Use Case: RAG with PDF - Google Cloud Reference Architecture

Let's create a reference architecture for a document query system (RAG-based GenAI system) on the Google Cloud platform. The system, which currently processes and analyses a PDF about the impact of the Indian Premier League on Test cricket in India, will be reimagined using Google Cloud services including VertexAI to improve scalability, performance, and cost-effectiveness.

Notebook of RAG with PDF standalone Use Case

Key Google Cloud Services and Migration Strategy:

1. Document Storage:

- Migrate from local PDF storage to **Google Cloud Storage** for secure, scalable document storage.

2. Text Processing and Chunking:

- Utilize **Cloud Run Functions** to handle PDF parsing and text chunking, triggered by Cloud Storage events when new documents are uploaded.

3. Text Embedding:

- Use **VertexAI Text Embeddings** **API or Huggingface based Embedding Model deployed as VertexAI Endpoint** for generating embeddings, offering efficient and cost-effective text embedding generation without managing infrastructure.

4. Vector Database:

- Replace the local FAISS implementation with **Vertex AI Vector Search** (formerly Matching Engine) for efficient similarity search at scale.

5. Large Language Model (LLM):

- Use **VertexAI Gemini 1.5/1.0 Pro** for text generation.

6. Question & Answer Pipeline:

- Implement the retrieval-augmented generation process using Python, LangChain or LlamaIndex with **Cloud Run Functions** to orchestrate the workflow between Vector Search, Text Embeddings API, Gemini 1.5 Pro, and other Google Cloud services.

7. API Layer:

- Create an API using **Cloud API Gateway** and **Cloud Run Functions** to handle user requests and responses.

8. Front-end and Load Balancing:

- Implement **Cloud Load Balancing** for global load distribution and **Cloud CDN** for content delivery optimization.

9. Authentication and Authorization:

- Use **Cloud Identity Platform** for secure user authentication and authorization.

10. Data Processing Pipeline:

- Use Dataflow for ETL processes
- Cloud Run Functions for Data Fetching & Embedding generation

11. Monitoring and Analytics:

- Implement Cloud Monitoring, Cloud Logging, and Looker for comprehensive monitoring, logging, and analytics.

12. Security and Compliance:

- Leverage Cloud KMS for secret management
- Cloud Armor for web application firewall and DDoS protection
- Cloud DLP (Data Loss Prevention) for sensitive data handling

13. Scalability and Reliability:

- Utilize **Cloud Run** for serverless container deployment
- Cloud Load Balancing with multiple regions
- Cloud Storage with multi-region configuration for high availability

Additional Considerations:

Development and Testing:

- Use **Cloud Workstations** for development environments
- Cloud Build for CI/CD pipelines
- Artifact Registry for container image storage

Cost Optimization:

- Use Cloud Functions and Cloud Run for serverless compute to optimize costs
- Configure caching strategies to minimize API calls

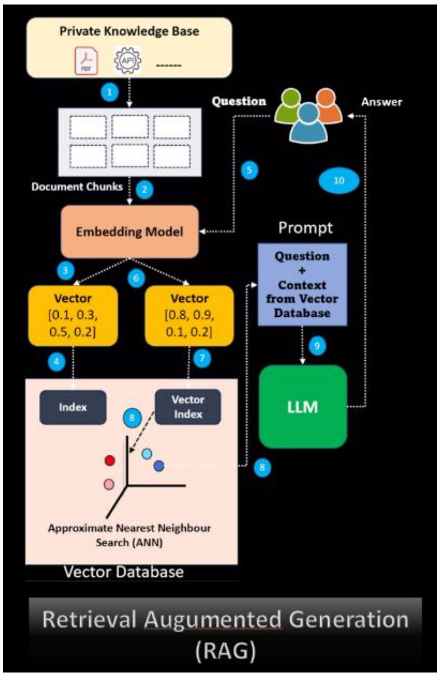
Best Practices:

