International Journal of New Media Studies (IJNMS), ISSN: 2394-4331
Volume 8 Issue 2, July-December, 2021, Impact Factor: 7.382

Cloud-Native Development: Review of Best Practices and Frameworks for Scalable and Resilient Web Applications

Srinivas Chippagiri[1*], Preethi Ravula[2]

[1] Member of Technical Staff, Salesforce Inc, Seattle USA
[2] PhD Researcher, University of Maryland College Park, MD USA

***Corresponding Author:** cvas22 [at] gmail [dot] com

ABSTRACT

Recent innovations in cloud computing have prompted a sea change away from monolithic systems to cloud-native architecture, enabling developers to build adaptable, scalable, and modular applications. Key technologies such as serverless computing, containerization, and microservices architecture are used in cloud-native development, enhancing operational efficiency, flexibility, and the deployment speed of applications in dynamic business environments. While these technologies allow organizations to build applications with improved adaptability, scalability and resilience, they also introduce challenges, including managing complex service dependencies, diagnosing failures, optimizing performance, and ensuring security in highly distributed systems. Moreover, managing stateful services and ensuring data consistency in the cloud environment can be particularly challenging. In this paper, the fundamental principles of cloud-native development are explored with focus on resilience, scalability, and elasticity while addressing the challenges faced by developers. Further, best practices like continuous integration and deployment (CI/CD), infrastructure as code (IaC), and security-focused DevSecOps practices are emphasized. Furthermore, a review of essential tools and frameworks, such as Kubernetes, Prometheus, and AWS CloudWatch, that assist in orchestrating, monitoring, and optimizing cloud-native systems is provided. The insights provided aim to guide organizations in adopting cloud-native technologies to build secure, high-performance applications.

Keywords—Cloud-native development, microservices architecture, continuous integration, continuous deployment (CI/CD)

INTRODUCTION

Technological advancements in cloud computing infrastructures has led to a sharp rise in cloud-native applications. Previously, applications are packed with services that operate in a container as microservices and are controlled on elastic infrastructure; cloud-native applications operate in a container-based environment. Cloud infrastructure administration has never been easy. Furthermore, orchestration has become a crucial component of these cloud-native apps as it enables a variety of functions, including scheduling, scalability, anomaly detection, resource management, and more[1].

The advent of business digital transformation has made the migration of enterprise apps to cloud platforms feasible. With the advancement of cloud computing technologies, MSA has emerged as a popular web application design. Complex software systems may be independently built and implemented using MSA, which breaks them up into single-function service components. The previous service-oriented architecture (SOA) and MSA are comparable. Although it does not highlight the heavy-duty service bus in the SOA architecture, it does further develop the servicing notion. However, since microservices include so many components, the intricate relationships between services and the rapid modifications of microservices-based system versions inherently raise the likelihood of failure and make issue detection more challenging[2].

Reduced hosting costs and more accessible and effective computer resources have led to a meteoric rise in the popularity of cloud-based apps. Performance and scalability testing and evaluation must be a part of the development lifecycle for any software system to be as scalable and effective as possible. This will ensure that cloud services meet SLA standards and provide the groundwork for future optimization[3].

The rise of enterprise digital transformation has further driven the migration of applications to cloud platforms. The adoption of microservice architecture (MSA), which divides complex systems into independently deployable single-function services, has become prevalent. However, the numerous components, complex interdependencies, and frequent updates in MSA environments increase the risk of failures and complicate problem diagnosis. Addressing these challenges requires adopting best practices and robust frameworks to build scalable and resilient cloud-native systems[4].

Motivation of the Study

The motivation for this study stems from the growing adoption of cloud-native application development as organizations increasingly migrate to cloud environments to enhance scalability, flexibility, and cost efficiency. Application development and deployment have been radically altered by the rapid development of contemporary technologies like

serverless computing, containerization, microservices architectures, and DevOps approaches. However, the inherent complexity of designing, managing, and optimizing cloud-native applications necessitates a comprehensive understanding of best practices, frameworks, and tools to achieve resilience, performance, and seamless scalability. This study aims to address these challenges by reviewing critical aspects of cloud-native development, enabling organizations to effectively leverage these advancements to gain a competitive advantage in a dynamic digital landscape.

Structure of the paper

The structure of this paper is as follows: An introduction to cloud-native development is given in section II. In section III, the best practices for building scalable cloud-native web applications are discussed; in section IV, the frameworks and tools for cloud-native web applications are discussed; section V follows in which relevant literature and case studies are presented; and conclusions and recommendations for future studies are presented in Section VI.

