

# Cloud-Native Retrieval-Augmented Generation for Field Technicians in Energy Utilities

Michael Kerscher  
South East Technological University  
Waterford, Ireland  
20115673@setu.ie

## Abstract

Field technicians in energy utilities require timely access to safety-critical operational knowledge, yet documents such as SOPs, manuals and incident logs are fragmented across heterogeneous systems. Retrieval-Augmented Generation (RAG) can ground large language models (LLMs) in organisation-specific sources, but little is known about how managed cloud RAG services behave in utility contexts or how they compare with on-premises alternatives. This paper implements a cloud-native RAG architecture on Google Vertex AI and qualitatively evaluates its ingestion, retrieval and grounded-generation behaviour using a representative synthetic corpus. To contextualise these findings, a criteria-based comparison is conducted between managed (Vertex AI, Azure) and on-premises RAG deployments. Results indicate that managed RAG can ingest heterogeneous documents and produce coherent grounded responses with minimal engineering effort, while platform suitability depends strongly on ecosystem alignment, governance requirements and operational capacity. The study concludes that managed cloud RAG is technically feasible for utility field support but should be adopted with awareness of regional maturity and vendor-lock-in constraints.

## CCS Concepts

• Computing methodologies → Artificial intelligence.

## Keywords

retrieval-augmented generation, cloud computing, field service, energy utilities, mobile support

## ACM Reference Format:

Michael Kerscher. 2025. Cloud-Native Retrieval-Augmented Generation for Field Technicians in Energy Utilities. In *Proceedings of the Mobile Computing Master Conference*. ACM, New York, NY, USA, 5 pages. <https://doi.org/10.1145/nnnnnnnn.nnnnnnnn>

## 1 Introduction

Field technicians in energy utilities operate under time pressure and safety-critical conditions, yet essential operational knowledge standard operating procedures (SOPs), manuals, incident histories and GIS/SCADA information—is fragmented across heterogeneous systems and often inaccessible in the field. This fragmentation leads to inconsistent troubleshooting, inefficiencies and increased operational risk.

Large Language Models (LLMs) can process natural-language and multimodal inputs, but without access to organisation-specific knowledge their outputs remain generic or prone to hallucinations.

Retrieval-Augmented Generation (RAG) addresses this limitation by grounding model outputs in authoritative enterprise documents. While prior work analyses RAG architectures and retrieval strategies, little is known about the practical behaviour of *managed* cloud RAG services in industrial settings or how they compare with on-premises deployments in terms of governance, operational effort and suitability for medium-sized utilities.

This paper investigates these questions by implementing a cloud-native RAG architecture on Google Vertex AI and qualitatively evaluating its ingestion, retrieval and grounded generation behaviour using a representative synthetic corpus. To contextualise the findings, a criteria-based comparison is conducted between managed GCP RAG, managed Azure RAG and on-premises RAG.

The study is guided by three research questions:

- RQ1** How can fragmented utility knowledge be integrated into a cloud-native RAG architecture suitable for context-aware field support?
- RQ2** How does a managed RAG service behave with respect to ingestion, retrieval and grounded generation for representative utility-domain documents?
- RQ3** What architectural and operational considerations emerge when positioning managed cloud RAG (Vertex AI) relative to alternative deployment options such as Azure and on-premises RAG for medium-sized utilities?

**Hypothesis.** Building on the limitations identified in Project 1, this study explores the following hypothesis:

*Integrating enterprise knowledge into a cloud-native RAG architecture on Google Cloud will improve the contextual relevance and perceived reliability of LLM responses in utility field-support scenarios compared to baseline LLM usage without access to enterprise knowledge.*

The hypothesis is examined qualitatively through ingestion, retrieval and grounded-generation tests conducted with a representative synthetic corpus.

**Contributions.** This paper provides:

- (1) A domain-grounded architecture for cloud-native RAG in utility field support.
- (2) A qualitative empirical characterisation of a managed RAG workflow on Vertex AI.
- (3) A structured comparison of managed and on-premises RAG deployments based on six evaluation criteria.

