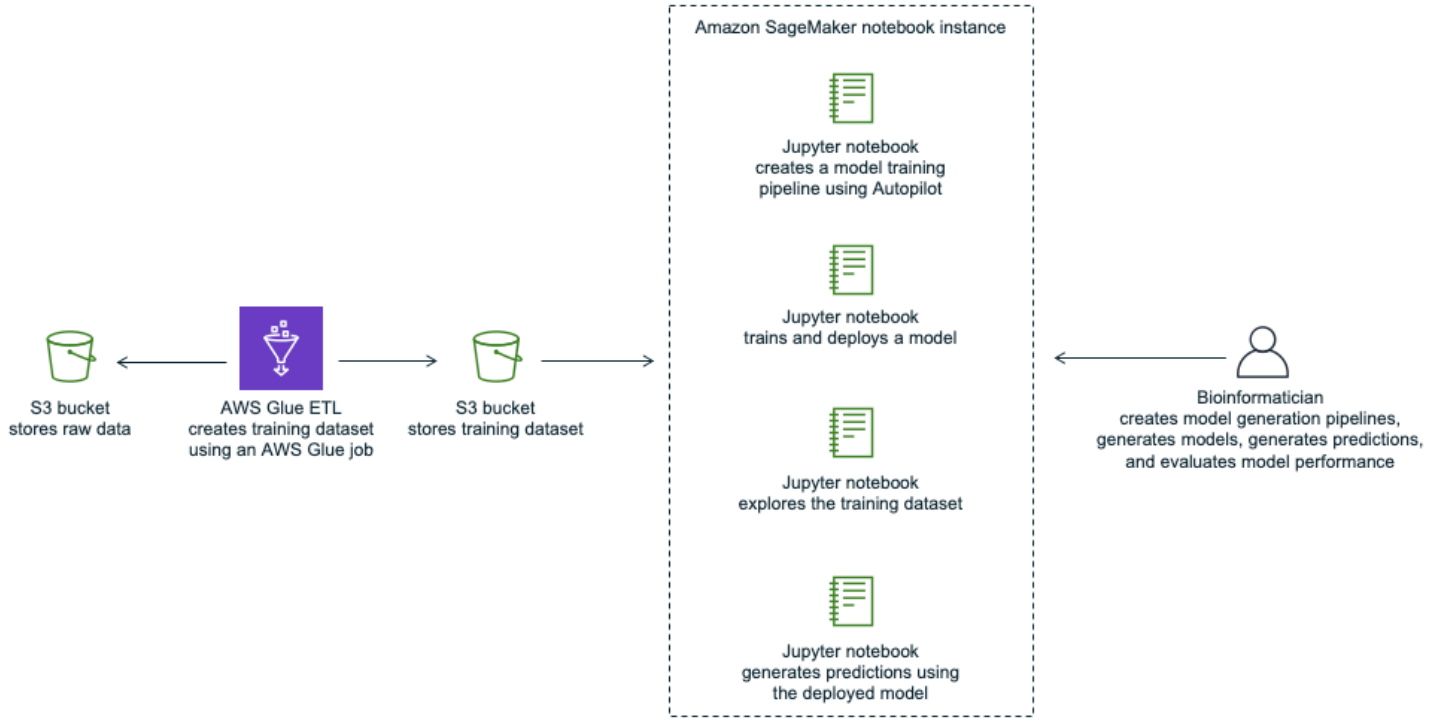

Figure 4: Tertiary analysis with machine learning using Amazon SageMaker reference architecture

An AWS Glue job is used to create a machine learning training set. Jupyter notebooks are used to generate machine learning model generation pipelines, explore the datasets, generate predictions, and interpret the results.

An AWS Glue job ingests data used for training a machine learning model, adds features used for training, and writes the resulting dataset to an Amazon S3 bucket. A Jupyter notebook is run to generate a machine learning pipeline using Amazon SageMaker Autopilot, generating two notebooks. These two notebooks capture the data structures and statistics, feature engineering steps, algorithm selection, hyperparameter tuning steps, and the deployment of the best performing model. Users have a choice to either use the plan recommended by SageMaker Autopilot or modify the generated notebook to influence the final results. A prediction notebook is used to generate predictions and evaluate model performance using a test dataset.

Note

Note: To access an AWS Solutions Implementation providing an AWS CloudFormation template to automate the deployment of the solution in the AWS Cloud, see the Genomics Tertiary Analysis and Machine Learning Using Amazon SageMaker.

