electronics

MDPI

Article

Optimization of DevOps Transformation for Cloud-Based Applications

Ahmed Mateen Buttar [1], Adeel Khalid [1], Mamdouh Alenezi [2], Muhammad Azeem Akbar [3,*], Saima Rafi [4], Abdu H. Gumaei [5] and Muhammad Tanveer Riaz [6,7]

[1] Department of Computer Science, University of Agriculture Faisalabad, Faisalabad 38000, Pakistan
[2] Software Engineering and Disruptive Innovation (SEDI), College of Computer and Information Sciences, Prince Sultan University, Riyadh 11586, Saudi Arabia
[3] Software Engineering Department, LUT University, 15210 Lahti, Finland
[4] Department of Informatics and Systems, University of Murcia, 30100 Murcia, Spain
[5] Department of Computer Science, College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia
[6] Department of Mechanical, Mechatronics and Manufacturing Engineering, University of Engineering and Technology Lahore Faisalabad Campus, Faisalabad 38000, Pakistan
[7] Department of Electrical, Electronic, and Information Engineering, Università di Bologna, 40136 Bologna, Italy
* Correspondence: azeem.akbar@lut.fi or ahmedmatin@hotmail.com

Abstract: Rapid software development is critical for meeting company objectives and competing more effectively in the competitive IoT infrastructure. DevOps is a growing technique that enables enterprises to provide high-quality software capabilities through automation, to improve team communication, and to increase efficiency across the software product lifecycle. Research problem: Due to the increased demand for new products and technologies, a huge overwork shifted on the organizations for introducing software with pace and to become stable to compete with others. Due to this, the majority of organizations prefer an automated system for product development and require cloud-based applications. The git version control system is used for version management and Docker is used to package code and provide libraries. AWS services are leveraged to deploy an application as a cloud. Jenkins is used as a CI/CD pipeline to manage various phases of development and to make the development process continuous. The ELK stack is used to monitor and visualize the execution of code. In light of the findings, DevOps is an efficient method for cloud application deployment and resource selection based on the relative importance of each optimized objective in terms of value parameters such as cost, memory, and CPU capacity, and that the method can be tailored to specific application requirements. The findings of this analysis indicate that an application can be deployed to the cloud using DevOps techniques. The proposed approach cost 60% less at full weight 1.0 and 11.3% less with no weight compared to the benchmark solution's 15.078%

Keywords: DevOps; cloud pipelines; continuous integration; continuous development

Citation: Buttar, A.M.; Khalid, A.; Alenezi, M.; Akbar, M.A.; Rafi, S.; Gumaei, A.H.; Riaz, M.T. Optimization of DevOps Transformation for Cloud-Based Applications. *Electronics* **2023**, *12*, 357. https://doi.org/10.3390/electronics12020357

Academic Editor: Antonio Brogi

Received: 29 October 2022
Revised: 22 December 2022
Accepted: 4 January 2023
Published: 10 January 2023

1. Introduction

Modern technology contributes to our life by providing tools and it's also helping us to improve our working conditions. Due to the growing demand for new products and technologies, businesses are distributing software at a faster speed and are more consistent than others. As a result, many businesses opt for an automated product development program and require cloud-based applications. As a solution, Cloud collaboration using DevOps makes organizations very powerful, as they not only facilitate the development of software products but also facilitate the implementation and control of the deployment process. DevOps is a set of practices that combines software development (Dev) and IT operations (Ops). Leited et al. [1] described that "DevOps is a collaborative and multidisciplinary effort within an organization to automate continuous delivery of new software

versions, while guaranteeing their correctness and reliability". DevOps along with things like the CI/CD pipeline is to avoid frequent code changes during development work and to complete the project promptly.

DevOps can run hundreds of tests within one day and receive feedback from clients after each delivery. This will help the organization to explore additional features that can be implemented in the project and minimize configuration issues. According to Jambunathan et al. [2] when it comes to cloud deploying applications, DevOps is crucial. DevOps' primary skills for handling cloud-based computing include automated application deployment, Infrastructure as a Code, and delivering servers. DevOps is a vital part of cloud computing that manages infrastructure, application deployment, and application functionalities in a variety of contexts. From the perspective of infrastructure providers, cloud computing has been viewed as an economy of scale, and in many instances, cloud economics has been used as a deciding factor between private and public clouds [3]. In environmental constraints, DevOps aid in making a high-quality product, continuous delivery and helping end-user with quality software [4]. DevOps also enables quick responses to changing client requirements. DevOps allows developers and operations to work in a shared environment. DevOps opens the possibility of removing organizational and cultural divides, as well as lowering the cost of fault detection in the early phases [5].

Besides from benefits of DevOps, shifting from local infrastructure to microservices, combining different domains tools, using new tools, different tool kits between dev and ops teams are challenges. According to Ellen et al. [6], the major difficulties for DevOps acceptance in the software business include poor communication, ingrained organizational culture, market limits, scalability, and diverse ecosystems. Hence, to resolve the issues that DevOps teams may experience during continuous integration, deployment, and testing. Our study uses Jenkins as a helping tool for solving the problem related to the deployment. We have also used the Jenkins pipeline to manage various phases of deployment process. Furthermore, we have selected a method for implementing DevOps using Continuous Delivery, and we build a specific Continuous Delivery system design based on top of the Amazon Web Services (AWS) cloud.

As a result, to address a challenge; how can IoT applications be developed, deployed, and managed in a multi-cloud environment while remaining within the overall parameters of the existing organization ecosystem? The motivation is to propose an algorithm to support cloud-based applications to employ DevOps techniques. The findings of this analysis indicated that an application can be deployed to the cloud using DevOps techniques.

2. Background and Related Work

2.1. Background

Cois et al. [7] explored that an application is implemented in the production environment once the development team has completed all of its development work and the operations team has created and configured the application's deployment environment independently. Soni [8] described that Coding, designing, integrating, testing, debugging, infrastructure configuring, access control, establishing runtime environment, and deploying experience in different environments are all phases of application development. Barna et al. [9] described a method for creating an AMS for data-intensive containerized applications that are multi-tiered and multi-layer. They used an auto process model which generates quality measures relying on the software's architecture, and they design for possible reactive system actions if the performance model detects any potential issues.

According to Guerriero et al., [10] the aim of using DevOps for cloud applications is to help the cloud applications in controlling their run time processes and in minimizing the execution cost. However, an Autonomous Management System (AMS) is required to facilitate DevOps activities. Various strategies were proposed in past by researchers for scaling installations over several clouds in Multi and Hybrid Cloud Autonomous Structures. For example, an auto-scaling approach [11] for Hybrid clouds aim of providing efficient resource consumption. The -self-organized technique is designed to accommodate the

varying strains of application domains while still adhering to service level agreements such as timeframes, price-based, and functionality-based requirements. The CI/CD pipeline is a set of actions that were performed according to defined rules and are automatically triggered by CI/CD exercises [12]. With CI/CD pipeline methods, you can avoid doing time-consuming tasks manually. This will help the organization to explore additional features that can be implemented in the project and minimize configuration issues. Elastically Ruling the Cloud [13] is used for applying adaptive modifications in Federated clouds, leveraging the Rule Interchange Format (RIF) to enable rules to be reused on multiple rule engines. Adopting DevOps administration for business applications by merging variable costs and existing network management technologies, we were able to improve the task in cloud-based applications

DevOps provides more comprehensive support for cloud application deployment using a variety of tools that offer automation and continuous integration (CI). For any modern business, software is becoming more and more important. The management of software change in DevOps environment has evolved into a critical skill for a digital organization due to the high volatility of software and its surroundings. In fact, the constantly evolving technical features of cloud-based software may have an equal impact on its economic aspects [14]. Tsilionis et al. [15], mentioned advantages of utilizing cloud-enabled platforms/resources include, among other things, cost-effective access to significant on-demand processing, storage, and network capacities while being able to group a pool of qualified individuals via virtualized environments throughout the entire development process [16].

