Unified Self-Service Cloud Portal with Cloud Connectors for Multi-Platform Orchestration

Aryan Kothari, V.R.N.S Nikhil, Namana Rohit, Beena B. M
Department of Computer Science and Engineering
Amrita School of Engineering, Bengaluru
Amrita Vishwa Vidyapeetham, India
bl.en.u4aie22071@bl.students.amrita.edu, bl.en.u4aie22005@bl.students.amrita.edu
bl.en.u4aie22062@bl.students.amrita.edu, bm_beena@blr.amrita.edu

***Abstract*—In today's rapidly evolving technological landscape, organizations increasingly rely on multi-cloud environments to leverage the best features of various cloud service providers. Managing resources across different platforms like AWS, Azure, Google Cloud poses significant challenges due to the heterogeneity of interfaces and services. This paper presents a unified self-service cloud portal empowered by custom-built cloud connectors, enabling seamless orchestration and management of cloud resources across multiple platforms. The solution emphasizes green computing principles to optimize resource utilization and reduce environmental impact.**

***Index Terms*—Cloud Computing, Multi-Cloud Management, Cloud Connectors, Self-Service Portal, Green Computing, Resource Orchestration.**

I. Introduction

The advent of cloud computing has revolutionized how organizations deploy, manage, and scale their IT infrastructure. With a plethora of cloud service providers (CSPs) like Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform (GCP), enterprises often adopt a multi-cloud strategy to avoid vendor lock-in and optimize costs . However, this approach introduces complexity in managing disparate platforms, each with its own interfaces, APIs, and management tools [**?**].

The lack of a unified management interface leads to increased operational overhead, security risks, and inefficiencies. Administrators and users must navigate different portals, learn various APIs, and develop platform-specific expertise [**?**]. This fragmentation hampers agility and contradicts the cloud's promise of streamlined operations.

To address these challenges, we propose a unified self-service cloud portal that leverages custom cloud connectors to orchestrate tasks across multiple cloud platforms. By abstracting the underlying complexities, the portal offers a single pane of glass for resource management, enhancing productivity and reducing the learning curve for users.

II. Objective

The primary objective of this project is to develop a unified self-service cloud portal that enables users to manage and orchestrate cloud resources across multiple platforms seamlessly. The specific aims include:

- **Develop Cloud Connectors**: Create modular connectors for AWS, Azure, GCP, OpenStack, and VMware that interface with their respective APIs.
- **Unified Interface**: Design a user-friendly web portal that abstracts the complexities of different cloud platforms, providing a consistent user experience.
- **Task Orchestration**: Implement functionality for automating deployment, management, and coordination of cloud resources.

III. Literature Survey

The state of the art for the paper entitled "Cloud-Native Orchestration Framework for Network Slice Federation Across Administrative Domains in 5G/6G Mobile Networks" by Maria et al [1] discusses the preceding work in network slicing and orchestration. Standards systems that exist in 5G/6G traditional networks for instance have employed usage of virtualization and cloud native approaches such as Open RAN (O-RAN) to optimize resource utilization as well as administrative domain mobility.[11] Though single operator domain scenarios are well researched, various issues arise in multi-domain scenarios such as user mobility, sharing of-resources and handover slice continuity. Such frameworks have been suggested before, such as GSMA's Operator Platform (OP), but often these frameworks are areas of research, and few have practical field studies. Related work has been introduced based on the integration with cloud-native characteristics that include Kubernetes for microservice orchestration. Nevertheless, prior research fails to provide hassle-free systematic methods consisting of standard interfaces and reality test environments towards inter-operator concerns.[12] The paper enlightens this area by introducing a concept of the cloud-native Slice Federation as a Service (SFaaS) architecture and proved on the experimental 5G platform that it can provide proper service and performance in the context of federated scenarios.

The paper "A multi-dimensional extensible cloud-native service stack for enterprises" by Jian Lin et al [2] suggests the new trends in cloud-native value proposition focus on building compact, highly-usable and scalable solutions that are being designed for various businesses. Some of the early IT practices have transformed into containers and microservices solutions which are focusing on infrastructure sharing, scalability, and automation. A lot of focus is placed on extensibility problems in applications, infrastructure, tenancy, and workflows, as can be seen in systems like Kubernetes and new ideas like the cloud-native light-cone model. Among the important findings of the research, diverse scheduling modes, runtime environment compatibility, and the ability to provide strong multi-tenant isolation while supporting dynamic workflows such as CI / CD are provided.[10] Various proposed architectures and implementations like OMStack include aspects of container orchestration, big data processing, and AI workflows, and are valuable and functional for enterprises, thus resource optimized and efficient.

