Assistant for Effortless Cloud Resource Management and Experimentation

Carmel Prosper Sagbo, Ellon Berhanu, Birhanu Shimelis Girma, Ngoga Alexis
Carnegie Mellon University Africa, Kigali, Rwanda
Email: {csagbo, eberhanu, bgirmash, nalexis}@andrew.cmu.edu

I. Abstract

***Abstract*—With the rise of AI and cloud technologies, social goods services providers and startups, as well as big companies, have the requirements to deliver high-performance and efficient solutions to their clients. However, with the increasing demand and requirements, it is challenging for these service providers or individuals to run their experiments on-premise or using local resources due to the high financial cost, lack of time, and quality of specialists in infrastructure management. It is therefore important to find an approach for them to cost-effectively and efficiently analyze different architecture propositions to effortlessly run their business within budget limits and systems requirements. The current solution leverages state-of-the-art solutions through Large Language Models (LLMs) to create a user-friendly web platform that assists startups and company developers in generating effortless architecture fitting a limited budget with high performance using the recommended solution based solely on AWS cloud services. It combines knowledges from Cloud providers documentation data and the Retrieval Augmented Generation technique to provide AI-assisted recommendation for data storage mainly on AWS.**

Github Repository, **Video Link**

II. Introduction

Cloud technologies and platforms have matured significantly in the last decades, offering a variety of services with critical components that reduce infrastructure and human resource management costs [1]. These advancements enable a seamless transition from on-premise systems to cloud-based or hybrid environments [2]. This shift represents a pivotal change in how developers and startups deliver their services to end users using cloud computing architectures. The cloud paradigm offers a model characterized by on-demand, low-cost shared resources capable of handling high availability, scalability, and low latency in minimal time [3]. However, achieving these benefits requires expertise in efficiently allocating resources such as memory, storage, and bandwidth to various applications [3].

Efficient resource management is a significant challenge for cloud service providers (CSPs) and an even greater obstacle for individuals or organizations lacking cloud computing expertise. Cloud computing offers a wide variety of solutions, empowering developers and IT enthusiasts to realize their projects and social good initiatives. However, the first step toward building these solutions often involves hiring experts. This becomes problematic when immediate expertise is needed or when financial and time constraints limit the feasibility of hiring consultants. As [3] discuss, cloud resource management plays a crucial role in product management for service providers, yet it demands a deep understanding of the documentation and tools provided by each CSP.

The emergence of artificial intelligence (AI) technologies, enabled by cloud computing, has unlocked powerful capabilities for solving complex problems, including those related to resource management. Among these innovations, large language models (LLMs) offer a remarkable ability to process information and provide actionable insights, even to individuals without prior expertise. For example, [4] highlight how AI models can ingest and analyze massive volumes of data quickly. Similarly, GPT-based models demonstrate human-like capabilities in understanding and generating text across a wide range of tasks [5].

In this context, LLMs present a compelling solution to cloud resource management challenges for non-experts. By processing project requirements and applying advanced algorithms alongside existing domain knowledge, these models can generate customized recommendations and go beyond by executing code. The concept of Retrieval-Augmented Generation (RAG) further enhances this capability by combining the extensive knowledge encoded in LLMs with real-time, domain-specific information from external sources. As [5] explain, RAG enables more accurate, relevant, and timely responses, making it a transformative tool for addressing cloud resource management issues.

III. Objectives

- Simplify the process of cloud resource management for users without specialized expertise.
- Provide cost-efficient and optimized recommendations for data storage architectures.
- Automate deployment and provisioning of cloud resources.
- Ensure system reliability, scalability, and security.

IV. System Overview

The system integrates multiple components to provide effortless cloud resource management and experimentation as can be seen in the system architecture in Fig 1:

- **Chat Agent:** Processes and interprets user input using natural language understanding capabilities. This component acts as the primary interface for users to submit

cloud resource management requests, enabling intuitive interaction with the system's backend services.

