National College of Ireland

Management of Self-Healing Systems for Multi-Cloud Deployments on Kubernetes

MSc Research Project
Cloud Computing

Vaishnavi Udayrao Deshpande
Student ID: x23183209

School of Computing
National College of Ireland

Supervisor: Sai Emani

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Vaishnavi Udayrao Deshpande
Student ID:	x23183209
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Sai Emani
Submission Due Date:	12/12/2024
Project Title:	Management of Self-Healing Systems for Multi-Cloud Deployments on Kubernetes
Word Count:	2025
Page Count:	9

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Vaishnavi Udayrao Deshpande
Date:	12th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Management of Self-Healing Systems for Multi-Cloud Deployments on Kubernetes

Vaishnavi Udayrao Deshpande
x23183209

1 Introduction

This Configuration Manual details out the configuration of the software tools and settings required to replicate the experimental setup used for the project work 'Management of Self-Healing Systems for Multi-Cloud Deployments on Kubernetes'. This includes the configurations for Kubernetes, Jenkins, Datadog, AWS, GCP, and Azure along with the necessary environment setup to ensure successful deployment, monitoring, and management of the self-healing framework.

Magalhães and Silva (2013); Pandi et al. (2023); Mfula and Norminen (2018); Pellegrini et al. (2017); Samarakoon et al. (2023); Sharma (2022); Ye et al. (2016)

2 System Requirements

The designed system requires some of the tools and software and it is important to make sure that the following software and Services are available and configured correctly before starting the setup.

- **Operating System:** In this research study, AWS EC2 is used which uses : Amazon Linux 2023.6.20241121
  But Ubuntu 20.04 or equivalent Linux distribution can also be configured.
- **Cloud Providers:** AWS, GCP, Azure accounts with appropriate permissions and IAM Roles
- **Kubernetes:** Kubernetes 1.22.1 or later
- **Jenkins:** Latest stable version
- **Datadog:** For Monitoring and Observability
- **Docker:** For Containerization
- **Kubectl:** Kubernetes Command Line Tool for interacting with Kubernetes
- **Cloud Command Line Tools:** AWS CLI, GCP CLI, Azure CLI
- **Helm:** For Kubernetes package management

3 Cloud Provider Setup

As the system designed is for multiple clouds, make sure that all of the clouds are set-up correctly. For this framework, three of the most popular public clouds are used which are AWS, GCP and Azure. This section provides the detailed configuration for each od the clouds below.

3.1 AWS Setup

AWS is primarily used in this design work. AWS EKS, ECR and EC2 instnace, are the services used. Elastic Kubernetes Service is set up along with the Elastic Container registry which is used for storing the application images. These application images then pulled by kubernetes cluster to deploy that application.

- **IAM Permissions:** Create necessary IAM Roles and Policies for seamless interaction with EKS and other AWS Resources as well. Following table gives out the details of required IAM permissions and roles for AWS integration with other tools.
  - **IAM Permissions:** Create necessary IAM Roles and Policies for seamless interaction with EKS and other AWS resources. The following table outlines the required IAM permissions and roles for AWS integration.

Cloud	**Role/Service Account**	**Key Permissions/Roles**
AWS	Jenkins Role	eks:DescribeCluster, ecr:*, ec2:*, autoscaling:DescribeAutoScalingGroups
	EKS Cluster Role	AmazonEKSClusterPolicy
	Node Group Role	AmazonEKSWorkerNodePolicy, AmazonEKS$_{C}NI_{P}olicy$, $AmazonEC2ContainerRegistryReadOnly$

Table 1: IAM Roles and Permissions for AWS

- **EKS Cluster:** Create an EKS Cluster in your AWS project environment using the AWS management console or AWS CLI.
  - Set-up VPC, Subnets as well as Security Groups for Kubernetes nodes
- **AWS SSO:** For this study, AWS SSO is used and hence, configuration of AWS SSO authentication is required. AWS accounts will not require AWS SSO configuration setup.