Conclusion

In this paper we have detailed an approach to build a human next-generation sequencing (NGS) platform from instrument to interpretation using AWS services. We focused on common concerns expressed by technical leaders in companies building genomics report pipelines or data lakes in the AWS Cloud—capturing and optimizing cost; securing sensitive information, compliance, and operational excellence; and performing analytics using machine learning. To learn more about the accompanying AWS Solutions Implementations for this paper, visit the home page for the following solutions.

Note

To access guidance providing an AWS CloudFormation template to automate the deployment of the secondary analysis solution in the AWS Cloud, see Genomics Secondary Analysis Using AWS Step Functions and AWS Batch.

To access an AWS Solution Implementation providing an AWS CloudFormation template to automate the deployment of the tertiary analysis and data lakes solution in the AWS Cloud, see the Guidance for Multi-Omics and Multi-Modal Data Integration and Analysis on AWS Implementation Guide.

To access guidance providing an AWS CloudFormation template to automate the deployment of the tertiary analysis and machine learning solution in the AWS Cloud, see Genomics Tertiary Analysis and Machine Learning using Amazon SageMaker.

Appendix A: Genomics report pipeline reference architecture

The following shows an example end-to-end genomics report pipeline architecture using the reference architectures described in this paper.

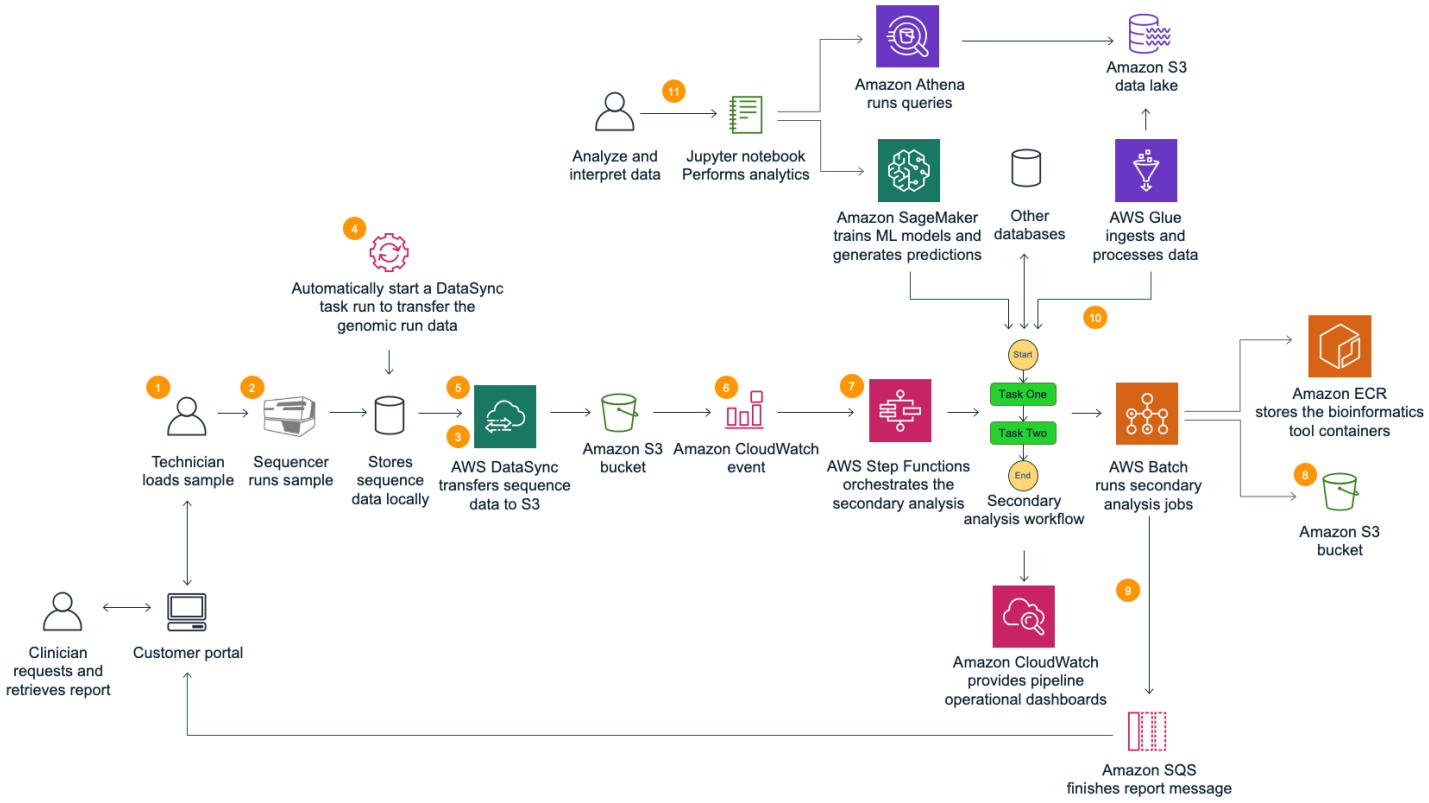

Figure 5: Genomics report pipeline reference architecture

1. A technician loads a genomic sample on a sequencer.
2. The genomic sample is sequenced and written to a landing folder that is stored in a local on-premises storage system.
3. An AWS DataSync sync task is preconfigured to sync the data from the parent directory of the landing folder on on-premises storage, to a bucket in Amazon S3.