Jaatun et al. [17] described that when it comes to moving applications to the cloud, security is a key concern for developers. To keep the market stable, they must make data very secure. DevOps plays a crucial role in cloud security. While some unauthorized sources try to gain access to data, DevOps responds quickly to the developer team. Almeida et al. [18] discussed that managing cloud resources or prices is a multifaceted issue, improper resource allocation may have an impact on the software product's performance as well as its profitability and DevOps can play an important role continuously monitoring the utility of the customer. According to Arulkumar and Lathamanju [19] DevOps is a tool that relies on automation throughout the lifecycle. DevOps is a method for rapidly developing applications while maintaining consistency and security. Due to their continuous delivery of software, the technique continuously performs designing, testing, and rewriting code and receives continuous feedback from customers. Many product deliveries are completed in a single day. This aids in the development of high-quality software at a rapid speed. Large-scale and distributed cloud applications [20] often face two major challenges: (i) a lack of compatibility among cloud systems, and (ii) difficult maintenance and evolution management. Interoperability issues across cloud solutions lead to vendor support and limit the creation of multi-cloud applications. This makes it difficult for cloud application providers to make use of the unique features of existing cloud solutions, such as improving performance, availability, and affordability. The second difficulty is maintaining and evolving such complex cloud apps, particularly in a multi-cloud DevOps environment.

Wettinger et al. [21] worked on artifacts for DevOps and their transformation using TOSCA as standardized metamodel. The artifacts described here allows DevOps to be integrated with model and application deployment. It allows you to classify artifacts by node and environment. They utilize open-source building components to handle all model and framework integration. Shin and Williams [22] explained that one of the most serious issues with software security is that it is difficult to anticipate all threats that the system will face. Furthermore, even in firms that are particularly vulnerable to espionage, discovered flaws typically go unresolved for years. Despite this much less vulnerability is reported than conventional software flaws. These are the reasons why many people believe that software security isn't a huge deal and that associated vulnerabilities aren't given the same priority as other software flaws. Basiri et al. [23] highlighted that the goal of the DevOps paradigm is to reduce the software delivery cycle. The goal is for every modification to be

easily integrated into the production system while maintaining excellent quality. Netflix is already using chaos engineering techniques, which enable ideas like continuous integration and continuous deployment in the scope of DevOps. Some other studies also worked on DevOps adoption in software organization for example Akbar et al. [24] discussed DevOps from perspective of identification of DevOps success factors and Rafi et al. [25] proposed a DevOps business model for small startups and work from home environment. Venkateswaran and Sarkar [26] proposed a model for deploying applications on hybrid clouds as well as determining the optimum hosting match. They concentrated on the algorithm for selecting the best cloud to conduct the operation regarding the data center's geographical position.

2.2. Available Tools and Related Work

CI/CD tools are used to automate the development of builds from code and their subsequent deployment to the domain controller. Code integration is one of the most widely used strategies in the software development industry because it reduces the problem of integrating source code when software is built across multiple sites allowing developers to quickly build, evaluate, analyze, and deploy software [27].

Docker is a platform that enables developers to assemble code, libraries, and other configuration files into a single image which can be used to run an application as a container as seen in Figure 1. Developers build their software, frameworks, and modules into Docker containers, which they can then deploy to testers and operational experts [2].

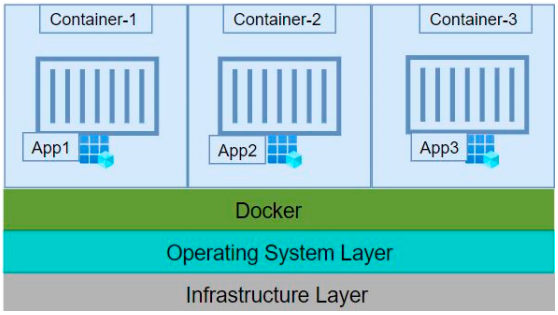

Figure 1. Containerised Application with Docker [2].

The CICD pipeline [28] is made up of many phases like Designing, configuration management, Production Support, Continuous integration, Iterative Development, and Continuous Deployment. The initial stage of the pipeline is Continuous Integration, which allows developers to often merge their work into the repository. Continuous Deployment is a process of systematically delivering ready code, whereas Continuous Delivery seems to be a method of planning and ready code for deploying.

2.3. Version Control with Git

Version control system (VCS) adoption will help you to have complete access to your code. All the access controls are set and access is given to limited number of people i.e., who works on it, when does he commit it, and what changes are made to the code?. It improves code security and ensures that code is delivered quickly. VCS stores the code in a designated location and is accountable for its availability. The most popular VCSs are AWS CloudWatch [29], Git [30], Mercurial [31], and SVM [32]. It is more versatile and freer. Git is the most popular among all. It also offers hosting services. Git is used by the majority of new businesses and students. In our work, we also use git to maintain our source code. The developer writes their code and git help them to track, commit and finally push the code to a dedicated storage place called GitHub

2.4. Create Container Using Docker

Docker is an open-source tool for creating a container. It helps indirectly in developing, building, deploying, and executing software in isolation [33]. It does so by creating

containers that completely wrap a software. You can put an application in a container with the help of Docker and then can be deployed easily. The isolation provided by the container gives a layer of security to the containers. Docker Architecture consists of Docker CLI, Docker host, and Docker Registry. You use Docker build command on Docker CLI to build Docker images. The command gets API of code and gives it to Daemon. In next step Daemon read the command and check which image needs to build and then build the required image. Docker swarm [34] is an important feature of Docker that helps to orchestrate the Docker container. Orchestration allows to manage and maintain multiple containers. This is especially helpful in software development where we may be making use of micro-services architecture as it breaks down the software into small chunks.

2.5. Use Jenkins for Continuous Integration

Jenkins is the most popular tool in the software development industry because it is free for use. It is flexible and provides amazing supportability. Jenkins supports thousands of plugins and no other continuous integration tool support such a huge number of plugins. The Jenkins automatically detect from source code manager then clone it and then move it to the other process like build, test and deploy

The Jenkins pipeline [35] is shown in Figure 2 where the Developer pushes their written code to the source code manager and then automatically pulls through Jenkins with the help of jobs. Jobs are processes or instructions that explain how to perform an action or automate a process.

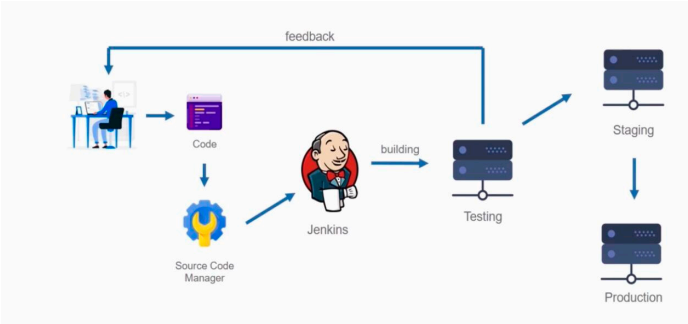

Figure 2. Jenkins pipeline (Li et al. [36]).