The paper ”Cloud Infrastructure Self Service Delivery System using Infrastructure as Code” by Ankita Dalvi et al [3] focusing on the importance of IaC and self-service delivery systems for the efficient running of cloud systems. The shift to multi-cloud solutions based on digital transformations as well potentials gaps and risks of vendor lock-in requires intelligent and automated approaches to provisioning of services. COEs like the Cloud Centers of Excellence or CCoE[9] are critical to empower developers as much as to use the cloud infrastructure, and instead, use a set of IaC templates that are officially approved to avoid human error exponentially: E.g. Automation to COE tends to CCOE. IaC has been discussed in numerous studies as a means to automate such tedious tasks as provisioning and deprovisioning, decrease operational downtime and increase security by addressing misconfigurations[4].

The proposed method in the paper by ”Cloud-as-a-Gift: Effectively Exploiting Personal Cloud Free Accounts via REST APIs” embrace self-service where, through standard templates, the developers implement application requests and dynamically allocate resources that meet the business need of speed with compromises on the traditional compliance with organizational policies and security frameworks.[13] Such systems exemplify the shifts experienced with cloud management from more rigid infrastructures to more streamlined, effective and even, now, more friendly developers approaches. These forms of levers support the current evolution of the multicloud problems and future developments in cloud-native systems.[5]

The method in paper ”Online Self-Service Learning Platform for Application-Inspired Cloud Development and Operations (DevOps) Curriculum” by Roshan et al provides an increasing trend of utilizing cloud-native and self-service learning and operation in cloud computing and DevOps.[14] Current paradigms in education like Mizzou Cloud DevOps [MCD] develop working models and incorporate advanced tools including kubernetes, jenkins, and kubeflow etc into practical learning. Automation is highlighted as particularly valuable for creating and managing large, multiple cloud infrastructures, especially with the help of Infrastructure as Code. Issues which include Resource Management, Operational Delays, and lack of Interoperability are pitted against proactive solutions like Power-Usable Learning Modules, and Centralized Management System. It allows these systems to promote learning at one’s own pace while following different compliance and security measures. The enhancement of tangible, real-life oriented assignments also narrows the gap between virtual learning,work-related practices and skills[6].

The paper ”SecIngress: An API gateway framework to secure cloud applications based on N-variant system” by H Chen et al revisiting cloud application security have emphasised the need for redundancy, diversity and strong frameworks in tackling emerging and escalating threats in cyber-security. Despite these advantages, models based on SaaS delivery inherently have issues with static execution environments of applications that can be leveraged by adversaries.[15] The N-variant system solves this through concurrent running of multiple diversified version of the application and checks for inconsistency. Several recent studies have shown that it is very efficient in preventing attack types such as code injections and privilege escalations, but applying this approach in cloud applications is not easy. New solutions including the SecIngress an API gateway framework are shown to have improvements in addressing such issues using approaches like two-stage timeout processing as well as AHP voting for security improvements without performance loss. These frameworks sit nicely with containerized environments such as Kubernetes and present solutions that are flexible and elastic for cloud providers. Having reviewed different studies, the effectiveness of such systems in minimizing opportunities for threats and risks[7].

The paper ”Network Traffic Impact on Cloud Usage at Different Providers” by Gusev et al Cloud computing performance related literature is centered on the effects that network traffic may have on data transfer and data processing and performance of CSPs.[16] Literature covers Amazon Web Services (AWS) mainly for being a superior network provider because of its capability to transfer data with high speeds between its services like EC2 and S3 services. The examined comparison of AWS as a provider with the European provider Scaleway showed that the AWS achieved a faster and more economical data transfer, especially in cases of large datasets and frequent storage retrievals. Thus, while AWS provides extensive possibilities for the inter-service communication, the expenses have to be adjusted: it’s possible to make a certain number of necessary intermediate storing options, for instance, S3. Second, although there are cheaper providers like Scaleway, their average usage rates and raw performance are somewhat lower than AWS in some cases.[17] Studies also stress that it is becoming increasingly possible to create multiple clouds to achieve the lowest latency and cost and to cover more regions, which indicates a trend that the future cloud resources are managed through a combination of two cloud models[8].