Understanding Cloud-Native Development

The process of creating and executing apps that fully use cloud computing's benefits is known as cloud-native application development. The practice of creating apps that are meant to be utilized in the cloud by the start is known as cloud native development. Cloud native applications are built using cloud technologies like container orchestrators, microservices etc[5].

These applications are typically built using modern cloud technologies like container orchestrators (e.g., Kubernetes), microservices architectures, and serverless computing, enabling seamless deployment and management across distributed environments. Cloud-native applications are characterized by their ability to automatically adapt to changing workloads, ensuring optimal performance and cost efficiency[6]. CI/CD pipelines, which promote a culture of cooperation and fast iteration, are also often used in the development process. The following programs were developed specifically for use in the cloud, as shown in Figure 1:

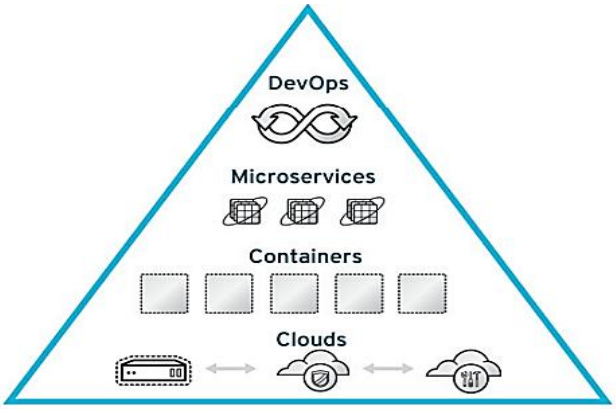

Fig. 1. A pyramid of modern cloud-native applications

The layers of cloud-native development. At the base is cloud infrastructure, enabling scalability and flexibility. Figure 1 mentions the containers for portability and resource efficiency, followed by microservices, which ensure modular, independent development[7]. At the top is DevOps, integrating development and operations for streamlined, continuous delivery. The characteristics of cloud-native are discussed below:

Microservices Architecture

The majority of cloud-native apps are constructed as a collection of loosely linked microservices. Each microservice handles a particular task and uses well-defined APIs to interact with other microservices [8]. Software applications are increasingly being developed and deployed using the Microservice Architecture (MSA) paradigm, which consists of a collection of small, granular services that may be connected via RESTful APIs or other lightweight communication methods. Microservices are easily comprehensible, compact components that provide the services' business capabilities. SOA is the inspiration for MSA, a cloud-native architectural framework. Generally, microservices are arranged as a collection of small, granular services that may be created, tested, and deployed using various technical stacks across many platforms [9].

Containerization

A program and all of its dependencies may be safely housed inside a lightweight and portable container. Developers may guarantee that their apps work consistently across development, production, and other environments by using containers [10].

As a lightweight and effective method of application packaging, deployment, and management, containerization has become an important technology in cloud-native application development. Containers guarantee that programs execute consistently across development, production, and other contexts by providing a consistent runtime environment [11].

Serverless Computing
The current trend in business application designs towards containers and microservices has led to the advent of serverless computing, sometimes called FaaS, as an attractive new paradigm for cloud application deployment [12]. Developers can concentrate on creating code with serverless computing since it removes the need to manage servers or scale resources. This method has several advantages, such as less operational overhead, better scalability, and cost efficiency[13]. The capacity to scale up or down is a major plus of serverless architecture.

DevOps
DevOps is a methodology that combines agile and lean principles with software development. This method encourages the development and operations teams to work together to continuously build high-quality software. The goal of the activities that make up DevOps—i.e., continuous planning, integration, deployment, testing, and monitoring—is to facilitate the rapid and reliable development, testing, and deployment of software updates via the promotion of close cooperation among developers, testers, and operators[8].

Cloud Native Application and requirements
Cloud-native applications are characterized by their basic properties: scalability and reliability. The scalability of a CNA depends on its flexibility to modify its capacity by adding or removing resources, as well as on its ability to make use of the cloud's on-demand self-service, quick elasticity, and measurable service. To be resilient, a CNA must be able to withstand the loss of virtualized resources, services, or commodity hardware. CNAs are designed with core and supporting functionality sections to accomplish both of these goals[14]. To enable individual scalability and governance in response to demand in each component of the program, the core is partitioned into fine-grained microservices. Included in the assistance are methods for monitoring and management..