## 2 Related Work

### 2.1 RAG for Knowledge-Intensive Settings

Retrieval-Augmented Generation (RAG) grounds large language models (LLMs) in domain-specific knowledge by retrieving relevant document segments and using them as evidence for generation [3]. Surveys identify retrieval quality (chunking, embeddings, ranking, corpus coverage) and grounding as key determinants of correctness and explainability [8]. Recent extensions investigate multimodal or structured retrieval using images, diagrams, sensor data and graph relations [1, 10]. These studies provide foundations for modern RAG pipelines, but they analyse technical components rather than the behaviour of *managed* cloud RAG services.

### 2.2 Cloud-Native vs. On-Prem RAG Platforms

Cloud-native platforms such as Google Vertex AI RAG and Azure AI Search with Azure OpenAI offer managed ingestion, embedding, vector search and LLM inference [9]. Their goal is to reduce operational burden through abstracted vector stores (e.g. RagManagedDb, Azure Cognitive Search). In contrast, on-prem or self-managed stacks combine open-source vector databases (Milvus, Qdrant, Weaviate, pgvector) with local or external LLMs to maximise control and data sovereignty [4]. Prior work highlights trade-offs between managed and self-managed deployments—notably lock-in, maturity variations and security risks [5, 7]. However, no study systematically compares these deployment options for utility field operations.

### 2.3 Knowledge and Field-Service Context in Utilities

Utilities operate distributed infrastructures where technicians depend on SOPs, manuals, incident logs and asset data. Prior research shows that these sources are often fragmented across ERP, SCADA/GIS and document repositories, causing inconsistent troubleshooting and inefficiencies [6]. Industrial AI studies emphasise multimodal assistance (photos, labels, diagrams) and context-aware reasoning based on spatial context, asset identifiers and sensor values [1, 2]. While this work motivates the need for unified and grounded access to heterogeneous knowledge, no existing research evaluates managed cloud RAG services for this domain or compares them with Azure- or on-premises RAG approaches. This gap motivates the contribution of this paper.

## 3 System & Methodology

### 3.1 Use Case & Knowledge Gaps

Field technicians in medium-sized energy utilities operate under time pressure and handle safety-relevant incidents in geographically distributed infrastructure. Required knowledge sources—SOPs, manuals, incident logs, GIS/SCADA outputs—are fragmented across heterogeneous systems and often inaccessible in the field. This leads to inconsistent troubleshooting, inefficiencies and reliance on tacit knowledge.

The envisioned mobile support application captures multimodal inputs (text, speech, images) and lightweight context (GPS, device identifiers) and forwards queries to a cloud-native RAG backend that returns grounded, actionable guidance.

*Typical scenarios.* Troubleshooting a malfunctioning asset; retrieving emergency procedures (e.g., gas leak); inspecting historical incident patterns.

*Core knowledge gaps motivating RAG..*

- fragmented information across ERP, SCADA/GIS and document repositories,
- slow or imprecise access to long SOPs and manuals,
- limited reuse of historical troubleshooting knowledge,
- missing integration of contextual data (GPS, asset ID, sensor values),
- unstructured field documentation,
- high cognitive load from combining heterogeneous sources manually.

RAG directly addresses these gaps through semantic retrieval and grounded response generation.

### 3.2 Requirements & Constraints

The RAG-based support system must meet fundamental requirements for field usability and cloud-native deployment.

*Functional requirements.*

- **Multimodal input:** text, speech-to-text and images.
- **Context capture:** GPS coordinates, timestamps, device identifiers.
- **Semantic retrieval:** SOPs, manuals and incident logs.
- **Grounded generation:** responses with source references.
- **Backend API & observability:** IAM-protected REST endpoint with logs for latency, errors and user interactions.

*Constraints.*

- strict access control via cloud Identity and Access Management (IAM),
- interactive latency suitable for field use,
- EU data residency requirements,
- elastic scalability through managed services.

### 3.3 System Architecture

The system follows a three-layer cloud-native architecture optimised for low operational overhead and separation of concerns.

*Application Layer.* Mobile client capturing multimodal inputs and context.

*Integration Layer.* A lightweight backend (Cloud Run) performs request validation, metadata enrichment and IAM-authenticated calls to the RAG API.

