TQ Solution

Technical Specifications

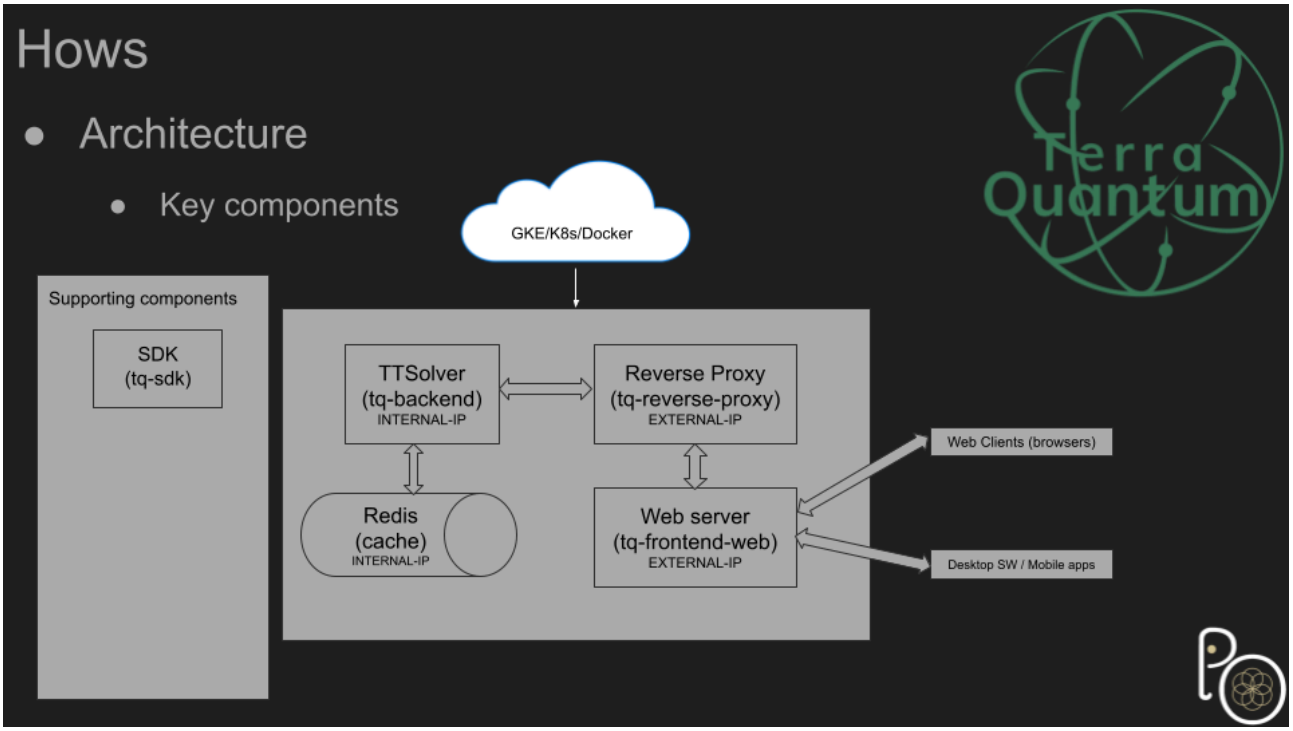
Introduction

This document outlines the technical architecture and software technologies used in the TQ Solution project, designed for scientists who may not be familiar with software development concepts. The project aims to provide a secure, scalable, performant, and reliable system for calculating minima using TTSolver.

Problem Statement

The task is to create software that allows users to find minima of multidimensional functions using the TTOpt solver. The solution includes both a locally installable application and a web-based version. The key requirement is to keep the TTOpt solver code hidden from users while providing a straightforward interface.