4. A run completion tracker script running as a cron job, starts a DataSync task run to transfer the run data to an Amazon S3 bucket. An inclusion filter can be used when running a DataSync task run, to only include a given run folder. Exclusion filters can be used to exclude files from data transfer. In addition, consider Incorporating a zero-bite file as a flag when uploading the data. Technicians can then indicate when a run has passed a manual QA check by placing an empty

file in the data folder. Then, the watcher application will only trigger a sync task if the success file is present.

5. DataSync transfers the data to Amazon S3.
6. An Amazon CloudWatch Events is raised that uses an Amazon CloudWatch rule to launch an AWS Step Functions state machine.
7. The state machine orchestrates secondary analysis and report generation tools which run in Docker containers using AWS Batch.
8. Amazon S3 is used to store intermediate files for the state machine execution jobs.
9. Optionally, the last tool in the state machine execution workflow uploads the report to the Laboratory Information Management System (LIMS).
10. An additional step is added to run an AWS Glue workflow to convert the VCF to Apache Parquet, write the Parquet files to a data lake bucket in Amazon S3 and update the AWS Glue Data Catalog.
11. A bioinformatic scientist works with the data in the Amazon S3 data lake using Amazon Athena via a Jupyter notebook, Amazon Athena console, AWS CLI, or an API. Jupyter notebooks can be launched from either Amazon SageMaker or AWS Glue. You can also use Amazon SageMaker to train machine learning models or do inference using data in your data lake.

Appendix B: Research data lake ingestion pipeline reference architecture

The following reference architecture shows an example end-to-end research data lake data ingestion AWS Glue pipeline using the data lake reference architectures described in this paper. The AWS Glue workflows enable you to construct data pipelines using extract, transform, and load (ETL) functions, crawlers, and triggers.

