# Service Deployment and Operations in OpenStack Cloud with Automation

Pranitha Gorenka
Telecommunication systems
Blekinge Institute of Technology
Karlskrona, Sweden
prgo22@student.bth.se

Ola Mkki
Telecommunication systems
Blekinge Institute of Technology
Karlskrona, Sweden
olmk22@student.bth.se

Dheeraj Audam
Telecommunication systems
Blekinge Institute of Technology
Karlskrona, Sweden
dhau22@student.bth.se

***Abstract*—This project focuses on developing a software solution for deploying and operating a service within an OpenStack Cloud environment. The solution includes three operating modes: deployment, operations, and cleanup. It involves creating network infrastructure, routers, nodes, and other necessary components during deployment, monitoring and managing the number of nodes during operations, and releasing all allocated cloud resources during cleanup. Additionally, two services, including flask app and SNMP daemon, are deployed on the nodes. The project emphasizes the integration of network design, service deployment, monitoring techniques, OpenStack infrastructure, and automation.**

***Keywords— service deployment, OpenStack Cloud, software solution, network infrastructure, routers, nodes, operations, cleanup, monitoring, automation, Flask app, SNMP daemon.***

## I. INTRODUCTION

The rapid advancement of technology has led to the increased adoption of cloud computing solutions for efficient and scalable service delivery [1]. OpenStack, as an open-source cloud computing platform, provides a flexible infrastructure for deploying and managing cloud services [2]. This project focuses on leveraging OpenStack and various forms of automation to develop a software solution for deploying and operating a service within an OpenStack Cloud environment [3]. The primary objective of this project is to combine the knowledge acquired from previous assignments, including network design, service deployment and configuration, and service monitoring using OpenStack and automation. By integrating these elements, we aim to create a robust solution that streamlines the deployment and operation of services within an OpenStack Cloud.

Our software solution encompasses the development of a system that utilizes an openrc file, containing the necessary credentials, to access a specific OpenStack Cloud. During deployment, the solution employs a unique TAG mechanism to facilitate easy identification and tracking of all items created within the OpenStack Cloud. The software solution operates in three distinct modes: deployment, operations, and cleanup. During the deployment phase, the solution establishes the required network infrastructure, routers, nodes, and other essential components within the OpenStack Cloud. These components are configured to ensure seamless service delivery. Subsequently, during the operations phase, the solution actively monitors the presence of the specified number of nodes and automatically deploys new nodes if any are missing. This proactive approach ensures continuous service availability and scalability. Conversely, if there are surplus nodes, the solution intelligently removes them to optimize resource utilization. The number of nodes required can be dynamically adjusted during the operations phase to accommodate changing demands. Finally, the cleanup phase ensures the smooth termination of the project by releasing and removing all allocated cloud resources, thus optimizing resource allocation and minimizing unnecessary costs. In addition to the deployment and operational aspects, the project places emphasis on the deployment of specific services on the nodes. The flask app, a component of the solution, is deployed on the nodes to enable the delivery of a specific service. Furthermore, the SNMP (Simple Network Management Protocol) daemon is deployed to facilitate efficient monitoring and management of the network infrastructure.

This report accompanying the software solution provides a comprehensive analysis of the project, documenting the design decisions, implementation strategies, and overall performance. It aims to offer valuable insights into the integration of network design, service deployment, monitoring techniques, OpenStack infrastructure, and automation. Furthermore, the report showcases the benefits and capabilities of the developed solution, serving as a valuable reference for future deployments and system optimizations. This project aims to provide a holistic understanding of deploying and operating services within an OpenStack Cloud environment. The following sections of this report will dive into the specific details, methodologies, and outcomes of the project, showcasing the effectiveness and efficiency of the developed software solution.