Architecture Overview

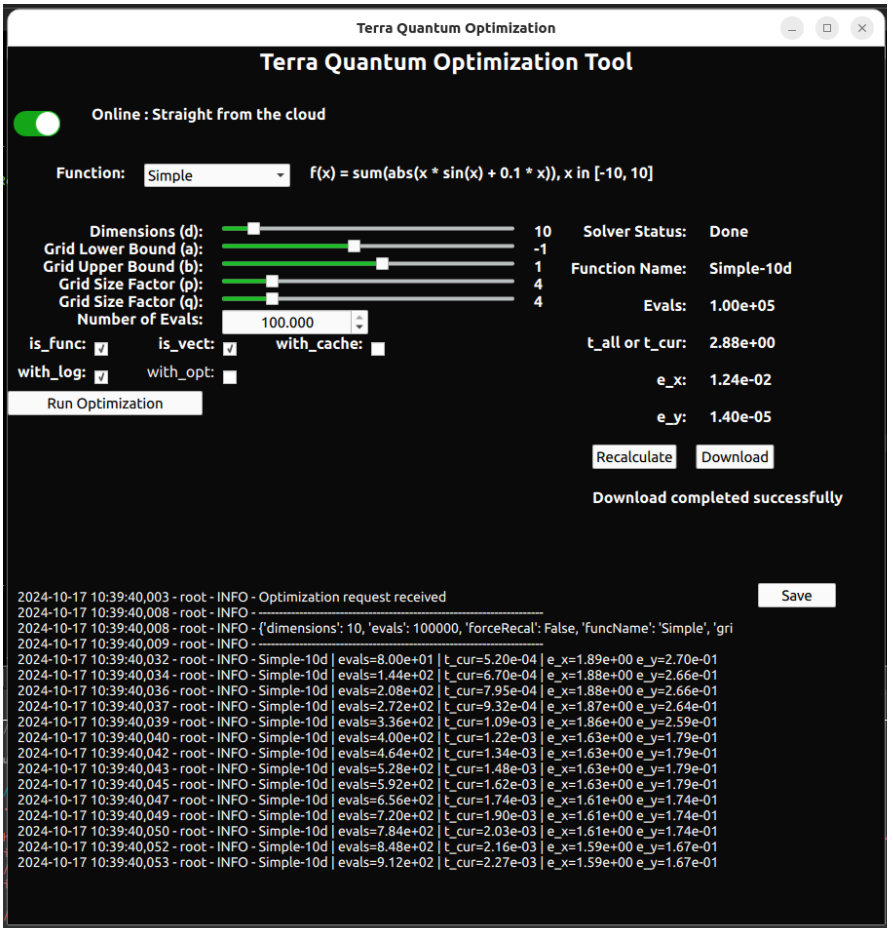


The TQ Solution architecture consists of the following key components:

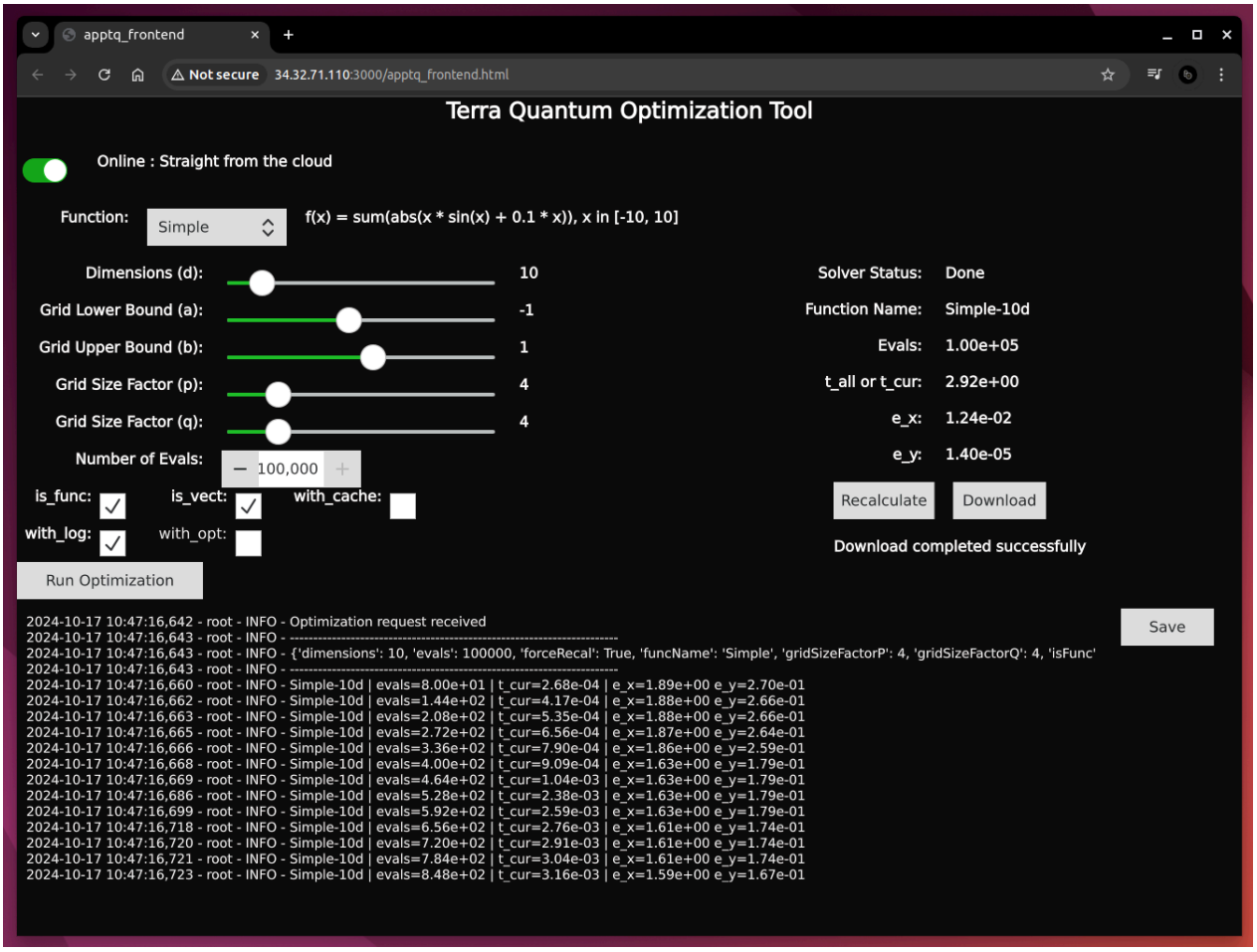
Frontend Web UI (tq-frontend-web)