In this research work, AWS Cloud is used extensively. Resources like EC2, EKS, and ECR are used with required roles and permissions. EC2 instance is created with t2-medium for fast processing. Once the instance is ready, it is connected via bash terminal and Jenkins is installed in the same. Once the Jenkins is installed, it is accessed using Public IPv4 address

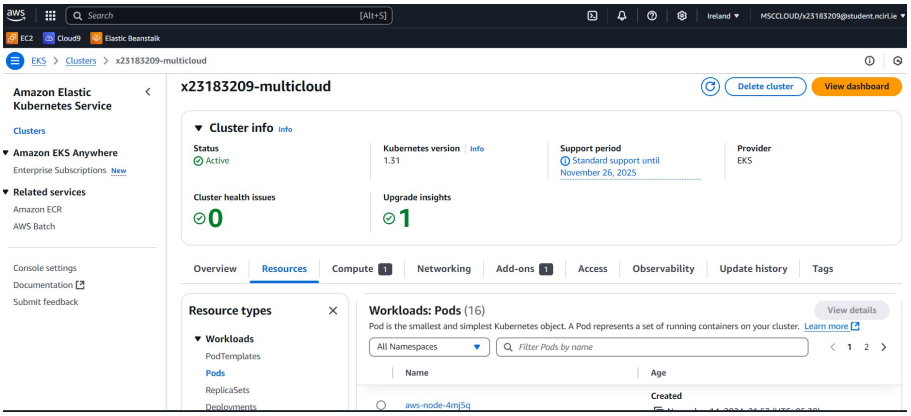

Figure 1: EKS Setup

3.2 GCP Setup

For the GCP setup, GCP account is created. For purpose of this study, a sample project in the GCP account is created and used. As a DevOps best practice, a Service account is created which is given a 'contributor' role in the project which makes it easy for GKE cluster to communicate with Jenkins which is in the EC2.

- **GKE Cluster:** GKE cluster is created inside the GCP using GCP console. It can be created through GCP CLI as well. Once the cluster is setup, Kubernetes API needs to be enabled in the project and it should have all the required appropriate permissions.
- **Service Account:** For this study, a specific service account is created which has required permissions and roles to interact with Jenkins, and GKE Clusters. This will make sure that the developers and devops engineers won't have elevated permissions individually.
- **Google Cloud IAM:** Set up Google Cloud IAM roles and permissions to interact with Kubernetes clusters and other resources for the service account that is created. Following table shows the roles and policies for GCP:
  - **IAM Permissions:** Create necessary roles and permissions for seamless interaction with GKE and other GCP resources. The following table outlines the required IAM permissions and roles for GCP integration.

Cloud	**Role/Service Account**	**Key Permissions/Roles**
GCP	Jenkins Service Account	roles/container.admin, roles/storage.admin, roles/logging.logWriter, roles/monitoring.metricWriter
	GKE Cluster Service Account	roles/container.clusterViewer, roles/compute.networkAdmin

Table 2: IAM Roles and Permissions for GCP

3.3 Azure Setup

For this project, Azure for Students account is used which provides you free credit of $100. AKS Cluster, artifact repository, resource group, are the services that are required to be created. Once the resource group is set up, the IAM permissions which allow the cluster to be integrated with other resources are given to the group.

- **AKS Cluster:** A Kubernetes cluster is created using the Azure console in the AKS with 2 node pools, both having Standard_DS2_v2 sizes
  - Set up networking, Security Groups, VNet
- **Azure IAM:** Configure Azure active directory roles and permissions for accessing the AKS cluster. Following table gives the idea of required IAM roles and permissions for this setup:
  - **IAM Permissions:** Create necessary roles and permissions for seamless interaction with AKS and other Azure resources. The following table outlines the required IAM permissions and roles for Azure integration.

Cloud	**Role/Service Account**	**Key Permissions/Roles**
Azure	Jenkins Service Principal	Contributor, Azure Kubernetes Service Cluster User Role
	AKS Managed Identity	Reader, Azure Kubernetes Service Node Pool Contributor

Table 3: IAM Roles and Permissions for Azure

- **Azure Resource Group:** The resource group contains the AKS cluster and Networking, storage and associated infrastructures. The Jenkins service principle and Managed identity requires contributor-level access to the resource group to interact with AKS for deployments.
  All components which are required for Kubernetes deployments, such as Virtual machines, public IPs, loadbalancers, reside within the same resource group for centralized managements.