Cloud containers are a suitable method for achieving CNA at the implementation level. Unlike virtual machines, containers can be started and stopped just like any other native system process[9]. Additionally, by using groups of containers across nodes, the management of containers may be synchronized with that of virtual machines, allowing for the use of current infrastructure management solutions.

Based on the considerations outlined, the primary requirements of CNA can be defined as follows:

1) **Resilience**

- The main objective is to make sure the application is resilient so it can work and be accessible on the cloud.
- Scalability takes operating cost reduction and load variance into account.
- Redundant resources are often used to improve cloud resilience.
- A strategic business choice must be made to strike a balance between cost reduction and redundancy.

2) **Elasticity**

- CNA should adjust capacity dynamically by adding or removing resources to meet QoS requirements during load variations.
- This ensures avoidance of over-provisioning and under-provisioning.
- Cloud-native applications must leverage cloud capabilities such as measured services, on-demand self-service, and rapid elasticity[15].

To build an application that is both functional and accessible in the cloud, resilience must be achieved first. Scalability, on the other hand, deals with variations in demand and the reduction of operating costs. The use of redundant resources is a common practice for cloud resilience [16]. It is a business choice to determine the trade-off between reducing operating costs and redundancy.

Cloud-native application characteristics
These following are the main characteristics of cloud-native applications:

Service-based Architectures
The building blocks of cloud-native apps are collections of independent (micro)services. Independent creation and operation of each service is made possible since each service in an application exists in its own right. Concurrently, services often communicate with one another and with other services inside an application; these services are found by taking use of capabilities offered by the application runtime[17][18]. This opens the door to building cloud-native apps and composing services.

API-based Interactions
API-based service-to-service connections are used in cloud-native applications. Each service in an application exposes its capabilities via an API, and each service in turn connects to and uses the APIs of the other services in the application.

The APIs used in cloud-native applications should all adhere to widely recognized standards, such as REST over HTTP, and each component should have its own API.8.

Infrastructure as Code
Every aspect of cloud-native apps, including deployment, administration, scaling, and monitoring, is highly automated. Infrastructure as code, or machine-readable files that enable the specification of the intended configuration for an application and its components, is usually used to accomplish such automation ..

Key Features of Cloud-Native Applications:
This section outlines the essential aspects of cloud-native applications, focusing on their key features and functionalities.

Performance:
The public cloud's built-in capabilities, which may outperform their non-native counterparts in terms of performance. You may, for instance, manage an I/O system that has autoscaling and load-balancing capabilities.

Efficiency
Cloud-native apps should make better use of underlying resources by using cloud-native capabilities and APIs.. That results in either reduced operational expenses or improved performance.

Cost
It usually costs less to operate applications that are more efficient. You may save money by making better use of the resources you have, as cloud providers charge you monthly according to your consumption.

Scalability: The native cloud APIs provide you immediate access to the platform's autoscaling and load-balancing capabilities when you develop your apps [19].

Best Practices For Building Scalable Cloud-Native Web Applications
Building scalable cloud-native web applications necessitates following best practices to ensure resilience, maintainability, and the ability to handle growing user demands effectively. These practices include designing for elasticity, leveraging microservices architecture, implementing robust monitoring and logging systems, optimizing performance through caching and load balancing, ensuring security at every layer, and adopting automation for CI/CD. Scalable web applications must efficiently manage increasing traffic while maintaining high performance, reliability, security, and cost-effectiveness.

BEST PRACTICES FOR BUILDING SCALABLE CLOUD-NATIVE WEB APPLICATIONS

Continuous Integration and Continuous Deployment (CI/CD)

Automatic software integration is a key component of continuous integration, which typically entails constructing and testing modified source code at regular intervals. The term "frequency" refers to the regularity with which software is developed and tested; for instance, with each version control commit.0.

Software must be constantly changeable and deployable, and Continuous Integration is a part of Continuous Delivery. The majority of the time, this need is met by implementing an automated staging environment[20].

Continuous Deployment ensures that software is immediately deployed to production as soon as it is committed to the version control system branches specific to production environments and passes the automated tests to be ready for production[21].

Deployment and Continuous Delivery are often used interchangeably and might be confused, but it's crucial to distinguish between the terms in academic settings. Figure 2 illustrates the connections between Continuous Integration, Deployment, and Delivery.