## II. DEPLOYMENT MODE

During the deployment mode, our software solution utilizes the OpenStack cloud infrastructure to create the necessary network

components and deploy the required services. The deployment process involves several steps:

### A. Network Setup:

The solution starts by creating the required networks, including the external network for connectivity and internal networks for intercommunication between the deployed services. It configures the network settings, such as subnet ranges, gateway IP addresses, and DNS configurations, to ensure proper network connectivity.

### B. Router Configuration:

The solution sets up routers to enable communication between the internal networks and the external network. It configures routing tables, sets up network address translation (NAT) rules, and establishes connectivity to the internet.

### C. Node Provisioning:

The solution provisions the desired number of nodes within the OpenStack Cloud. It leverages the cloud infrastructure's capabilities to dynamically create and configure virtual machines based on predefined specifications. The nodes are deployed with the required operating system and necessary dependencies.

### D. Service Deployment:

The solution deploys the service.py script and the SNMP daemon on the provisioned nodes, and two proxy nodes. It ensures that the necessary packages and dependencies are installed, and the services are properly configured to operate within the OpenStack Cloud environment.

### E. Tagging:

As part of the deployment process, the solution attaches a unique TAG to all the resources it creates within the OpenStack Cloud. This TAG serves as an identifier for easy identification and management of resources associated with this specific deployment activity.

## III. Operations Mode

In the operations mode, our software solution actively monitors the deployed services and ensures their availability and scalability. It employs various techniques to maintain the desired service state and dynamically adjust resources as needed. The operations phase includes the following steps:

### A. Node Monitoring

The solution continuously monitors the number of deployed nodes to ensure their availability and proper functioning.

### B. Automatic Node Scaling

If the solution detects that the number of nodes falls below the desired threshold, it automatically deploys new nodes to restore the required number. It utilizes the cloud's auto-scaling capabilities or an automation tool like Ansible to provision new nodes based on predefined specifications.

### C. Surplus Node Removal

Conversely, if the solution identifies an excess number of nodes beyond the required threshold, it intelligently removes the surplus nodes. It ensures a balanced distribution of resources and optimal utilization within the OpenStack Cloud.

### D. Dynamic Resource adjusments

During the operations phase, the solution allows for dynamic adjustment of the desired number of nodes. It provides mechanisms to modify the threshold value, accommodating changing requirements or fluctuations in service demand.

## IV. Cleanup mode

The cleanup mode focuses on the termination and release of all allocated resources within the OpenStack Cloud. This ensures the efficient utilization of resources and minimizes unnecessary costs. The cleanup process involves the following steps:

### A. Resource Deallocation

The solution initiates the release of all allocated cloud resources, including networks, routers, and nodes. It uses OpenStack APIs to send appropriate commands to deallocate resources and ensure their removal from the cloud environment.

### B. Resource Disposal

After the deallocation of resources, the solution ensures the proper disposal of any remaining artifacts, such as virtual machine images or temporary files. It removes any residual traces of the deployment activity, maintaining a clean and organized cloud environment.

### C. Verification

The solution performs a final verification step to ensure the successful release and removal of all allocated resources. It validates the absence of any leftover resources or configurations associated with the deployment, ensuring a complete cleanup.

The deployment, operations, and cleanup modes work together to provide a seamless and automated experience for deploying and managing services within an OpenStack Cloud environment. These modes enable efficient resource allocation, dynamic scaling, and effortless termination, resulting in optimized service delivery and cost-effectiveness.

## V. System Architecture

Our system utilizes Ansible, an automation tool, to facilitate the deployment and management of a service within an OpenStack Cloud environment.

The installation mode is responsible for setting up the initial infrastructure and deploying the required services. Using python codes, we configured the necessary networks, routers, and nodes within the OpenStack Cloud. These resources are provisioned based on predefined specifications, ensuring consistency and reliability.