IV. Proposed Innovation

Our innovation lies in the development of a modular, extensible framework that combines multi-cloud orchestration with green computing strategies. Key features include:

A. Modular Cloud Connectors

We introduce custom-built cloud connectors written in Python, leveraging the SDKs and APIs of various CSPs. Each connector is designed to interface seamlessly with the unified portal, translating generic commands into platform-specific API calls. This modularity allows for easy addition of new providers and simplifies maintenance.

B. Unified Self-Service Portal

The web-based portal offers a consistent user interface for managing resources across all integrated platforms. Users can perform tasks such as provisioning virtual machines, managing storage, and configuring networks without needing to understand the underlying APIs or tools of each CSP.

C. Green Computing Integration

By implementing intelligent resource scheduling, auto-scaling based on demand, and workload consolidation, the system optimizes resource utilization. This reduces unnecessary compute cycles and energy consumption, contributing to environmental sustainability.

V. Methodology

The project follows a structured methodology comprising design, development, integration, and testing phases.

A. System Architecture

The architecture consists of three main components:

1) **Unified Portal**: A web application built using Flask (Python) and Bootstrap, providing the user interface.
2) **Cloud Connectors**: Modular Python modules interfacing with each CSP's API.
3) **Orchestration Layer**: Manages communication between the portal and connectors, handling tasks and workflows.

B. Development Process

1) Implementation of EC2 Functionality: The EC2 functionality allows users to create, list, and terminate virtual machines on AWS. The algorithm for creating an EC2 instance is as follows:

Algorithm 1 Create EC2 Instance

```
Require: AWS Access Key, Secret Key, Region, Image ID,
    Instance Type, Key Pair Name
1: Initialize EC2 client using AWS credentials
2: Call run_instances() with provided parameters
3: Retrieve InstanceId from the response
4: Return InstanceId ClientError
5: Log error message
6: Return failure response
```

Listing instances and terminating instances follow similar patterns, utilizing AWS SDK functions like `describe_instances()` and `terminate_instances()`.

2) Implementation of S3 Functionality: The S3 functionality enables users to create, list, and delete storage buckets. The algorithm for creating an S3 bucket is as follows:

Algorithm 2 Create S3 Bucket

```
Require: AWS Access Key, Secret Key, Region, Bucket
    Name
1: Initialize S3 client using AWS credentials
2: if Region is 'us-east-1' then
3:     Call create_bucket() with BucketName
4: else
5:     Call create_bucket() with BucketName and
       CreateBucketConfiguration
6: end if
7: if Bucket creation is successful then
8:     Return success response
9: else
10:    Return failure response
11: end if
```

Deleting a bucket involves first deleting all its contents to prevent errors due to non-empty buckets.

3) Development of Flask API: The Flask API serves as the intermediary between the unified portal and the AWS services. The general process for creating API endpoints is as follows:

Algorithm 3 Flask API Endpoint Creation

```
Require: Flask Framework, AWS Connector Functions
1: Define route with appropriate HTTP method (GET, POST)
2: Define function to handle the request
3: Parse input data from the request
4: Call the corresponding AWS connector function with input
   data
5: if Operation is successful then
6:     Return JSON response with success message or data
7: else
8:     Return JSON response with error message
9: end if
```

VI. Cloud-Specific Integration Methodologies

Amazon Web Services (AWS)

APIs for AWS services are built using the AWS SDKs and REST APIs to manage **S3**, **EC2**, **Rekognition**, **CloudFront**, and **CloudWatch**. The S3 API handles storage operations like uploading, retrieving, and deleting objects. EC2 APIs manage instance lifecycle tasks such as starting, stopping, and scaling virtual machines. For Rekognition, APIs are designed to process and analyze image or video inputs, providing insights such as object detection or face recognition. CloudFront APIs manage CDN configurations and distributions, while CloudWatch APIs enable logging, monitoring, and metric analysis. Authentication is handled using **AWS Signature Version 4** for secure API access.