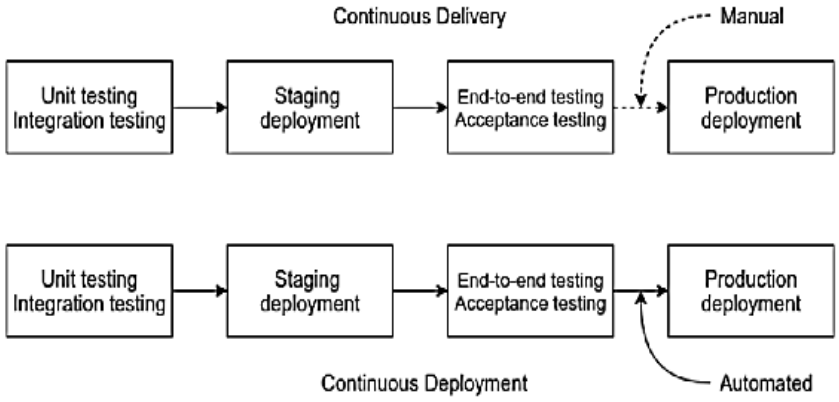

Fig. 2. Relations of Continuous Integration, Delivery and Deployment

Companies cannot hence implement Continuous Deployment without Continuous Integration and Delivery systems[22].

Infrastructure as code
Infrastructure as code makes it possible to distribute software components continuously and automatically throughout their whole lifespan, including installation, starting, stopping, and terminating. Repeatable end-to-end deployment automation may be created by specifying an application's infrastructure and components in deployable models that are reusable and maintained [23][24]. The following are some advantages of cloud-native development:

- **Scalability and Flexibility:** IaC guarantees that applications can manage a range of workloads by enabling developers to swiftly scale infrastructure up or down in response to demand [25].
- **Consistency and Reproducibility:** IaC lessens the likelihood of configuration drift by ensuring that infrastructure is reliably distributed across environments, including production, testing, and development [**26**].

Monitoring and Logging
An important part of cloud-native development is monitoring and logging, which allow developers to keep tabs on the availability, performance, and dependability of infrastructure and apps. Teams are able to recognize and resolve problems more rapidly with the use of monitoring and logging, which give insight into the system's health [27]. Several benefits in cloud-native development, including:

- **Use Monitoring Tools:** Use monitoring tools, such as Prometheus, Grafana, or AWS CloudWatch, to collect and visualize metrics related to application performance and infrastructure health.
- **Set up Alerts:** Define alerting rules based on your SLIs and SLOs to notify teams of any issues that require immediate attention[3].
- **Monitor Application Dependencies:** Monitor the performance of external dependencies, such as databases or third-party services, to identify and mitigate potential issues.

Orchestration
When there is a certain sequence that has to be followed, orchestration becomes necessary. In microservice orchestration, a service consumer or integration hub acts as a middleman to coordinate several services. Both production and testing environments make extensive use of Kubernetes [28].

Orchestration technologies used in cloud-native settings, like Kubernetes, also have security implications. Securing the Kubernetes control plane, establishing secure configuration processes, and safeguarding essential components like the data store are all part of this. Additionally, it entails safeguarding container orchestration and deployment procedures to avoid unauthorized access or modifications to deployments[29].

DevSecOps Practices
Embracing security principles across the development, deployment, and operations lifecycle is crucial for cloud-native service security. It highlights the importance of security automation, Continuous Security Testing, and security monitoring as components of the DevSecOps methodology, and how they should be integrated into the development process throughout [30].

Frameworks And Tools For Cloud-Native Web Applications

Some of the most important frameworks for developing cloud-native web applications are Kubernetes and Docker, which are used for container orchestration and containerization, respectively, and Spring Boot, which is used for constructing microservices. These frameworks streamline scalability, reliability, and efficient resource management in cloud environments.

Microservices Frameworks
A wide variety of frameworks are available for use with various computer languages. Four separate microservices frameworks were chosen for the toll system's implementation: Go Micro (Go), Molecular (JavaScript with Node.js), Spring Boot/Spring Cloud (Java), and others [31].

- Distributed system development using Go's built-in support for RPC and event-driven communication is made easier with Go Micro. This framework is offered as a collection of modular components that may be added or removed from a project according to the needs of the application. Application Development Frameworks Based on Microservices: Assessing and Contrasting[32].
- Molecular is a Node.js runtime–based microservice framework for JavaScript that is based on Chrome's V8 JavaScript engine. Making event-driven apps with reduced latency was the driving concept behind this framework's creation. Gateways, databases, serialization, and other functionalities are all available as pluggable modules from Molecular. The transporter, the service, and the service broker are the three primary parts of the framework[33].
- The Spring ecosystem includes the projects Spring Boot and Spring Cloud. In contrast to Spring Boot, which is an opinionated method for developing standalone Spring applications, Spring Cloud is a collection of tools for rapidly developing several common patterns in distributed systems, including configuration management, service discovery, and more. That being said, Spring Boot is more of a full-stack development environment than anything tailored for creating microservices [34].