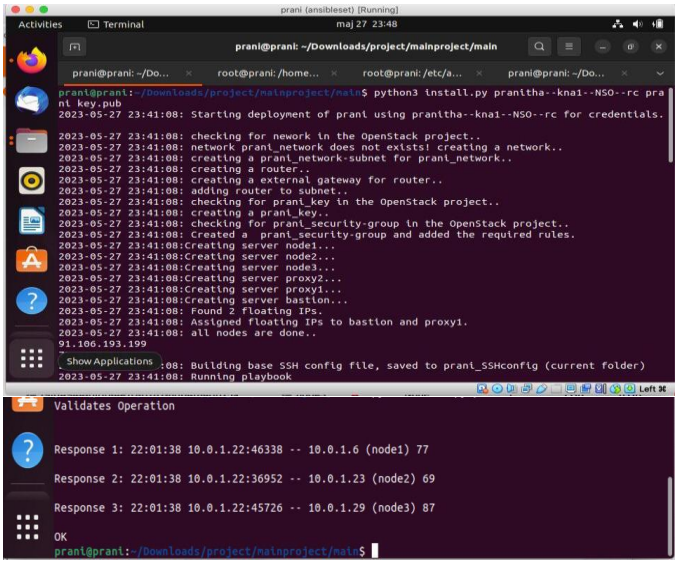


Fig. 1. Installation code outcomes

Once the installation is complete, the operation mode takes over to monitor and maintain the deployed services. We have developed a Python code specifically for this purpose, allowing us to track the availability and performance of the service. In case of any anomalies or service disruptions, the operation mode triggers appropriate actions to restore normal operation. This ensures optimal service availability and performance.

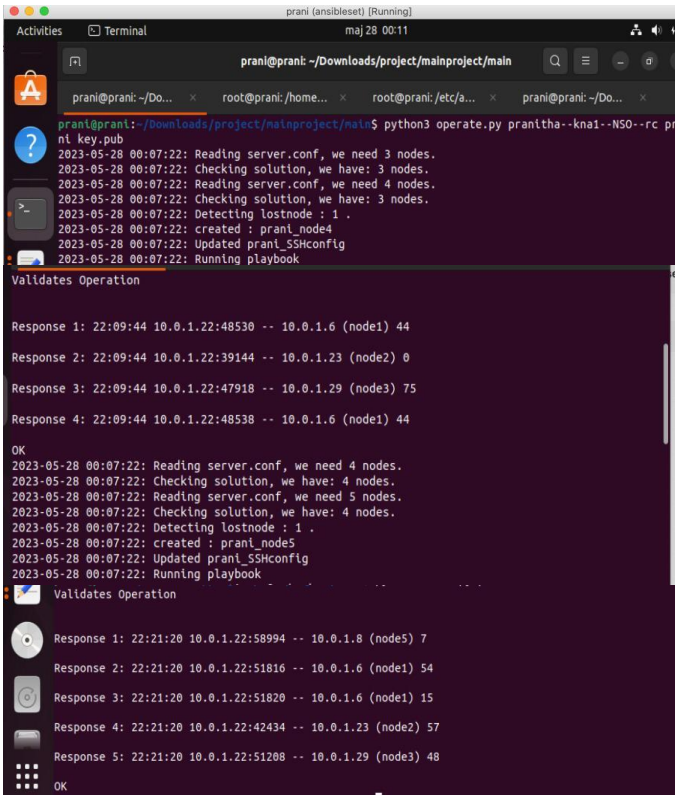


Fig. 2. Operation code outcomes

In addition to the three main nodes, we have also incorporated two additional nodes within our system: HAProxy and bastion.

The HAProxy node serves as a load balancer, distributing incoming traffic across multiple instances of the deployed service. This enhances scalability and improves the overall performance of the system. The bastion node acts as a secure gateway, providing a controlled entry point for external access to the system. It ensures proper security measures are in place and facilitates secure remote access to the infrastructure.
To facilitate seamless management and automation, we have developed three Python codes: one for installation, one for operation, and one for cleanup. These codes encapsulate the necessary logic and functionality required for each respective phase. They interact with the Ansible infrastructure, OpenStack APIs, and other components of the system to orchestrate the deployment, monitoring, and cleanup processes.