Microsoft Azure

Azure APIs integrate services like **Text-to-Speech**, **Virtual Machines**, and **Storage** using the Azure SDKs and REST interfaces. The Text-to-Speech API generates speech from text input, with customization options for voices and languages. APIs for Virtual Machines enable users to create, start, stop, and manage VMs programmatically. Storage APIs allow operations on Blob and Queue storage, supporting both structured and unstructured data management. **Azure Active Directory** ensures API security with **OAuth 2.0** and role-based access control, while **ARM templates** automate provisioning via the APIs.

Google Cloud Platform (GCP)

GCP APIs are implemented using the **Google Cloud SDK** and client libraries to provide seamless interactions with **Text-to-Text Translation**, **Virtual Machines**, and **Storage**. The Text-to-Text Translation API supports multilingual text processing and language detection. Compute Engine APIs handle VM instance management, including provisioning, resizing, and termination. Cloud Storage APIs enable secure data uploads, downloads, and management operations. All APIs are secured using **OAuth 2.0** tokens and fine-grained **IAM policies**, while Cloud Monitoring APIs allow real-time insights into usage and performance metrics.

VII. Implementation

Due to time constraints, we implemented a partial prototype focusing on AWS integration. The AWS connector and the unified portal demonstrate the core functionalities of the system.

A. AWS Connector

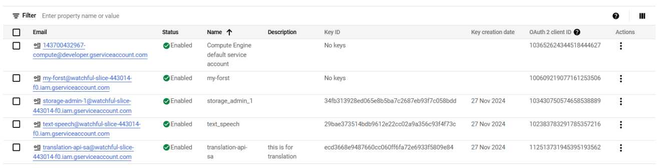
The AWS connector provides functions to:

- Create, list, and terminate EC2 instances.
- Create, list, and delete S3 buckets.

1) Handling AWS Credentials Securely: To ensure security, AWS credentials are managed through environment variables or AWS configuration files, avoiding hardcoding sensitive information in the code.

B. Unified Portal Interface

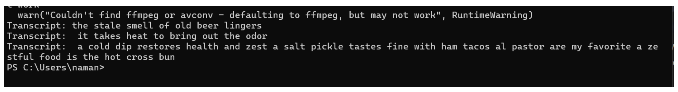
The portal's front-end is developed using HTML, CSS, and JavaScript, with Bootstrap for styling. It communicates with the Flask API to perform operations.



Filter Enter property name or value

	Email	Status	Name ↑	Description	Key ID	Key creation date	OAuth 2 client ID	Actions
☐	143700432967-compute@developer.gserviceaccount.com	✓ Enabled	Compute Engine default service account		No keys		103652624344518444627	⋮
☐	my-forst@watchful-slice-443014-f0.iam.gserviceaccount.com	✓ Enabled	my-forst		No keys		100609219077161253506	⋮
☐	storage-admin-1@watchful-slice-443014-f0.iam.gserviceaccount.com	✓ Enabled	storage_admin_1		34fb313928ed065e8b5ba7c2687eb93f7c058bdd	27 Nov 2024	103430750574658538889	⋮
☐	text-speech@watchful-slice-443014-f0.iam.gserviceaccount.com	✓ Enabled	text_speech		29bae373514bdb9612e22cc02a9a356c93f4f73c	27 Nov 2024	102383783291785357216	⋮
☐	translation-api-sa@watchful-slice-443014-f0.iam.gserviceaccount.com	✓ Enabled	translation-api-sa	this is for translation	ecd3668e9487660cc060ff6fa72e6933f5809e84	27 Nov 2024	112513731945395193562	⋮

Fig. 1. Service Account



```
  warn("Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may not work", RuntimeWarning)
Transcript: the stale smell of old beer lingers
Transcript:  it takes heat to bring out the odor
Transcript:  a cold dip restores health and zest a salt pickle tastes fine with ham tacos al pastor are my favorite a ze
stful food is the hot cross bun
PS C:\Users\naman>
```

Fig. 2. Speech to Text

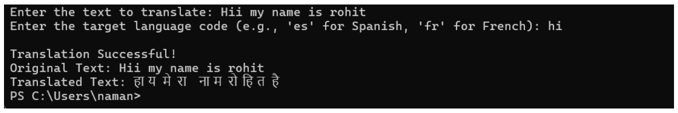
VIII. Testing and Results

We conducted tests to validate the functionality of the AWS connector and the unified portal.