4 Kubernetes Configuration

The Kubernetes configuration for this management of self-healing multi-cloud deployments on Kubernetes project is designed to deploy as well as manage the application consistently across AWS EKS, GCP GKE, Azure AKS clusters. This section explains the configurations done for setting up Kubernetes across all three cloud platforms.

- **Cluster Configuration:** Ensure that the Multi-cloud kubernetes clusters (EKS, GKE, AKS) are configured to allow the communication between them.
  - Define Name-spaces and resource quotas to manage all the deployments.

- **AWS:** use any storage class for persistent volume. Ensure that IAM roles and policies 1 are properly attached. Along with this, configure security groups to allow NodePort.
- **GCP:** Use standard or SSD storage class for persistent volumes. Assign proper roles as per mentioned in the table 2 to the service account used by Jenkins. Open the NodePort range (30000-32767) to allow traffic to application.
- **Azure:** Use Azure disks or Azure files for persistent volumes. Configure the network security groups (NSG) for the AKS nodes to allow NodePort access. Ensure that AKS has managed identity with required permissions as mentioned in the table 3

- **Self-Healing Mechanisms:** Deploy Kubernetes operators to manage the life-cycle of the self-healing systems. 2
  - Setup the Horizontal Pod Autoscalers (HPA) to manage the pod scaling based on the resource utilization.
  - implement the health checks for automated recovery of failing pods.

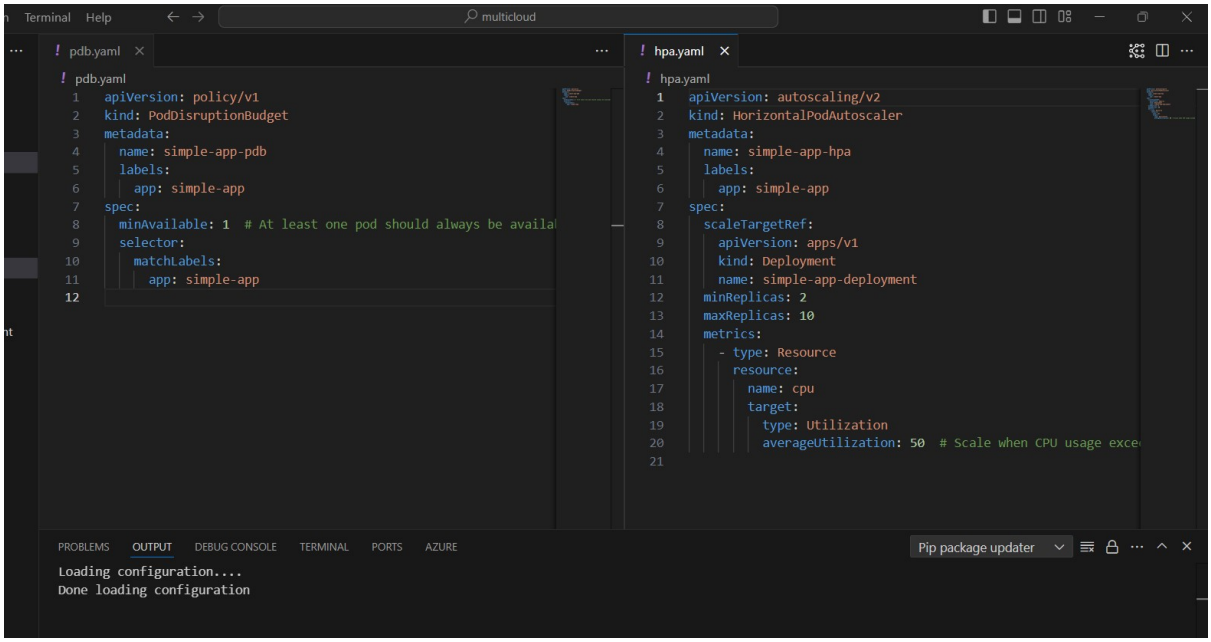

Figure 2: PDB & HPA .yaml files

5 CI/CD Configurations

5.1 jenkins Setup

- **Jenkins Installation:** EC2 instance is set up on AWS which will host the Jenkins. Once the EC2 is configured and running, connect to the instance using bash terminal in local system or using tools like Putty. After connecting, install the Jenkins on instance and start using Jenkins console via Public IPv4 address of EC2. 3