- **Purpose**: Serves as the user interface for interacting with the system
- **Access**: Has an External IP, accessible from the internet
- **Technologies**: Web technologies (Qt WASM - Webassembly)
- **Features**:
    - Dropdown for function selection
    - Sliders for input parameter adjustment
    - “Run Optimization" button
    - Results display with status (Processing, Done, Error)
    - Download button for complete info logs
    - Text panel for viewing logs

Local standalone app



Web app



Reverse Proxy (tq-reverse-proxy)

- **Purpose**: Acts as an intermediary between Frontend and Backend
- **Access**: Has an External IP, accessible from the internet
- **Function**: Routes requests to appropriate backend services
- **Security**: Handles only known endpoints, providing an additional security layer

TTSolver Backend (tq-backend)

Link: https://github.com/pprajap/tq_backend/blob/main/docs/api_reference.md

- **Purpose**: Performs calculations for finding minima
- **Access**: Internal IP only, not accessible from outside the cluster
- **Technologies**: Specific backend technology (e.g., Python)
- **Features**:
    - Communicates with Redis Cache (or in-memory storage) for data persistence
    - Processes optimization requests
    - Generates detailed logs of the optimization process

Redis Cache (not implemented in current version)

- **Purpose**: Provides fast, in-memory data storage for the Backend
- **Access**: Internal IP only, not accessible from outside the cluster

- **Technology**: Redis (in-memory data structure store)
- **Note**: Currently replaced by an unordered_map directory inside the backend service

