Integrating Jenkins for Efficient Deployment and Orchestration across Multi-Cloud Environments

S. Senthil Pandi, Dr. P.Kumar and R.M.Suchindhar

Department of Computer Science and Engineering Rajalakshmi Engineering College Chennai, India

E-mail : mailtosenthil.ks@gmail.com kumaar@rajalakshmi.edu.in 230711011@rajalakshmi.edu.in

Abstract- This paper shows a comparative study of terraform and kubernetes as orchestration tools for cost optimization capabilities of these tools and provide insights into effective reducing of cloud expenditure. The primary objective of the research is to evaluate and compare the cost reduction capabilities of these tools in the cloud-based environment. This methodology involves planning and carrying out evaluates in a multiple clouds environment while simulating real-time workload and environment. Kubernetes and integrating with jenkinsto highlights its resource allocation efficiency and auto-scaling features as ways to reduce cost. The study also explores how advanced machine learning algorithms and predictive analytics might be incorporated into the orchestration process. It offers a path for businesses to take on a transformational journey toward cutting-edge cost-cutting tactics, reinventing the way multicloud settings are managed, and optimizing the return on cloud expenditures.

Keywords— Cloud Computing, Multiple Clouds, Terraform, Kubernetes, Jenkins.

I. Introduction

In the current cloud computing environment, industries are rapidly expanding their use of multi cloud models, adopting them, and benefiting from it.Multiple cloud environments present significant management and optimization problems.To accomplish cost and optimize return on investment, resources are essential.The purpose of this research is to examine Terraform and Kubernetes as cost-effective multicloud deployment orchestration technologies. Modern cloud infrastructure management and container orchestration now wouldn't exist without the two potent tools Terraform and Kubernetes, respectively. With the help of Terraform, users may design and deploy cloud resources across different providers in a consistent and effective way. Terraform is a flexible infrastructure as code (IAC) solution. Users are given the ability to declare their intended infrastructure state and then automatically realize that vision thanks to its declarative approach to infrastructure management, which relies on arguments and blocks in its language syntax.A container orchestration platform called Kubernetes enables the deployment, scaling, and administration of containerized applications automatically. It has sophisticated features including load balancing, auto-scaling, and self-healing capacities. Users can specify their intended state and have containerized apps managed automatically with Kubernetes' declarative approach to application deployment and management.Despite having diverse use cases, Terraform and Kubernetes can be combined to manage infrastructure and applications in a cloud environment.The purpose of the study is to provide some light on the benefits, drawbacks, and inventive differences between these tools. The study aims to identify novel tactics and methodologies that can produce cost reductions and improve efficiency in multi cloud installations through experimentation and the analysis of key cost optimization variables. The usage of Kubernetes for container orchestration and terraform for infrastructure provisioning will be investigated in the study. The study will also look at how these technologies may work together to manage cloud environments' infrastructure and applications. The study will offer recommendations for best practices in multi cloud installations and provide a thorough analysis of the cost optimization capabilities of Terraform and Kubernetes.

Organizations can learn important lessons and develop effective methods for achieving cost optimization in their multiple clouds deployments by performing tests, reviewing important KPIs, and identifying innovative points of differentiation. Through this research, businesses may enhance multi cloud orchestration techniques and equip themselves to get the most out of their cloud expenditures. Organizations can successfully apply cost optimization techniques by utilizing tools like Terraform, Kubernetes, and AWS DevOps.

Organizations may achieve cost optimization in their multi cloud deployments and realize the full potential of their cloud investments with the appropriate strategy and technologies.