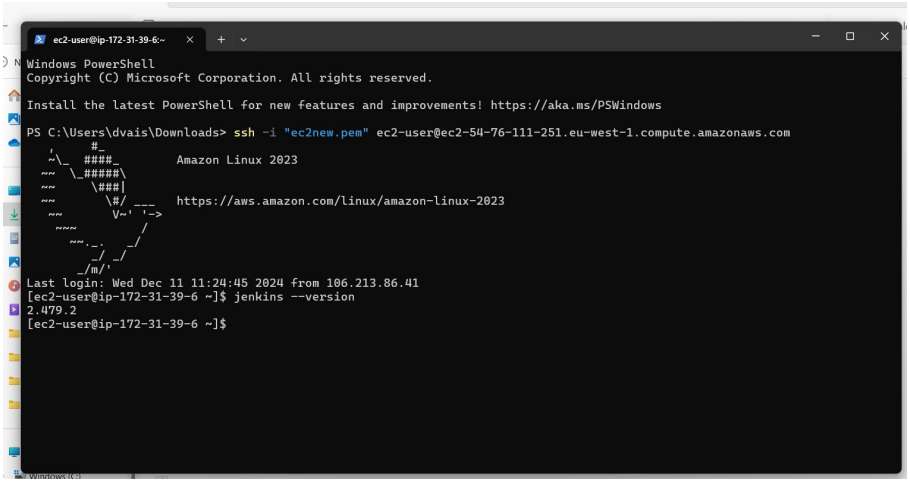

Figure 3: Jenkins on EC2

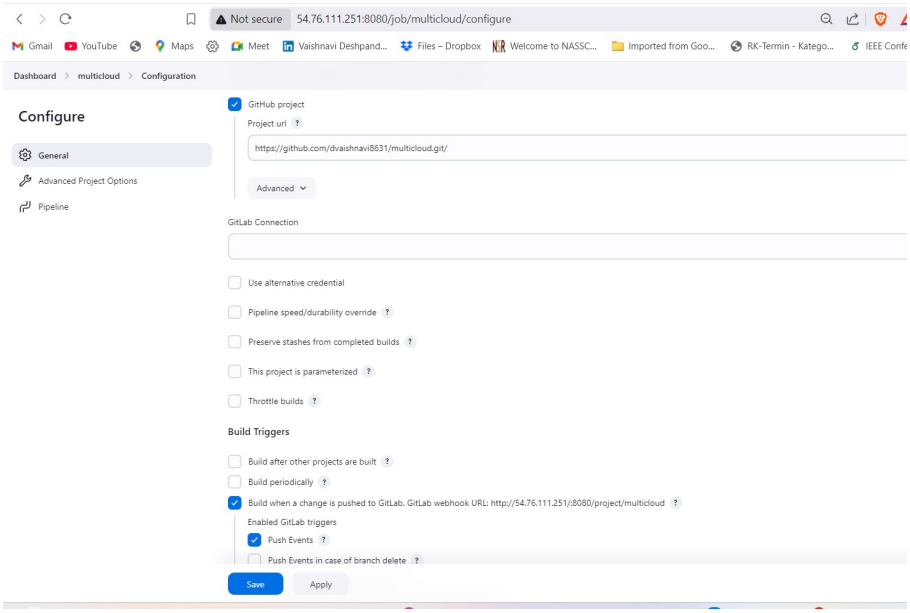

Figure 4: Jenkins Configuration

- **Pipeline setup:**

  Install the required plugins in the Jenkin's management console and add the keys and Tokens as well as passwords which are required to integrate Jenkins with AWS, GCP, Azure, Github, Datadog, etc.

  - Using Jenkins console, create a new pipeline and Configure the pipeline for deploying the application on EKS, GKE, and AKS clusters.
  - Make sure to include the steps for building the docker images for application, pushing them to the respective container registries (ECR, GAR, ACR), and deploying those to Kubernetes using **kubectl** and Helm.
  - Configure the pipeline to trigger when there is 'push' event in the GitHub repository which is has the application files. 4

6 Monitoring

6.1 Datadog Integration

Datadog is an open-source platform for monitoring purpose where custom dashboards for custom metrics can be created. Once the deployment of application is done and the system is up and ready, install the datadog agents on each of the clusters using bash terminal. And using the Datadog console, create the custom metrics and dashboards to visualize the system and kubernetes clusters efficiently.

