# Microservices-Based Python Strava Data Analysis Application Architecture

This document outlines the architecture for a microservices-based Python desktop application that analyzes running and swimming exercise data from Strava. The system provides AI-powered feedback and includes exposure to Docker, Kubernetes, Terraform, Helm, Kafka, and other modern tools.

## 1. High-Level Architecture Overview

The architecture follows a microservices design, ensuring separation of concerns, scalability, and independent deployment of services. Each microservice is containerized with Docker and orchestrated via Kubernetes, with Helm for deployment configuration. Terraform provisions the cloud infrastructure. Kafka serves as the event streaming backbone.

High-Level Components:  
• Frontend UI Service (Desktop Client)  
• API Gateway  
• Strava Data Collection Service  
• Data Pipeline + Analytics Service  
• AI Agent Service  
• Data Visualization Service  
• Supporting infrastructure (Databases, Kafka, Monitoring tools)

## 2. Service Breakdown

### Frontend UI Service

Provides a desktop client for visualizing workout stats and receiving AI feedback. Built with PyQt5/PySide6 for a native desktop UI, or Electron for a web-based UI packaged as a desktop app. Communicates with backend services over HTTP or gRPC.

### API Gateway

Routes requests from the frontend to backend microservices. Implemented with FastAPI or NGINX as a reverse proxy. Handles authentication and API versioning.

### Strava Data Collection Service

Authenticates with Strava API, fetches workout logs, and stores raw data. Publishes events to Kafka for further processing.

### Data Pipeline + Analytics Service

Processes raw Strava data using Pandas/NumPy and stores processed metrics in a time-series database (TimescaleDB/PostgreSQL). Optionally uses Scikit-learn for predictive analytics.

### AI Agent Service

Receives processed metrics, queries AI APIs (e.g., OpenAI) for personalized feedback, and returns this feedback to the frontend.

### Data Visualization Service

Generates charts and dashboards using Matplotlib/Plotly/Bokeh, exposing visualizations via an API for the frontend to consume.

## 3. Infrastructure & Tooling

• Docker – Containerize each microservice.  
• Kubernetes – Orchestrate and scale services.  
• Helm – Manage Kubernetes deployment configurations.  
• Terraform – Provision cloud infrastructure.  
• Kafka – Event streaming for data processing.  
• PostgreSQL + TimescaleDB – Time-series metrics storage.  
• Prometheus + Grafana – Monitoring and dashboards.  
• MinIO – Store large files (e.g., GPX/FIT exports).  
• Airflow – Optional ETL scheduling.  
• gRPC – Optional high-performance service-to-service communication.