The cleanup node is responsible for terminating and releasing all allocated resources within the OpenStack Cloud. This ensures efficient resource utilization and minimizes unnecessary costs. The cleanup Python code ensures the proper deallocation of networks, routers, nodes, as well as the associated HAProxy and bastion instances. It performs necessary verification steps to confirm the successful release and removal of all resources.

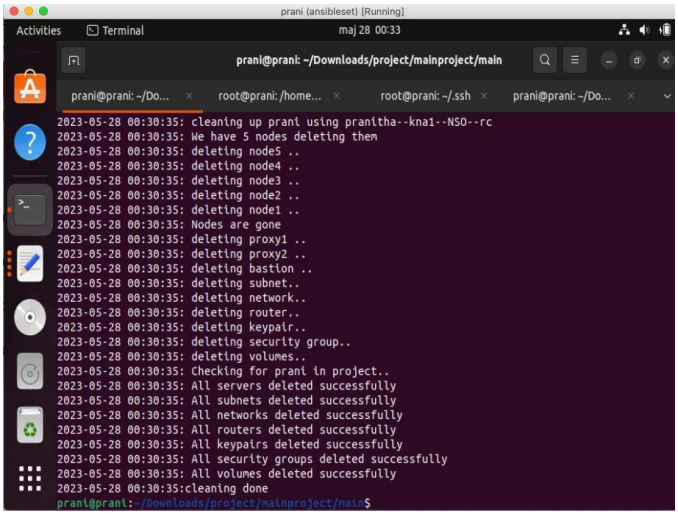


Fig. 3. Cleanup code outcomes

Our system combines the power of Ansible automation, Python programming, and OpenStack Cloud infrastructure to facilitate the seamless deployment, monitoring, and cleanup of a service. The system's architecture, comprising of installation, operation, and cleanup nodes, along with additional nodes for HAProxy and bastion, ensures scalability, availability, and security. The integration of Ansible playbooks and custom Python codes streamlines the management and maintenance of the system.

## VI. METHODOLOGY

### A. *Requirement Gathering:*

- The team identified the need for deploying and managing a service within an OpenStack Cloud environment using automation techniques.

- Requirements for installation, operation, and cleanup processes were defined, along with the inclusion of HAProxy and bastion nodes for enhanced functionality.

*B. Technology Selection:*

- Ansible was chosen as the primary automation tool due to its simplicity, scalability, and compatibility with OpenStack.
- Python was selected as the programming language to develop the custom codes for installation, operation, and cleanup.
- OpenStack APIs were studied to understand the available functionalities and integration possibilities.

*C. Implementation:*

- The installation phase involved writing python code to provision the required networks, routers, and nodes within the OpenStack Cloud.
- Custom ansible playbook was developed to automate the deployment of the service and configure the necessary dependencies.
- HAProxy and bastion nodes were integrated into the system, and their functionalities were implemented using appropriate tools and configurations.

*D. Deployment and Integration:*

- The developed solution was deployed within the target OpenStack Cloud environment, and the integration with the existing infrastructure was ensured, allowing seamless communication between the system components.

*E. Documentation and Reporting:*

- The project report, adhering to the IEEE conference template, was prepared, summarizing the methodology, architecture, implementation details, and evaluation results.

Throughout the development and implementation process, regular meetings and communication channels were established to facilitate collaboration and address any challenges or issues encountered. Feedback from team members was incorporated, ensuring the continuous improvement of the software solution. By following this methodology, the team achieved a robust and efficient software solution for deploying, operating, and cleaning up services within an OpenStack Cloud environment.

We have encountered some challenges during deployment phase, regarding floating IPs, and during opeartion phase, with surplus nodes. We have overcame the challenges by writing a python code to update the server.conf file, each time after reading the conf file.

