GA4GH Projects for GSoC 2025

For more information about GSoC 2025, please visit https://summerofcode.withgoogle.com/about. Note the following while filling out the below form.

- Create a copy of the project information template from below and fill out project information.
- All the fields in the form (table) are necessary for submission.
- Project's difficulty is defined by the following matrix:
  - ~90 hours is small project
  - ~175 hours is medium project
  - ~350 hours is large project
- Provide as much information as possible to help contributors decide and submit proposals.
- For any queries/clarifications, please contact Jimmy Payyappilly: Jimmy Payyappilly

Table of Contents

Projects

Please share the details of the projects below

Project Name: Leverage Cat-VRS in a clinical decision support pipeline

Workstream	Genomic Knowledge Standards
Mentor names/emails	• Bob Dolin, MD (bdolin@elimu.io) • Daniel Puthawala, PhD (Daniel.Puthawala@nationwidechildrens.org) • Akash Rampersad (akashjrampersad@gmail.com)
Difficulty	MEDIUM to LARGE
Requirements	Required: • Python Required to be learned as part of the project: • Working understanding of the GA4GH Cat-VRS specification • Working understanding of the HL7 CQL specification Other technologies student will be exposed to: • HL7 FHIR Genomics Reporting Implementation Guide ('FHIR Genomics') • https://github.com/ga4gh/cat-vrs-python
Workload	175h - 350h (approximately 20h/week)

Project Description

Project Overview

This project seeks to test the use of GA4GH knowledge standards in a clinical decision support pipeline.

Representative knowledge, encoded per the GA4GH Cat-VRS specification, will be obtained from public knowledge bases ClinVar and CIViC. That knowledge will be translated into HL7 CQL format and loaded into a CQL execution engine.

Sample (publicly anonymized or synthetic) patient data will be created. Clinical and genetic data will be FHIR-formatted and placed on a public FHIR server.

The CQL execution engine will then be fed FHIR-formatted genomic data and will attempt to automatically associate that data with knowledge. The CQL engine will generate FHIR-formatted annotated genomic data, where those annotations are derived from the public knowledge sources.

Stretch goals for the project are to encapsulate this pipeline in a prototype API, and to surface the annotated data via a simple user-facing app.

Project Milestones

- Obtain representative knowledge, encoded per the GA4GH Categorical Variation Representation specification (Cat-VRS).
- Develop Cat-VRS to CQL translator
- Obtain or create sample patient data, encoded per the FHIR Genomics specification.
- Develop execution framework using a public CQL engine
- Stretch goals
    - Develop a prototype API
    - Surface annotated genome data via a user-facing app

Project Timeline

GSOC master timeline (coding is ~June/July/Aug, ~13 weeks)

- May 8 - Jun 1: Accepted students will begin to familiarize themself with Cat-VRS and CQL. Students will meet with mentors to learn more about the scope and details of the project.
- Jun 2 - Sep 1: Weekly scrum call, attacking each Project Milestone in order. Roughly:
    - 2 weeks: Obtain or create representative knowledge, encoded per the GA4GH Cat-VRS specification.
    - 6 weeks: Develop Cat-VRS to CQL translator.
    - 1 week: Obtain or create sample patient data, encoded per the FHIR Genomics specification.
    - 2 weeks: Develop execution framework using a public CQL engine.
    - 2 weeks: Stretch goals
        - Develop a prototype API.
        - Surface annotated genome data via a user-facing app.

Project Requirements

Project requirements are listed above. Baseline Python knowledge is required. For the other items, we will give preference to those students who express an interest in self-learning and who agree to use the May 8 - June 1 period to familiarize themselves with relevant technologies and standards. It is important to stress that students must be self-motivated for self-learning (with mentorship guidance).

Project Name: Extension of the GA4GH-SDK

Workstream	Cloud Workstream
Mentor names/emails	• **Pavel Nikonorov (pavel@genxt.network)** • Secondary mentors TBA
Difficulty	LARGE
Requirements	Required: • Rust • Python Required to be learned as part of the project: • GA4GH API specifications & implementations Other technologies students will be exposed to: • Attested-TLS (Confidential Computing; TEEs)
Workload	175h/350h

Project Description

Provide detailed information on the project for contributors to understand the scope of the project, tasks and expectations. Sub-sections included could be project overview, **milestones**, **timelines**, and impact.

