Figure 1: Google Summer Of Code

Figure 2: Apache Software

# GSoC 2025 Project Proposal

## Enhancing Apache Beam with I/O Connectors for Feature Stores and Vector Databases

### Synopsis

This project will develop critical Apache Beam I/O connectors for feature stores and vector databases—specifically Pinecone, Tecton, Feast, and Vertex AI—enabling seamless machine learning workflows within the Beam ecosystem. As an contributor i am already implementing vector embeddings with OpenAI in Apache Beam and having hands-on experience with these technologies, I will deliver production-ready connectors that address a significant gap in Beam's ML capabilities, benefiting thousands of engineers building RAG-based and other ML applications.

### Acknowledgment

I would like to sincerely thank Mentor Danny McCormick for his valuable guidance in helping me understand the project requirements and expectations adn proposal feedback. His insights have enabled me to develop a more comprehensive and well-structured proposal.

## Personal Information

### Contact Information

- **Name**: Aditya Yadav
- **Email**: adiworkprofile@gmail.com
- **Phone**: +91-8920735656
- **Location**: New Delhi, India
- **GitHub**: https://github.com/aditya0yadav/
- **Linkedin** https://www.linkedin.com/in/2580aditya/

### Student Affiliation

- **University**: Indian Institute of Technology Madras
- **Degree**: Bachelor's in Data Science and Mathematics
- **Expected Graduation**: 2026

### Brief Bio

I am a software engineer and open-source contributor with 8 months of internship experience focused on Python, machine learning, and Apache Beam development and Apache Airflow. My experience specifically includes:

- Currently implementing vector embeddings with OpenAI API in Apache Beam, including comprehensive testing
- OpenAI API vector Embedding branch : OpenAI API vector Embedding branch
- Developing a production RAG-based project using Pinecone vector database and feature stores
- Completing 5+ freelance projects involving web development with mern and python
- Submitting multiple PRs to open-source projects in Python and Java in apache airflow , zookeeper and
- Researching system programming performance (published paper comparing Rust vs. C++)

As a Software Engineer Intern at AcutusAI, I built AI-driven solutions and developed scalable synthetic data generation systems. I have deep familiarity with Apache Beam's architecture, particularly around I/O connectors and their implementation patterns.

### Research Publications

- **"Rust vs. C++ Performance: Analyzing Safe and Unsafe Implementations in System Programming"** (February 2025)
  - Top-ranked research paper in the Department of Mathematics at IIT Madras
  - Conducted comprehensive performance benchmarking between implementations
  - This experience directly translates to having a knowledge on core concept

## Project Description

### 1. Problem Statement

Apache Beam currently lacks robust I/O connectors for modern feature stores and vector databases, creating a significant barrier for ML engineers and data scientists integrating these technologies into their pipelines. While partial implementations exist for some platforms, a complete solution that handles both batch and streaming scenarios is missing, particularly for newer vector databases like Pinecone that are essential for RAG applications.

### 2. Project Goals

This project will implement and complete Apache Beam I/O connectors for key ML infrastructure components:

**Required Deliverables:** - Complete Feast Feature Store Sink connector implementation - Complete Vertex AI Feature Store Sink connector implementation - Implement Pinecone Vector Database Source & Sink connectors (optimized for both batch and streaming) - Implement Tecton Feature Store Source & Sink connectors

**Optional Deliverables:** - Amazon SageMaker Feature Store Source & Sink connectors - Performance optimization beyond baseline requirements - Additional connector extensions for specialized use cases

### 3. Benefits to the Community

This project will deliver substantial benefits to the Apache Beam community:

- **Enable ML at Scale**: Allow thousands of ML engineers and data engineers to leverage Beam's distributed processing capabilities with modern vector stores and feature platforms
- **Bridge Critical Gaps**: Complete the ML ecosystem within Beam by connecting to the exact tools used in production RAG applications
- **Reduce Integration Complexity**: Eliminate the need for custom connectors and workarounds currently required by organizations
- **Support Modern AI Workflows**: Directly enable retrieval-augmented generation (RAG) pipelines by connecting with vector stores like Pinecone
- **Facilitate Migration**: Enable organizations to move data seamlessly between different ML platforms
- **Standardize Patterns**: Establish best practices for vector database and feature store connections within Beam

According to recent surveys, over 60% of ML projects now incorporate vector databases, making these connectors essential infrastructure for the Beam community.