Kubernetes and Container Orchestration Frameworks
Container orchestration frameworks like Kubernetes have become essential for deploying and managing cloud-native applications. Here's an overview of Kubernetes and related frameworks:

- **Kubernetes (K8s):** Containerized workloads and services may be easily managed using Kubernetes, an adaptable, scalable, and open-source platform that allows declarative setup and automation. Its rise is quick, and its environment is extensive. Kubernetes promotes

its services, support, and software to a large audience[35]. Application bundling and running are made easier using containers. If you're looking for a management solution that can make scaling your workload with containers easier, look no further. You may use the same Docker images with Kubernetes[36].

Docker Swarm Mode: Although Docker is mostly used for creating virtualized containers on individual computers, it also offers a platform for orchestrating containers called Docker Swarm Mode. This platform offers a set of tools for managing a cluster of containers. As the official clustering solution for Docker containers, it benefits from being deeply ingrained in the Docker ecosystem and making use of its own API. A swarm is an ensemble of Docker hosts operating in swarm mode[37].

LITERATURE REVIEW

A survey of the research on Cloud-Native Development with an emphasis on Highly Available and Scalable Web Apps is provided in this section. Table I provides a brief description of the studies that were examined.

In this study, Brunner et al. (2015) provide an example of a commercial application that is operating as a CAN and provide experimental proof of the design's benefits. Moreover, they introduce Dynamite, a containerized CNA application auto-scaler. There is little extra engineering required for CNA, according on our testing conducted on a Vagrant host, a private OpenStack installation, and a public Amazon EC2 testbed[14].

In this study, Chang and Fink (2017) showcase a tool that uses cloud execution log visualizations for a different purpose: to help with program comprehension and to create application documentation based on runtime data. A new timeline visualization, an improved way to summaries and display the results of many JSON objects, and interaction approaches to make moving between functions easier are all part of our solution. The composition, performance, data flow, and data structure of a serverless cloud application are all explained by these characteristics taken together. They provide some preliminary user comments from a number of knowledgeable developers who contributed to the tool's conception and creation[38].

In this study, Kratzke and Peinl (2016) looked at industrial cloud standards, public CSP, and cloud-native application design approaches. According to every conclusion, the majority of cloud service categories seem to encourage vendor lock-in scenarios, which might be particularly troublesome for business structures. At first, this can seem discouraging. On the other hand, provided a reference model for cloud-native apps that only use a limited number of highly standardized IaaS services. Cloud technology may be codified using the reference model. It may direct research and development, adoption, categorization, and technology identification procedures for cloud-native applications and enterprise architectural engineering approaches that are mindful of vendor lock-in[9].

In this research, Astyrakakis et al. (2019) introduces a novel, fully automated tool for OpenStack Kubernetes cluster deployment and monitoring. They also provided a solution that can automatically validate cloud-native apps. Comparing the assessment of the suggested toolbox to alternative manual methods, Kubernetes clusters were deployed with very short overall timeframes. An application that was containerized and had the Kubernetes HPA enabled took around 11 minutes to validate, whereas an application that was containerized and had the HPA deactivated took about 3 minutes[39].

In this study, Imadali and Bousselmi (2018) provide a software platform that is native to the cloud, enabling MNOs to make their networking resources, mobile services, and cloud computing available to 5GaaS over-the-top companies.

They also detail the prototype's standard-based viability and provide our open-source Cloud Native VNF API design, which is an implementation of the suggested design principles[40]

Table : 1 Presents the Comparative Table based on Cloud-Native Development and Frameworks for Scalable and Resilient Web Applications

Reference	Study On	Methodology	Key Findings	Challenges	Limitations
[14]	Case study of a business application running as a Cloud-Native Application (CNA)	Experimental evaluation of CNA advantages using Dynamite auto-scaler on Vagrant, OpenStack, and Amazon EC2 testbeds.	Demonstrated CNA requires minimal additional engineering, and Dynamite enhances auto-scaling for containerised applications.	Resource management in varied cloud environments.	Limited scope to specific experimental testbeds (e.g., Vagrant, OpenStack, Amazon EC2).
[38]	Visualisation of cloud execution logs for program understanding	Developed a tool with timeline visualisation, JSON	Improved understanding of serverless cloud applications'	Ensuring usability across diverse cloud applications.	Initial feedback limited to expert developers; broader validation