A. Test Cases

1) **Instance Creation**: Successfully launched an EC2 instance from the portal.
2) **Instance Termination**: Terminated the EC2 instance via the portal.
3) **Bucket Creation**: Created an S3 bucket using the portal interface.
4) **Bucket Deletion**: Deleted the S3 bucket through the portal.
5) **Error Handling**: Attempted to create an instance with invalid parameters and verified appropriate error messages.

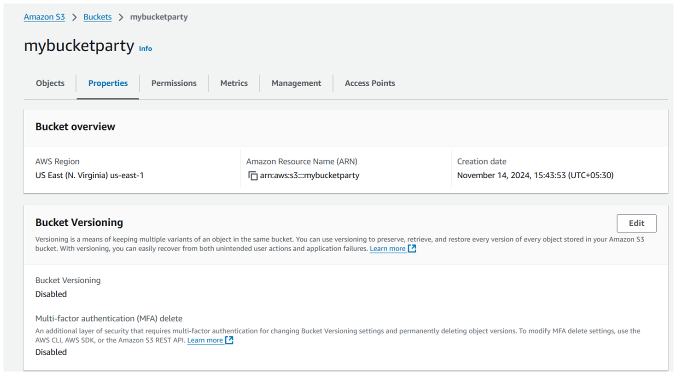
By implementing these ideas in a novel environment, it provides invaluable demonstrations of improved cloud resource automation, speech processing and text translation. Virtual machine instances were effectively created, listed and terminated using Google Cloud Platform (GCP) services to demonstrate simple infrastructure management. The operations related to the creation of buckets, uploading and deleting files was responsible for the efficient cloud storage operations. We implemented a scalable content delivery solution using a load balancer behind backend buckets as well as CDN configurations. In the context of speech processing, we pre processed audio files into mono format and converted files into mono format, improving compatibility, before transcribing them using high accuracy speech recognition capabilities. Firstly, the combination of multilingual translation APIs allowed for dynamic language translation of user defined inputs into the language they desired. The results here show that this system is capable of automating tasks, optimizing resource use, and integrating AI enabled functionality for real world applications, both in a scalable, and efficient manner across various domains.



```
Enter the text to translate: Hii my name is rohit
Enter the target language code (e.g., 'es' for Spanish, 'fr' for French): hi

Translation Successful!
Original Text: Hii my name is rohit
Translated Text: हा य मे रा  ना म रो हि त है
PS C:\Users\naman>
```

Fig. 3. Translation



Amazon S3 > Buckets > mybucketparty

mybucketparty Info

Objects | Properties | Permissions | Metrics | Management | Access Points

Bucket overview

AWS Region	Amazon Resource Name (ARN)	Creation date
US East (N. Virginia) us-east-1	arn:aws:s3:::mybucketparty	November 14, 2024, 15:43:53 (UTC+05:30)

Bucket Versioning Edit

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning
Disabled

Multi-factor authentication (MFA) delete
An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. Learn more
Disabled

Fig. 4. S3 Bucket

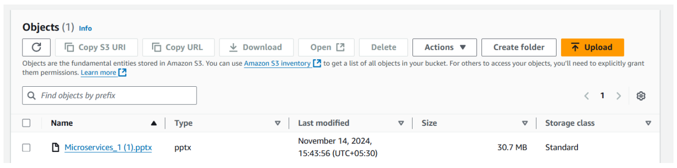
B. *Screenshots of Test Cases*

Fig 4, Fig 5 & Fig 6 show the effect of the code in creating AWS Instances In this study, we achieved the development of a Python based solution that seamlessly works with the Google Sheets API to find out the data from the Source page and then update it to the Target page automatically. The script authenticated and communicated with the spreadsheet by configuring of a service account with the appropriate permissions. It demonstrated being able to read existing information and writing new entries to the interactive sheet as it updates dynamically, which indicates the possibility of integrating Python applications with cloud based spreadsheets. These results validate the use of Google cloud services for easy, automated data handling in different computational and analytical workflows.

IX. Discussion on Green Computing

A. *Environmental Impact*

Data centers and cloud services contribute significantly to global energy consumption. By optimizing resource utilization through intelligent orchestration, our system reduces the energy required to perform computing tasks. This contributes to lower carbon emissions and aligns with sustainability initiatives.