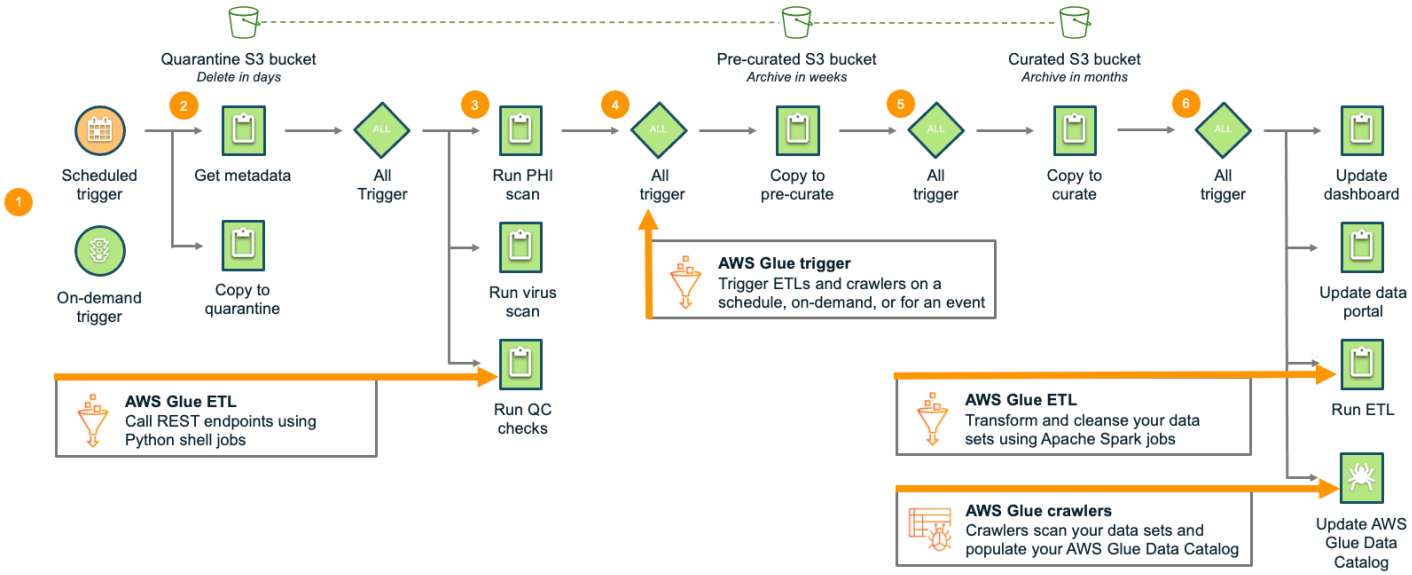

Figure 6: Data pipeline using AWS Glue workflows

1. An AWS Glue trigger is run either on-demand or on a schedule.
2. The dataset is first copied to a quarantine bucket to be scanned for personal health information (PHI) or viruses.
3. A trigger then launches Glue Python shell jobs that make REST calls to Personal Health Information (PHI) and virus scanning services that scan the dataset which resides in the Amazon S3 quarantine bucket. A quality control (QC) process is run to confirm that the data is in the agreed upon format and schema.
4. If the dataset passes the scans and QC validation, the data is copied to a pre-curated bucket where the dataset resides without changes.
5. The dataset is then copied to a curated bucket where it is reorganized and filtered based on the study or research project.

6. A trigger then launches jobs to update a research project dashboard, a research portal database, extract transform, and load (ETL) processes to transform the data and write it to the data lake in Apache Parquet format. AWS Glue crawlers crawl the data, infer the schema, and update the AWS Glue meta data catalog. The data is made available for query using big data query engines such as Amazon Athena. Data governance is managed with Identity and Access Management (IAM) or with AWS Lake Formation.

Appendix C: Genomics data transfer, analytics, and machine learning reference architecture

The following genomics reference architecture describes the AWS services used in this paper to ingest, store, archive, analyze, prepare, and interpret genomics data to gain insights and make predictions.