	and documentation generation.	summarisation, and interaction techniques for runtime data analysis.	composition, performance, and data flow with positive feedback from expert developers.		not covered.
[9]	Design principles, public cloud services, and industrial cloud standards for cloud-native apps.	Developed a reference model relying on standardised IaaS services to address vendor lock-in concerns.	Highlighted vendor lock-in risks in public cloud services and proposed a model for codifying cloud technologies and guiding enterprise architectures.	Overcoming vendor lock-in in real-world implementations.	Focuses on IaaS and lacks detailed discussion of PaaS and SaaS contexts.
[39]	Automated deployment and monitoring of Kubernetes clusters over OpenStack.	Developed and evaluated an automated tool for Kubernetes deployment and validation with HPA-enabled/disabled setups.	Achieved faster Kubernetes cluster deployment times and efficient validation processes with notable time reductions compared to manual approaches.	Adapting the tool for larger, more complex deployments.	Limited evaluation scope to Kubernetes clusters and specific configurations.
[40]	Cloud-native software platform enabling MNOs to expose assets for 5GaaS.	Developed an open-source Cloud Native VNF API design and evaluated feasibility from a standards perspective.	Provided a viable prototype for 5GaaS with a design enabling integration of networking resources, mobile services, and cloud computing.	Balancing standardisation and innovation requirements.	Focused on 5GaaS-specific use cases; broader applicability to other industries not demonstrated.

CONCLUSION AND FUTURE WORK

Advancements in cloud-native application development enables the design, development, and implementation of cutting-edge applications that fully leverage cloud computing capabilities.

Scalability, reliability, and cost-effectiveness are attained by these systems by employing technologies such as, microservices, containerization, and serverless computing integrated with DevOps procedures and automation. Implementing best practices, including CI/CD, infrastructure as code, and robust monitoring, ensures streamlined operations and rapid innovation. Tools such as Kubernetes, Docker, and Spring Boot, developers help addressing the complexities of distributed environments while delivering highly performant and adaptive solutions.

Future research in cloud-native development can focus on enhancing scalability and resilience through advanced AI-driven orchestration and predictive resource allocation. Integrating cutting-edge technologies, such as edge computing and 5G, can further optimize cloud-native applications for low-latency use cases.

Moreover, investigation of zero-trust security models and the application of blockchain for secure microservices communication in improving the security posture of cloud-native environments can be explored in future studies.

REFERENCES

[1]. R. Chowdhury, C. Talhi, H. Ould-Slimane, and A. Mourad, “A Framework for Automated Monitoring and Orchestration of Cloud-Native applications,” in 2020 International Symposium on Networks, Computers and Communications (ISNCC), IEEE, Oct. 2020, pp. 1–6. doi: 10.1109/ISNCC49221.2020.9297238.

[2]. J. Qiu, Q. Du, K. Yin, S. Zhang, and C. Qian, “applied sciences A Causality Mining and Knowledge Graph Based Method of Root Cause Diagnosis for Performance Anomaly in Cloud Applications,” 2020, doi: 10.3390/app10062166.

[3]. T. Atmaca, T. Begin, A. Brandwajn, and H. Castel-Taleb, “Performance Evaluation of Cloud Computing Centers with General Arrivals and Service,” IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 8, pp. 2341–2348, 2016, doi: 10.1109/TPDS.2015.2499749.

[4]. P. Cappaert and A. Redei, “A scalable cloud native platform for interactive museum exhibits,” in EPiC Series in Computing, 2020. doi: 10.29007/6b3j.

[5]. A. S. Ramakrishna Garine, Rajeev Arora, Anoop Kumar, “Advanced Machine Learning for Analyzing and Mitigating Global Supply Chain Disruptions during COVID-19,”

SSRN, pp. 1–6, 2020.

[6]. A. Kumar, R. Vij, M. Gupta, S. Sharma, and S. Singh, “Risk assessment of exposure to radon concentration and heavy metal analysis in drinking water samples in some areas of Jammu & Kashmir, India,” J. Radioanal. Nucl. Chem., vol. 304, no. 3, pp. 1009–1016, Jun. 2015, doi: 10.1007/s10967-015-3967-y.

[7]. A. Kumar, R. Garine, A. Soni, R. K. Arora, R. C. Dublin, and I. Researcher, “Leveraging AI for E-Commerce Personalization : Insights and Challenges from 2020,” pp. 1–6, 2020.