1. Implement retry logic for API calls
2. Use batch processing for large document sets
3. Implement proper error handling and monitoring
4. Set up appropriate IAM roles and permissions
5. Regular backup and disaster recovery planning

This Google Cloud migration will transform the solution into a cloud-native, serverless architecture, offering:

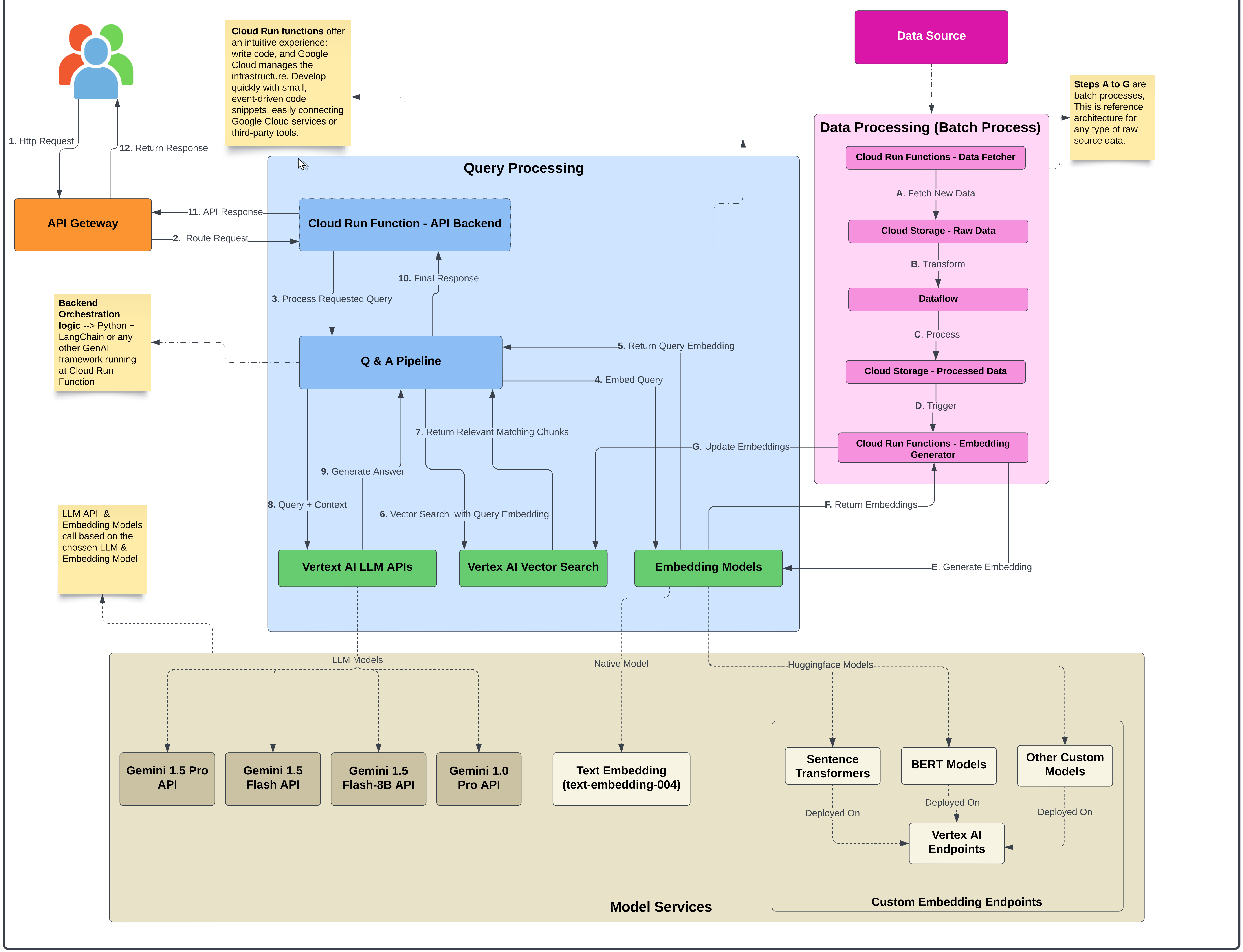
- Better scalability through managed services
- Enhanced security with Cloud-native security tools
- Cost optimization through pay-as-you-go pricing
- Reduced operational overhead
- Integrated AI capabilities through Vertex AI
- Simplified management and monitoring

