**Project: Building a Scalable, Serverless Genomics Analysis Pipeline on AWS**

**## 1. The Motivation: Solving the Data Deluge in Bioinformatics**

In the world of life sciences, the challenge is no longer about generating DNA data, but about **processing it**. A single human genome sequence can be over 100 gigabytes of raw data. For researchers to analyze this data to find links to diseases like cancer, they need massive computational power, which is both expensive and complex to manage.

The **main reason** for this project is to build a cost-effective, automated, and scalable platform that allows researchers to run complex genomics analysis pipelines without needing to be cloud infrastructure experts.

**# 2. The Solution: A High-Level Overview**

This project builds an end-to-end, serverless pipeline on AWS that automates a standard genomics workflow. A user can provide a raw DNA data file, and the system will automatically process it through a multi-step analysis and produce a final report of genetic variants.

**## 3. Project Scope**

* **IN SCOPE:**
  + Automate a 3-step bioinformatics pipeline (Quality Control -> Alignment -> Variant Calling).
  + Use Docker to containerize all scientific tools.
  + Leverage AWS Batch for on-demand, serverless compute.
  + Use AWS Step Functions to orchestrate the entire workflow.
* **OUT OF SCOPE:**
  + Building a web-based user interface (UI).
  + Creating new bioinformatics algorithms.
  + Performing scientific interpretation of the final results.

**## 4. The Architecture & Tech Stack**

The workflow is built on a foundation of powerful AWS services:

* **Docker:** To package our scientific tools (FastQC, BWA, GATK) into portable "workers."
* **Amazon ECR:** Our private repository to store the Docker container images.
* **AWS Batch:** The core "compute engine" that automatically provisions servers and runs our jobs at scale.
* **AWS Step Functions:** The "brain" or "factory manager" that orchestrates the entire multi-step workflow.
* **Amazon S3:** Used as the data source for raw DNA files and as the destination for final results.
* **Amazon CloudWatch:** For centralized logging and monitoring.
* **AWS IAM:** To manage all service permissions securely.
* **Terraform:** To define and manage our entire infrastructure as code (IaC).

**## 5. Project Roadmap: From Foundation to Final Product**

This project is broken down into four distinct phases:

**Phase 1: Foundational Engine (COMPLETED)**

* **Goal:** Build and validate the core, general-purpose compute platform.
* **Milestones Achieved:**
  1. **Containerized** the first tool (FastQC) into a Docker image.
  2. **Pushed** the image to a private **Amazon ECR** repository.
  3. **Configured** all components of **AWS Batch** (Compute Environment, Job Queue, Job Definition).
  4. **Successfully executed** a test job, proving the end-to-end plumbing works.

**Phase 2: Workflow Orchestration (NEXT STEPS)**

* **Goal:** Introduce the "brain" of the pipeline to manage the sequence of jobs.
* **Key Tasks:**
  1. Design and build a state machine in **AWS Step Functions**.
  2. Integrate the state machine with AWS Batch to trigger our existing fastqc-job-def.
  3. Implement data handling to pass an input DNA file from S3 to the Batch job.
  4. Configure the job to save its results to a dedicated S3 bucket.

**Phase 3: Building the Complete 3-Step Pipeline**

* **Goal:** Expand our single-step pipeline to the full, three-step scientific workflow.
* **Key Tasks:**
  1. **Containerize BWA and GATK:** Create Dockerfiles for the remaining two tools and push their images to ECR.
  2. **Create New Job Definitions:** Add new blueprints in AWS Batch for the BWA and GATK workers.
  3. **Chain the Workflow:** Update the Step Functions state machine to run the three jobs in the correct sequence (FastQC -> BWA -> GATK), passing the output of one step as the input to the next.

**Phase 4: Professionalization (Infrastructure as Code)**

* **Goal:** Make the entire project professional, reproducible, and easy to deploy.
* **Key Tasks:**
  1. **Write Terraform Code:** Convert all the AWS resources we created manually into Terraform (.tf) files.
  2. **Automate Deployment:** Ensure the entire infrastructure can be created with terraform apply and destroyed with terraform destroy.
  3. **Finalize Documentation:** Thoroughly document the project in the README.md, including architecture, setup, and usage instructions.

**## 6. Final Outcome**

The final result will be a fully automated, production-grade platform for running complex bioinformatics analysis on AWS. This project demonstrates a deep, practical understanding of advanced cloud architecture, serverless computing, containerization, and DevOps (IaC) principles.