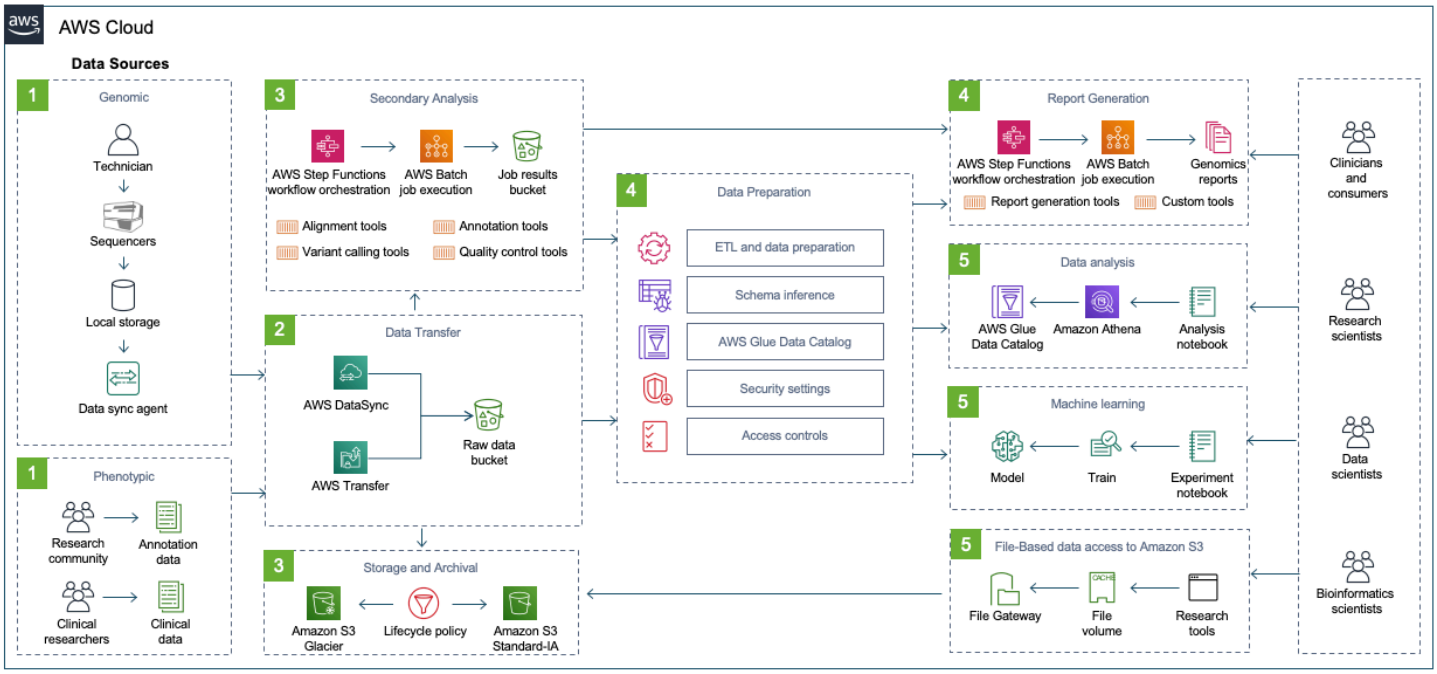

Figure 7: Genomics data transfer, analytics, and machine learning reference architecture

Appendix D: Compliance resources

Genomics data is generally considered the most private of personal data. From a regulatory perspective it is certainly considered Protected Health Information (PHI) or a special class of Personal Data.

Privacy, reliability, and security must be kept in mind at every stage, from data creation, collection and processing, to storage and transfer. Customers need to have a solid understanding of regulatory privacy requirements, for example, GINA, HIPAA, the EU's GDPR or local equivalents, and comply with them at every stage of data handling.

Although customers are ultimately accountable for their own regulatory compliance, AWS does take steps to help.

In the case of Health Insurance Portability and Accountability Act of 1996 (HIPAA) in the US, cloud service providers (CSPs) such as AWS are considered business associates. For customers subject to this regulation, the Business Associate Agreement (BAA) is an AWS contract that is required under HIPAA rules to ensure that AWS appropriately safeguards protected health information (PHI). Customers who execute an AWS BAA may use any AWS service in an account designated as a HIPAA Account, but they may only process, store, and transmit PHI using the HIPAA-eligible services defined in the AWS BAA. For the latest list of HIPAA-eligible AWS services, see the HIPAA Eligible Services Reference webpage. Throughout this whitepaper we have used HIPAA-eligible services.

Additional relevant US regulations include:

- Genetic Information Nondiscrimination Act of 2008 (GINA). GINA was used to modify the HIPAA Privacy Rule to strengthen the privacy protections for genetic information by implementing section 105 of Title I of the Genetic Information Nondiscrimination Act of 2008 (GINA) [1].
- Health Information Technology for Economic and Clinical Health Act (HITECH)
- The Health Information Trust Alliance (HITRUST) Common Security Framework (CSF)

For more information about AWS' compliance programs for HIPAA, HITECH and HITRUST, refer to the HIPAA compliance program webpage.

As for EU regulations, AWS acts as both a data processor and a data controller under the GDPR which clearly states in recital 34 that genetics data is considered personal data. Under the shared responsibility model, AWS is responsible for securing the underlying infrastructure that supports

the cloud, and customers and APN partners, acting either as data controllers or data processors, are responsible for any personal data they put on the cloud.

We can confirm that all AWS services can be used in compliance with the GDPR. This means that, in addition to benefiting from all of the measures that AWS already takes to maintain services security, customers can deploy AWS services as a key part of their GDPR compliance plans. For more details, see our GDPR services readiness announcement in the AWS Security Blog.

For further information about AWS' compliance program for GDPR, refer to the GDPR compliance program webpage.

Depending on where the genomics data is used in the customers business, from drug discovery to clinical trial patient recruitment, GxP regulations may apply. In particular, Title 21 CFR part 11 in the US or Eudralex volume 4 Annex 11 in the EU.

For further information about AWS' compliance program for GxP, refer to the GxP compliance program webpage.

For further information about any of the numerous AWS compliance programs, refer to the AWS Compliance Programs webpage.