[8]. M. Waseem, P. Liang, and M. Shahin, “A Systematic Mapping Study on Microservices Architecture in DevOps,” J. Syst. Softw., vol. 170, 2020, doi: 10.1016/j.jss.2020.110798.

[9]. N. Kratzke and R. Peinl, “ClouNS-a Cloud-Native Application Reference Model for Enterprise Architects,” in Proceedings - IEEE International Enterprise Distributed Object Computing Workshop, EDOCW, 2016. doi: 10.1109/EDOCW.2016.7584353.

[10]. C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, “Cloud container technologies: A state-of-the-art review,” IEEE Trans. Cloud Comput., 2019, doi: 10.1109/TCC.2017.2702586.

[11]. C. Pahl, “Containerization and the PaaS Cloud,” IEEE Cloud Comput., 2015, doi: 10.1109/MCC.2015.51.

[12]. R. A. P. Rajan, “A review on serverless architectures-Function as a service (FaaS) in cloud computing,” Telkomnika (Telecommunication Comput. Electron. Control., vol. 18, no. 1, pp. 530–537, 2020, doi: 10.12928/TELKOMNIKA.v18i1.12169.

[13]. I. Baldini et al., “Serverless computing: Current trends and open problems,” in Research Advances in Cloud Computing, 2017. doi: 10.1007/978-981-10-5026-8_1.

[14]. S. Brunner, M. Blochlinger, G. Toffetti, J. Spillner, and T. M. Bohnert, “Experimental Evaluation of the Cloud-Native Application Design,” Proc. - 2015 IEEE/ACM 8th Int. Conf. Util. Cloud Comput. UCC 2015, pp. 488–493, 2015, doi: 10.1109/UCC.2015.87.

[15]. R. Goyal, “THE ROLE OF BUSINESS ANALYSTS IN INFORMATION MANAGEMENT PROJECTS,” Int. J. Core Eng. Manag., vol. 6, no. 9, pp. 76–86, 2020.

[16]. G. Toffetti, S. Brunner, M. Blöchlinger, J. Spillner, and T. M. Bohnert, “Self-managing cloud-native applications: Design, implementation, and experience,” Futur. Gener. Comput. Syst., 2017, doi: 10.1016/j.future.2016.09.002.

[17]. Jana. I, Oprea. A, “AppMine: Behavioral Analytics for Web Application Vulnerability Detection,” arxiv, 2019, doi: 10.48550/arXiv.1908.01928.

[18]. M. Wurster, U. Breitenbücher, A. Brogi, F. Leymann, and J. Soldani, “Cloud-native Deploy-ability : An Analysis of Required Features of Deployment Technologies to Deploy Arbitrary Cloud-native Applications,” no. Closer, pp. 171–180, 2020, doi: 10.5220/0009571001710180.

[19]. D. S. Linthicum, “Cloud-Native Applications and Cloud Migration: The Good, the Bad, and the Points between,” IEEE Cloud Comput., 2017, doi: 10.1109/MCC.2017.4250932

[20]. M. Z. Hasan, R. Fink, M. R. Suyambu, and M. K. Baskaran, “Assessment and improvement of elevator controllers for energy efficiency,” in Digest of Technical Papers - IEEE International Conference on Consumer Electronics, 2012. doi: 10.1109/ISCE.2012.6241747.

[21]. M. Z. Hasan, R. Fink, M. R. Suyambu, and M. K. Baskaran, “Assessment and improvement of intelligent controllers for elevator energy efficiency,” in IEEE International Conference on Electro Information Technology, 2012. doi: 10.1109/EIT.2012.6220727.

[22]. A. Hakli, D. Taibi, and K. Systa, “Towards cloud native continuous delivery: An industrial experience report,” Proc. - 11th IEEE/ACM Int. Conf. Util. Cloud Comput. Companion, UCC Companion 2018, pp. 314–320, 2018, doi: 10.1109/UCC-Companion.2018.00074.

[23]. M. Rodriguez-sanchez, “Cloud native Application Development - Best Practices : Studying best practices for developing cloud native applications , including containerization , microservices , and serverless computing,” vol. 1, pp. 18–27.

[24]. M. Z. Hasan, R. Fink, M. R. Suyambu, M. K. Baskaran, D. James, and J. Gamboa, “Performance evaluation of energy efficient intelligent elevator controllers,” in IEEE International Conference on Electro Information Technology, 2015. doi: 10.1109/EIT.2015.7293320.