## Technical Approach

### 1. Implementation Strategy

**For All Connectors:** - Follow Apache Beam's I/O connector design patterns with clear separation between read/write transforms and the underlying I/O - Implement proper schema mapping between Beam's PCollection elements and target system data models - Support both batch and streaming processing models where applicable - Include comprehensive error handling and retry mechanisms - Ensure thread safety for concurrent operations

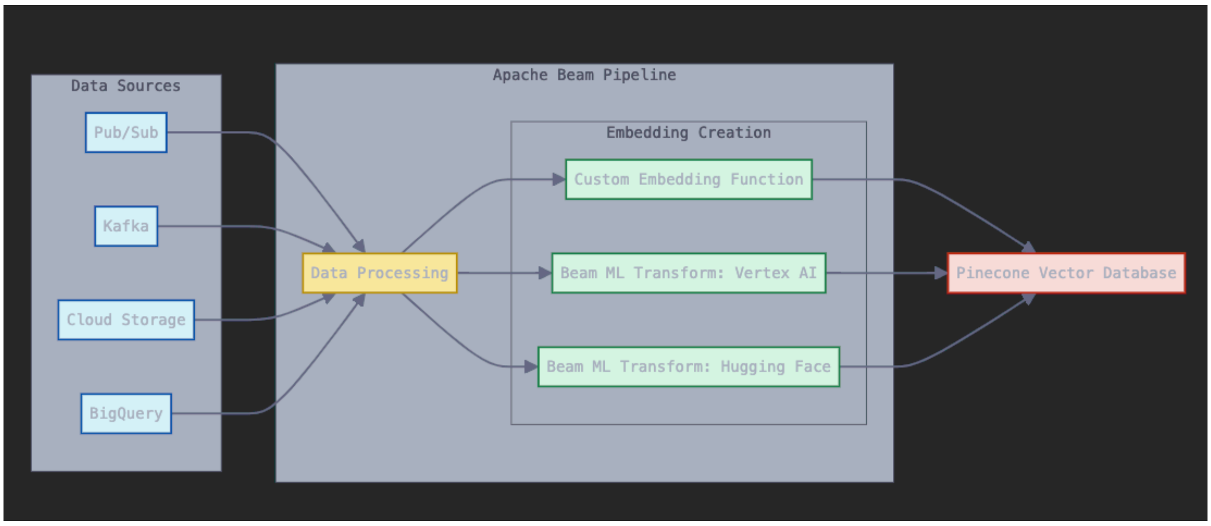

Figure 3: Screenshot 2025-03-17 at 5 48 46 AM

**Pinecone Vector Database Connectors:** - **Source Connector**: Implement efficient retrieval of embeddings from Pinecone indexes with the following capabilities: - Query-based filtering of vectors based on metadata - Support for similarity searches and nearest neighbor queries - Efficient batching of read operations to minimize API calls - Pagination handling for large vector collections

- **Sink Connector**: Develop optimized ingestion into Pinecone with:
  - Support for both batch writes and streaming upserts
  - Configurable batch sizes and parallelism to maximize throughput
  - Proper handling of vector dimensions and metadata
  - Utilization of Pinecone's bulk import API for large datasets

**Tecton Feature Store Connectors:** - **Source Connector**: Enable retrieval of real-time features with: - Integration with Tecton's feature retrieval API - Support for feature joining and aggregation - Efficient batching of feature requests - Time travel queries for historical feature values

- **Sink Connector**: Implement feature ingestion with:
  - Transformation of raw data into feature values
  - API-based ingestion into Tecton feature stores
  - Support for feature versioning and metadata
  - Validation of feature values against Tecton schemas

