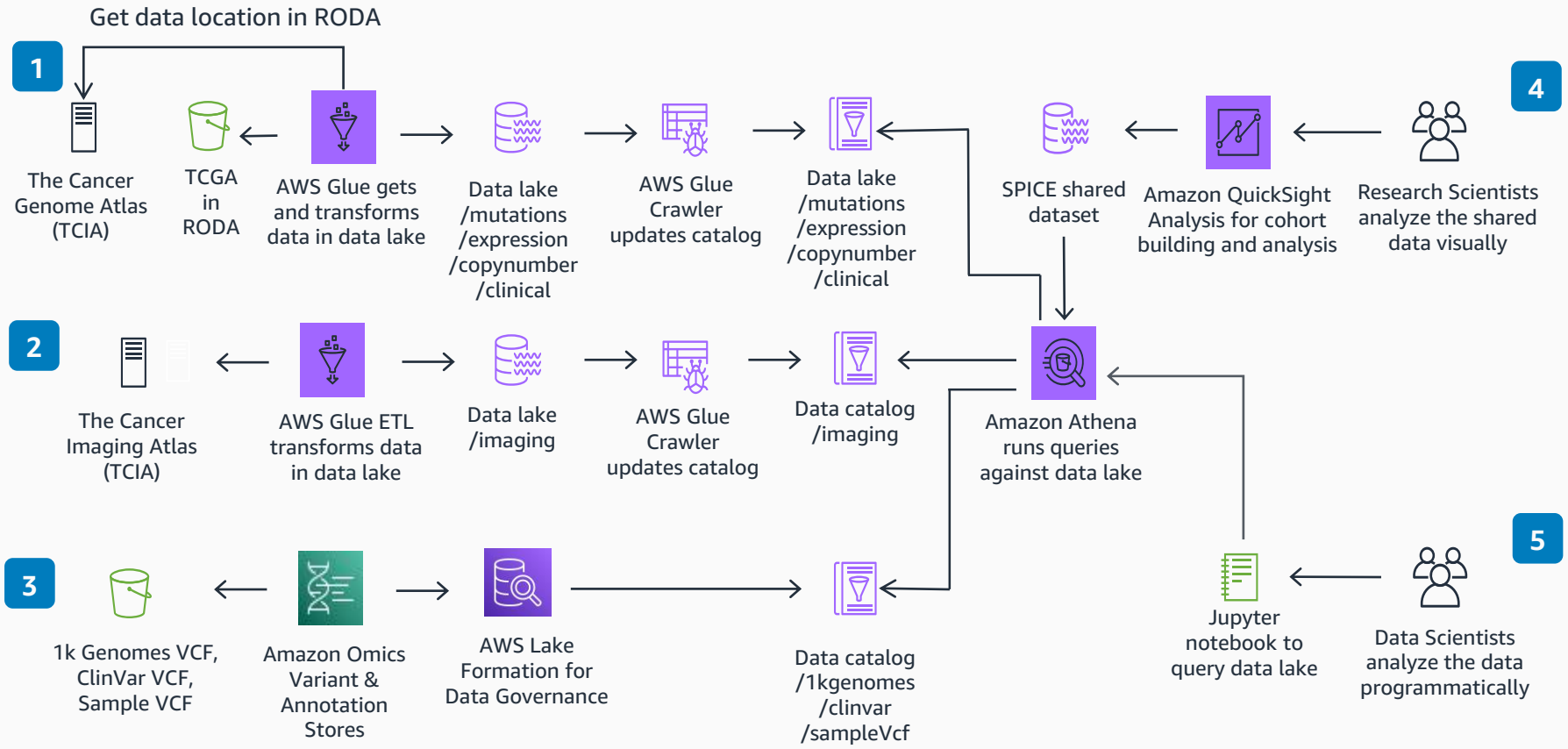
Guidance for Multi-Omics and Multi-Modal Data Integration and Analysis on AWS – Architecture

Prepare genomic, clinical, mutation, expression and imaging data for large-scale analysis and query against a data lake.



1. Ingest, format, and catalog data from The Cancer Genome Archive (TCGA). The raw data is pulled from the Registry of Open Data on AWS (RODA) through the TCGA API. The data is transformed in an **AWS Glue** extract, transform, and load (ETL) job and cataloged by an **AWS Glue** Crawler. This makes the data available for query in **Athena**.

2. Data from The Cancer Imaging Atlas (TCIA) is ingested, formatted, and catalogued. The raw data is pulled from RODA through the TCIA API. The data is transformed in an **AWS Glue** ETL job and cataloged by an **AWS Glue** Crawler. Image locations can be queried and displayed using **SageMaker** Notebooks.

3. VCF data from the One Thousand Genomes project, a sample VCF, and ClinVar Annotation VCF is ingested into **Amazon Omics** Variant and Annotation Stores and made available as tables in **Lake Formation**.

4. Research scientists analyze the multi-modal data through a visual interface in **QuickSight**. The data is cached in a SPICE (Super-fast, Parallel, In-memory Calculation Engine) database, optimizing query performance.