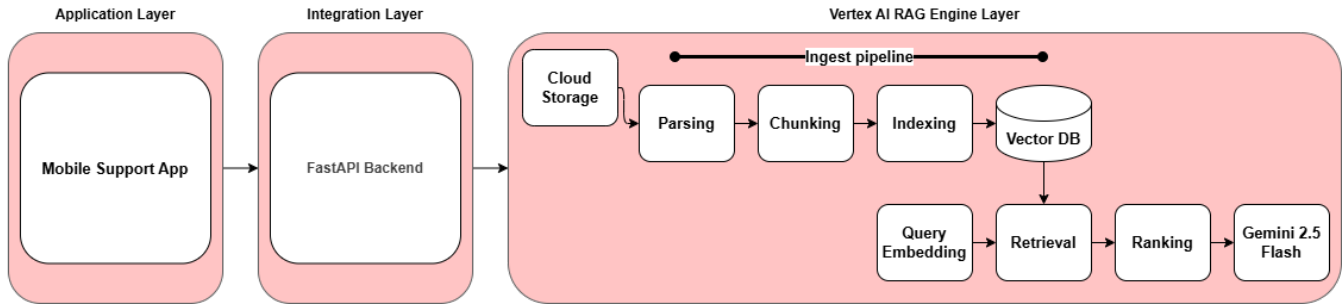
*RAG Engine Layer.* A managed service executes ingestion (parsing, chunking, embedding) and provides vector retrieval and grounded LLM generation. The prototype ran on Vertex AI in europe-west3 using text-embedding-005 for all embeddings.

### 3.4 Ingestion Pipeline

The ingestion pipeline (Fig. 1) follows the RAG structure:

*Storage* → *Parsing* → *Chunking* → *Embedding* → *Vector Indexing*

## Cloud-Native RAG Architecture (Google Cloud Vertex AI)



**Figure 1: Cloud-native architecture consisting of a mobile support app, a lightweight backend and a managed RAG engine on Vertex AI.**

Documents in TXT, CSV, JSON and YAML format were parsed using the default Vertex AI extractors. Custom chunking was applied with a chunk size of 512 tokens and an overlap of 100 tokens. All chunks were embedded using text-embedding-005 and stored in a managed RagManagedDb vector database. No custom tokenisation rules, filters or preprocessing steps were used.

### 3.5 Implementation Environment & Procedure

The prototype was implemented on Google Cloud in the project gemini-support-app using Vertex AI in region europe-west3 (Frankfurt). All experiments were executed in Cloud Shell with the Python Vertex AI SDK.

#### Environment.

- Google Cloud project: gemini-support-app.
- Region: europe-west3 (Vertex AI, managed RAG).
- Embedding model: text-embedding-005.
- Generative model: gemini-2.5-flash.
- Vector backend: managed RagManagedDb.

**Corpus creation and import.** A RAG corpus with display name energy-support-corpus was created programmatically using the Vertex AI RAG SDK. The corpus was backed by RagManagedDb and configured to use text-embedding-005 for all embeddings. Documents were stored in a Cloud Storage bucket and imported via the managed ingestion API. During import, the custom chunking configuration (chunk size 512, overlap 100) was applied and all resulting chunks were embedded and indexed automatically.

**Retrieval query.** To probe retrieval behaviour, a German emergency procedure query was submitted:

*"Wie wird bei einem Gasleck laut Notfallprozedur reagiert?"*

The retrieval API was called with  $\text{top}_k = 3$  and no re-ranking or metadata filters. Returned contexts were printed and inspected manually to assess whether the top-ranked chunks corresponded to the relevant SOP sections.

**Grounded generation.** For grounded generation, a retrieval tool over the same corpus was attached to gemini-2.5-flash. The model was invoked with a semantically equivalent German query

and automatic grounding enabled. The generated answer was inspected for step-wise correctness, alignment with the retrieved SOP content and absence of hallucinated actions.

**Summary.** These steps define a minimal, reproducible procedure for recreating the prototype behaviour: (i) provision a managed RAG corpus with text-embedding-005, (ii) import a synthetic SOP/manual corpus from Cloud Storage with the specified chunking configuration, (iii) issue realistic field-support queries in German with  $\text{top}_k = 3$ , and (iv) inspect retrieved contexts and grounded responses qualitatively.