2.6. Create Cloud Using AWS

AWS is a cloud system that provide cloud for applications. We can run virtual machines on it. If the system has not dual booted or have no services of Ubuntu then we can use AWS to run Ubuntu [37] virtual machine. After creating login on AWS console, we have to go to services and use EC2 to launch virtual machine that run on AWS server but we can access it on our local system.

2.7. Continuous Monitoring

The monitoring is used according to the software requirement. For instance, you can monitor either matrix or log. For monitoring matrix, in this research we have used Prometheus and Grafana [38] for metrics as these are free and open-source tools. These tools combine to create a dashboard that help to identify software performance through different metrics. We have uesd ELK [39] stack for collecting logs in this research. The logs are basically collected from source and converted into visualization that help to understand the issues emerge in the using software.

However, there is a need robust and systematic way to of git version control system that could be for management and Docker operation and provide libraries. AWS services are leveraged to deploy an application as a cloud. Jenkins is used as a CI/CD pipeline to manage various phases of development and to make the development process continuous. The ELK stack is used to monitor and visualize the execution of code. In light of the findings, DevOps is an efficient method for cloud application deployment and resource

selection based on the relative importance of each optimized objective in terms of value parameters such as cost, memory, and CPU capacity, and that the method can be tailored to specific application requirements.

3. Materials and Methods

We perform experiments to analyze and illustrate the efficiency of the suggested algorithm compared to the standard points by formulating a multi-objective optimal solution for creating deployment initiatives using ANOVA approach to determine the relationship between two groups. The results are presented in form of ANOVA table which break down the components of variation in the data into variation between treatments and error or residual variation [40]. To address the optimal solution, the Decision Maker takes inputs from several other components. These parameters correspond to vendor services (instances), information hubs, the applications, and the client footprints. The Resource Trader receives datacenter id, instance type (specified by memory space, amount of CPUs, and storage), accessible number of instances, cost per hour, and evaluation inputs relating to vendor capabilities. Application node data (i.e., endpoint id, number of CPUs, memory capacity, disk usage) and node interdependence are also retrieved from the Application Repo. The Resource Trader also obtains information center-related parameters, such as data centre id, geo position (elevation, longitudinal), administrator, and data centre time stamp. The Information Analyzer provides information on the client footprint, such as the geo-positioning (elevation, longitudinal) of the POI (points of interest) as well as the expected volume of traffic in that area

The Algorithm 1 used for optimizing objectives is based on Genetic Algorithm (GA). GA is a type of genetic computation that replicates the natural evolution process. The solution becomes closer to a more ideal (desired) answer with each generation (i.e., iteration). This algorithm can also be used to solve a multi-objective optimization issue. The proposed algorithm which creates an application deployment plan defining the mapping of virtual machines to instances, is specified as follows:

Algorithm 1. Optimizing objectives

```
begin:
    function setApplication(objectives)
    set i = 1
    set itr = 0
    while(itr ≤ 500)
        for(i = 1; i ≤ maximum)
            function generatePlansRandom()
        end
    end #end while loop
    for (i = 1; 500)
        function determineFitness(value_cost, CPU_no, memory,
        user_node_distance, inter_node_distance)
            #determine fitness of each individual
        function tournamentSelection(value_cost, CPU_no, memory,
        user_node_distance, inter_node_distance)
            #select individual based on tournament selection

        function inheritenceMutationCrossover()
            #perform inheritance, mutation and crossover on selected individuals
    end #end for loop
    function bestDeploymentProcedure()
        #choose the fitness individual with best deployment procedure

    function generatePlan(best_Individual)
        #generate best deployment plan
end #end code
```

4. Results

Design Algorithm Solution Based on Individual

This research conduct on three different clouds with the help of ten samples (instance types). The hierarchical representation in Figure 3 describes the solution individuals (variables) and mapping the virtual machine to instances. Available instance with different specifications provided at various locations. The variation in specification for application which is being deployed on cloud using DevOps needed four nodes of virtual machine (VM).

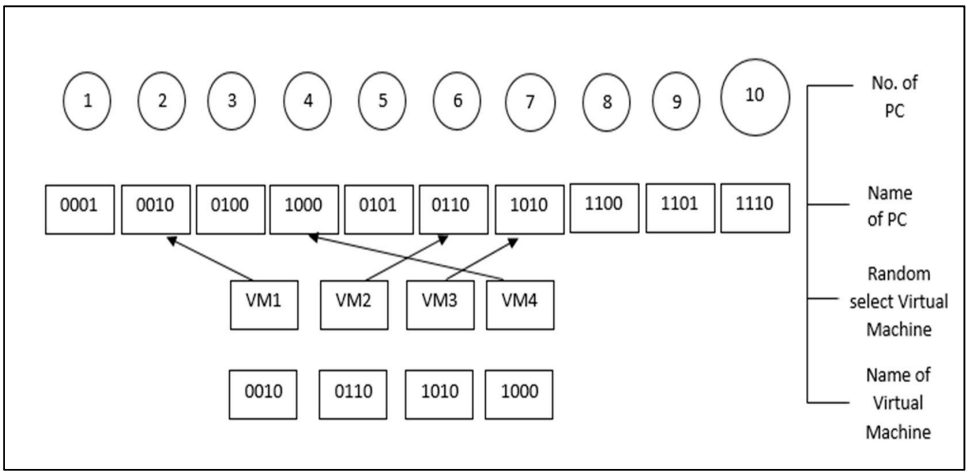

Figure 3. Random selections of V.M machine using algorithm.

We designed an automated system to overcome these challenges and demonstrated a significant reduction in time and effort for deployment and CI of the application using their proposed system. The total capacity provided to virtual machines on three different clouds AWS, Azure and Google is shown in Figure 4 and Table 1. The Table 1 shows that there is a significance between clouds regarding capacity, and variant regarding time. It shows that p-value = 2×10^{-16} of cloud shows highly significant results for total capacity while time is not significant

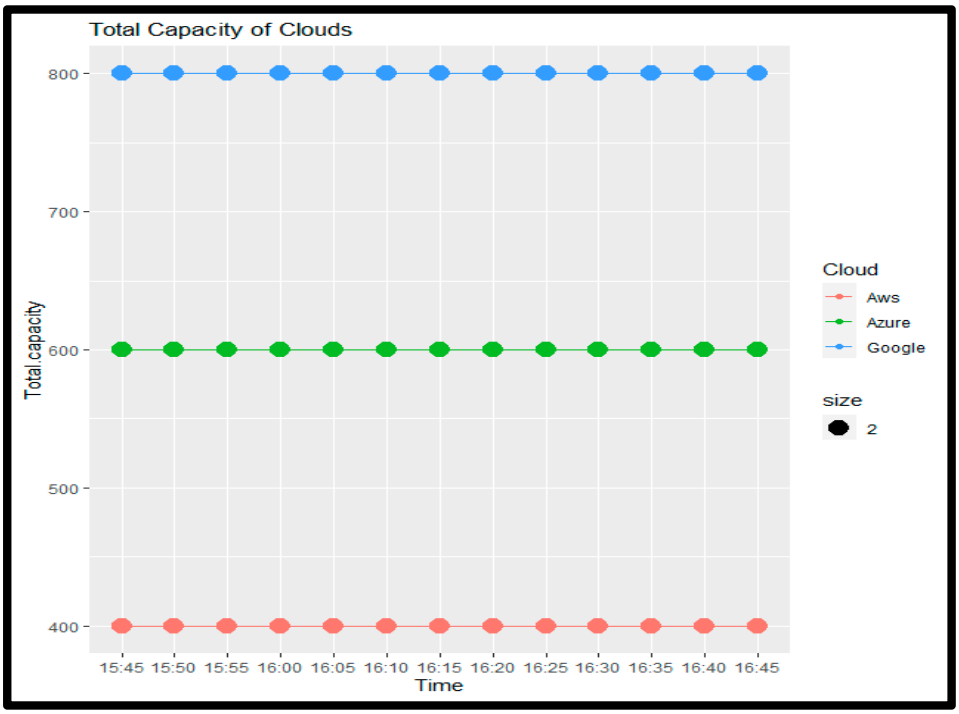

Figure 4. Total capacity and variant of clouds regarding time.