[25]. M. Wurster et al., “The essential deployment metamodel: a systematic review of deployment automation technologies,” in Software-Intensive Cyber-Physical Systems, 2020. doi: 10.1007/s00450-019-00412-x.

[26]. V. V. Kumar, F. W. Liou, S. N. Balakrishnan, and V. Kumar, “Economical impact of RFID implementation in remanufacturing: a Chaos-based Interactive Artificial Bee Colony approach,” J. Intell. Manuf., 2015, doi: 10.1007/s10845-013-0836-9.

[27]. V. Kumar and F. T. S. Chan, “A superiority search and optimisation algorithm to solve RFID and an environmental factor embedded closed loop logistics model,” Int. J. Prod. Res., vol. 49, no. 16, 2011, doi: 10.1080/00207543.2010.503201.

[28]. V. V. Kumar, M. K. Pandey, M. K. Tiwari,

and D. Ben-Arieh, "Simultaneous optimization of parts and operations sequences in SSMS: A chaos embedded Taguchi particle swarm optimization approach," J. Intell. Manuf., 2010, doi: 10.1007/s10845-008-0175-4.

[29]. V. V Kumar, "An interactive product development model in remanufacturing environment : a chaos-based artificial bee colony approach," Missouri University of Science and Technology, 2014. [Online]. Available: https://scholarsmine.mst.edu/cgi/viewcontent.cgi?article=8243&context=masters_theses

[30]. V. V Kumar, M. Tripathi, M. K. Pandey, and M. K. Tiwari, "Physical programming and conjoint analysis-based redundancy allocation in multistate systems: A Taguchi embedded algorithm selection and control (TAS&C) approach," Proc. Inst. Mech. Eng. Part O J. Risk Reliab., vol. 223, no. 3, pp. 215–232, Sep. 2009, doi: 10.1243/1748006XJRR210.

[31]. V. V. Kumar, S. R. Yadav, F. W. Liou, and S. N. Balakrishnan, "A digital interface for the part designers and the fixture designers for a reconfigurable assembly system," Math. Probl. Eng., 2013, doi: 10.1155/2013/943702.

[32]. H. DInh-Tuan, M. Mora-Martinez, F. Beierle, and S. R. Garzon, "Development Frameworks for Microservice-based Applications: Evaluation and Comparison," in ACM International Conference Proceeding Series, 2020. doi: 10.1145/3393822.3432339.

[33]. V. V. Kumar, F. T. S. Chan, N. Mishra, and V. Kumar, "Environmental integrated closed loop logistics model: An artificial bee colony approach," in SCMIS 2010 - Proceedings of 2010 8th International Conference on Supply Chain Management and Information Systems: Logistics Systems and Engineering, 2010.

[34]. V. V. Kumar, A. Sahoo, and F. W. Liou, "Cyber-enabled product lifecycle management: A multi-agent framework," in Procedia Manufacturing, 2019. doi: 10.1016/j.promfg.2020.01.247.

[35]. Eshwari H M, Rekha B S, and G. N. Srinivasan, "Hybrid Cloud Technologies: Dockers, Containers and Kubernetes," Int. Res. J. Eng. Technol. , vol. 7, no. 6, pp. 7628–7634, 2020.

[36]. V. Kumar, V. V. Kumar, N. Mishra, F. T. S. Chan, and B. Gnanasekar, "Warranty failure analysis in service supply Chain a multi-agent framework," in SCMIS 2010 - Proceedings of 2010 8th International Conference on Supply Chain Management and Information Systems: Logistics Systems and Engineering, 2010.

[37]. V. Kumar, M. Tripathi, S. Tyagi, and M. K. Tiwari, "An integrated real time optimization approach (IRTO) for physical programming based redundancy allocation problem," Proc. 3rd Int. Conf. Reliab. Saf. …, 2007.

[38]. K. S. P. Chang and S. J. Fink, "Visualizing serverless cloud application logs for program understanding," in Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC, 2017. doi: 10.1109/VLHCC.2017.8103476.

[39]. N. Astyrakakis, Y. Nikoloudakis, I. Kefaloukos, C. Skianis, E. Pallis, and E. K. Markakis, "Cloud-Native Application Validation & Stress Testing through a Framework for Auto-Cluster Deployment," in 2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2019, pp. 1–5. doi: 10.1109/CAMAD.2019.8858164.

[40]. S. Imadali and A. Bousselmi, "Cloud native 5g virtual network functions: Design principles and use cases," in Proceedings - 8th IEEE International Symposium on Cloud and Services Computing, SC2 2018, 2018. doi: 10.1109/SC2.2018.00019.