- **RAG Vector Store:** Utilizes a sophisticated retrieval mechanism to fetch relevant cloud documentation and service information. The vector store enables context-aware responses by dynamically retrieving and matching user queries with the most relevant cloud service documentation.
- **API Agent Tool** Handles critical backend operations, facilitating communication with cloud APIs and managing the deployment and configuration of cloud resources. This tool ensures seamless interaction between the user's requirements and AWS service provisioning.
- **Backend API:** Serves as the critical interface with AWS, translating user requests into specific resource deployment instructions. It manages the authentication, validation, and execution of cloud resource provisioning processes.
- **End User Interface:** Provides an intuitive chat-based platform that allows users to interact with the system, submit project requirements, and receive architectural recommendations. The interface supports a conversational approach to cloud resource management.
- **AWS Services:** Represents the core infrastructure resources such as S3 and DynamoDB that can be dynamically configured and deployed based on user requirements.

The necessity of these tools becomes apparent when examining the current landscape of cloud infrastructure management. Traditional approaches often require extensive technical expertise, creating significant barriers for organizations and individuals with limited cloud computing knowledge. Cloud resource management presents complex challenges: navigating intricate service configurations, understanding cost-optimization strategies, and managing scalable deployments demand specialized skills that most organizations cannot easily acquire. This system addresses these challenges by leveraging LLM-powered recommendations and a structured, intelligent approach to cloud resource provisioning. By integrating advanced retrieval techniques with cloud service APIs, the platform transforms complex infrastructure decisions into intuitive, guided experiences. The result is a solution that breaks down technical barriers, enabling startups, individual developers, and resource-constrained organizations to design, deploy, and manage cloud architectures with unprecedented ease and confidence.

This architecture combines advanced AI-driven recommendation techniques with cloud service APIs to simplify and streamline the process of cloud resource management, making it accessible to users without specialized infrastructure expertise.

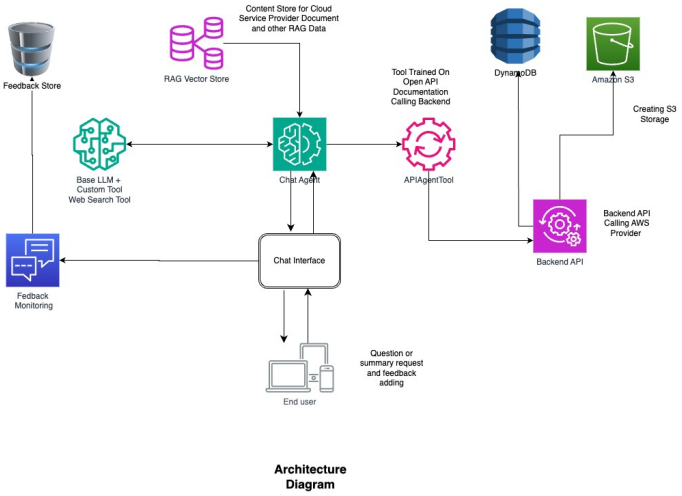

Fig. 1. System Architecture Components

V. Data

A. Description and Processing

Large Language Models (LLMs), such as GPT-3.5 or GPT-4, are pre-trained models on vast datasets and are capable of completing a wide range of tasks. However, delivering accurate and context-specific responses requires these models to access domain-specific knowledge. Given the rapidly evolving nature of the tech domain, particularly in software engineering, LLMs must be provided with the latest information and appropriate data to meet the needs of current users and startup products.