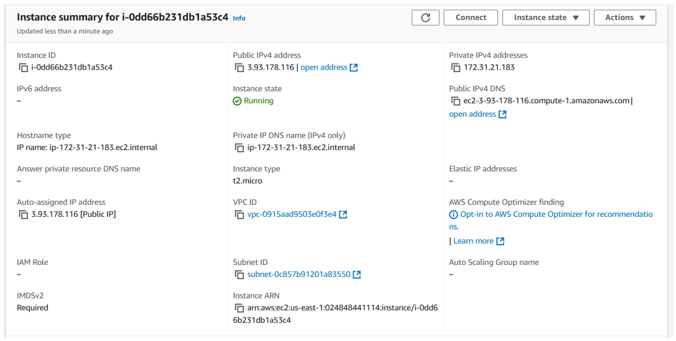


Objects (1) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
Microservices_1 (1).pptx	pptx	November 14, 2024, 15:43:56 (UTC+05:30)	30.7 MB	Standard

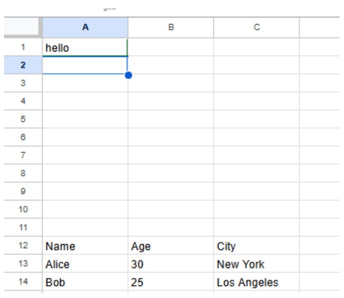
Fig. 5. S3 Instance



Instance summary for i-0dd66b231db1a53c4 Info
Updated less than a minute ago

Instance ID i-0dd66b231db1a53c4	Public IPv4 address 3.93.178.116 | open address	Private IPv4 addresses 172.31.21.183
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-3-93-178-116.compute-1.amazonaws.com | open address
Hostname type IP name: ip-172-31-21-183.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-21-183.ec2.internal	
Answer private resource DNS name –	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address 3.93.178.116 [Public IP]	VPC ID vpc-0915aad9503e0f3e4	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role –	Subnet ID subnet-0c857b91201a83550	Auto Scaling Group name –
IMDSv2 Required	Instance ARN arn:aws:ec2:us-east-1:024848441114:instance/i-0dd66b231db1a53c4	

Fig. 6. EC2 Instance creation



	A	B	C
1	hello		
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12	Name	Age	City
13	Alice	30	New York
14	Bob	25	Los Angeles

Fig. 7. Adding Data to SpreadSheet

B. *Scalability and Extensibility*

The modular design of our cloud connectors allows for easy integration of additional cloud providers. This flexibility is crucial for organizations looking to adapt to changing technological landscapes without overhauling their management systems.

C. *Economic Benefits*

Optimizing resource usage not only benefits the environment but also reduces operational costs. Efficient scaling and resource management minimize wasteful spending on idle or underutilized resources.

D. *Future Work*

Expanding the system to include more advanced features like predictive scaling using machine learning algorithms could further enhance efficiency. Integrating real-time monitoring and analytics can provide deeper insights into resource utilization patterns.

X. Conclusion

The unified self-service cloud portal with custom cloud connectors demonstrates a viable solution for managing multi-cloud environments efficiently. By abstracting the complexities of individual CSPs, the system enhances user experience and operational efficiency. The integration of green computing practices further adds value by optimizing resource usage and contributing to environmental sustainability.

While the current implementation focuses on AWS, the modular design allows for easy extension to other cloud platforms. Future work will involve integrating additional providers, enhancing security features, and implementing more advanced green computing strategies.

References

[1] Dalgitsis, M., Cadenelli, N., Serrano, M. A., Bartzoudis, N., Alonso, L., Antonopoulos, A. (2024). Cloud-native orchestration framework for network slice federation across administrative domains in 5G/6G mobile networks. IEEE Transactions on Vehicular Technology.

[2] A. Verma, D. Malla, A. K. Choudhary and V. Arora, "A Detailed Study of Azure Platform Its Cognitive Services," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 2019, pp. 129-134, doi: 10.1109/COMITCon.2019.8862178.

[3] Lin, J., Xie, D., Huang, J., Liao, Z., Ye, L. (2022). A multi-dimensional extensible cloud-native service stack for enterprises. Journal of Cloud Computing, 11(1), 83.

[4] A. Dalvi, "Cloud Infrastructure Self Service Delivery System using Infrastructure as Code," 2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2022, pp. 1-6, doi: 10.1109/ICCCIS56430.2022.10037603.