The **GA4GH-SDK** is an evolving Rust-based toolkit designed to streamline interactions with GA4GH API standards such as **ServiceInfo**, **DRS**, **TRS**, **TES**, and **WES**. Building on prior efforts, this 2025 GSoC project aims to **extend and enhance** the GA4GH-SDK by focusing on DRS support and cross-API compatibility by using DRS identifiers with TES Task submission.

Current Development Status: The project already has ServiceInfo and TES support implemented, as well as the extension subsystem allowing for loadable extensions such as GENXT Confido, an implementation of the Attested-TLS (aTLS) protocol powered by Confidential Computing (Trusted Execution Environments). The project also has CLI implemented on top of the SDK that provides UI/UX similar to conventional cloud consoles. New contributors should first familiarise themselves with the existing development status, features, architecture, and codebase. This stage is critical for identifying opportunities for further development and ensuring alignment with the project's strategic goals.

Project Milestones:

A good proposal could cover these or a similar set of milestones. The number of milestones included in a proposal will define the size of the project (medium or large).

- **Initial Setup and Planning:** Understand GA4GH API standards and identify libraries/tools for potential leverage or contribution. Set up ELIXIR TESK in an aTLS-enabled environment (provided by GENXT) and test existing SDK/CLI functionality (see the SDK documentation and/or consult with the mentors). Decide on the number of additional API clients to implement within the following priority: DRS (required), TRS, ServiceRegistry, and WES. For each client ensure having a backend service instance for development purposes (use the GA4GH Starter Kit and/or consult with the mentors).
- **Core Library Development:** implement one or more API clients as decided during the planning stage, provide documentation, test coverage, and CI/CD support.
- **CLI Tool Development:** for each new API client implement CLI support.
- **Python bindings:** Enhance the lib's accessibility by extending Python bindings for new functionality.
- **Documentation and Community Feedback:** Produce comprehensive documentation with an **end-to-end demonstrator** of working with DRS instances and TES Tasks with DRS identifiers, as well as potentially other API clients implemented during the project. Engage with the GA4GH community for feedback and ongoing development.
- **CLI Tool Test Coverage (stretch Goal):** Implement comprehensive automated testing to validate CLI commands, parameters, and outputs through both unit and integration tests, ensuring reliable performance and easy maintenance.

Contributors are encouraged to outline a timeline in their application, taking into account the described milestones and potential stretch goals.

Impact: By joining this project, you will substantially contribute towards a future where **AI-driven** biomedical research aligns with the **GA4GH vision of interoperability and secure data sharing** for the benefit of human health. Through advanced protocols like **attested TLS** for hardware-enforced privacy policy protection across federated environments, your contributions will safeguard sensitive genomic and medical data while enabling groundbreaking discoveries in a globally connected life sciences ecosystem. In doing so, you'll empower researchers and data holders to collaborate, ensuring **Responsible AI** remains at the core of next-generation biomedical research.

Project Name: Secure Data Handling with Crypt4GH, Confidential Computing, DRS and TES

Workstream	Cloud Workstream
Mentor names/emails	• **Joris Vankerschaver** • Secondary mentors TBA
Difficulty	Hard
Requirements	Microservices, Python. Familiarity with confidential computing is a plus.
Workload	175h

Project Description

Overview:

In this project, you will combine GA4GH technologies for accessing data (DRS) and running computational tasks (TES), with standards for encrypting data at rest (Crypt4GH) and handling data securely during processing in remote VM-based Trusted Execution Environments (Confidential Computing) verified with Attested-TLS (aTLS) protocol. More specifically, you will build a proof-of-concept solution that:

1. DRS server provides Crypt4GH-encrypted data upon successful aTLS verification of the TES instance.
2. Given a DRS ID, the TES Task executor retrieves a Crypt4GH-encrypted data file from file storage (optionally, upon successful aTLS verification of the DRS instance).
3. Executes a computational task on the data using a Task Execution Service (TES) backend, running within a Confidential Computing environment. The environment should decrypt the data prior to processing it, and encrypt the outputs of the task.
4. Returns a DRS ID to the encrypted outputs that the client can use to securely receive the data.