To address this, the approach implemented here relies on Retrieval-Augmented Generation (RAG), which works by integrating external data sources. The current solution incorporates scraped web data and whitepaper documentation in PDF format for AWS products, such as:

- AWS S3 documentation [6]
- AWS RDS documentation [6]
- AWS Dynamo DB documentation [6]
- Terraform AWS documentation
- Cloud computing crash course content from Carnegie Mellon University

These documents are pre-processed, tokenized, and stored in an online MongoDB cluster. These data provide the extracted content as a chunk of text data, which creates a search index using the MongoDB indexing technique. This enables similarity check queries on existing knowledge. For real-time updates and accurate cost analysis, the implemented LLM agent uses a search tool called the Tavily Search Tool. This tool provides up-to-date recommendations and insights, particularly for dynamic cost calculations.

VI. Model and Techniques

A. Large Language Model for Cloud Resource Management

At the core of the system is OpenAI's GPT-3.5 [7], integrated via LangChain to provide AI-driven recommendations for cloud resource management. This LLM is specifically

tailored to interact seamlessly with backend tools and the Retrieval-Augmented Generation (RAG) framework. By leveraging custom prompts designed using a ChatPromptTemplate, the system effectively guides the LLM in generating recommendations for architecture design, cost estimation, and deployment processes. These prompts include precise instructions for tasks such as selecting cloud storage services, optimizing costs, and automating resource provisioning, ensuring accurate, context-aware, and actionable responses.

B. Retrieval-Augmented Generation (RAG) Framework

The RAG framework enhances the system's capability to provide accurate recommendations by integrating real-time, domain-specific knowledge from AWS documentation and additional resources. Fig 2 shows the sequence diagram and the workflow is divided into two main phases.

- Knowledge Ingestion: Cloud provider documentation (AWS) is segmented into manageable chunks based on topics such as S3, DynamoDB, and cost optimization strategies. These chunks are embedded using vectorization and stored in a FAISS (Facebook AI Similarity Search) vector database for efficient retrieval.
- Query Processing: When a user submits a query through the chat interface, it is vectorized and matched against the stored knowledge base. Relevant document chunks are retrieved, forming a context-rich prompt for the LLM. This ensures that responses are grounded in the latest and most relevant documentation.

The RAG architecture significantly mitigates the risk of hallucinations and enhances the system's reliability in generating domain-specific recommendations.

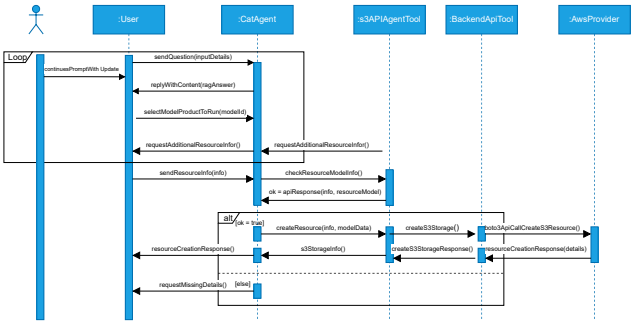

Fig. 2. Sequence diagram of the cloud resource management system

C. Architecture Recommendation Tool

This tool uses predefined templates and user-provided inputs, such as project requirements and budget constraints, to recommend optimized cloud architectures. The process involves:

- Input Collection: User inputs (e.g., data size, expected workload, and budget) are submitted via the chat interface.
- Template Matching: The system matches inputs to predefined templates for AWS services (e.g., S3, DynamoDB).
- Custom Recommendations: Tailored architectures are generated using the RAG-enhanced knowledge base and validated against AWS best practices.

D. Cost Projection Tool

The cost projection tool provides real-time estimates for user-selected cloud architectures. It calculates costs using AWS pricing APIs and preloaded data, considering factors such as data transfer, storage size, and compute time.

The results are visualized in an interactive dashboard, enabling users to compare multiple options and make informed decisions.

E. Resource Deployment Automation

Resource deployment is handled by an API agent integrated with AWS APIs. The agent generates configuration files (e.g., JSON, YAML) for services such as S3 and DynamoDB and automates provisioning. The workflow includes:

1) Validation: User inputs are validated against AWS service constraints.
2) Configuration File Generation: Templates are auto-filled with user inputs to create deployment files.
3) Execution: Resources are deployed programmatically using the AWS SDK.