Appendix E: Optimizing data transfer, cost, and performance

When transferring genomics data to long term storage, filter and validate the data before data transfer to Amazon S3. You can create a run completion tracker that starts an AWS DataSync task run to transfer the run data to an Amazon S3 bucket. The run completion tracker script can be run as a cron job. An inclusion filter can be used when running a DataSync task run, to only include a given run folder. Exclusion filters can be used to exclude files from data transfer. In addition, consider Incorporating a zero-bite file as a flag when uploading the data. Technicians can then indicate when a run has passed a manual QA check by placing an empty file in the data folder. Then, the run completion tracker will only trigger a sync task if the success file is present.

This type of application is useful, for example, to monitor a HiSEQ X flow cell directory which may contain up to 1.4 million files or more. A percentage of those files, approximately 450,000, may not be needed for subsequent analytics. These extraneous files are in the form of thumbnails and can be excluded from the run data uploaded.

Laboratory personnel typically perform a manual and, often, an automated quality control (QC) check to verify that the Genomics Sequencer functioned correctly and that the data meets laboratory quality standards. For example, in whole genome sequencing using a HiSEQ X system, a minimum number of cycles must be reached for the data to be of sufficient quality. This data resides in a text file and can be checked using a basic script. Also, laboratory personnel must verify that the data meets laboratory quality standards using quality control software that can check the performance of the instrument, reagents, and consumables. After passing QC, the lab may place a ‘qc’ file in the run data directory indicating that it has passed QC. Only data folders that have the 'qc' file will trigger a DataSync task execution for upload. By adding the QC file to the flow cell folder, unnecessary information including test runs, titration runs, and qualification runs (for Good Laboratory Practices (GLP)/Clinical Laboratory Improvement Amendment (CLIA)/College of American Pathologist (CAP) certified laboratories) can be excluded from upload. This prevents unwanted data from being uploaded to Amazon S3 and is an important step to keep storage costs down.

Appendix F: Optimizing storage cost and data lifecycle management

Filtering the data files to be transferred to Amazon Simple Storage Service (Amazon S3) minimizes storage cost. In a production laboratory, unnecessary data can account for up to 10% of the total data produced. Consider optimizing storage by writing instrument run data to an Amazon S3 bucket configured for Infrequent Access (IA) then archive the data to Glacier Deep Archive.

Enable data lifecycle management to optimize storage costs. Identify your Amazon S3 storage access patterns to optimally configure your S3 bucket lifecycle policy. Use Amazon S3 analytics *storage class analysis* to analyze your storage access patterns. After storage class analysis observes the access patterns of a filtered set of data over a period of time, you can use the analysis results to improve your lifecycle policies. For genomics data, the key metric is identifying the amount of data retrieved for an observation period of at least 30 days. Knowing the amount of data that was retrieved for a given observation period helps you decide the length of time to keep data in infrequent access before archiving.

For example, a customer performs genomics sequencing on-premises where each run contains six samples for a given scientific study. The run produces 600GB of Binary Base Call (BCL) file data that is transferred, using AWS DataSync, from on-premises to a bucket in Amazon S3, where objects are written using the Amazon S3 Standard-Infrequent Access storage tier. Demultiplexing and secondary analysis are run producing six 100GB FastQ files and one 1GB Variant Call File (VCF), all stored in the same S3 bucket. The BCL files, FastQ files and VCF files are tarred and archived to Glacier Deep Archive after 90 days. A copy of the VCF files remain in the infrequent access tier for twelve months since VCF files are frequently used for a year. After a year, they are deleted from the S3 bucket. Upon request, an entire study is restored from Glacier to Infrequent Access, making the data available in the original S3 bucket, and through the Storage Gateway. To request a restore of a study, the customer attempts to retrieve the data through the Storage Gateway which triggers a restore action. An email is sent to the person that requested the restore when the data is available in the Infrequent Access bucket and through the Storage Gateway. You can learn more about automating data restore from AWS Glacier in Automate restore of archived objects through AWS Storage Gateway.

Appendix G: Optimizing secondary analysis compute cost

To optimize compute cost, first consider the resource requirements for a given tool that runs in your secondary analysis workflow. There are a variety of tools available to help with Amazon Elastic Compute Cloud (Amazon EC2) instance right-sizing for a given workload in AWS. The basic idea is to gather sufficient data about your tool performance over a period of time to capture the workload and business peak resource utilization. For more information about EC2 instance right sizing, see Identifying Opportunities to Right Size in the *Right Sizing: Provisioning Instances to Match Workloads* whitepaper.