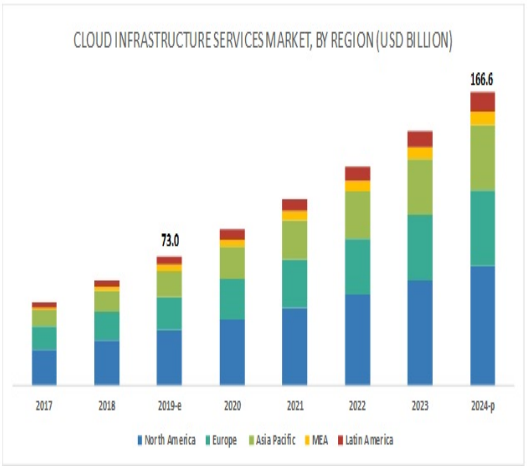


Figure 1.Market for Cloud architectural Services

The market share breakdown for architectural services by region is shown in Figure 1 Europe has the highest percentage, followed by Asia, America, Africa, and Oceania, based on the information.

II.Literature Survey

K. Hughes et al[1] emphasizes the dynamic deployment of intricate virtual scientific infrastructures on private and public clouds utilizing a software platform in the existing system.Lack of programming interface standards in the J. M. R. Poovizhi and R. Devi.[2] defining a cross cloud system drives up prices and restricts the flexibility and mobility of applications and virtual infrastructures.A Hybrid Cloud Orchestrator to Deploy Complex Applications S. Srithar et.al [3] With insights into its capabilities and advantages for application deployment and orchestration, it demonstrates how Roboconf addresses the difficulties of installing and maintaining applications in hybrid cloud settings.Insights on the architecture of cloud computing, market-oriented resource allocation, customer-driven service management, computational risk management, and the connectivity of clouds to form global cloud exchanges and markets are provided by H. Iqbalet. al[5] and F. Olariu et. al[6]. Along with presenting a case study on using storage clouds for high-performance content distribution, they also examine the distinction between workloads for High Performance Computing (HPC) and workloads for internet-based services.

In the proposed system it highlights setting up of complex virtual infrastructure on both public and private clouds using tools like Terraform for dynamic infrastructure provisioning and management.In this system, the constraints are addressed, the flexibility and portability of application-and-infrastructure are improved, and insights into design choices are provided.This concept of dynamically deploying and managing infrastructure resources is covered in the paper by A.Jothi[7]. In the suggested solution, Kubernetes' scalability and flexibility for container orchestration across several cloud platforms is merged with Jenkins' advantages as a flexible tool for continuous integration and delivery.It emphasizes the idea of cloud computing [8,9,10] as a way to realize the goal of computing as the fifth utility. It offers a thorough understanding of the cloud computing architecture, resource allocation based on the market, customer-driven service management, computational risk management, and cloud connectivity [11,12]. The paper [13,14] by Ramya and senthilselvi explained about load balancing in cloud computing environment.

III.Methodology

A. Multiple cloud orchestration - Jenkins

Software development processes are built, deployed, and automated using Jenkins, an open-source automation server. Scalability, extensibility, and flexibility are all features ofits architecture. A crucial part of the continuous integration and continuous delivery (CI/CD) process is played by Jenkins, a widely used open-source automation server. Its main function is to orchestrate multiple operations like code creation, testing, deployment, and monitoring in order to automate and streamline the software development lifecycle.

Figure 2 demonstrates how Jenkins pulls code from github, automatically runs a Jenkins script created, and uses terraform to perform jobs that build infrastructure in the appropriate cloud. Jenkins integrates the cloud by acting as the focal point for organizing and managing cloud-related activities. Through plugins and integrations, it interfaces to cloud computing infrastructures including AWS, Azure, Google Cloud, and OpenStack. Jenkins makes use of these integrations to seamlessly deploy, scale, and manage cloud services.