Table 1. ANOVA Table for Total Capacity.

ANOVA Table for Total Capacity						
	Df	**Sum Sq**	**Mean Sq**	**F Value**	**Pr (>F)**	
Cloud	2	1,040,000	520,000	$3.90 \times 10^{+30}$	$<2 \times 10^{-16}$	***
Time	12	0	0	$1.00 \times 10^{+00}$	0.478	
Residuals	24	0	0			

Residual standard error: 3.651×10^{-13} on 24 degrees of freedom. Multiple R-squared: 1, Adjusted R-squared: F-statistic: $5.572 \times 10^{+29}$ on 14 and 24 DF, p-value: $< 2.2 \times 10^{-16}$. Interpretation: p-value = 2×10^{-16} of cloud shows highly significant results for total. capacity while time is not significant.

The Figure 5 shows the used space of instance on AWS, Azure and Google clouds. The Table 2 describe the ANOVA table results for cloud and time. It interprets that p-value = 2.1×10^{-09} of cloud and 1.7×10^{-10} for time shows highly significant results for used space.

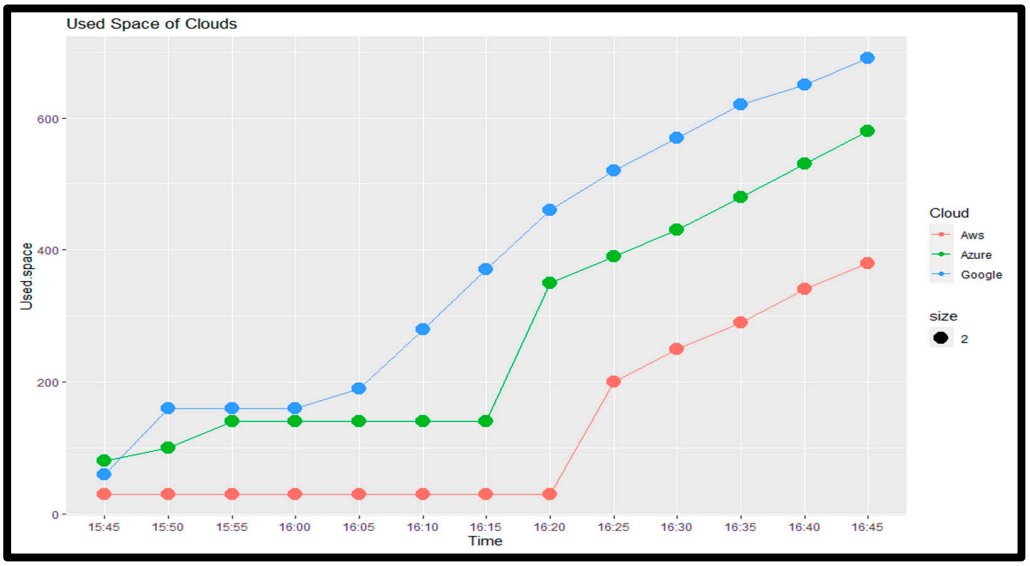

Figure 5. Used space of clouds.

Table 2. ANOVA Table for used space.

ANOVA Table for Used Space						
	Df	**Sum Sq**	**Mean Sq**	**F Value**	**Pr (>F)**	
Cloud	2	397,492	198,746	$5.13 \times 10^{+01}$	2.17×10^{-09}	***
Time	12	1,123,159	93,597	$2.41 \times 10^{+01}$	1.72×10^{-10}	***
Residuals	24	93,041	3877			

Residual standard error: 62.26 on 24 degrees of freedom. Multiple R-squared: 0.9423, Adjusted R-squared: 0.9087. F-statistic: 28.02 on 14 and 24 DF, p-value: 1.805×10^{-11}. Interpretation: p-value = 2.1×10^{-09} of cloud and 1.7×10^{-10} for time shows highly significant results for used space. Multiple R^2 = 94% show that prediction accuracy of this model is best fit for used space.

The Figure 6 and Table 3 shows the instance utilization on clouds at an appropriate time frame. It describes that p-value = 0.01 of cloud shows significant and 1.6×10^{-15} for time shows highly significant results for instance utilize data entry. Therefore, on the basis of results in Figure 6, it is concluded that AWS use less space for instance creation than other clouds.

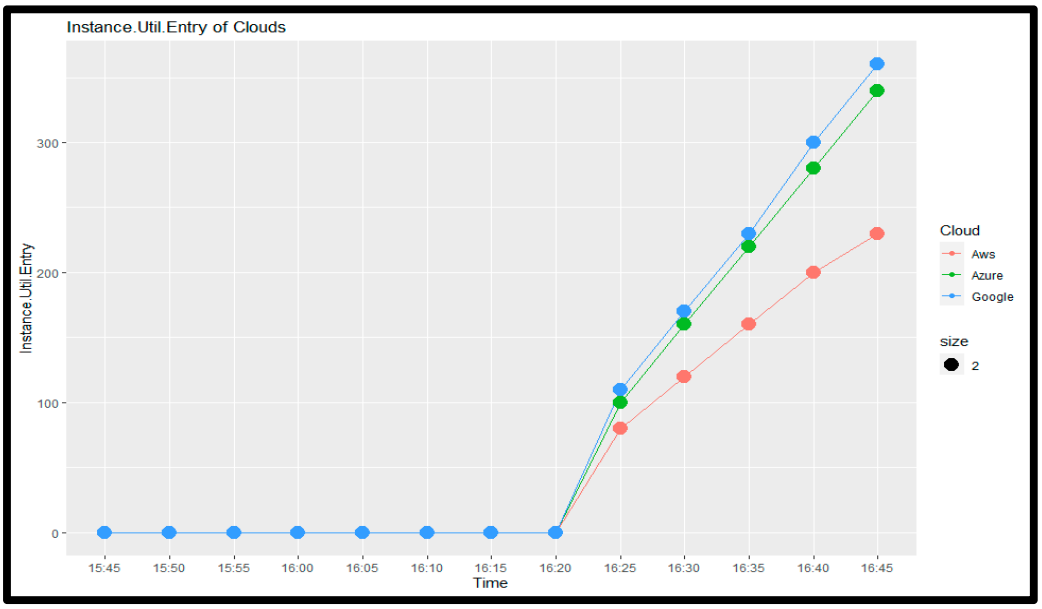

Figure 6. Instance. Utilized data Entry on clouds.

Table 3. ANOVA Table for instance utilized data on clouds.

ANOVA Table for Instance Utilized Data						
	Df	**Sum Sq**	**Mean Sq**	**F Value**	**Pr (>F)**	
Cloud	2	6292	3146	$5.46 \times 10^{+00}$	0.0111	*
Time	12	470,574	39,215	$6.80 \times 10^{+01}$	1.6×10^{-15}	***
Residuals	24	13,841	577			

Residual standard error: 24.01 on 24 degrees of freedom. Multiple R-squared: 0.9718, Adjusted R-squared: 0.9553. F-statistic: 59.06 on 14 and 24 DF, *p*-value: 4.019×10^{-15}. Interpretation: *p*-value = 0.01 of cloud shows significant and 1.6×10^{-15} for time shows highly significant results for Instance Util. data Entry. Multiple R^2 = 97% show that prediction accuracy of this model is best fit for instance utilized data entry.