A successful prototype will demonstrate that data can securely be handled at rest and in transit (via Crypt4GH) and can be handled securely during processing (via Confidential Computing), and can form the basis for an eventual production-ready solution that puts data security and privacy first and foremost.

As this project ties together several different GA4GH standards and underlying implementations, while also bridging the gap with non-GA4GH standards for data security, it is highly recommended to start from established services, such as DRS-Filer for DRS (for which a proof-of-concept integration with Crypt4GH already exists), TESK for TES, and the GENXT Confido library for Attested-TLS.

For this project, good Python programming skills and familiarity with microservices are desirable. Experience with confidential computing and cryptographic standards such as Crypt4GH is not required, and can be acquired over the course of the project.

Expected outcomes:

1. A working proof-of-concept implementation that can be demonstrated to developers and other interested parties.
2. Design docs and specs contributing to the planning for a production-ready solution.
3. Wherever possible, contributions to existing open source projects to facilitate the integration of Crypt4GH and confidential computing.

Project Name: GA4GH Cloud Secrets Management with Python Starlette, Microsoft Dapr and HashiCorp Vault

Workstream	Data Security, Cloud
Mentor names/emails	• **Alex Kanitz** <alexanderkanitz@gmail.com> • Secondary mentors TBA
Difficulty	Hard
Requirements	• Strong Python programming skills. • Experience with Flask, Starlette, FastAPI or other Python microservices • Experience with Docker and Kubernetes • Understanding of security best practices • Familiarity with Dapr and/or HashiCorp Vault is a plus
Workload	350h

Project Description

This project addresses the critical need for secure and manageable secrets access within cloud-native bioinformatics workflows. The goal is to create a reusable Starlette middleware component that integrates with the Dapr (Distributed Application Runtime) secret stores API, providing a consistent and secure way for ELIXIR Cloud & AAI's FOCA-based microservices to access secrets, with HashiCorp Vault as the initial backend. This approach simplifies secrets management for developers, improves security, and promotes portability across different cloud environments. This is a potential blueprint for GA4GH-powered cloud stacks in general for handling secrets management.

The contributor will:

1. **Develop Starlette Middleware:** Create a Python middleware for Starlette that uses the Dapr Python SDK to interact with the Dapr secret stores API. This middleware will handle authentication to Dapr and provide a simple interface for services to retrieve secrets (e.g., via configuration, request context, or a decorator).
2. **Integrate with proTES:** Integrate the middleware with the proTES service, an existing FOCA-based backend, demonstrating a practical application. This involves modifying proTES to use the middleware for secret retrieval. Note that this requires updating FOCA to use Connexion 3 and updating proTES to use the updated FOCA.

This will be done prior to the start of the project and is not a project requirement - the contributor should assume this to be in place!

3. **Deploy a Test Environment:** Set up a local development environment using Docker Compose and a production-like environment using Kubernetes, including:
    - proTES with the integrated middleware.
    - A Dapr sidecar for proTES.
    - A HashiCorp Vault instance.
4. **Write Tests:** Develop unit, integration, and end-to-end tests to validate the functionality, security, and error handling of the middleware and its integration.
5. **Write Documentation:** Create clear and concise documentation, including examples, to enable developers to use the created middleware and integrate secrets into their services.

Expected Outcomes (with rough **timeline**, to be adapted by contributor):

- A reusable, well-documented Starlette middleware for Dapr-based secrets access (week 3)
- A working integration with proTES (week 5)
- Deployment configurations (Docker Compose and Kubernetes) (week 7)
- A comprehensive test suite (week 9)
- Improved developer experience and security posture when handling secrets within FOCA-compliant services (week 11)

Project Name: Integrate Elixir Component to Krini

Workstream	Cloud Workstream
Mentor names/emails	• Javed Habib <jh4official@gmail.com>
Difficulty	Medium
Requirements	• Strong programming skills in Python and JavaScript (React for web component integration). • Experience with containerization and orchestration (Docker, Kubernetes, Helm). • Knowledge of infrastructure management (Terraform, Mikube). • Familiarity with API integration (REST, Weskit, Tesk). • Understanding of CI/CD pipelines (GitHub Actions, Jenkins).
Workload	175h