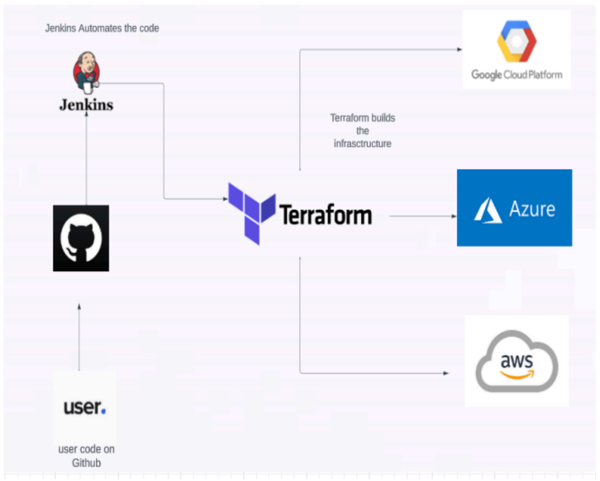


Figure 2.Work flow of Jenkins

Jenkins, for instance, can leverage code changes or user-defined triggers to cause the deployment of virtual machines, container clusters, or serverless operations. With the help of this seamless connectivity, development teams may take use of the cloud's scalability, flexibility, and automation capabilities to speed up the supply of software while preserving consistency and dependability in the deployment procedure.

B. Terraform

Within a Jenkins CI/CD pipeline, Terraform is essential for automating the provisioning of cloud infrastructure. Determining, developing, updating, and maintaining cloud resources across various cloud providers are all part of its job description. With Terraform, users can use a declarative configuration language to design your cloud architecture as code. The desired state of your infrastructure, including resources like virtual machines, networks, storage, and more, is specified in Terraform configuration files, which often have a.tf extension. The provisioning process is outlined in this infrastructure code.Terraform supports several cloud providers, including AWS, Azure, Google Cloud, and OpenStack. It is cloud provider agnostic. You provide a set of Terraform configurations that are consistent, and terraform takes care of converting those configurations into API calls unique to the selected cloud provider.Terraform examines your configuration files when you run terraform apply and compares the desired state with the present state of your cloud infrastructure. In order to get the infrastructure into the state that is intended, it then builds, changes, or deletes resources as necessary. To carry out these tasks, Terraform talks with the cloud provider's API.Terraform commands (such as terraform init, terraform plan, terraform apply, and terraform destruct) can be run as build steps or shell commands within a Jenkins pipeline. Jenkins can give Terraform the authorization and credentials it needs to communicate with cloud providers.

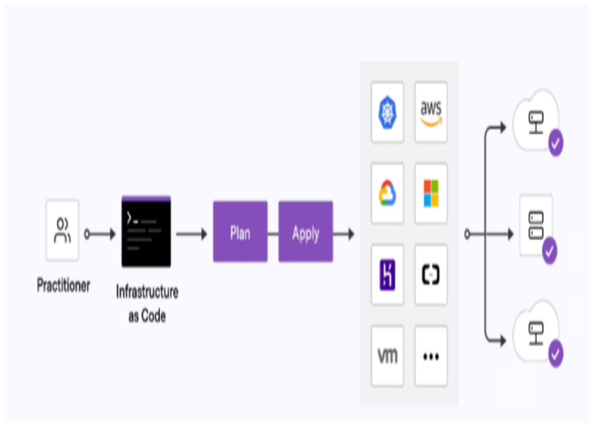


Figure 3 . Terraform Workflow

Figure 3 depicts how the Terraform workflow works. After writing the necessary code and provisioning the necessary infrastructure, Terraform creates the infrastructure while using particular commands, such as plan apply.

C. System architecture

The suggested system architecture uses a number of cloud providers, including AWS, Azure, and GCP, to build a Kubernetes cluster that is incredibly durable and adaptable. This multi-cloud strategy decreases the chance of outage while ensuring redundancy. Each cloud provider hosts a different component of the Kubernetes cluster; for instance, AWS manages the control plane, Azure manages worker nodes, and GCP acts as a backup or disaster recovery site. By exploiting the advantages of each cloud provider, this distributed configuration improves resource consumption while also improving fault tolerance.Additionally, terraform is used as the go-to infrastructure as code (IaC) technology, facilitating the provisioning and management of cloud resources across many platforms without any hitches. This architecture shows the adaptability and compatibility of Kubernetes and Terraform with multiple cloud environments in addition to achieving high availability and scalability.