The architecture leverages Google Cloud's comprehensive set of AI and machine learning services, particularly VertexAI, which provides a unified platform for both traditional ML and modern GenAI applications. This allows teams to focus on improving the document query system.


Private Knowledge Base
PDF
API

1
Question
Answer
Document Chunks
2
5
10
Embedding Model
Prompt
Question + Context from Vector Database
3
6
Vector [0.1, 0.3, 0.5, 0.2]
Vector [0.8, 0.9, 0.1, 0.2]
4
7
9
Index
Vector Index
8
LLM
8
Approximate Nearest Neighbour Search (ANN)
Vector Database
Retrieval Augumented Generation (RAG)

GenAI Simplified Reference Architecture: RAG on Google's Vertex AI



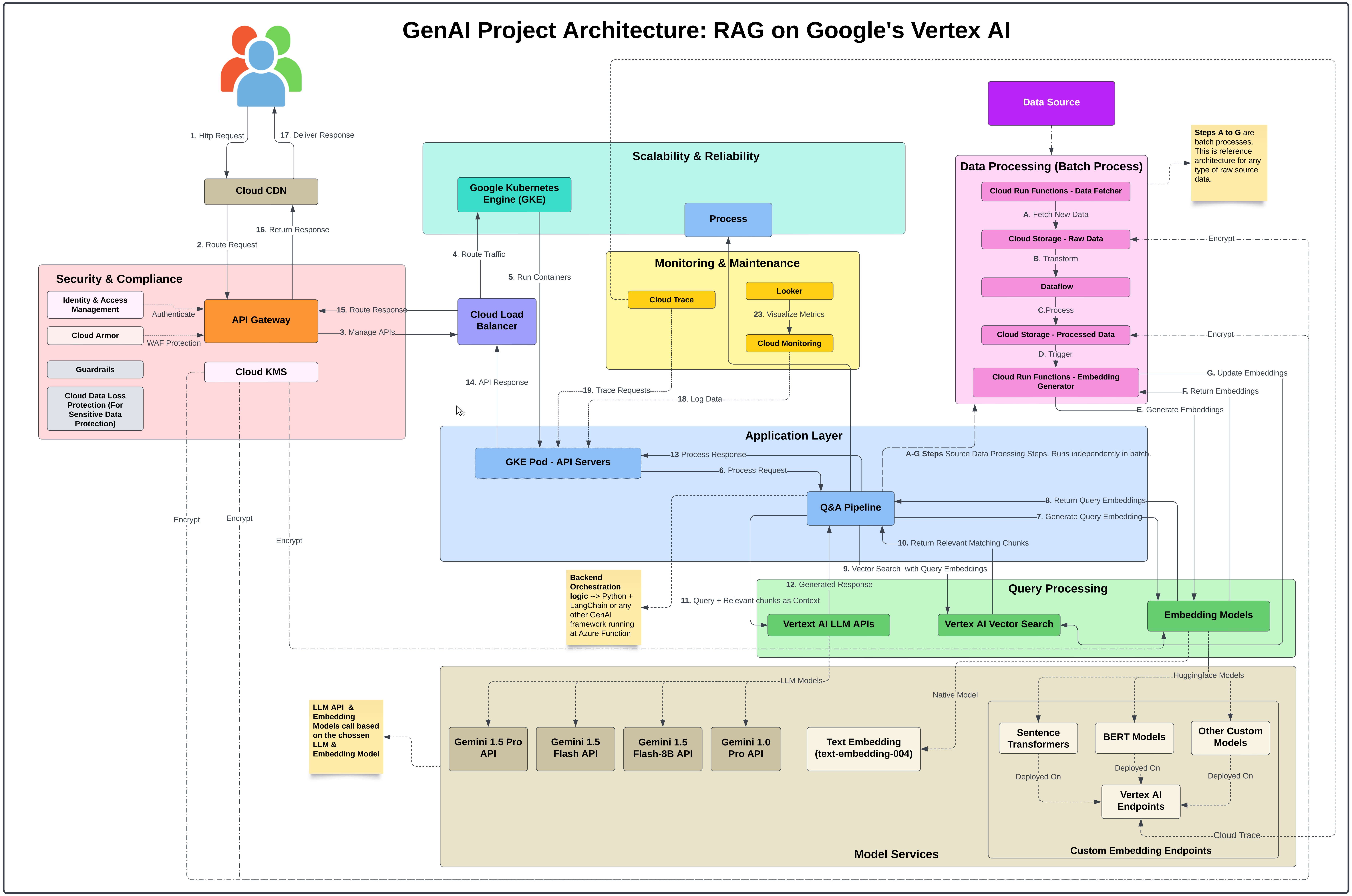
Cloud Run functions offer an intuitive experience: write code, and Google Cloud manages the infrastructure. Develop quickly with small, event-driven code snippets, easily connecting Google Cloud services or third-party tools.