### 3.6 Evaluation Design

The evaluation adopts a structured but qualitative design centred on technical feasibility. Three dimensions were assessed:

- **Ingestion feasibility:** ability to parse and embed heterogeneous documents without custom pipelines.
- **Retrieval plausibility:** whether retrieved chunks correspond to relevant SOP sections for realistic field queries.
- **Groundedness:** whether generated responses remain aligned with retrieved context.

Semantic retrieval used  $\text{top}_k = 3$  with no re-ranking, no metadata filters and no scoring thresholds. Grounded generation used gemini-2.5-flash with default decoding parameters (temperature,  $\text{top}_p$ , max tokens) via the Vertex AI RAG tool wrapper, with automatic grounding enabled.

In scientific terms, the independent variable was the query type (e.g., emergency procedure, maintenance request). Dependent variables were retrieval plausibility and groundedness of responses. Controlled variables included corpus content, embedding model and chunking setup. Uncontrolled variables included cloud-region latency fluctuations and internal service behaviour.

### 3.7 Dataset Limitations

The prototype uses a synthetic corpus representing SOPs, manuals and incident logs. While adequate for validating technical ingestion and retrieval behaviour, such data lacks the noise, inconsistencies and versioning issues typical in real utility documentation. Findings

should therefore be interpreted as technical feasibility insights rather than as operational performance benchmarks.

### 3.8 Evaluation Framework

To position the behaviour of Vertex AI RAG relative to Azure RAG and on-premises RAG, a qualitative evaluation framework was developed grounded in two sources: (i) platform-comparison dimensions commonly used in cloud-native AI literature, and (ii) organisational constraints specific to medium-sized utilities.

The framework consists of six criteria:

- (1) **Architecture:** degree of end-to-end integration, pipeline transparency, supported modalities, and flexibility of ingestion and retrieval components.
- (2) **Security & Compliance:** IAM mechanisms, auditability, data residency, and suitability for regulated infrastructures (e.g. EU-based utilities).
- (3) **Operational Complexity:** deployment effort, maintenance burden, monitoring requirements and the amount of DevOps expertise required.
- (4) **Cost & Scalability:** elasticity of vector search and model inference, cost transparency, and expected scaling behaviour under increasing load.
- (5) **Field Applicability:** latency expectations, robustness of ingestion pipelines, multimodal support and the ability to integrate contextual metadata such as GPS and asset identifiers.
- (6) **Strategic Alignment:** vendor lock-in risks, ecosystem compatibility (e.g. Microsoft 365 vs. Google Cloud), and long-term maintainability.

These criteria serve two methodological purposes:

- First, they provide a structured lens for interpreting the prototype results (ingestion, retrieval, groundedness) in terms of operational relevance for utilities.
- Second, they enable a principled comparison of Vertex AI RAG with Azure and on-premises RAG in the Discussion section without requiring full empirical benchmarking, which was beyond the scope of the study.

The framework does not prescribe numerical metrics; instead, it supports a qualitative assessment aligned with the feasibility-oriented nature of this research.

## 4 Results

### 4.1 Prototype Execution

The prototype provides a factual characterisation of the behaviour of a managed cloud-native RAG system on Google Vertex AI across ingestion, retrieval and grounded generation.

**4.1.1 Corpus Creation.** A synthetic corpus of SOPs, manuals, checklists and incident notes (TXT, CSV, JSON, YAML) was ingested into Vertex AI RAG. The system (i) created the corpus and metadata structures, (ii) parsed all file types without custom preprocessing, (iii) produced token-overlapping chunks using the configured parameters (512 tokens, 100-token overlap), and (iv) embedded all chunks using text-embedding-005. These steps confirm that the ingestion and indexing pipeline processed all document types successfully.

**4.1.2 Retrieval Behaviour.** For the German emergency-procedure query, the retriever returned (i) the correct SOP segment as the top-ranked result, (ii) secondary segments referencing related procedures, and (iii) no irrelevant items among the top 3 results. This demonstrates that the retrieval component accessed and ranked relevant passages for a structurally simple operational query.

**4.1.3 Grounded Generation.** Using the retrieved chunks, Gemini 2.5 Flash produced step-wise procedural guidance consistent with the SOP text. All generated responses referenced retrieved information; no hallucinations or unsupported steps were observed. Generation latency was subjectively perceived as interactive, though no measurements were recorded.