F. Interactive Feedback System

An interactive feedback module collects user responses to recommendations and deployment outcomes as shown in Fig 3. Features include:

- Feedback Capture: Users can rate responses and provide comments via thumbs-up/down icons or free-text input.
- Visualization Dashboard: Aggregated feedback is displayed to system administrators, enabling continuous improvement.

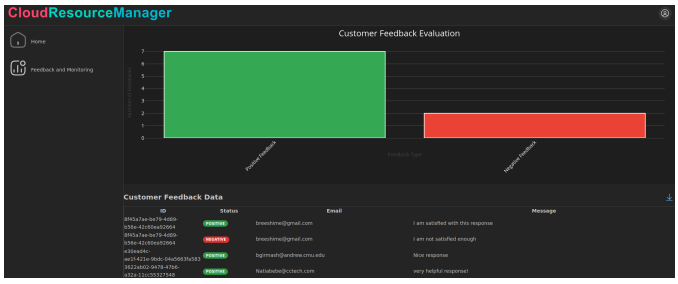

Fig. 3. Feedback and monitoring dashboard that shows the real time user feedback to the chat agent response

G. Dashboard and Visualization

Built with Dash, Fig 4. shows the dashboard that visualizes the services available to the users.

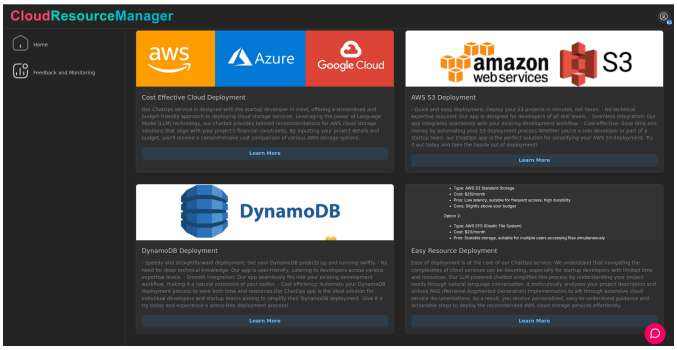

Fig. 4. Dashboard of the cloud resource management system

VII. User Guide

The Cloud Resource Management system has been designed to simplify and optimize the process of deploying and managing cloud resources. This guide provides a comprehensive overview of how users can interact with the platform and leverage its features to achieve their goals efficiently.

A. Purpose of the Software

The platform assists users in generating and deploying cloud architectures tailored to their specific project descriptions, such as budget constraints and performance requirements. It also provides cost projections, facilitates deployment automation, and ensures scalability and security throughout the process.

B. Accessing the System

To access the platform, users must first register by creating an account. The registration process requires basic information such as name, email address, a secure password, resource platform (AWS, GCP, Azure), Company name, address, and AWS credentials. Once registered, users can log in with their credentials. For security purposes, access is restricted to authorized individuals only.

Upon successful login, users are presented with the main interface, which consists of a chat-based interaction area and a menu (Home) providing access to additional functionalities such as feedback and monitoring system.

C. Using the Chat Interface

The chat interface serves as the primary point of interaction for users. It allows them to submit queries and receive recommendations in a conversational format. Users can:

- Request architecture recommendations by specifying details such as storage requirements, workload type, and cost constraints.
- Inquire about the costs of specific cloud services, such as S3 or DynamoDB.
- Receive guidance on deploying resources automatically with pre-configured templates.

For example, a user may input, ”What is the most cost-efficient storage option for 1TB of data with high retrieval frequency?” The system responds with tailored recommendations and associated cost projections, enabling the user to make informed decisions.

D. Generating and Deploying Architectures

To generate a custom architecture, users are guided step-by-step by the system. After entering project-specific details, the platform generates an optimized architecture, considering factors such as performance, scalability, and cost. Users can review the recommendations, which include detailed resource specifications and associated costs.