Steps A to G are batch processes, This is reference architecture for any type of raw source data.

Backend Orchestration logic --> Python + LangChain or any other GenAI framework running at Cloud Run Function

LLM API & Embedding Models call based on the chossen LLM & Embedding Model

GenAI Project Architecture: RAG on Google's Vertex AI



Comprehensive Google Cloud Vertex AI-Based Architecture Flow Details

1. User Interaction

- The process begins with a user sending an HTTPS request to the system through the User Interface.

2. Cloud Load Balancing

Cloud Load Balancing is a global entry-point that:

- Provides global load distribution of workloads
- Offers built-in DDoS protection
- Enables multi-region load balancing
- Performs SSL/TLS termination
- Supports HTTP(S), TCP/SSL, and UDP protocols

3. Cloud Armor

Cloud Armor is a web application firewall (WAF) that:

- Provides DDoS and application-layer security
- Enables preconfigured rules for OWASP Top 10 risks
- Supports custom rules and IP allow/deny lists
- Enables adaptive protection using machine learning
- Provides bot management capabilities

4. API Gateway

API Gateway is an API management service that:

- Secures, monitors, and manages APIs
- Provides authentication and authorization
- Implements rate limiting and quota management
- Enables API versioning and documentation
- Supports OpenAPI and gRPC protocols

5. Google Kubernetes Engine (GKE)

Google Kubernetes Engine is a managed container orchestration service that:

- Simplifies deployment and management of containerized applications
- Provides automated scaling and upgrades

- Integrates with Cloud IAM for authentication and RBAC
- Supports advanced networking features with Cloud VPC
- Enables autopilot mode for hands off operations

6. GKE Pods - API Servers

GKE Pods running API Servers are:

- Containerized applications (e.g. Python + LangChain) running as pods within GKE
- Designed to handle incoming API requests
- Responsible for managing the Question-Answering process
- Scalable units that can be increased or decreased based on demand

7. Question-Answering Pipeline

- The core logic for processing user queries and generating responses
- Integrates with Vertex AI for model inference
- Handles context retrieval and prompt engineering

8. Vertex Vector Search

Vertex Vector Search (formerly Matching Engine) is a fully managed vector similarity search service that:

- Provides fast and scalable nearest neighbour search
- Supports multiple distance metrics (cosine, dot product, Euclidean)
- Enables efficient storage and retrieval of embeddings
- Integrates seamlessly with Vertex AI embeddings
- Offers automatic scaling and optimization