By setting the instance type attribute in your AWS Batch compute environment to optimal, AWS Batch will dynamically provision the optimal quantity and type of compute resources (for example, CPU or memory optimized instances) based on the volume and specific resource requirements of the batch jobs submitted. You can also specify a specific set of instance types that are optimal for your given tools. Also, if a Spot Instances type is unavailable, AWS Batch can use another instance type defined for that job. Keep in mind that AWS Batch can pick large instance types and schedule a number of jobs on that same instance, so it is not limited to one tool for one instance. It can be cost-effective to bin pack jobs on a larger instance than placing one job in one instance with the lowest cost.

AWS Batch has built-in failover support from Spot Instances compute environments to an On-Demand compute environment, allowing organizations to take advantage of lower compute cost on Spot Instances when it's available. If you specify the maximum price when configuring your AWS Batch compute environment that is using Spot Instances, AWS Batch will only schedule jobs in that compute environment when Spot Instance pricing is below your maximum price (as a percent of the On-Demand price). For more information about AWS Batch compute environment, see Creating a Compute Environment in the *AWS Batch User Guide*. If you choose to set up a queue with two compute environments—Spot Instances and On-Demand Instances—jobs will be submitted to the On-Demand queue when the Spot Instances price is above the maximum price or if Spot instances are not available. By default, the maximum price is 100% of EC2 On-Demand prices.

Appendix H: Handling secondary analysis task errors and workflow failures

If your Amazon Elastic Compute Cloud (Amazon EC2) Spot Instances are interrupted, jobs running on that instance will fail. Amazon EC2 terminates, stops, or hibernates your Spot Instance when the price exceeds the maximum price for your request or capacity is no longer available. If retries are enabled for the given task in the AWS Step Functions state machine and there are more retries left, the job will be resubmitted to AWS Batch. If a Spot Instance is available, it will be used to run the task. If there are no Spot Instances available and you have an On-Demand compute environment next in the compute environments list for the queue, AWS Batch will provision an On-Demand Instance to run the task. Verify that you have configured failover from a Spot Instance compute environment to an On-Demand Instance compute environment for a given job queue. Also verify you have configured retries on tasks in your Step Functions state machine.

You can also specify a try/catch block to handle complex retry scenarios or to execute custom code such as submitting a ticket to an issue tracking system with custom code in the catch block. For more information about how AWS Step Functions handle errors, see Error Handling in Step Functions in the *AWS Step Functions Developer Guide*.

Many organizations build in the ability to resume a secondary analysis state machine execution when it was interrupted due to a job failure. The idea is to skip jobs that have already produced their output files, resuming the state machine execution at the job that was next when resources were interrupted. This is commonly referred to as *checkpointing*. One approach is to add a gate clause to the beginning of your tool wrapper code to check for the job outputs and return if they already exist. If the output already exists, the tool immediately returns, allowing failed state machine executions to fast-forward to the point of failure.

Appendix I: Monitoring secondary analysis workflow status, cost, and performance

Monitoring and alerting are an important part of operational excellence. Both AWS Step Functions and AWS Batch produce a variety of operational metrics and events to monitor secondary analysis status and performance and alert you if there is an issue requiring remediation.

First, establish a baseline of activity for your secondary analysis workflow and then set alarms that will notify personnel if secondary analysis behavior is outside of established baselines. At a minimum, monitor the Step Functions metrics—`ActivitiesStarted`, `ActivitiesTimedOut`, `ExecutionsStarted`, and `ExecutionsTimedOut`. The AWS Solutions Implementation provides an Amazon CloudWatch dashboard that plots these metrics over time and provides counter widgets for each one. This dashboard can be used for monitoring your secondary analysis workflows and for establishing a workflow performance baseline. Step Functions also provides events for `ExecutionFailed`, `ExecutionTimedOut`, and `ExecutionAborted` so that you can create CloudWatch alarms that notify personnel when either of these events take place. CloudWatch alarms can be configured to send an email or publish to a Simple Notification Service (Amazon SNS) topic, notifying subscribers of the issue.

To monitor cost over time, tag your tasks in your Step Functions state machine so that you can track them in AWS Cost Explorer. AWS Cost Explorer helps you track costs over time and compute an average cost per sample metric for a given secondary analysis workflow type, by tag. As of the date of this publication, Spot Instances costs cannot be associated with a given container instance in a compute environment. Job costs can be averaged in AWS Cost Explorer by tag.