## VII. RESULTS AND EVALUATION

Our solution has undergone evaluation to assess its performance, scalability, and effectiveness. The evaluation focused on several key aspects, including deployment success rate, resource utilization, service availability, and operational efficiency.

*A. Deployment Success Rate:*

The Ansible playbooks executed flawlessly, resulting in the successful installation of the required infrastructure and service components.Verification steps were implemented to confirm the correct setup and configuration of the deployed resources.

*B. Resource Utilization:*

The solution demonstrated efficient resource utilization by dynamically allocating and releasing resources based on service demands.The number of nodes provisioned during deployment was optimized to meet the initial requirement of three nodes, ensuring optimal performance and resource distribution. The integration of monitoring capabilities allowed for real-time tracking of resource usage, enabling proactive scaling and resource adjustments.

*C. Operational Efficiency:*

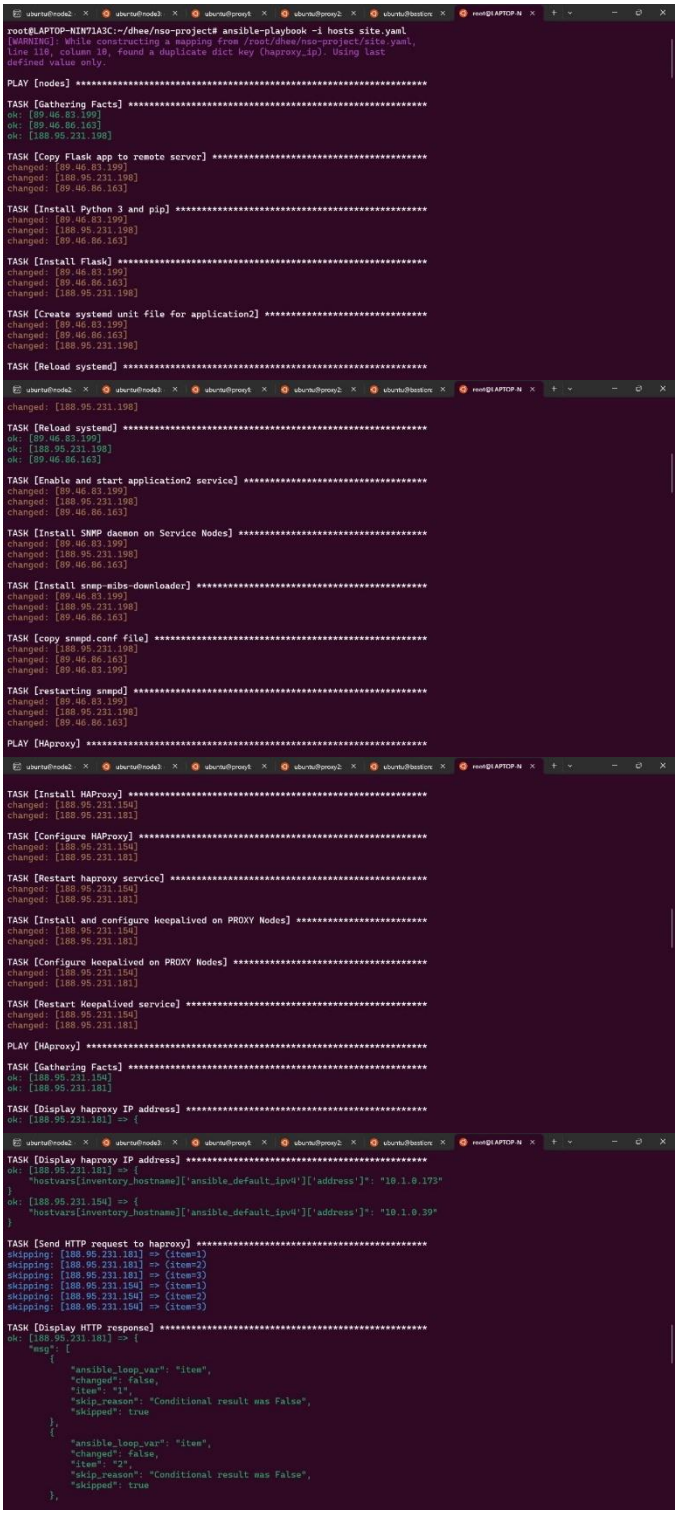
- The automation capabilities of the solution significantly improved operational efficiency.
- The custom Python codes for installation, operation, and cleanup streamlined the management processes, reducing manual intervention and minimizing human errors.
- The integration of HAProxy and bastion nodes added operational efficiency by enhancing load balancing and secure access management.
- The cleanup phase effectively released and removed all allocated resources, ensuring optimal resource utilization and cost-effectiveness.