- **Datadog Agent setup:** Install the datadog agent on each of the Kubernetes clusters to collect metrics, logs and traces from infrastructure and applications deployed. In the EC2 instance, switch to the AWS cluster context and install datadog agent and deploy it on the EKS. Once done, follow the same steps for GCP GKE and Azure AKS. 5

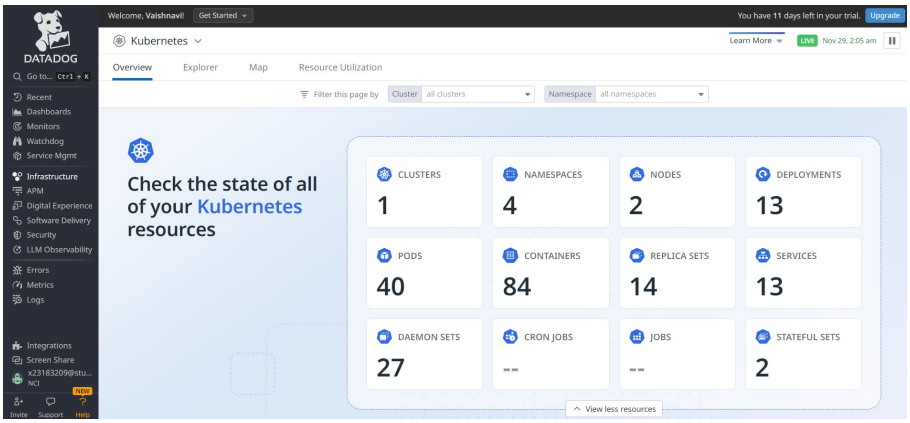

Figure 5: Datadog Dashboard

- **Dashboards:** Create custom datadog dashboards to monitor key metrics such as pod health, resource utilization (CPU, Memory), and network performance as well. TO create the dashboards, go to new dashboard and decide the metrics that are required to be monitored. Once decided, create the metric and datadog will start monitoring that service or resource. 6

7 Self-Healing System Configuration

- **Failure Detection:** Make sure to define failure conditions such as pod crashes, high resource utilization, and network issues. Here, PDB and HPA are configured using yaml files along with Kubernetes in the Github repository.

- **Automated recovery:** Set up Kubernetes to automatically recover from failures such as by restarting pods, scaling services, or shifting workloads between clusters.