[5] T. Aubonnet and N. Simoni, "Self-Control Cloud Services," 2014 IEEE 13th International Symposium on Network Computing and Applications, Cambridge, MA, USA, 2014, pp. 282-286, doi: 10.1109/NCA.2014.48.

[6] R. Gracia-Tinedo, M. S. Artigas and P. G. López, "Cloud-as-a-Gift: Effectively Exploiting Personal Cloud Free Accounts via REST APIs," 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, 2013, pp. 621-628, doi: 10.1109/CLOUD.2013.47.

[7] Neupane, Roshan Lal, et al. "Online Self-Service Learning Platform for Application-Inspired Cloud Development and Operations (DevOps) Curriculum." IEEE Transactions on Learning Technologies (2024).

[8] Zhou, Dacheng, et al. "SecIngress: an API gateway framework to secure cloud applications based on N-variant system." China Communications 18.8 (2021): 17-34.

[9] V. Bidikov, M. Gusev and V. Markozanov, "Network Traffic Impact on Cloud Usage at Different Providers," 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 2022, pp. 847-852, doi: 10.23919/MIPRO55190.2022.9803730.

[10] R. C. Pushpaleela, S. Sankar, K. Viswanathan and S. A. Kumar, "Application Modernization Strategies for AWS Cloud," 2022 1st International Conference on Computational Science and Technology (ICCST), CHENNAI, India, 2022, pp. 108-110, doi: 10.1109/ICCST55948.2022.10040356.

[11] R. Bundela, N. Dhanda and R. Verma, "Load Balanced Web Server on AWS Cloud," 2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), Greater Noida, India, 2022, pp. 114-118, doi: 10.1109/ICCCIS56430.2022.10037657.

[12] M. Villamizar et al., "Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures," 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Cartagena, Colombia, 2016, pp. 179-182, doi: 10.1109/CCGrid.2016.37.

[13] C. Baun, M. Kunze and V. Mauch, "The KOALA Cloud Manager: Cloud Service Management the Easy Way," 2011 IEEE 4th International Conference on Cloud Computing, Washington, DC, USA, 2011, pp. 744-745, doi: 10.1109/CLOUD.2011.64.

[14] R. Khande, S. Rajapurkar, P. Barde, H. Balsara and A. Datkhile, "Data Security in AWS S3 Cloud Storage," 2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT), Delhi, India, 2023, pp. 1-6, doi: 10.1109/ICCCNT56998.2023.10306922.

[15] R. Gohil and H. Patel, "Comparative Analysis of Cloud Platform: Amazon Web Service, Microsoft Azure, And Google Cloud Provider: A Review," 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kamand, India, 2024, pp. 1-5, doi: 10.1109/ICCCNT61001.2024.10725914.

[16] A. S. Muhammed and D. Ucuz, "Comparison of the IoT Platform Vendors, Microsoft Azure, Amazon Web Services, and Google Cloud, from Users' Perspectives," 2020 8th International Symposium on Digital Forensics and Security (ISDFS), Beirut, Lebanon, 2020, pp. 1-4, doi: 10.1109/ISDFS49300.2020.9116254.

[17] DA. Verma, D. Malla, A. K. Choudhary and V. Arora, "A Detailed Study of Azure Platform Its Cognitive Services," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 2019, pp. 129-134, doi: 10.1109/COMITCon.2019.8862178.

[18] L. N. Hyseni and A. Ibrahimi, "Comparison of the cloud computing platforms provided by Amazon and Google," 2017 Computing Conference, London, UK, 2017, pp. 236-243, doi: 10.1109/SAI.2017.8252109.

[19] A. Verma, D. Malla, A. K. Choudhary and V. Arora, "A Detailed Study of Azure Platform Its Cognitive Services," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 2019, pp. 129-134, doi: 10.1109/COMITCon.2019.8862178.

[20] P. Kaushik, A. M. Rao, D. P. Singh, S. Vashisht and S. Gupta, "Cloud Computing and Comparison based on Service and Performance between Amazon AWS, Microsoft Azure, and Google Cloud," 2021 International Conference on Technological Advancements and Innovations (ICTAI), Tashkent, Uzbekistan, 2021, pp. 268-273, doi: 10.1109/ICTAI53825.2021.9673425.