In Figure 7 it is clearly seen that when instance create at 16:20 it utilizes resources on different clouds. Compared to other clouds, the AWS cloud uses the minimum resources. Similar behavior shown by clouds for instance Lang UUID (Universally unique identifier of instance created by Lang library of Java) as shown in Figure 7 and Table 4. It interprets that *p*-value = 0.00 of cloud and 2×10^{-16} for time shows highly significant results for instance Lang UUID. The result shows AWS is suitable clouds for application deployment.

Table 4. ANOVA Table for instance Lang. UUID.

ANOVA Table for Instance Lang. UUID						
	Df	**Sum Sq**	**Mean Sq**	**F Value**	**Pr (>F)**	
Cloud	2	565	283	$5.89 \times 10^{+00}$	0.00828	**
Time	12	154,506	12,876	$2.68 \times 10^{+02}$	$<2 \times 10^{-16}$	***
Residuals	24	1151	48			

Residual standard error: 6.926 on 24 degrees of freedom. Multiple R-squared: 0.9926, Adjusted R-squared: 0.9883. F-statistic: 230.9 on 14 and 24 DF, *p*-value: $< 2.2 \times 10^{-16}$. Interpretation: *p*-value = 0.00 of Cloud and 2×10^{-16} for Time shows highly significant results for Instance Lang. UUID. Multiple R^2 = 99% show that prediction accuracy of this model is best fit for Instance Lang. UUID.

The optimization parameters are cost, CPU processing, memory allocation and user node distance in multi-cloud application deployment.

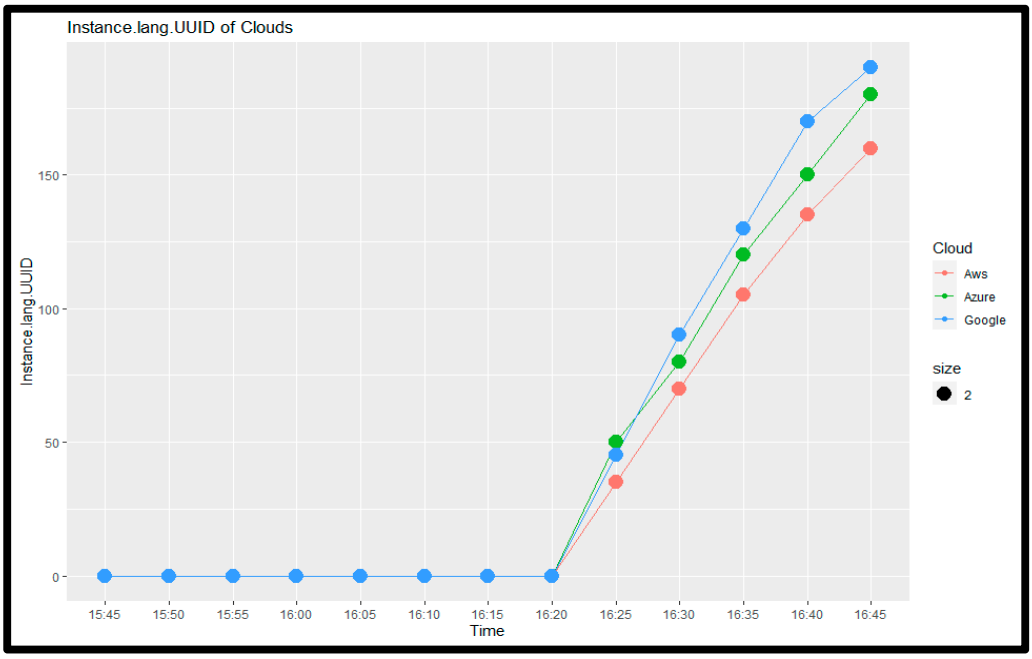

Figure 7. Instance Lang. UUID of clouds.

When cost has no weight, the proposed algorithmic solution has a cost value of 16.10, which is greater than the baseline solution. The cost has dropped to 13.347 in the second instance of applying a weight of 0.2 to cost (equal weight scenarios) that is already 11.3% cheaper than the baseline solution's 15.078%. Additionally, when the cost has the full weight of 1.0, the cost value drops to 5.92%, resulting in a 60% cost reduction over the baseline solution. In the same way, as the cost parameter is reducing, the user-node distance parameter is reduced by 9.26% in the equal weight condition and by 27.42% in the full weight situation when compared to the baseline solution. In the Table 5 the cost and clouds for different virtual machines are projected. Further in Table 6 the distance on clouds for different virtual machines are presented. Similarly, the Table 7 explains the Memory on clouds for different virtual machines.

Table 5. ANOVA Table for Cost on clouds for different virtual machines.

ANOVA Table for Cost						
	Df	**Sum Sq**	**Mean Sq**	**F Value**	**Pr (>F)**	
Cloud	2	1128	564	83.186	4.22×10^{-05}	***
Virtual Machine	3	168.3	56.1	8.274	0.0149	*
Residuals	6	40.7	6.8			

Residual standard error: 2.604 on 6 degrees of freedom. Multiple R-squared: 0.9696, Adjusted R-squared: 0.9442. F-statistic: 38.24 on 5 and 6 DF, *p*-value: 0.0001786. Interpretation: *p*-value = 4.22×10^{-13} of cloud and 0.01 for Virtual Machines (VM) shows highly significant results for cost. Multiple R^2 = 96% show that prediction accuracy of this model is best fit for cost.

Table 6. ANOVA Table for Distance on clouds for different virtual machines.

ANOVA Table for Distance						
	Df	**Sum Sq**	**Mean Sq**	**F Value**	**Pr (>F)**	
Cloud	2	779.4	389.7	$1.16 \times 10^{+02}$	4.57×10^{-13}	***
Time	12	145.4	12.1	$3.61 \times 10^{+00}$	0.0036	**
Residuals	24	80.6	3.4			

Residual standard error: 1.832 on 24 degrees of freedom. Multiple R-squared: 0.9199, Adjusted R-squared: 0.8731. F-statistic: 19.68 on 14 and 24 DF, *p*-value: 8.206×10^{-10}. Interpretation: *p*-value = 4.57×10^{-13} of cloud and 0.00 for Virtual Machines shows highly significant results for distance. Multiple R^2 = 92% show that prediction accuracy of this model is best fit for distance.

Table 7. ANOVA Table for Memory on clouds for different virtual machines.

ANOVA Table for Memory						
	Df	**Sum Sq**	**Mean Sq**	**F Value**	**Pr (>F)**	
Cloud	2	2243.9	1121.9	227.3	2.21×10^{-06}	***
Virtual Machine	3	290.3	96.8	19.6	0.00168	**
Residuals	6	29.6	4.9			

Residual standard error: 2.222 on 6 degrees of freedom. Multiple R-squared: 0.9884, Adjusted R-squared: 0.9788. F-statistic: 102.7 on 5 and 6 DF, *p*-value: 9.989×10^{-06}. Interpretation: *p*-value = 2.21×10^{-06} of cloud and 0.00 for Virtual. Machines shows highly significant results for memory. Multiple R^2 = 98% show that prediction accuracy of this model is best fit for memory.

The Table 8 represents the optimization parameters for a range of weight and their comparative result shown in Figures 8–10 respectively for cost, memory and user node distance parameters.