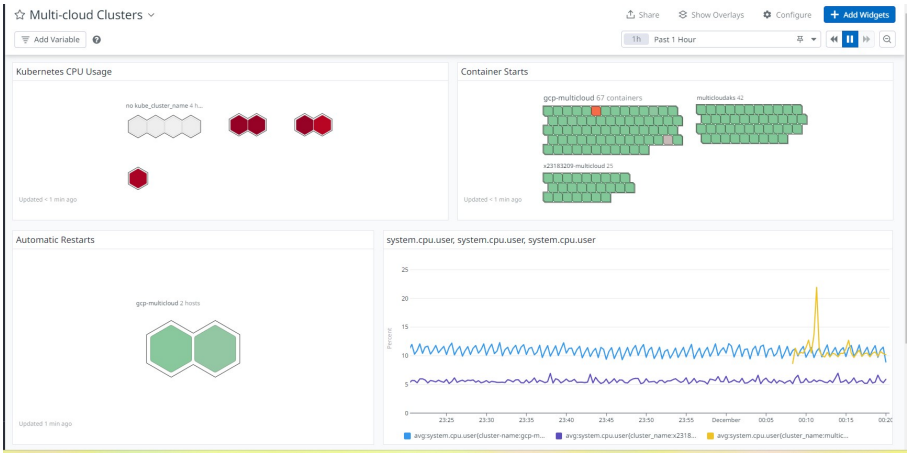

Figure 6: Datadog metrics

```
[ec2-user@ip-172-31-39-6 ~]$ kubectl get pods -o wide -n default
NAME                                  READY   STATUS    RESTARTS   AGE    IP           NODE                                                NOMINATED NODE   READINESS GATES
datadog-agent-lnqcw                   3/3     Running   0          46h    10.32.0.32   gke-gcp-multicloud-default-pool-691607e2-v8bb   <none>           <none>
datadog-agent-nr8rv                   3/3     Running   0          46h    10.32.1.51   gke-gcp-multicloud-default-pool-691607e2-qt0j   <none>           <none>
simple-app-deployment-5b6bc6fb46-77bld   1/1     Running   0          115s   10.32.0.39   gke-gcp-multicloud-default-pool-691607e2-v8bb   <none>           <none>
simple-app-deployment-5b6bc6fb46-jvlpv   1/1     Running   0          12h    10.32.1.54   gke-gcp-multicloud-default-pool-691607e2-qt0j   <none>           <none>
[ec2-user@ip-172-31-39-6 ~]$ kubectl delete pod simple-app-deployment-5b6bc6fb46-jvlpv -n default
pod "simple-app-deployment-5b6bc6fb46-jvlpv" deleted
[ec2-user@ip-172-31-39-6 ~]$ kubectl get pods -o wide -n default
NAME                                  READY   STATUS    RESTARTS   AGE     IP           NODE                                                NOMINATED NODE   READINESS GATE
S
datadog-agent-lnqcw                   3/3     Running   0          46h     10.32.0.32   gke-gcp-multicloud-default-pool-691607e2-v8bb   <none>           <none>
datadog-agent-nr8rv                   3/3     Running   0          46h     10.32.1.51   gke-gcp-multicloud-default-pool-691607e2-qt0j   <none>           <none>
simple-app-deployment-5b6bc6fb46-4zk72   0/1     Pending   0          4s      <none>       <none>                                              <none>           <none>
simple-app-deployment-5b6bc6fb46-77bld   1/1     Running   0          3m25s   10.32.0.39   gke-gcp-multicloud-default-pool-691607e2-v8bb   <none>           <none>
```

Figure 7: pods recreation

8 Testing Setup

After configuring all the component, test the setup by simulating failure scenarios (e.g. pod crashes, node failures) and verify that the self-healing mechanisms detect and recover from the failures.

- **Test 1:** Simulate a pod failure and encure it restarts automatically.
- **Test 2:** Simulate high CPU Usage and verify that the HPA scales the pods accordingly.
- **Test 3:** Simulate a node failure and check the workload is distributed to healthy nodes.

9 Conclusion

This manual provides the configuration needed to replicate the management of multi-cloud self-healing processes for deployments on Kubernetes. By following these steps, researchers should be able to set up a resilient and automated infrastructure which is capable of recovering from failures in multi-cloud environments.

References

Magalhães, J. P. and Silva, M. L. (2013). A framework for self-healing and self-adaptation of cloud-hosted web-based applications, *IEEE International Conference on Cloud Computing Technology and Science* .

Mfula, H. and Norminen, K. J. (2018). Self-healing cloud services in private multi-clouds, *2018 International Conference on High Performance Computing Simulation* .

Pandi, S. S., Kumar, P. and Suchindhar, R. M. (2023). Integrating jenkins for efficient deployment and orchestration across multi-cloud environments, *International Conference on Innovative Computing, Intelligent Communication and Smart Electrical Systems (ICSES)* .

Pellegrini, R., Rottmann, P. and Strieder, G. (2017). Preventing vendor lock-ins via an interoperable multi-cloud deployment approach, *The 12th International Conference for Internet Technology and Secured Transactions (ICITST-2017)* .

Samarakoon, S., Bandara, S., Jayasanka, N. and Hettiarachchi, C. (2023). Self-healing and self-adaptive management for iot-edge computing infrastructure, *Moratuwa Engineering Research Conference (MERCon)* .

Sharma, V. (2022). Managing multi-cloud deployments on kubernetes with istio, prometheus and grafana, *8th International Conference on Advanced Computing and Communication Systems (ICACCS)* p. 10.

Ye, F., Wu, S., Huang, Q. and Wang, X. A. (2016). The research of enhancing the dependability of cloud services using a self-healing mechanism, *International Conference on Intelligent Networking and Collaborative Systems* .