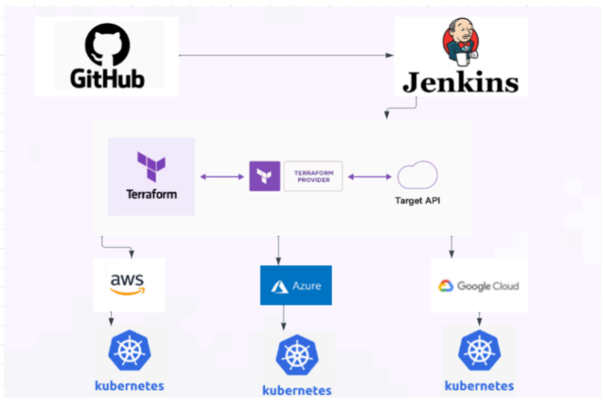


Figure 4. Automated Workflow

The architecture in Figure 4 illustrates an automated workflow where code modifications start a build process and terraform is used to provision the necessary infrastructure resources. On Kubernetes clusters in GCP, Azure, and AWS, developers can now deploy and manage their apps via this.The following steps are demonstrated by the architecture:
Git: Git is a version control tool that is employed to manage source code. Code modifications can be pushed to a Git repository by developers.

Jenkins: Jenkins is an automation server that keeps an eye out for changes in the Git repository. Jenkins starts a build process when changes are found.

Terraform: To provision and manage infrastructure resources during the building process, utilize Terraform. Declarative configuration files are used to specify the infrastructure's ideal condition. Terraform is in charge of installing infrastructure resources, including Kubernetes clusters, on cloud infrastructures like GCP, Azure, and AWS. These platforms offer the infrastructure services required for managing resources and running applications.

D. AWS Control Plane

The Kubernetes control plane node, which is in charge of cluster administration and API server operations, is hosted by AWS. The dependable infrastructure of AWS is advantageous to this node.

Subnets and VPC: The architecture of the AWS Virtual Private Cloud (VPC) guarantees network segregation, and many subnets offer redundancy.

Security Groups: Security groups manage traffic to and from the Kubernetes cluster nodes on an inbound and outgoing basis, hence boosting security.

For Multi-Cloud IaC, using Terraform
Terraform is used in infrastructure as code (IaC) to reliably define and deploy infrastructure resources across AWS, Azure, and GCP.

Flexibility: The cluster may be dynamically scaled, altered, and managed as necessary thanks to Terraform's multi-cloud interoperability.

Kubernetes and Container Orchestration:
Several different cloud providers' Kubernetes clusters work together to orchestrate and scale containerized applications in an efficient manner.Kubernetes' multi-cloud features allow for the flexible deployment of applications wherever resources are most readily accessible or most affordable.High availability, fault tolerance, and flexibility are all features of this multi-cloud Kubernetes architecture across AWS, Azure, and GCP. It makes use of each cloud provider's distinct features to build a stable and dispersed Kubernetes cluster.

IV.Results & Discussions

The software delivery pipeline is streamlined by Jenkins' easy connection with Kubernetes clusters, which empowers automated CI/CD procedures. The dynamic infrastructure required for deploying, scaling, and managing application containers is provided by Kubernetes. Jenkins makes use of this setup to coordinate and automate different phases of the software development lifecycle.