Table 8. Parameter values for various weights [41].

Optimization Parameters	**Weights**		
	Zero (0.0)	**Equal (0.2)**	**Full (1.0)**
Cost	16.10	13.34	5.92
CPU	8.43	10.75	11.99
Memory (GB)	30.20	39.20	47.84
User-node distance(million)	96.75	87.11	69.68
Inter-node distance	14.85	18.41	0.00

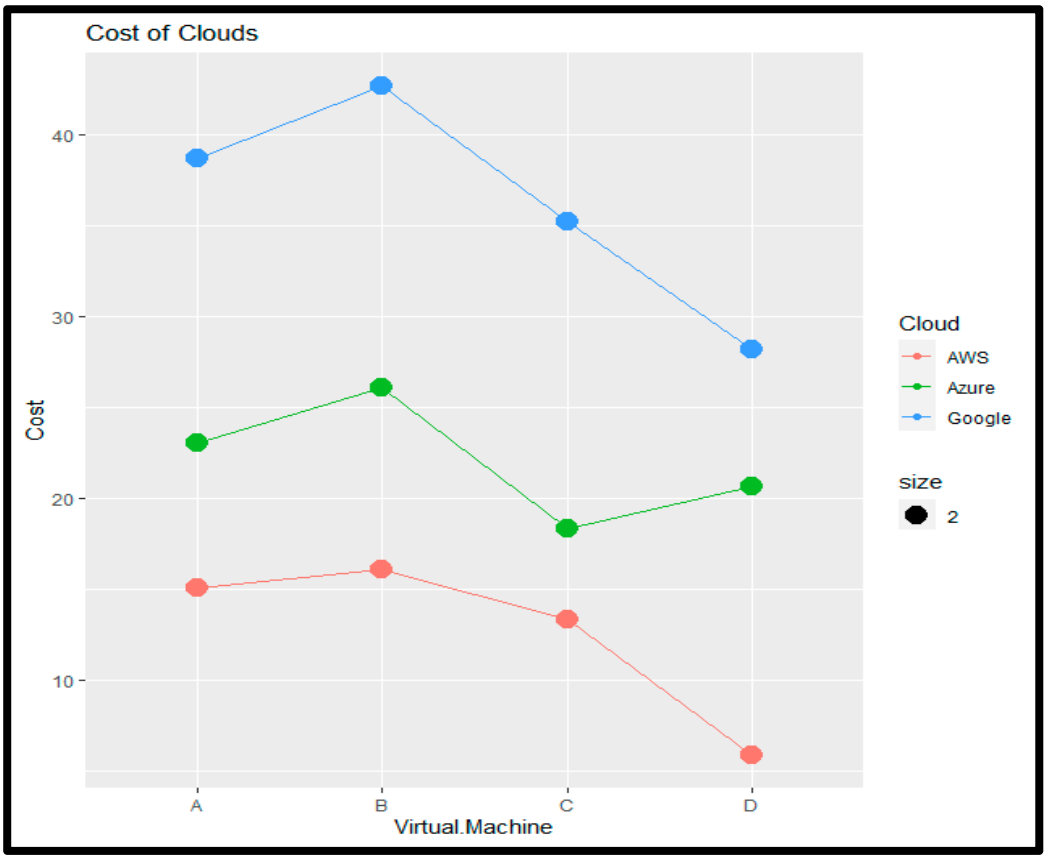

Figure 8. Cost ($/hr) for VMs on clouds.

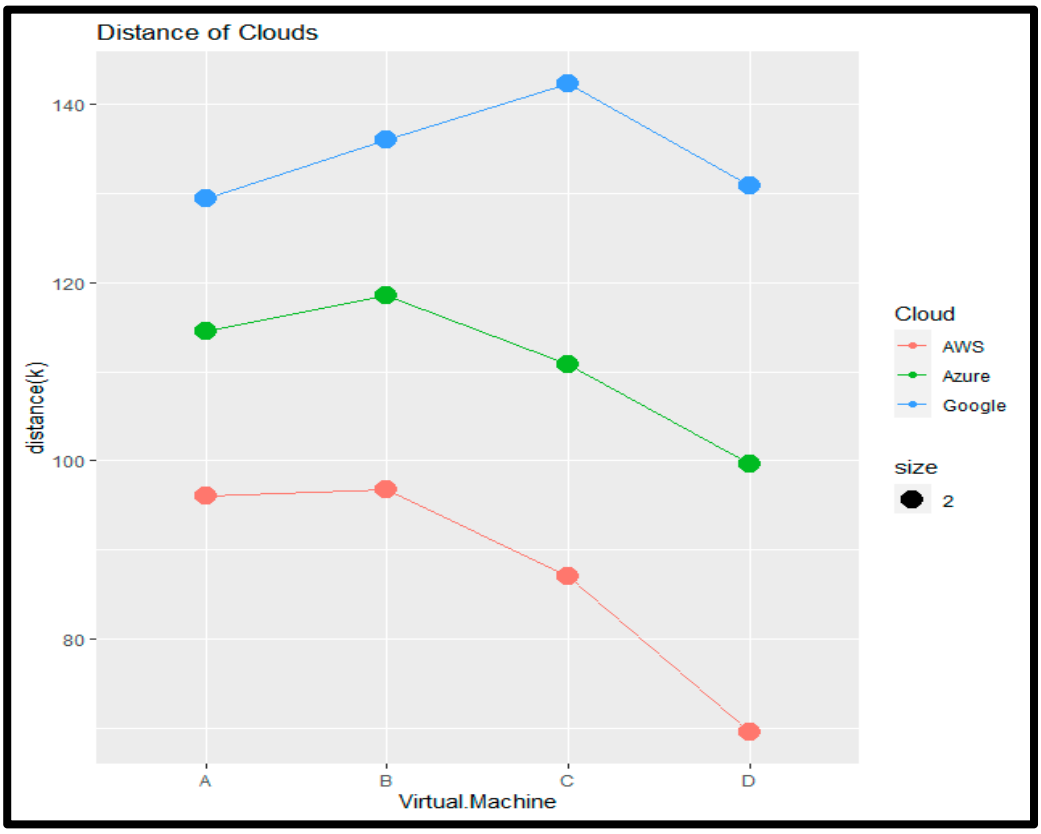

Figure 9. Distance (K) for different VMs on clouds.