Appendix J: Scaling secondary analysis

To scale secondary analysis in your account using AWS Step Functions and AWS Batch, there are a few optimizations that can be made and service limits that may need to be increased.

You should provide the Amazon EC2 instances in your AWS Batch compute environments with access to reference genome files either through a shared file system like Amazon Elastic File System (EFS) or Amazon FSx for Lustre, or mount an Elastic Block Store (EBS) volume on each instance that contains the reference genome files. This avoids downloading the reference files for each job which will decrease job runtime, limit data transfer volume, and limit put or get requests to Amazon S3.

Configure your VPC and AWS Batch compute environments to use multiple Availability Zones (AZs) so you have access to a bigger pool of instances. Configure your compute environment to use instance types or instance families that are optimal for your workload, for example, the Sentieon aligner and variant caller both run optimally with 40 cores and 50 GB of RAM and are compute bound so the c5.12xlarge and c5.24xlarge would provide optimal performance and maximum utilization of the compute instances. Try to include as many instance types as possible to increase the pool of instances that can be used in your compute environments.

You may need to increase your Amazon EC2 instance limits, Elastic Block Store (EBS) volumes limit, and move to a shared file system like Amazon Elastic File System (EFS) or Amazon FSx for Lustre to avoid request rate limits in Amazon S3.

Appendix K: Optimizing the performance of data lake queries

For the solution implementation using tertiary analysis and data lakes, we optimized Amazon Athena performance based on the recommendations in Top 10 Performance Tuning Tips for Amazon Athena. We partitioned the variant data on the sample ID field, sorted each sample on location, and converted the data to Apache Parquet format. Partitioning in this way allows cohorts to be built optimally based on sample IDs. New samples can be ingested efficiently into the data lake without recomputing the data lake dataset. The annotation data sources are also written in Apache Parquet format to optimize for performance. If you need to query by location (chromosome, position, reference, alternate–CPRA), either create a sample ID to location ID lookup table in a database like Amazon DynamoDB or create a duplicate of the data partitioned on location.

Appendix L: Optimizing the cost of data lake queries

The biggest challenge with cost optimizing a data lake in Amazon Simple Storage Service (Amazon S3) is understanding the data lake access patterns to lifecycle the data and optimize for cost. With data lakes, the access patterns are often irregular, making it difficult to capture a common set of access patterns and lifecycle the data. Amazon S3 Intelligent-Tiering solves this problem.

For a small monitoring and automation fee, S3 Intelligent-Tiering monitors access patterns and moves objects that have not been accessed for 30 consecutive days to the Infrequent Access tier. If the data is accessed later, it is automatically moved back to the frequent access tier. Enabling S3 Intelligent-Tiering is an easy way to cost optimize your data lake.

Contributors

Contributors to this document include:

- Ryan Ulaszek, Senior Solutions Architect, AWS (Principal Author)
- Bryan Coon, Senior Manager Solutions Architecture, AWS
- Ujjwal Ratan, Principal AI/ML Solutions Architect, AWS
- Michael Miller, Senior Startup Solutions Architect, AWS
- Patrick Combes, Head of Technology, HCLS, AWS
- Lee Pang, Principal Bioinformatics Architect, AWS
- Nihir Chadderwala, AI/ML Solutions Architect, AWS
- Lisa McFerrin, Principal Specialist World Wide Genomics, AWS
- Deven Atnoor, Principal HCLS Consultant, AWS
- Aaron Friedman, Principal Startup Solutions Architect, AWS
- Ian Sutcliffe, Senior Solutions Architect, AWS
- Scott Franks, Storage Services Solutions Architect, AWS
- Ariella Sasson, Senior Solutions Architect, AWS
- Sujaya Srinivasan, Senior Solutions Architect, AWS
- Juan Yu, Senior Analytics Specialist Solution Architect, AWS
- Roy Hasson, Senior Analytics Specialist, AWS

Document history

To be notified about updates to this whitepaper, subscribe to the RSS feed.

Change	Description	Date
Whitepaper updated	Bug fixes and numerous minor changes throughout.	April 1, 2022
Initial publication	Whitepaper first published.	November 23, 2020

Note

To subscribe to RSS updates, you must have an RSS plug-in enabled for the browser that you are using.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.