9. Vertex AI

Vertex AI is a unified ML platform that:

- Provides access to foundation models (PaLM 2, Gemini)
- Enables text generation, embeddings, and chat completions
- Offers model fine-tuning and customization
- Supports prompt engineering and tuning
- Provides enterprise-grade security and compliance
- Enables model monitoring and explainability

10. Response Flow

- The generated answer flows back through the system components to the user

Data Processing Pipeline

11. Cloud Run Functions

Cloud Run Functions for data processing:

- Serverless compute for event-driven data processing
- Automatic scaling based on workload
- Pay-only-for-what-you-use model
- Native integration with Google Cloud services

12. Cloud Storage – Raw Data

Raw Data Storage:

- Object storage for unstructured data
- Automatic replication and recovery
- Lifecycle management policies
- Strong consistency and durability
- Integration with IAM for access control

13. Dataflow

- Fully managed data processing service
- Handles both batch and streaming pipelines
- Automatic scaling and resource management
- Native integration with other Google Cloud services
- Supports Apache Beam programming model

14. Cloud Storage (Processed Data)

- Storage for processed and transformed data
- Enables versioning and access control
- Integrates with Cloud IAM
- Supports object lifecycle management

15. Cloud Run Functions - Embedding Generator

- Generates embeddings using Vertex AI
- Updates Vector Search indexes
- Handles data preprocessing and chunking
- Manages metadata storage

Monitoring & Maintenance

16. Cloud Monitoring

- Comprehensive monitoring solution
- Collects metrics, logs, and traces
- Provides customizable dashboards
- Enables alerting and notification
- Integrates with Cloud Logging

17. Cloud Trace

- Application performance monitoring
- Distributed tracing capabilities
- Latency analysis and optimization
- Integration with OpenTelemetry
- Automatic trace sampling

18. Looker

- Business intelligence and analytics
- Creates interactive dashboards
- Enables data exploration
- Supports embedded analytics
- Integrates with BigQuery

Security & Compliance

19. Cloud IAM

- Identity and access management
- Fine-grained access control

- Support for service accounts
- Integration with external identity providers
- Audit logging capabilities

20. Cloud Armor

- Web application firewall (WAF)
- DDoS protection
- Bot management
- Security rules and policies
- ML-based adaptive protection

21. Secret Manager

- Centralized secrets management
- Version control for secrets
- Automatic encryption
- IAM integration
- Audit logging

Scalability & Reliability

22. Cloud Autoscaler

- Automatic scaling of resources
- Support for multiple scaling metrics
- Predictive autoscaling capabilities
- Custom scaling policies
- Integration with monitoring

23. Regional/Multi-Regional Deployment

- High availability across regions
- Disaster recovery capabilities
- Global load balancing
- Data replication
- Cross-region failover

24. Cloud CDN

- Content delivery network
- Global edge caching
- SSL/TLS termination
- Cache optimization
- Analytics and monitoring

Interview Questions on Google Vertex AI Architecture for GenAI RAG Project

Q: How does this architecture ensure low latency and high availability for global users?

A: The architecture leverages Google Cloud's global load balancing and Content Delivery Network (CDN) to ensure low latency for users worldwide. Specifically:

- Cloud Load Balancing provides global load distribution across regions
- Cloud CDN caches content close to users
- Multiple regional deployments using Google Kubernetes Engine (GKE)
- Cloud Armor for DDoS protection
- Regional replicas for critical services using Cloud Spanner or Cloud SQL
- Multi-region deployment options for Vertex AI endpoints

Q: How does the system handle data processing and keep the question-answering capabilities up-to-date?