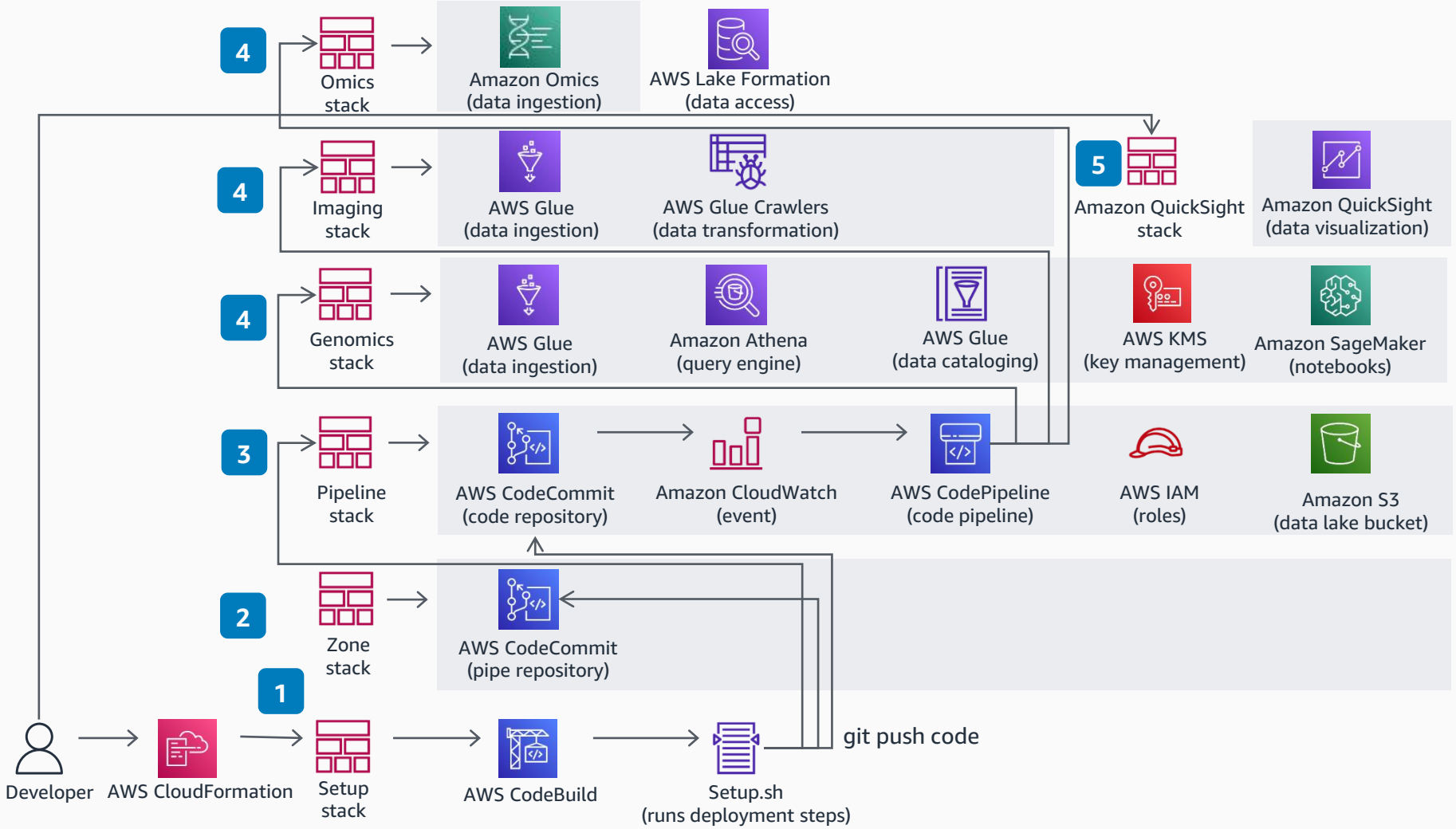
5. Data Scientists analyze the data with code using Jupyter notebooks provided through **SageMaker** Notebook environments.

Reviewed for technical accuracy December 14, 2022

AWS Reference Architecture

Guidance for Multi-Omics and Multi-Modal Data Integration and Analysis on AWS – CI/CD

Prepare genomic, clinical, mutation, expression and imaging data for large-scale analysis and query against a data lake.



1. Create an **AWS CodeBuild** project containing the setup.sh script. This script creates the remaining **AWS CloudFormation** stacks, code repositories, and code.

2. The landing zone (zone) stack creates the **AWS 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 **AWS 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 (genomics, imaging and omics) **CloudFormation** stacks. After the **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; a **CodeBuild** project for building code artifacts; a **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. An **Amazon Omics** Reference Store, Variant Store, and Annotation Store is provisioned, and a sample variant call file (VCF), a subset 1000 genomes VCF, and ClinVar Annotation VCF is ingested for analysis. Using **AWS Lake Formation**, a Data lake Admin can enable access of data in Omics Variant and Annotation Stores using **Amazon Athena** and **SageMaker**. An Amazon Omics Reference Store, Variant Store, and Annotation Store is provisioned to store publicly available variant and annotation data and make it available for query and analysis.

5. The imaging stack creates a hyperlink to a **CloudFormation** quick start, which can be launched to deploy the **Amazon QuickSight** stack. The **QuickSight** stack creates **Identity and Access Management (IAM)** and **QuickSight** resources necessary to interactively explore the multi-omics dataset.

Reviewed for technical accuracy December 14, 2022

AWS Reference Architecture