Table 1 Cloud Provider details

	AWS	Azure	GCP
Instance type	t3.medium	A standard B2s	n1-standard-2
Region	us-east-1	East US	us-central 1

OS	Ubuntu 20.04	Ubuntu 20.04	Ubuntu 20.04
Kubernetes version	Kubernetes 1.21	Kubernetes 1.21	Kubernetes 1.21
Terraform Version	Terraform 0.14	Terraform 0.14	Terraform 0.14

Table 1 represents the cloud provider, instance type, region, operating system, Kubernetes version, and Terraform version for each environment are described in these configurations. This information is essential for Terraform to efficiently provision and manage Kubernetes clusters across AWS, Azure, and GCP.These procedures lead to deployments that are affordable but do not sacrifice performance. Additionally, its multi-cloud deployment features highlight its flexibility and adaptability, making it a useful tool for businesses looking for a consistent method of deploying Kubernetes clusters across a variety of cloud providers. The script is positioned as a potent solution for managing modern cloud architecture thanks to its support for many clouds and resource efficiency.Overall, by automating tasks, improving scalability, and enabling uniform deployments across multi-cloud Kubernetes clusters, this integration speeds up the delivery of software.The performance evaluation of terraform provisioned kubernetes cluster via jenkins in multicloud through pipeline showcase the finding of certain values such as response time,Resourceutilization,throughput values of the different clouds.

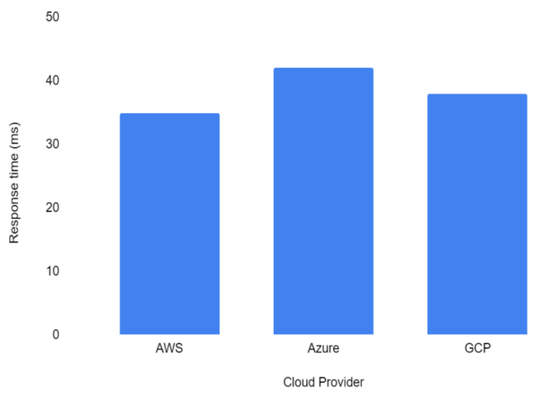


Figure 5. Response time using different cloud provider

The Figure.5 shows the multi-cloud response time. The amount of time it takes for a system to react to a user's request or inquiry is referred to as the response time. It is an essential performance parameter that gauges how quickly and effectively a system reacts.calculating the amount of time it takes for a request to be submitted and for a response to be returned. The start and finish timestamps of the request processing can be captured to do this.Response time is helpful for performance improvement to optimize the setup, which may improve user experience in multi cloud. Aws had the fastest response time, measured at 35 milliseconds, followed by Gcp at 38 milliseconds and Azure at 42 milliseconds.This demonstrates AWS's dominance against several clouds.

The Figure.6 show the many cloud's requested throughput.The amount of data that can be processed in a specific length of time is referred to as throughput. It evaluates the system's ability to manage a specific workload. A system's capacity to handle more requests or process more data in a given amount of time is indicated by its throughput.The throughput, which may be interpreted as the volume of transactions per minute or requests handled per second. The throughput is calculated using Throughput = Number of requests processed / Time period.

Throughput of 1200 represents requests per second in aws and let assume that time period is 60 seconds (1 minute)
Throughput = 1300/60 = 21

AWS outperformed both Azure and GCP, which reported throughputs of 1102 and 920 requests per second, respectively, by achieving the maximum throughput of 1300 requests per second. These results imply that, in comparison to other cloud providers, GCP has superior scalability and can manage a higher amount of requests.

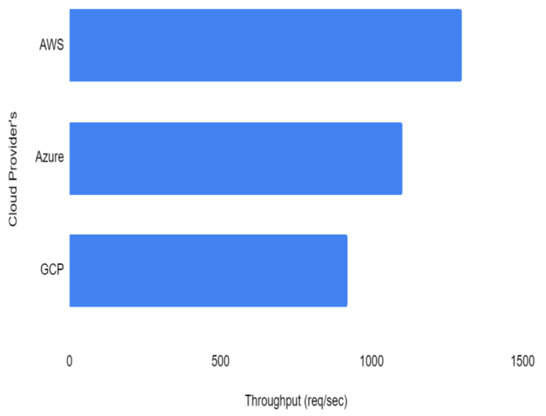


Figure 6. Throughput of the multi cloud platform

The practice of effectively allocating and managing resources in order to optimize their usage and decrease waste is known as resource optimization. It entails examining resource utilization patterns, spotting inefficiencies, and putting plans into action to enhance resource allocation and output.Organizations may discover resource allocation by tracking resource usage, which might result in effective resource optimization and cost savings.