Once the architecture is finalized, users can proceed to deploy resources. The system automatically initiates the process via the AWS back-end APIs. Upon completion, the platform provides users with access credentials, deployment logs, and verification details.

E. Analyzing Costs and Projections

The cost analysis feature allows users to visualize the financial implications of their selected architectures. For instance, if a user is evaluating storage options, the dashboard displays cost comparisons between S3 and DynamoDB based on the specified workload.

F. Providing Feedback

The system includes an integrated feedback mechanism, allowing users to evaluate the accuracy and relevance of the recommendations provided. Users can rate their experience using thumbs-up/down icons or leave detailed comments. This feedback is aggregated and displayed on the feedback and monitoring dashboard and saved to dynamoDB for further processing, enabling continuous improvements to the platform.

G. Troubleshooting Common Issues

If users encounter errors during deployment, the platform provides interactive troubleshooting prompts. For example, if a deployment fails due to insufficient AWS account permissions, the system notifies the user and provides guidance on resolving the issue.

H. Example Use Case

A startup requires a cost-efficient storage solution for hosting 1TB of data while maintaining frequent access. Using the system, the user:

1) Logs into the platform and submits project requirements via the chat interface.
2) Reviews the generated architecture, which recommends S3 with lifecycle policies to optimize costs.
3) Deploys the resources using the backeng APIs.
4) provides resource deployment confirmation for their records.

The process is completed in minutes, saving the startup significant time and effort compared to manual configuration.

I. Summary

This platform streamlines cloud resource management by offering an intuitive interface, automated deployment tools, and robust cost analysis capabilities. Users can achieve their goals without requiring advanced expertise in cloud infrastructure, making the system an invaluable asset for businesses and developers alike.

VIII. Resources Needs and Provider

Computing:

- AWS Storage resources.
- Access to the resources that are not available using free tiers of each cloud service are expected to be provided by the client (Prof. Charlie Wiecha).

OpenAI Access:

- Provided by the client already.

IX. Plan for Success

1) Measuring Outcomes Criteria: Note: Every milestone verification numbering corresponds to the milestone number in the previous table from the Project Plan section. The details are explained in Table I. All sharable deliverables will be communicated with the client on the team's Slack group chat and email with Prof. Charlie Wiecha (cwiecha@andrew.cmu.edu)

X. Conclusion

This handbook outlines a novel approach to simplifying cloud resource management using AI. By automating recommendations and provisioning, the system reduces complexity, enhances efficiency, and lowers operational costs. Future work includes expanding compatibility with other cloud platforms and integrating advanced analytics for further optimization.

References

[1] T. Khan, W. Tian, and R. Buyya, "Machine Learning (ML)-Centric Resource Management in Cloud Computing: A Review and Future Directions," May 2021, arXiv:2105.05079 [cs]. [Online]. Available: http://arxiv.org/abs/2105.05079

[2] E. Afgan, A. Lonie, J. Taylor, and N. Goonasekera, "CloudLaunch: Discover and Deploy Cloud Applications," May 2018, arXiv:1805.04005 [cs]. [Online]. Available: http://arxiv.org/abs/1805.04005

[3] S. R. Swain, A. K. Singh, and C. N. Lee, "Efficient Resource Management in Cloud Environment," Jun. 2022, arXiv:2207.12085 [cs]. [Online]. Available: http://arxiv.org/abs/2207.12085

[4] "AI, Cloud computing, Machine learning, Resource management, Automation, Auto scaling," *Computer Science and Engineering*, 2024.

[5] A. Ramachandran, "Retrieval augmented generation (rag) for large language models leveraging enterprise data (sap, salesforce, workday)," 08 2024.