**4.1.4 Summary of Findings.** Across ingestion, retrieval and grounded generation, the managed RAG pipeline:

- processed heterogeneous document formats without custom ETL,
- retrieved the correct SOP passage for a realistic field query,
- produced grounded and procedure-aligned guidance,
- operated with interactive responsiveness.

These results constitute qualitative evidence of the technical feasibility of using managed cloud RAG for operational knowledge access. Quantitative performance comparisons were out of scope.

## 5 Discussion

### 5.1 Interpretation of Findings

The results demonstrate that a managed cloud RAG service can integrate fragmented utility knowledge into a unified retrieval and grounding workflow. The successful ingestion of heterogeneous file types and the correct retrieval of SOP segments indicate that default parsing, chunking and embedding configurations are sufficient to support basic operational queries. In contrast to prior work that focuses on optimising retrieval pipelines [3, 8], these findings show that end-to-end RAG behaviour in managed platforms can support operational use cases even without custom model tuning.

The grounded responses generated by Gemini 2.5 Flash further show that managed RAG pipelines can provide procedure-aligned output for safety-relevant tasks—an observation that complements research on multimodal and context-aware field-support systems [2, 6]. However, the absence of noise, inconsistencies and versioning in the synthetic corpus limits the generalisability of this behaviour.

### 5.2 Principles and Relationships

Three main relationships emerge:

- (1) **Fragmentation vs. grounding:** RAG effectively unifies SOPs, manuals and incident notes that are otherwise distributed across heterogeneous systems.
- (2) **Integration vs. performance:** System usefulness depends more on ingestion robustness and retrieval quality than on LLM generation capabilities.
- (3) **Platform alignment vs. suitability:** The choice between Vertex AI, Azure or on-prem RAG is driven by ecosystem alignment, governance demands and operational capacity.

### 5.3 Exceptions and Unsettled Points

Two limitations create uncertainty: (i) region-specific maturity issues observed in europe-west3, and (ii) the lack of quantitative retrieval and latency metrics. These unsettled points prevent general claims about comparative performance.

### 5.4 Implications for Energy Utilities

For medium-sized utilities, managed cloud RAG offers a practical starting point for integrating operational knowledge. Adoption should proceed incrementally—beginning with SOPs and manuals—while establishing governance mechanisms for document eligibility, access control, versioning and traceability. These implications align with digital-transformation findings emphasising the centrality of workflow and governance structures.

### 5.5 Conclusions from the Findings

Taken together, the results indicate that managed cloud RAG is a technically feasible mechanism for addressing fragmented operational knowledge in utility field support, provided that (i) ingestion workflows are stable, (ii) governance is established and (iii) ecosystem alignment is considered. These conclusions are supported by the observed ingestion robustness, retrieval plausibility and groundedness of responses in the prototype.

### 5.6 Interpretation with Respect to the Hypothesis

The findings support the hypothesis qualitatively: managed cloud RAG improved the contextual relevance and procedural correctness of responses by grounding output in retrieved SOP segments. While these observations indicate that enterprise knowledge integration enhances response quality compared to baseline LLM behaviour, quantitative confirmation remains future work.

## 6 Conclusions and Outlook

### 6.1 Contributions

This paper contributes three top-level achievements:

- (1) **A cloud-native architecture for field-support RAG:** a domain-grounded design that integrates fragmented utility knowledge via managed ingestion, vector retrieval and grounded generation on Vertex AI.
- (2) **A qualitative empirical characterisation of managed RAG behaviour:** evidence that default parsing, chunking and embedding configurations successfully ingest heterogeneous SOPs, manuals and incident logs and enable relevant retrieval and grounded generation.
- (3) **A qualitative positioning of managed (Vertex AI) RAG:** relative to Azure and on-premises RAG deployments, based on architectural and governance criteria reported in prior work.

These contributions provide feasibility-oriented insights into how cloud-managed RAG can support context-aware operational knowledge access.