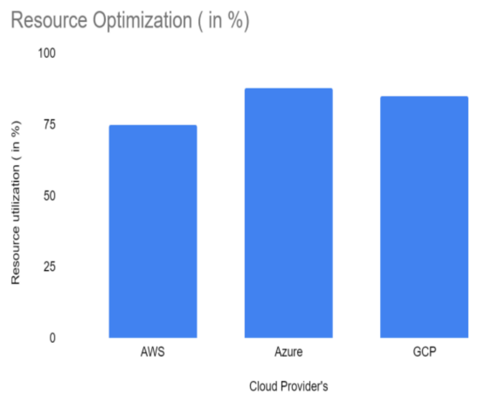


Figure.7 Resource Optimization

We can enhance the overall performance of the multi cloud by ensuring provisioned and balanced resource optimization. Furthermore, the examination of resource use showed that AWS had the greatest optimization rate, at 75%, followed by 78% utilized Azure, and 75% got to GCP. This Figure 7 shows that AWS makes efficient use of resources, making it an affordable choice for Kubernetes installations. Organizations contemplating multi-cloud deployments may get important insights from these performance characteristics, which will help them decide on their resource allocation and cloud provider selection methods.

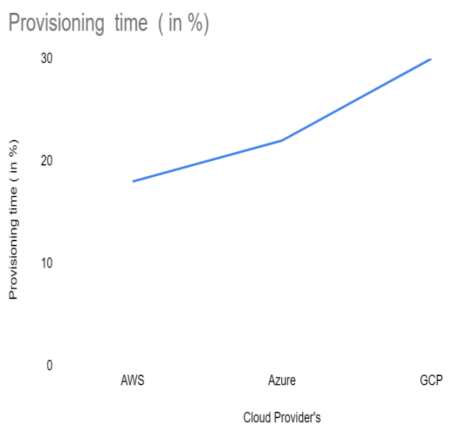


Figure.7 Provisioning Time

The average provision time Figure 7 for deploying Kubernetes clusters in multiple clouds with Terraform. The instance of AWS, creating a Kubernetes cluster with three nodes took an average of 20 minutes to complete. For a Kubernetes cluster with 3 nodes, Azure, on the other hand, showed a quicker provisioning time with an average execution time of 22 minutes. With an average execution time of 30 minutes for a Kubernetes cluster with 3 nodes, of GCP and so AWS displayed performance that was comparable to Azure and GCP. These results show how different cloud providers' provisioning effectiveness varies. Employing this knowledge, businesses contemplating multi-cloud deployments may choose their cloud providers wisely and plan their deployment methods while considering the trade-offs between provisioning speed and cost.

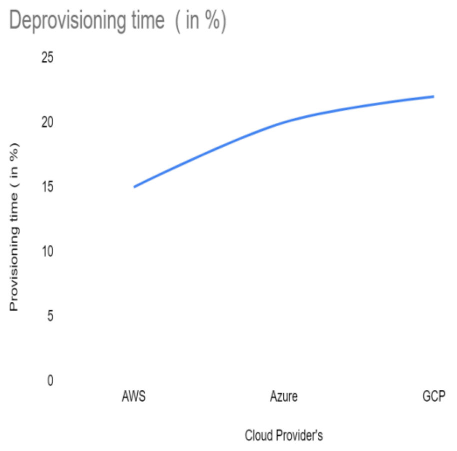


Figure.7 Deprovisioning Time

The Figure 7 shows the deprovisioning period for Azure, AWS, and GCP Kubernetes clusters. A Kubernetes cluster with three nodes for AWS took 10 minutes to deprovision. For a comparable cluster setup using GCP, the deprovision time was 20 minutes. For a Kubernetes cluster with 3 nodes, Azure demonstrated a deprovision time of 25 minutes. These results demonstrate how various cloud providers' deprovision times might vary, which can be important for businesses when thinking about how to manage and allocate resources for their Kubernetes clusters effectively.