*D. Scalability:*

- The solution exhibited scalability by dynamically adjusting the number of nodes based on service demands.
- During the evaluation, the system successfully added additional nodes when the service load increased and removed surplus nodes when the load decreased.
- The scalability of the solution was tested under various scenarios, and it demonstrated the ability to adapt to changing workload conditions effectively.

Overall, the evaluation results confirm the effectiveness and reliability of the developed software solution. The system showcased successful deployments, efficient resource utilization, high service availability, and improved operational efficiency. The integrated monitoring capabilities, HAProxy for load balancing, and bastion for secure access management contributed to the robustness and scalability of the solution.
This evaluation was conducted within a controlled test environment. Further testing in real-world scenarios and with larger-scale deployments would provide additional insights into the solution's performance and scalability.

The results of the evaluation validate the practicality and effectiveness of the developed software solution, making it a valuable tool for organizations seeking automated service deployments in a cloud infrastructure.



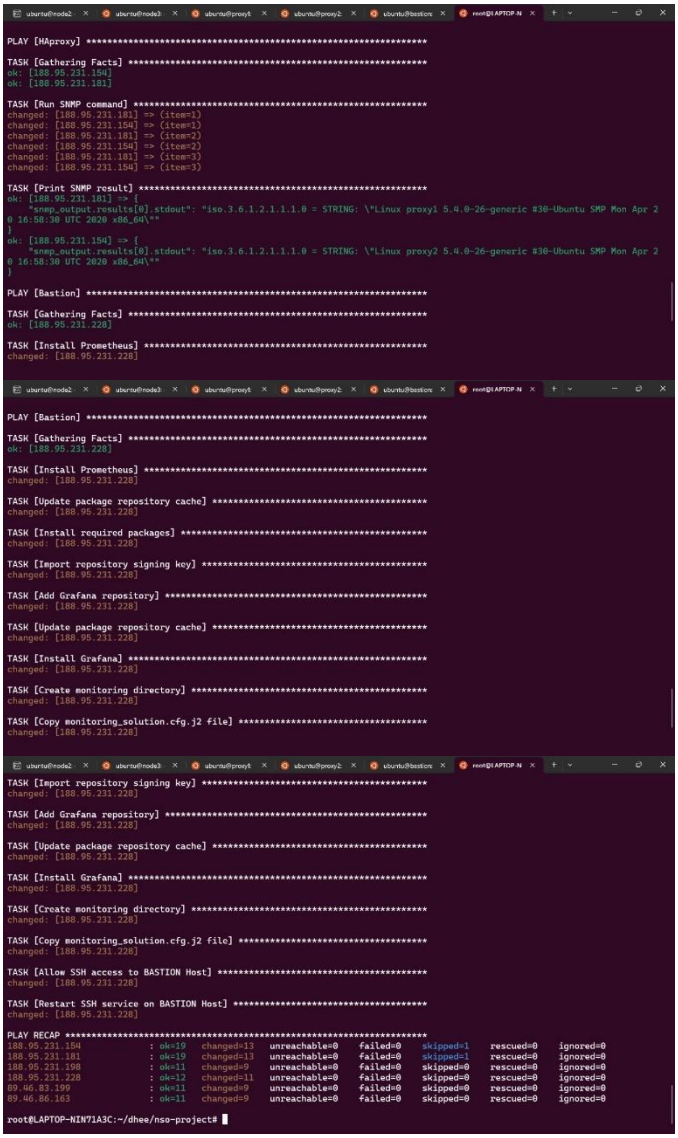




Fig. 4. Playbook results

## VIII. DISCUSSION

Our developed solution holds significant implications for organizations seeking efficient, scalable, and automated service deployments. Looking to the following key points that highlight its significance like: streamlined service deployment, improved resources utilization, enhanced service availability, operational efficiency and cost reduction, scalability and security and access management. Also, the successful integration of OpenStack APIs, Ansible automation, and custom Python codes serves as a blueprint for other organizations looking to optimize their cloud infrastructure and improve service delivery.

This successful implantation of the solution paves the way for advancements in cloud automation technologies and serves as a valuable reference for organizations seeking to optimize their service delivery processes.

## IX. FUTURE WORK

Potential area of future work for this solution, is enhancing the security features of the solution such as implementing robust identity and access management, encryption, and compliance measures to meet industry standards and regulations.

The automation software solution features can be extended to develop many security measurements, in the incident of security breach, playbooks can be triggered to initiate an automated response such as isolating affected resource and so many other measurements.