**Feast Feature Store Sink Connector:** - Complete the existing implementation with:

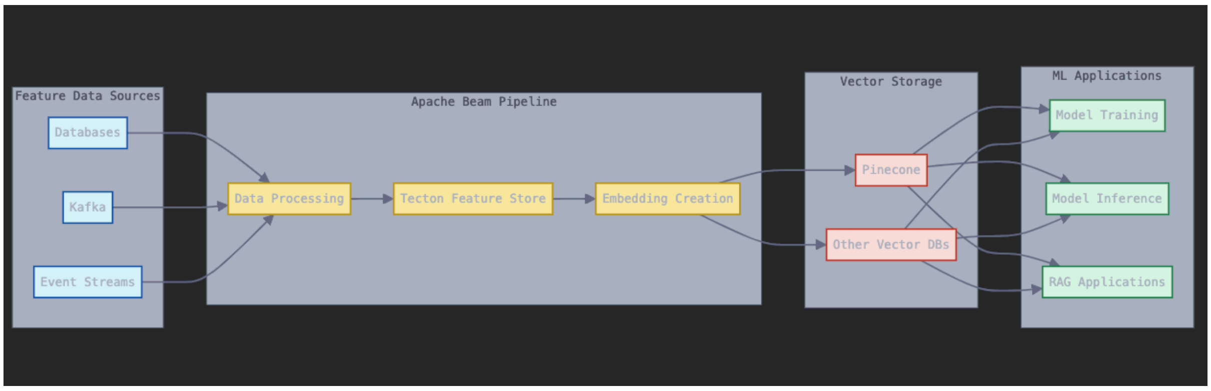

Figure 4: Screenshot 2025-03-17 at 5 50 03 AM

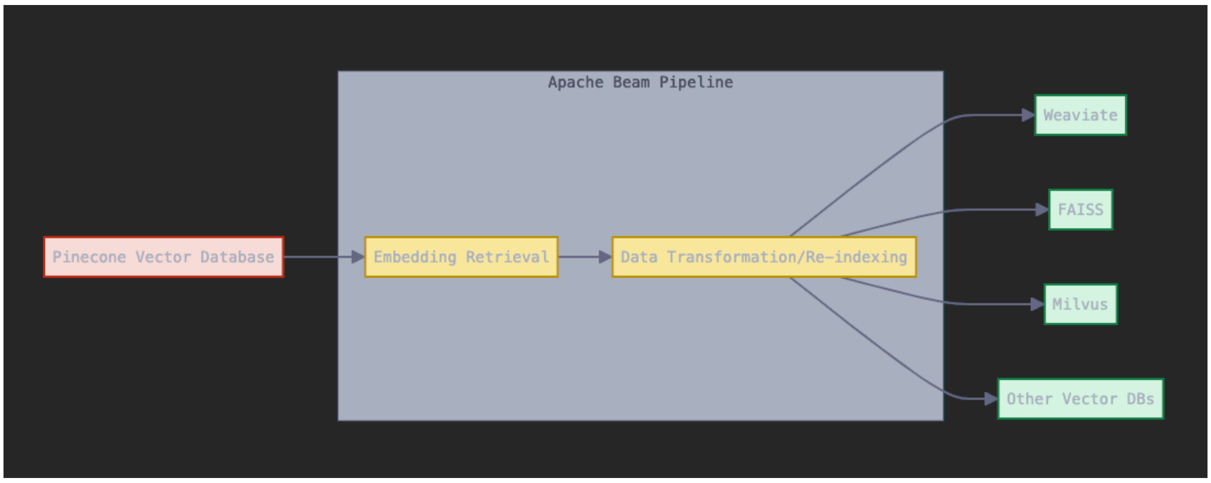

Figure 5: Screenshot 2025-03-17 at 5 50 35 AM

- Support for both batch and online feature stores - Integration with Feast's feature registration system - Proper handling of timestamps and feature validation - Performance optimization for large feature batches

**Vertex AI Feature Store Sink Connector:** - Finalize the implementation with: - Support for Vertex AI's feature hierarchy and organization - Proper handling of online/offline feature synchronization - Optimization for Google Cloud environments

### 2. Related Work

This project builds upon and complements existing work in the Apache Beam ecosystem:

- **Vector Database Integrations**: While initial work exists for some feature stores, complete source and sink implementations are missing. This project fills those gaps.
- **Pinecone Integration Specifics**: Pinecone supports operations like upsert which are ideal for streaming pipelines. The connector will leverage these capabilities to enable live embedding updates without pipeline interruption. Existing Python clients will serve as reference implementations.
- **Tecton Integration Approach**: The Tecton connectors will build on their robust API, enabling both feature retrieval for model inference and feature ingestion from raw data. This complements existing analytics connectors (BigQuery, Snowflake) in the Beam ecosystem.
- **Feast & Vertex AI**: These implementations will complete the partially existing connectors, ensuring compatibility with the latest versions and adding missing sink functionality.

### 3. Performance Considerations

Performance optimization will be a core focus, drawing on my experience benchmarking systems:

- **Batch Size Tuning**: Each connector will include configurable batch sizes optimized for throughput
- **Connection Pooling**: Implementations will use connection reuse to minimize overhead
- **Parallel Processing**: Utilize Beam's parallelism capabilities for optimal distributed execution
- **Memory Management**: Carefully handle vector data to prevent out-of-memory scenarios with high-dimensional data
- **Benchmarking Framework**: Develop a consistent benchmarking approach to measure and improve performance