[6] AWS documentation overview. [Online]. Available: https://aws.amazon.com/documentation-overview/

[7] OpenAI platform. [Online]. Available: https://platform.openai.com

TABLE I
DESCRIPTION AND SCHEDULE OF MAJOR ACTIVITIES, MILESTONES, AND DELIVERABLES.

Activity	Milestone	Hours
Generation of recommended and efficient architecture for AWS S3 based user needs	1) S3 bucket for storing cloud documentations and manuals 2) AWS S3 documentation vectorized and stored on MongoDB vector database 3) A RAG agent that can generate architectural recommendations for S3 related user project descriptions 4) A user interface built by using Dash and RAG integrated copilot with Chainlit	115
Deployment of chosen architectures for the user	1) A custom tool that can communicate with the backend 2) A backend API that deploys S3 resource on AWS	76
Generation of recommended and efficient architecture for DynamoDB database service based user needs	1) DynamoDB documentation added to existing raw data storage bucket for storing cloud documentations and manuals 2) DynamoDB documentation vectorized and stored on the existing MongoDB vector database 3) Integration of a new custom tool with the existing RAG agent for generating architectural recommendations that includes DynamoDB service options 4) A backend API that deploys DynamoDB resource on AWS	50
Provide deployment option for a chosen architecture from both AWS S3 and DynamoDB storage services based on user project description	1) A backend API that deploys either S3 or DynamoDB resources on the user provided AWS Account	50
Providing cost comparison for different storage architectures from AWS.	1) Summarized text description of cost comparison for each storage architectures	50
Generation of a detailed report on the cost projections for a chosen cloud architecture resources	1) Detailed downloadable report for cost projection of the chosen storage architecture	50
Delivering documentation and user manual for effective use of the cloud resource management system.	1) User manual 2) Demo of user interaction, showcasing the product's capabilities 3) Final report summarizing the project development process and outcomes 4) GitHub repository with proper README and guidance for developers	50
Collecting feedback from user on satisfaction of recommended deployment architectures and deployment process	1) Allowing feedback options from the chat interface by using positive and negative icons. 2) Enabling comment for each response of the Agent. 3) UI design and development of customer feedback visualization by graph and table and download 4) Downloadable user feedback data as CSV	50
Quality Assessment of mechanism for the service	1) Comparison with existing resource deployment methods such as AWS console and CLI performed	50
Testing procedures completed	1) Sample project descriptions for testing prepared 2) Expected architectural and design responses for each project description drafted 3) Comparison completed and results are analyzed	50

TABLE II
MILESTONE VERIFICATION TABLE FOR ACTIVITIES AND DELIVERABLES.

Activities (Deliverables)	**Milestone Verification Checks for Success**
Generation of recommended and efficient architecture for AWS S3-based user needs	1) S3 Bucket created, cloud documentations stored and access to the resource shared within the development team 2) Vector DB created and data from S3 fetched and vectorized 3) Verified with Demo of RAG agent and interactions with the user through the user interface
Generate and implement a recommended Data Storage with custom Credentials	1) Deployment of S3 API call script to be used by LLM tool 2) Verified when S3 resource deployment is successful
Generate and implement a recommended AWS Data storage system architecture with custom Credentials	1) S3 Bucket documentations 2) Vector DB updated, documentation fetched and vectorized 3) Verified with Demo of updated RAG agent and interactions with the user demonstrating the functionality on AWS
Providing cost comparison for different storage architectures from AWS	1) At least 3 AWS storage options compared 2) Cost breakdown per GB/month included 3) Access frequency per GB/month included 4) Demo of summarized text description based on user project description query
Provide cost projections with detailed reports for a chosen cloud data storage architecture	1) Verified by providing monthly cost estimates and itemized breakdown of AWS storage costs 2) Demo of showing detailed cost projection for user-chosen architecture
Delivering documentation and user manual for effective use of the cloud resource management system	1) PDF User manual produced 2) Video of demo produced 3) PDF final report produced 4) Project GitHub repository shared with the client