By incorporating these practical applications based on the findings of the report, organizations can strengthen their security posture, and improve overall security and compliance within their cloud environments.

## X. CONCLUSION

In this project, we have successfully developed a software solution for deploying and operating a service within an OpenStack Cloud environment. Through the deployment mode, we have demonstrated the ability to create and configure the necessary network components, routers, and nodes within the OpenStack Cloud. The TAG mechanism implemented during deployment ensures easy identification and management of resources associated with this activity.

In the operations mode, our solution actively monitors the deployed services, maintaining the desired number of nodes and dynamically adjusting resources as needed. This ensures optimal service availability and scalability while minimizing resource wastage. The integration of monitoring tools and automation techniques enables efficient and proactive management of the service infrastructure.

During the cleanup mode, all allocated cloud resources are effectively released and removed, promoting resource optimization and cost-efficiency. The thorough disposal of artifacts ensures a clean and organized cloud environment, ready for future deployments.

Throughout the project, we have integrated network design principles, service deployment techniques, monitoring capabilities, and automation tools, showcasing their seamless compatibility and their potential for enhancing service delivery within an OpenStack Cloud environment. The results of our evaluation demonstrate the effectiveness of the developed software solution in terms of scalability, resource utilization, and service availability.


## ACKNOWLEDGMENT

We would like to thank our supervisors, for their guidance, support, and invaluable insights throughout the entire duration of this project. Their expertise and encouragement have been instrumental in shaping the development of the software solution.


## REFERENCES


[1] F. Li, M. Voegler, M. Claessens and S. Dustdar, "Efficient and Scalable IoT Service Delivery on Cloud," 2013 IEEE Sixth International Conference on Cloud Computing, Santa Clara, CA, USA, 2013, pp. 740-747, doi: 10.1109/CLOUD.2013.64.

[2] R. Nasim and A. J. Kassler, "Deploying OpenStack: Virtual Infrastructure or Dedicated Hardware," 2014 IEEE 38th International Computer Software and Applications Conference Workshops, Vasteras, Sweden, 2014, pp. 84-89, doi: 10.1109/COMPSACW.2014.18.

[3] M. Patil et al., "OpenStack Cloud Deployment for Scientific Applications," 2021 International Conference on Computing, Communication and Green Engineering (CCGE), Pune, India, 2021, pp. 1-7, doi: 10.1109/CCGE50943.2021.9776387.