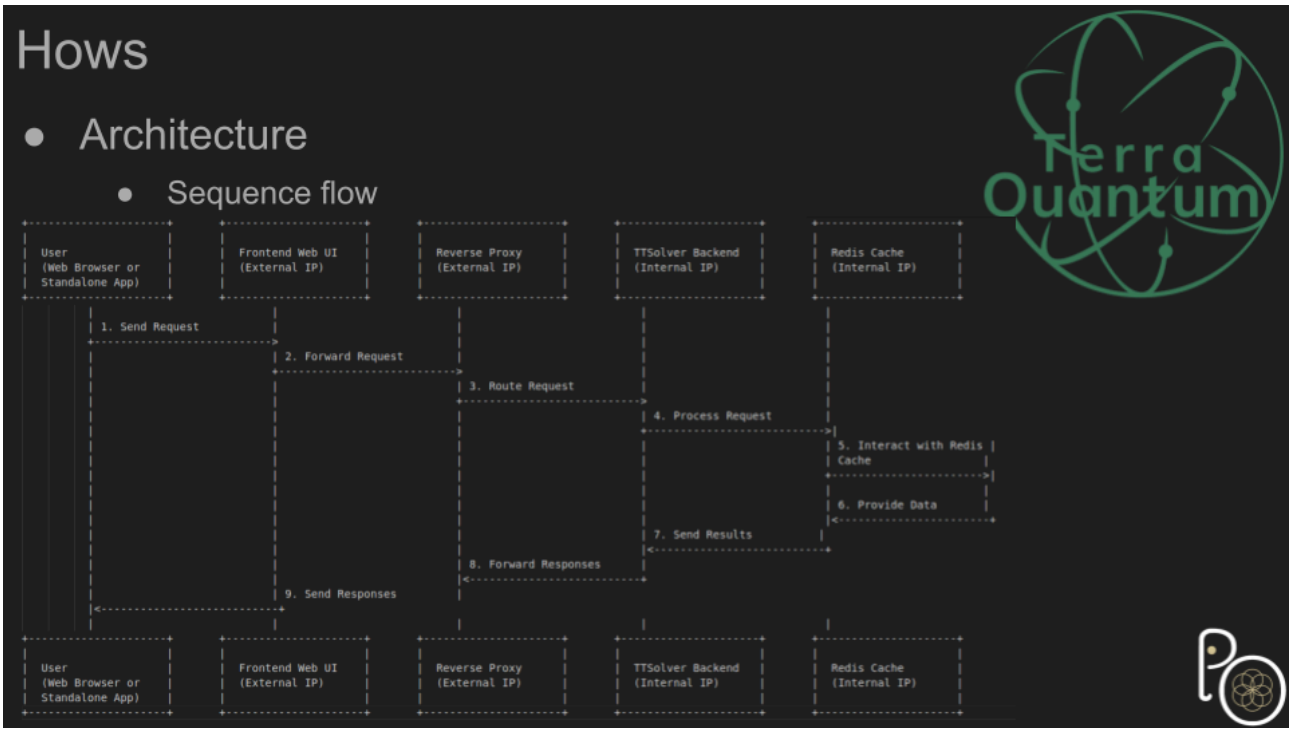
Kubernetes Cluster (hosted on Google Cloud Run)

- **Purpose**: Manages and orchestrates the containerized applications
- **Platform**: Google Cloud Run (GCR) with Google Kubernetes Engine (GKE)

SDK Module (tq-sdk)

- **Purpose**: Enables easy integration of the algorithm into other applications, improve performance by packaging libraries ported in C/C++
- **Status**: Work in Progress (WIP)

Sequence Diagram



Key Technologies Explained

Docker

Docker is used for containerizing the application components, ensuring consistency across different environments and simplifying deployment.

CI/CD Pipeline

Continuous Integration and Continuous Delivery/Deployment pipelines are implemented using GitHub Actions, automating the build, test, and deployment processes.
Link: https://github.com/pprajap/tq_solution/blob/main/README.md#setting-up-cicd

Kubernetes

Kubernetes is used for orchestrating the containerized applications, providing scalability and management capabilities.

Qt WebAssembly

With Qt for WebAssembly, you can distribute your application as a web application that runs in a browser sandbox. This approach is suitable for web distributed applications that do not require full access to host device capabilities

Security and Isolation

- TTSolver Backend has only an Internal IP, not accessible from outside the cluster
- Only the Frontend Web UI and Reverse Proxy have External IPs
- The Reverse Proxy acts as an additional security layer

Performance and Caching

- TTSolver saves previous calculations to improve performance on repeated requests
- Users have the option to force recalculation of minima

Tech Stack

- **Frontend**: Qt Desktop + Qt WebAssembly (same codebase for both)
- **Backend**: Python
- **Containerization**: Docker
- **Orchestration**: Kubernetes (GKE)
- **Cloud Platform**: Google Cloud Run
- **CI/CD**: GitHub Actions

- **Caching**: Recommend Redis but now In-memory storage (unordered_map)

Conclusion

The TQ Solution provides a platform for scientists to perform minima calculations using TTSolver. It offers both web-based and local application options, ensuring flexibility and ease of use. The architecture is designed to be secure, scalable, and performant, leveraging modern cloud and containerization technologies.