A: The system implements a robust data processing pipeline using Google Cloud services:

1. Cloud Functions or Cloud Run jobs trigger data ingestion
2. Cloud Storage stores raw data
3. Dataflow or Cloud Functions process and transform data
4. Vertex AI Feature Store maintains feature vectors
5. Cloud Scheduler manages periodic updates
6. Vertex AI Matching Engine indexes are updated with new embeddings
7. Vertex AI Model Registry maintains version control

Key advantages:

- Serverless architecture scales automatically
- Built-in monitoring and error handling
- Cost-effective with pay-as-you-go pricing

- Managed services reduce operational overhead
- Strong integration between services

Q: Describe the security measures implemented in this architecture.

A: The architecture implements comprehensive security measures:

- Identity and Access Management (IAM) for fine-grained access control
- Cloud Armor for web application firewall and DDoS protection
- Secret Manager for sensitive credential management
- VPC Service Controls for network security
- Cloud KMS for encryption key management
- Security Command Center for security monitoring
- Cloud Audit Logs for comprehensive audit trails
- Private Google Access for secure API calls
- Binary Authorization for container security
- Cloud DLP for sensitive data protection

Q: How does this architecture handle scaling under increased load?

A: The architecture scales through multiple mechanisms:

- GKE Autopilot automatically scales nodes and pods
- Cloud Run automatically scales containers
- Vertex AI endpoints scale based on traffic
- Cloud Load Balancing distributes traffic globally
- Memorystore scales horizontally for caching
- Pub/Sub automatically scales message processing
- Cloud Storage provides unlimited scaling for data
- Matching Engine scales vector search capacity

Q: Explain the role of Vertex AI in this architecture. What are its advantages?

A: Vertex AI serves as the core ML platform with several key components:

- Managed LLM endpoints for text generation
- Vector store and semantic search capabilities
- Model deployment and versioning

- Feature management and serving
- Model monitoring and evaluation

Advantages include:

- Unified ML platform for training and serving
- Integration with Google's foundation models
- Built-in monitoring and explainability
- Automated model deployment and scaling
- Enterprise-grade security and compliance
- Cost optimization through efficient scaling

Q: How does the system ensure efficient retrieval of relevant information for user queries?

A: The system leverages several Vertex AI components:

- Vertex AI Vector Search for efficient vector similarity search
- Semantic chunking and embedding generation
- Hybrid search combining vector and keyword approaches
- Query expansion and reformulation
- Relevance feedback mechanisms
- Caching of similar queries and results
- Dynamic reranking of search results

Q: Describe the observability and monitoring setup in this architecture.

A: The architecture uses multiple monitoring tools:

- Cloud Monitoring for metrics and alerts
- Cloud Logging for centralized logs
- Cloud Trace for request tracing
- Cloud Profiler for performance analysis
- Error Reporting for error tracking
- Cloud Operations suite for overall observability
- Custom dashboards in Cloud Monitoring
- Service Level Objectives (SLO) monitoring
- Vertex AI Model Monitoring for ML-specific metrics

Q: How does this architecture handle potential failures or outages in a single Google Cloud region?

A: The architecture implements several resilience measures:

- Multi-region deployment using GKE
- Global load balancing for traffic distribution
- Regional failover configurations
- Cloud Spanner multi-region replication
- Disaster recovery procedures
- Automated failover mechanisms
- Regular disaster recovery testing
- Regional backup and restore capabilities

Q: Explain the purpose of Cloud Load Balancing in this architecture. How does it differ from traditional load balancers?

A: Cloud Load Balancing provides:

- Global load distribution
- Automatic scaling
- Integrated security features
- Health checking
- SSL/TLS termination
- Traffic splitting capabilities
- Cross-region failover
- Integrated CDN

It differs from traditional load balancers by:

- Operating at global scale
- Providing integrated security
- Supporting multiple protocols
- Offering automatic scaling
- Requiring no pre-warming
- Providing integrated health checks

Q: How would you implement a feature to allow users to upload and process their own documents for question-answering?