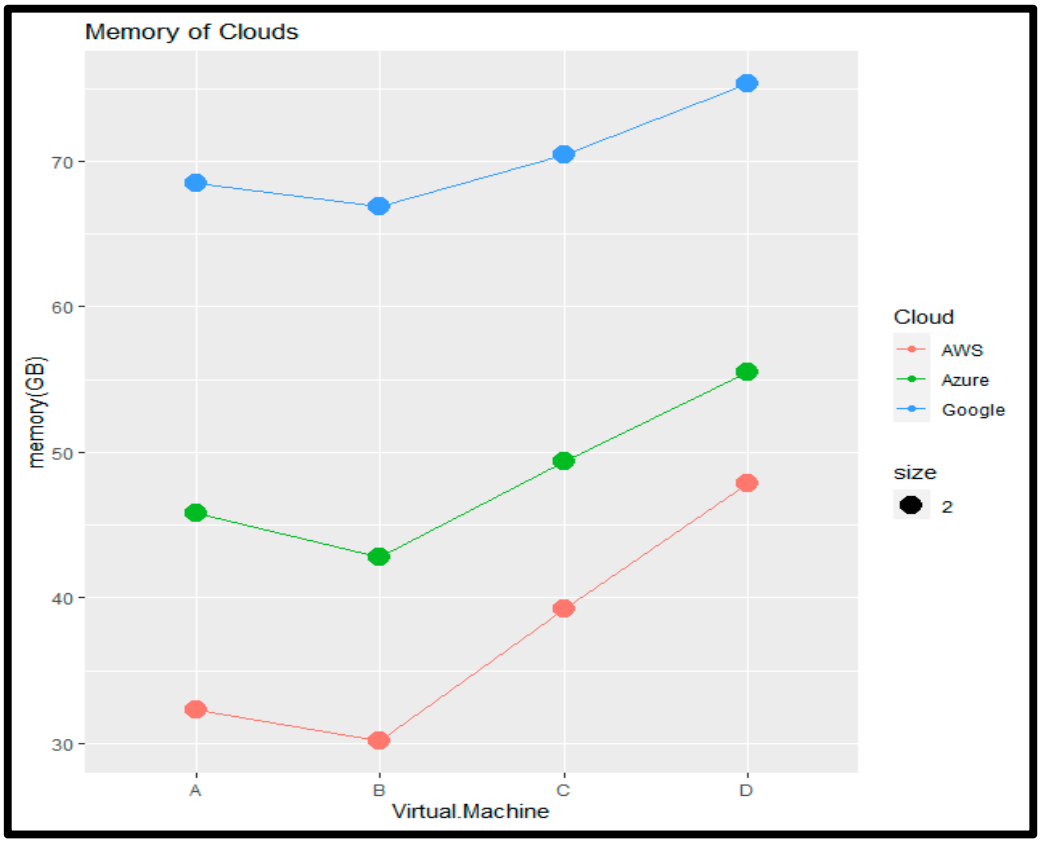

Figure 10. Memory (GB) for different VMs on clouds.

5. Summary, Conclusions and Future Work

The research established that DevOps is an efficient method for cloud application deployment and resource selection based on the relative importance of each optimised objective in terms of value parameters such as cost, memory, and CPU capacity, and that the method can be tailored to specific application requirements. This helps the product developing pace and more quality based software. The impact of DevOps implementation is so helpful for a project in digital transformation and compete with other available products. Shifting towards cloud makes the product fast to market and agile, this assists the DevOps to perform operations effectively, as it provides the efficient deployment environment.

The optimization parameters are cost, CPU processing, memory allocation and user node distance in multi-cloud application deployment. The Table 8 represents the optimization parameters for a range of weight and their comparative result shown in Figure 9 and 10 respectively for cost, memory and user node distance parameters. It is observed that it can by using appropriate weights, we can create various deployment plans with a cost ranging from 5.96 to 16.10, a CPU number between 8.43 and 11.99, a memory size among 30.2 GB and 47.84 GB, a user-node distance (calculated by multiplying by consumers at POIs) among 69,680,000 and 96,750,000, and lastly, an inter-node distance respectively 69,680,000 and 96,750,000. This indicates that by properly modifying the weights, it may achieve the desired trade-off between different parameters of the optimization problem

When cost has no weight, the proposed algorithmic solution has a cost value of 16.10, which is greater than the baseline solution. The cost has dropped to 13.347 in the second instance of applying a weight of 0.2 to cost (equal weight scenarios) that is already 11.3% cheaper than the baseline solution's 15.078%. Additionally, when the cost has the full weight of 1.0, the cost value drops to 5.92%, resulting in a 60% cost reduction over the baseline solution. In the same way, as the cost parameter is reducing, the user-node distance parameter is reduced by 9.26% in the equal weight condition and by 27.42% in the full weight situation when compared to the baseline solution

This paper is a baseline for organizations to develop a software and deploy it on cloud using DevOps techniques. The proposed algorithm will help companies in making decision for selecting best cloud for their software and application deployment using DevOps. They just use the algorithm and give parameter values and they will get the best suitable cloud for their application deployment.

In future, we will work on culture and environment effect on application deployment using DevOps and also will work on DevOps as a cloud provider for application implementation.

Author Contributions: Methodology, M.A.; Investigation, S.R.; Resources, M.A.A. and A.H.G.; Data curation, A.M.B.; Writing—original draft, A.K.; Writing—review & editing, M.T.R. All authors have read and agreed to the published version of the manuscript.

Funding: The authors would like to acknowledge the financial support from Prince Sultan University.

Institutional Review Board Statement: Novel Contribution.

Informed Consent Statement: Not Applicable.

Data Availability Statement: Data will be available on demand.

Acknowledgments: The authors would like to thank the participants who helped us in this research and shared their time and experience.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Leite, L.; Rocha, C.; Kon, F.; Milojicic, D.; Meirelles, P. A Survey of DevOps Concepts and Challenges. *ACM Comput. Surv.* **2019**, *52*, 1–35. [CrossRef]
2. Jambunathan, B.; Kalpana, Y. Design of devops solution for managing multi cloud distributed environment. *Int. J. Eng. Technol.* **2018**, *7*, 637–641. [CrossRef]