V.Conclusion

This study examined how AWS DevOps may be used for task deployment and load balancing in collaborative "cloud-edge" datacenters. Organizations may automate and streamline the software development and deployment process by utilizing the extensive suite of services and methodologies provided by AWS DevOps, ensuring effective and dependable application deployment in cloud.Examine cutting-edge load balancing techniques that dynamically allocate traffic in accordance with cloud provider performance indicators to ensure optimal resource use.With a focus on encryption, identity and access management (IAM), and regulatory compliance, expand the research to explore security and compliance issues in a multi-cloud Kubernetes context.To reduce downtime during failover, expand the architecture to include thorough disaster recovery plans and investigate effective data moving techniques between cloud providers.

References

[1] K. Hughes, P. di Pasquale, A. Babuscia and L. Fesq, "On-demand Command and Control of ASTERIA with Cloud-based Ground Station Services," 2021 IEEE Aerospace Conference (50100), Big Sky, MT, USA, 2021, pp. 1-15, doi: 10.1109/AERO50100.2021.9438199.

[2] J. M. R. Poovizhi and R. Devi, "The Performance Analysis and Comparison of Azure and AWS Hypervisor using Different Workloads in Virtualization," 2023 7th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2023, pp. 1636-1641, doi: 10.1109/ICICCS56967.2023.10142325.

[3] S. Srithar, G. Ramesh Kalyan, S. Karthic, M. Naveenkumar, P. Arulprakash and E. Vetrimani, "Cost-Effective Distributed Booster Load Balancer in Amazon Cloud Environment," 2022 7th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2022, pp. 613-619, doi: 10.1109/ICCES54183.2022.9835842.

[4] C. Li, C.Wang, and Y. Luo, ``An efficient scheduling optimization strategy for improving consistency maintenance in edge cloud environment,'' J.Supercomput., vol. 76, pp. 69416968, Sep. 2020.

[5] H. Iqbal, A. Singh and M. Shahzad, "Characterizing the Availability and Latency in AWS Network From the Perspective of Tenants," in IEEE/ACM Transactions on Networking, vol. 30, no. 4, pp. 1554-1568, Aug. 2022, doi: 10.1109/TNET.2022.3148701.

[6] F. Olariu, "Overcoming Challenges in Migrating Modular Monolith from On-Premises to AWS Cloud," 2023 22nd RoEduNet Conference: Networking in Education and Research (RoEduNet), Craiova, Romania, 2023, pp. 1-6, doi: 10.1109/RoEduNet60162.2023.10274946.

[7] A. Jyoti and M. Shrimali, ``Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing,'' Cluster Comput., vol. 23, no. 1, pp. 377395, Mar. 2020.

[8] Kumar, P., Vinodh Kumar, S., Priya, L. (2023). An Approach for Energy-Efficient Resource Allocation Through Early Planning of Virtual Machines to Servers in Cloud Server Farms. In: Doriya, R., Soni, B., Shukla, A., Gao, XZ. (eds) Machine Learning, Image Processing, Network Security and Data Sciences. Lecture Notes in Electrical Engineering, vol 946. Springer, Singapore. https://doi.org/10.1007/978-981-19-5868-7_54

[9] Kumar, P., Vinodh Kumar, S., Priya, L. (2022). QoS-Based Classical Trust Management System for the Evaluation of the Trustworthiness of a Cloud Resource. In: Suma, V., Baig, Z., Kolandapalayam Shanmugam, S., Lorenz, P. (eds) Inventive Systems and Control. Lecture Notes in Networks and Systems, vol 436. Springer, Singapore. https://doi.org/10.1007/978-981-19-1012-8_22

[10] P. Kumar, Sheila Anand,(2013). An approach to optimize workflow scheduling for cloud computing environment.Journal of Theoretical and Applied Information Technology,57(3), pp. 617-623.