Project Description

This project aims to integrate the Elixir Cloud component (Web Components) into Krini while ensuring seamless interoperability with essential APIs and services, including WES (maybe using Weskit), TES (using Tesk), DRS, and TRS. The integration process will involve containerization using Docker Compose, orchestration with Kubernetes (Helm charts), and infrastructure provisioning with Terraform to facilitate a Mikube setup. The final outcome will be a fully functional, scalable, and easily deployable system for Krini.
The project will involve:

- Integrating the Cloud component into Krini's architecture.
- Implementing WES (using Weskit) for workflow execution.
- Integrating TES (using Tesk) for task execution.
- Enabling support for DRS and TRS for data resolution and tool discovery.
- Developing Helm charts for Kubernetes-based deployment.
- Creating a Docker Compose setup for local development and testing.
- Writing Terraform configurations for initiating a Mikube-based setup.
- Implementing CI/CD pipelines for automated builds and testing.
- Writing documentation for setup, usage, and troubleshooting.
- Conducting comprehensive testing to validate integrations.

Stretch Goal

The APIs interact with each other and a complete end to end flow can be run.
Timeline

Week	Tasks
1–2	Research existing components; Set up Minikube with Terraform.
3–4	Develop Helm charts for WES (weskit) and TES (tesk).
5–6	Integrate DRS/TRS; Draft Docker Compose setup.
7–8	Embed Cloud component into Krini UI; Configure API gateways.
9–10	Write tests; Validate end-to-end workflows.
11–12	Finalize documentation; Optimize performance; Prepare demo.

Project Name: Web Components Authentication & Access Control Framework

Workstream	Cloud Workstream
Mentor names/emails	• Anurag Gupta <aguptaking@gmail.com> • Secondary mentors TBA
Difficulty	Hard
Requirements	• Proficiency in **Python** (Flask/FastAPI) for backend. • Strong experience with **JavaScript/TypeScript** and **Lit** for frontend. • Understanding of **OAuth2, JWT, OIDC**, and security best practices. • Experience with **RESTful APIs, API gateways, and cloud integrations**. • Familiarity with **Git, CI/CD, and automated testing**.
Workload	350h

Project Description

This project focuses on **enhancing and abstracting access control** for cloud-based services by improving the **FOCA Access Control API** and creating a **Lit-based web component package** (kind of like react)for seamless integration. The aim is to provide a **reusable, plug-and-play authentication and authorization system**, ensuring consistency across different cloud service components.

The project consists of two major phases:

1. Enhance and Abstract the Access Control API within FOCA

FOCA serves as an **abstraction layer** for cloud services, providing common functionalities, including **authentication and authorization**. This phase will focus on:

- **Refining and exposing FOCA's Access Control API** to ensure it can be easily integrated with frontend applications.
- **Standardizing access control workflows** to ensure compatibility with various cloud services.
- **Enhancing documentation and developer usability**, making it easier for other services to adopt the API.

The current documentation for FOCA's Access Control API can be found here.

2. Develop a Lit-Based Web Component Package

The second phase involves developing a **Lit-based web component package** that acts as a **frontend abstraction for authentication and access control**. This component will:

- **Provide a generic, reusable access control UI** that integrates with the FOCA API.
- **Support multiple authentication mechanisms dynamically**, allowing different services to configure access control easily.
- **Enable seamless integration with other cloud service components** by exposing a well-defined API for other packages.
- **Ensure flexibility and ease of use**, allowing cloud services to implement authentication without redundant frontend development.

The **first target integration** will be with the **GA4GH TES (Task Execution Service) client**, available in the cloud-components repository.

Expected Outcomes (with rough timeline, adaptable by contributor)

- **A frontend-compatible Access Control API** (week 4)
- **A reusable Lit web component** for authentication (week 8)
- **Integration with GA4GH TES client** (week 10)
- **Comprehensive documentation and security posture** (week 12)

Project Name: Template (copy and create, **do not modify!**)

Workstream	
Mentor names/emails	•
Difficulty	
Requirements	
Workload	90h/175h/350h
Project Description	
Provide detailed information on the project for contributors to understand the scope of the project, tasks and expectations. Sub-sections included could be project overview, **milestones**, **timelines**, impact.	