A: Implementation approach:

1. Use Cloud Storage for secure document upload
2. Implement Cloud Functions for document processing
3. Use Document AI for document understanding
4. Generate embeddings using Vertex AI
5. Store vectors in Matching Engine
6. Implement data isolation using namespaces
7. Use Cloud DLP for sensitive data detection
8. Set up IAM policies for access control
9. Implement usage quotas and monitoring
10. Add versioning and backup capabilities

Q: How would you modify this architecture to support multi-tenancy while ensuring data isolation and performance for each tenant?

A: Multi-tenancy implementation:

- Use Cloud Identity for tenant authentication
- Implement tenant isolation in Matching Engine
- Separate storage buckets or prefixes per tenant
- Use namespaces in GKE
- Implement tenant-specific rate limiting
- Set up resource quotas per tenant
- Use Cloud Spanner for tenant metadata
- Implement tenant-specific monitoring
- Configure tenant-specific backup policies
- Use VPC Service Controls for network isolation

Q: The current setup uses GKE for the API servers. In what scenarios might you consider using Cloud Run instead, and how would this change the architecture?

A: Consider Cloud Run when:

- You need simpler deployment
- Workloads are HTTP-based

- You want automatic scaling to zero
- You don't need complex orchestration
- Cost optimization is priority

Changes to architecture:

- Replace GKE with Cloud Run
- Use Cloud Run jobs for batch processing
- Implement Cloud Run services for APIs
- Modify networking setup
- Adjust monitoring configuration
- Update deployment pipelines

Q: The architecture uses Vertex AI for language model inference. How would you implement a fallback mechanism if it experiences downtime or throttling?

A: Fallback implementation:

1. Deploy backup models on GKE
2. Use Cloud Pub/Sub for request queuing
3. Implement circuit breaker pattern
4. Set up monitoring and alerts
5. Use Cloud Tasks for retry logic
6. Implement graceful degradation
7. Configure automatic failover
8. Set up cross-region redundancy
9. Monitor quota usage
10. Implement request prioritization

Q: How can you improve the relevance of retrieved passages in a RAG system using Vertex AI?

A: Relevance improvement strategies:

1. Use Vertex AI for custom embedding training
2. Implement semantic chunking strategies
3. Use hybrid search approaches
4. Apply dynamic reranking
5. Implement feedback loops

6. Use query expansion techniques
7. Apply contextual embeddings
8. Implement cross-encoders
9. Use sliding window chunking
10. Apply document structure awareness

Q: What strategies can be employed to handle queries that require multi-hop reasoning in a RAG system with Vertex AI?

A: Multi-hop reasoning strategies:

1. Implement iterative retrieval using Cloud Functions
2. Use knowledge graphs in Neo4j on GCP
3. Apply query decomposition
4. Implement chain-of-thought prompting
5. Use intermediate reasoning steps
6. Store reasoning paths in Memorystore
7. Implement verification mechanisms
8. Use context accumulation
9. Apply recursive retrieval
10. Implement answer validation

Q: How would you implement comprehensive guardrails in a RAG system built on Google Cloud to ensure safety, reliability, and compliance?

A: Guardrail implementation:

1. Input Validation
   - Cloud Functions for query validation
   - Cloud DLP for content scanning
   - Rate limiting with Cloud Armor
2. Content Safety
   - Content filtering
   - Toxic content detection
   - Output validation
3. Security
   - IAM for access control

    - Cloud KMS for encryption
    - Security Command Center integration

4. Compliance
    - Audit logging
    - Data residency controls
    - Privacy controls
    - GDPR compliance features
5. Reliability
    - Circuit breakers
    - Quota management
    - Error handling
    - Fallback mechanisms
6. Monitoring
    - Cloud Monitoring
    - Custom metrics
    - Alert policies
    - SLO tracking

These guardrails ensure:

- Safe and appropriate content
- Regulatory compliance
- System reliability
- Data protection
- User trust
- Operational excellence