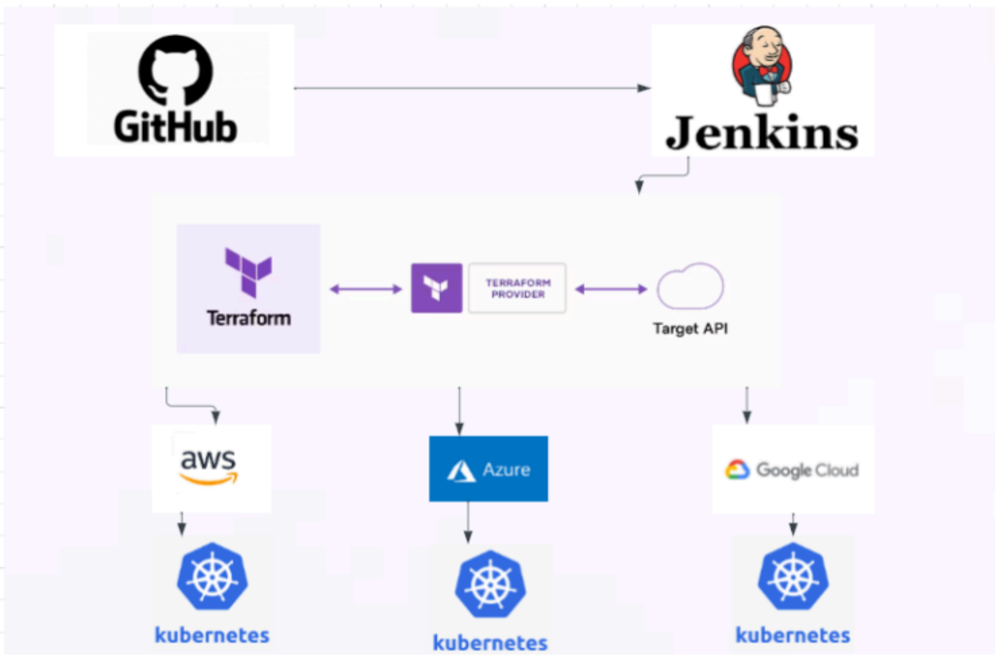
[11] S. Mishra, M. Kumar, N. Singh and S. Dwivedi, "A Survey on AWS Cloud Computing Security Challenges & Solutions," 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 2022, pp. 614-617, doi: 10.1109/ICICCS53718.2022.9788254.

[12] S. Sithiyopasakul, T. Archevapanich, B. Purahong, P. Sithiyopasakul, A. Lasakul and C. Benjangkaprasert, "Performance Evaluation of Infrastructure as a Service across Cloud Service Providers," 2023 International Electrical Engineering Congress (iEECON), Krabi, Thailand, 2023, pp. 58-63, doi: 10.1109/iEECON56657.2023.10127100.

[13] K. Ramya, Senthilselvi Ayothi, "Hybrid dingo and whale optimization algorithm-based optimal load balancing for cloud computing environment"2023, Emergngtelecommuncation

technologies, volume34,Issue 5,https://doi.org/10.1002/ett.4760

[14] K. Ramya, Senthilselvi, "Performance Improvement in Cloud Computing Environment by Load Balancing-A Comprehensive Review",REVISTA GEINTEC-GESTAO INOVACAO E TECNOLOGIAS,Vol. 11 No. 2 (2021)


GitHub
Jenkins
Terraform
TERRAFORM PROVIDER
Target API
aws
Azure
Google Cloud
kubernetes
kubernetes
kubernetes