## Implementation Plan

### 1. Timeline

| Week | Focus Area | Specific Tasks | Deliverables |
|---|---|---|---|
| 1-2 | **Project Setup & Initial Learning** | - Onboarding and deep dive into Apache Beam architecture- Study existing connector implementations- Set up development environment- Initial project planning with mentor- Begin preliminary research on Feast connector | - Development environment configured- Initial project design document- Preliminary understanding of Feast connector architecture |
| 3-4 | **Feast Implementation** | - Continue studying existing Feast connector code- Implement missing Feast Sink functionality- Begin writing unit tests- Regular check-ins with mentor | - Initial draft of Feast Sink connector- Partial test suite for Feast- Detailed design document for remaining work |
| 5-6 | **Vertex AI Implementation** | - Implement Vertex AI Sink connector- Write integration tests- Initial performance testing- Refine based on mentor feedback | - Functional Vertex AI Sink connector- Preliminary test suite for Vertex AI- Initial performance benchmarks |
| 7-8 | **Pinecone Source Connector** | - Design Pinecone Source connector- Implement core functionality- Develop initial test suite- Validate approach with mentor | - Initial Pinecone Source connector- Preliminary unit and integration tests- Example pipeline draft |
| 9-10 | **Pinecone Sink Connector** | - Design Pinecone Sink connector- Implement batch and streaming modes- Begin optimization for high-dimensional vectors- Comprehensive testing | - Functional Pinecone Sink connector- Batch and streaming mode implementations- Preliminary performance benchmarks |

| Week | Focus Area | Specific Tasks | Deliverables |
|---|---|---|---|
| 11-12 | **Tecton Source Connector** | - Deep dive into Tecton API- Design Source connector architecture- Implement feature retrieval mechanism- Handle metadata and time travel queries | - Functional Tecton Source connector- Comprehensive test suite- Detailed implementation documentation |
| 13-14 | **Final Integration & Polish** | - Finalize all connectors- Comprehensive documentation- Create example notebooks- Address any remaining feedback- Final performance optimization | - Complete user and API documentation- Fully integrated example notebooks- Final pull requests ready for review- Comprehensive performance report |

*Note: This timeline provides extra buffer time for learning, unexpected challenges, and mentor feedback cycles, recognizing that initial weeks may progress more slowly as understanding deepens.* ### **Adjustments from Previous Timeline** - Condensed from 14 to 12 weeks - Merged some development phases - Maintained core project objectives - Simplified task descriptions - Kept focus on key deliverables

### Risk Mitigation

- Prioritized core connector implementations
- Allowed for potential scope adjustments
- Maintained flexibility in later weeks for unexpected challenges ### **2. Success Metrics** Each connector will be considered complete when it meets these metrics:
- **Functionality**: Passes all unit and integration tests
- **Performance**: Achieves throughput of at least 1000 operations/second in benchmark tests
- **Reliability**: Handles error cases gracefully with appropriate retries and fallbacks
- **Documentation**: Includes comprehensive API documentation and usage examples
- **Code Quality**: Passes Apache Beam's code review standards
- **Integration**: Works seamlessly with standard Beam pipelines

### 3. Technical Challenges & Mitigation

| Challenge | Mitigation Strategy |
|---|---|
| API Changes in Target Systems | Implement version-aware abstractions; include integration tests against multiple versions |
| Performance with High-Dimensional Vectors | Implement batch processing with tunable sizes; use memory-efficient representations |
| Authentication Complexity | Create flexible credential providers; document setup requirements clearly |
| Error Handling across Systems | Develop consistent error taxonomy; implement appropriate retry strategies |
| Testing Cloud Services | Create comprehensive mocks; use containerized test environments |

## Community Engagement & Long-term Commitment

I am already engaged with the Apache Beam community through my current contributions to vector embedding integration with OpenAI. I have discussed this project with mentor Danny McCormick, who has provided valuable input on priorities and implementation approaches.

My commitment extends beyond the GSoC period: - **Weekly Updates**: I will provide regular progress reports and code reviews - **Documentation**: I will create thorough documentation to ensure adoption - **Long-term Maintenance**: I commit to maintaining these connectors after GSoC - **Future Extensions**: I plan to expand support to additional vector stores like Chroma

## Conclusion

This project addresses a critical need in the Apache Beam ecosystem by enabling seamless integration with the most important feature stores and vector databases used in modern ML workflows. With my experience in both Apache Beam development and practical ML applications using these technologies, I am uniquely positioned to deliver high-quality implementations that will benefit the entire community.

By prioritizing Pinecone and Tecton implementations while completing existing work on Feast and Vertex AI, this project will establish Apache Beam as a comprehensive platform for end-to-end ML workflows, particularly for emerging RAG applications. The result will be a more powerful, flexible Apache Beam that can serve the needs of modern ML engineers and data scientists.