### 6.2 Limitations

The study has several limitations. First, the evaluation uses a small synthetic corpus that does not reflect the noise, versioning, formatting inconsistencies or access-control constraints of real utility documentation. Second, the analysis focuses on a single cloud region and a single managed RAG provider; results may differ in other configurations. Third, no quantitative retrieval or latency metrics were recorded, and no user evaluation was conducted, limiting statements about performance, usability or workflow impact. Finally, the comparison with Azure and on-prem RAG is literature-based rather than empirically benchmarked.

### 6.3 Future Work

Future research should apply the RAG architecture to real enterprise datasets (SOPs, manuals, GIS/SCADA outputs, incident logs) to study ingestion robustness under realistic conditions. Enhancing context-awareness—e.g., via GPS, asset IDs or sensor values—and evaluating multimodal diagnostic workflows (images, diagrams, error codes) are promising directions. Incorporating bidirectional knowledge flows, where technician notes are summarised and reinjected into enterprise repositories, may support continuous documentation quality. Finally, systematic benchmarking of retrieval metrics, groundedness, latency and operational cost across managed and on-prem RAG deployments is needed to complement the qualitative findings and guide platform selection for utilities.

Overall, the qualitative findings provide initial support for the hypothesis and answer RQ1–RQ3, demonstrating the feasibility of using managed cloud RAG for integrating fragmented operational knowledge.

## References

- [1] Khalid M Alsaif, Aiiad A Albeshri, Maher A Khemakhem, and Fathy E Eassa. 2024. Multimodal large language model-based fault detection and diagnosis in context of industry 4.0. *Electronics* 13, 24 (2024), 4912.
- [2] Xuefeng Chen, Yaguo Lei, Yanfu Li, Simon Parkinson, Xiang Li, Jinxin Liu, Fan Lu, Huan Wang, Zisheng Wang, Bin Yang, et al. 2025. Large Models for Machine Monitoring and Fault Diagnostics: Opportunities, Challenges, and Future Direction. *Journal of Dynamics, Monitoring and Diagnostics* 4, 2 (2025), 76–90.
- [3] Mingyue Cheng, Yucong Luo, Jie Ouyang, Qi Liu, Huijie Liu, Li Li, Shuo Yu, Bohou Zhang, Jiawei Cao, Jie Ma, et al. 2025. A survey on knowledge-oriented retrieval-augmented generation. *arXiv preprint arXiv:2503.10677* (2025).
- [4] Yikun Han, Chunjiang Liu, and Pengfei Wang. 2023. A comprehensive survey on vector database: Storage and retrieval technique, challenge. *arXiv preprint arXiv:2310.11703* (2023).
- [5] Xinyi Hou, Jiahao Han, Yanjie Zhao, and Haoyu Wang. 2025. Unveiling the Landscape of LLM Deployment in the Wild: An Empirical Study. *arXiv preprint arXiv:2505.02502* (2025).
- [6] Olli Lehtonen, Timo Ala-Risku, and Jan Holmström. 2012. Enhancing field-service delivery: the role of information. *Journal of Quality in Maintenance Engineering* 18, 2 (2012), 125–140.
- [7] Yao Lu, Song Bian, Lequn Chen, Yongjun He, Yulong Hui, Matthew Lentz, Beibin Li, Fei Liu, Jialin Li, Qi Liu, et al. 2024. Computing in the era of large generative models: From cloud-native to AI-native. *arXiv preprint arXiv:2401.12230* (2024).
- [8] Agada Joseph Oche, Ademola Glory Folashade, Tirthankar Ghosal, and Arpan Biswas. 2025. A systematic review of key retrieval-augmented generation (rag) systems: Progress, gaps, and future directions. *arXiv preprint arXiv:2507.18910* (2025).
- [9] Dhavalkumar Patel, Ganesh Raut, Satya Narayan Cheetirala, Girish N Nadkarni, Robert Freeman, Benjamin S Glicksberg, Eyal Klang, and Prem Timsina. 2024. Cloud platforms for developing generative AI solutions: a scoping review of tools and services. *arXiv preprint arXiv:2412.06044* (2024).
- [10] Qinggang Zhang, Shengyuan Chen, Yuanchen Bei, Zheng Yuan, Huachi Zhou, Zijin Hong, Hao Chen, Yilin Xiao, Chuang Zhou, Junnan Dong, et al. 2025. A survey of graph retrieval-augmented generation for customized large language models. *arXiv preprint arXiv:2501.13958* (2025).