3. Rafi, S.; Yu, W.; Akbar, M.A. RMDevOps: A road map for improvement in DevOps activities in context of software organi-zations. In Proceedings of the Evaluation and Assessment in Software Engineering, Trondheim, Norway, 15–17 April 2020; pp. 413–418.
4. Zarour, M.; Alhammad, N.; Alenezi, M.; Alsarayrah, K. Devops Process Model Adoption in Saudi Arabia: An Empirical Study. *Jordanian J. Comput. Inf. Technol.* **2020**, *6*, 3. [CrossRef]
5. Akbar, M.A.; Rafi, S.; Alsanad, A.A.; Qadri, S.F.; Alsanad, A.; Alothaim, A. Toward Successful DevOps: A Decision-Making Framework. *IEEE Access* **2022**, *10*, 51343–51362. [CrossRef]
6. Ellen, L.; Riungu-Kalliosaari, L.; Mäkinen, S.; Lwakatare, L.E.; Tiihonen, J.; Männistö, T. *DevOps Adoption Benefits and Challenges in Practice: A Case Study*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 590–597. [CrossRef]
7. Cois, C.A.; Yankel, J.; Connell, A. Modern DevOps: Optimizing software development through effective system interactions. In Proceedings of the 2014 IEEE International Professional Communication conference (IPCC), Pittsburgh, PA, USA, 13–15 October 2014.
8. Soni, M. End to End Automation on Cloud with Build Pipeline: The Case for DevOps in Insurance Industry, Continuous In-tegration, Continuous Testing, and Continuous Delivery. In Proceedings of the 2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), Bangalore, India, 25–27 November 2015; pp. 85–89. [CrossRef]
9. Barna, C.; Khazaei, H.; Fokaefs, M.; Litoiu, M. Delivering elastic containerized cloud applications to enable DevOps. In Proceedings of the 2017 IEEE/ACM 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), Buenos Aires, Argentina, 22–23 May 2017; pp. 65–75. [CrossRef]
10. Guerriero, M.; Ciavotta, M.; Gibilisco, G.P.; Ardagna, D. SPACE4 Cloud: A DevOps Environment for Multi-cloud Applications. Short-Pap. In Proceedings of the 1st International Workshop on Quality-Aware DevOps, Bergamo, Italy, 28 May 2015; pp. 29–30.
11. Kang, H.; Yoonhee, J.K.; Rahm, J. A SLA Driven VM Auto-Scaling Method in Hybrid Cloud Environment. In Proceedings of the 2013 15th Asia-Pacific Network Operations and Management Symposium (APNOMS), Hiroshima, Japan, 25–27 September 2013; pp. 25–30.
12. Li, Y.; Xia, Y. Auto-scaling web applications in hybrid cloud based on docker. In Proceedings of the 2016 5th International Conference on Computer Science and Network Technology (ICCSNT), Changchun, China, 10–11 December 2016; pp. 75–79. [CrossRef]
13. Mor´n, D.; Vaquero, L.M.; Gal´n, F.; Moran, D.; Galán, F. Elastically Ruling the Cloud: Specifying Application's Behavior in Federated Clouds. In Proceedings of the IEEE 4th International Conference on Cloud Computing, Washington, DC, USA, 4–9 July 2011; pp. 89–96. [CrossRef]
14. Ghari, S. Devops for digital business: Optimizing the performance and economic efficiency of software products for digital business. In Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems, Pittsburgh, PA, USA, 18–23 May 2022; pp. 53–57.
15. Tsilionis, K.; Sassenus, S.; Wautelet, Y. Determining the Benefits and Drawbacks of Agile (Scrum) and DevOps in Addressing the Development Challenges of Cloud Applications. In Proceedings of the International Research & Innovation Forum, Athens, Greece, 15–17 April 2021; pp. 109–123.
16. Akbar, M.A.; Smolander, K.; Mahmood, S.; Alsanad, A. Toward successful DevSecOps in software development organizations: A decision-making framework. *Inf. Softw. Technol.* **2022**, *147*, 106894. [CrossRef]
17. Jaatun, M.G.; Cruzes, D.S.; Luna, J. DevOps for Better Software Security in the Cloud Invited Paper. In Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, 28 August–1 September 2017; p. 69. [CrossRef]
18. Almeida, F.; Simões, J.; Lopes, S. Exploring the Benefits of Combining DevOps and Agile. *Futur. Internet* **2022**, *14*, 63. [CrossRef]
19. Arulkumar, V.; Lathamanju, R. Start to Finish Automation Achieve on Cloud with Build Channel: By DevOps Method. *Procedia Comput. Sci.* **2019**, *165*, 399–405. [CrossRef]
20. Ferry, N.; Chauvel, F.; Song, H.; Rossini, A.; Lushpenko, M.; Solberg, A. CloudMF. *ACM Trans. Internet Technol.* **2018**, *18*, 1–24. [CrossRef]
21. Wettinger, J.; Breitenbücher, U.; Kopp, O.; Leymann, F. Streamlining DevOps automation for Cloud applications using TOSCA as standardized metamodel. *Futur. Gener. Comput. Syst.* **2016**, *56*, 317–332. [CrossRef]
22. Shin, Y.; Williams, L. Can traditional fault prediction models be used for vulnerability prediction? *Empir. Softw. Eng.* **2011**, *18*, 25–59. [CrossRef]
23. Blohowiak, A.; Basiri, A.; Hochstein, L.; Rosenthal, C. A Platform for Automating Chaos Experiments. In Proceedings of the IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Ottawa, ON, Canada, 23–27 October 2016; pp. 5–8. [CrossRef]
24. Akbar, M.A.; Mahmood, S.; Shafiq, M.; Alsanad, A.; Alsanad, A.A.A.; Gumaei, A. Identification and prioritization of DevOps success factors using fuzzy-AHP approach. *Soft Comput.* **2020**. [CrossRef]
25. Rafi, S.; Akbar, M.A.; Manzoor, A. DevOps Business Model: Work from Home Environment. In Proceedings of the International Conference on Evaluation and Assessment in Software Engineering, Gothenburg, Sweden, 13 June 2022; pp. 408–412.
26. Venkateswaran, S.; Santonu, S. Architectural partitioning and deployment modeling on hybrid clouds. *Softw. Pract. Exp.* **2018**, *48*, 345–365. [CrossRef]

27. Singh, V.; Peddoju, S.K. Container-based microservice architecture for cloud applications. In Proceedings of the 2017 International Conference on Computing, Communication and Automation (ICCCA), Greater Noida, India, 5–6 May 2017; pp. 847–852. [CrossRef]
28. Ghimire, R. Deploying Software in the Cloud with CI / CD Pipelines. 2020. Available online: https://www.theseus.fi/bitstream/handle/10024/345618/Thesis_Ramesh_Ghimire_1.pdf?sequence=2 (accessed on 27 October 2022).
29. AWS- Amazon Web Services, Amazon CloudWatch Developer Guide API Version 2010-08-01 Amazon CloudWatch: Developer Guide. 2010. Available online: https://s3.cn-north-1.amazonaws.com.cn/aws-dam-prod/china/pdf/acw-dg.pdf (accessed on 27 October 2022).
30. Knott, M. *Version Control with Git*; O'Reilly Media: Sebastopol, CA, USA, 2014.
31. Sullivan, B.O. Mercurial: The Definitive Guide Compiled from c3863298abc7. Available online: http://btn1x4.inf.uni-bayreuth.de/publications/dotor_buchmann/SCM/Mercurial/O%27Sullivan2009%20-%20Mercurial%20The%20defintive%20guide.pdf (accessed on 27 October 2022).
32. Jakkula, V. Tutorial on Support Vector Machine (SVM). 2011. Available online: http://www.ccs.neu.edu/course/cs5100f11/resources/jakkula.pdf (accessed on 27 October 2022).
33. Uphill, T.; Arundel, J.; Khare, N.; Saito, H.; Lee, H.C.C.; Hsu, K.J.C. *DevOps: Puppet, Docker, and Kubernetes*; Packt Publishing Ltd.: Birmingham, UK, 2017.
34. Jaramillo, D.; Nguyen, D.V.; Smart, R. Leveraging microservices architecture by using Docker technology. In Proceedings of the SoutheastCon 2016, Norfolk, VA, USA, 30 March–3 April 2016. [CrossRef]
35. Bowes, J. Jenkins Continuous Build System Executive summary. 2012. Available online: https://docplayer.net/5686123-Jenkins-continuous-build-system-jesse-bowes-csci-5828-spring-2012.html (accessed on 27 October 2022).
36. Li, Z.; Zhang, Y.; Liu, Y. Towards a Full-Stack DevOps Environment (Platform-as-a-Service) for Cloud-Hosted Applications. *Tsinghua Sci. Technol.* **2017**, *22*, 1–9. [CrossRef]
37. Shiwani, S. Performance Analysis of IPv4 v / s IPv6 in Virtual Environment Using UBUNTU. In Proceedings of the Interna-tional Conference on Computer Communication and Networks, Valencia, Spain, 9–13 May 2011; pp. 72–76.
38. Portnoy, J. Systems Monitoring with Prometheus and Grafana. Available online: https://flightaware.engineering/systems-monitoring-with-prometheus-grafana/ (accessed on 27 October 2022).
39. Beaver, D.; Hutchison, S. Elasticsearch, Logstash, and Kibana (ELK). 2015. Available online: https://resources.sei.cmu.edu/asset_files/presentation/2015_017_001_431205.pdf (accessed on 27 October 2022).
40. Padmanaban, S.; Khalili, M.; Nasab, M.A.; Zand, M.; Shamim, A.G.; Khan, B. Determination of Power Trans-formers Health Index Using Parameters Affecting the Transformer's Life. *IETE J. Res.* **2022**. [CrossRef]
41. Aryal, R.G.; Altmann, J. Dynamic application deployment in federations of clouds and edge resources using a multiobjective optimization AI algorithm. In Proceedings of the 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC), Barcelona, Spain, 23–26 April 2018; pp. 147–154. [CrossRef]