ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14959407

Modern Full Stack Development Practices for Scalable and Maintainable Cloud-Native Applications

Mohit Menghnani[1]

[1]Independent Researcher

Publication Date: 2025/03/04

Abstract: The widespread acceptance of the cloud-native concept and the emergence of several specialized cloud-native apps have focused industry attention on the web stacks of cloud-native apps. The integration of cloud-native and full-stack development tools allows for the rapid and smooth deployment, scaling, and maintenance of web applications. Full stack development today in the cloud native era is a hybrid of its technologies and ways of working to bring about scalability, efficiency and maintainability. Built in tools like AWS Amplify, Google Firebase, Heroku can be used for a smooth deployment; serverless computing (AWS Lambda, Google Cloud Functions) also cuts out all the overhead from infrastructure management. Moving applications to Dockerized containers that are supervised in Kubernetes leads to portability and consistency of applications across environments. The whole stack presents include different front-end frameworks (React, Angular), back-end technologies (Node.js, Django), and cloud-based databases (MongoDB, AWS DynamoDB) for making robust web applications. Furthermore, DevOps practices, CI/CD pipelines, and Infrastructure as Code (IaC) help with deployment, monitoring and scaling which makes the operation more efficient. Because cloud-native architecture is modularity and resilience-oriented, it provides some microservices and API-driven interactions. Despite the problems of emerge, such as security vulnerabilities and performance bottlenecks, best practices like containerization, serverless computing, and also the database optimization make possible reliable, scalable and secure cloud applications. In this paper, a modern full stack development approach is explored, including key technologies, challenges and solutions to increase the performance and maintenance in cloud-based environment.

Keywords*: Full-Stack Development, Cloud-Native Applications, Front-End Development, Back-End Development. Databases, Web Application.*

How to Cite: Mohit Menghnani. (2025). Modern Full Stack Development Practices for Scalable and Maintainable Cloud-Native Applications. *International Journal of Innovative Science and Research Technology*, 10(2), 1206-1216. https://doi.org/10.5281/zenodo.14959407.

I. INTRODUCTION

Cloud native applications are on the scene because of the sharp rise of the technological advancement in cloud computing infrastructures. 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[1][2]. Administration of the cloud infrastructure has never been easy. Also, orchestration is becoming an essential component of these cloud native apps to perform a lot of things like scheduling, scaling, anomaly detection, resource management, etc. [3]. Cloud platform migration of enterprise applications became possible due to business digital transformation. The process of creating and executing apps that fully use cloud computing's benefits is known as cloud native software development[4]. Developing web applications in software development exceeds basic source code creation to demand a structured environment that promotes collaborative work automation capabilities and elite scalability[5]. Modern stack developers design complete applications which require them to assemble the entire web application infrastructure from front end to back end systems. In order to guarantee that the program runs smoothly, developers must integrate many technologies[6]. A development environment should provide three components for efficient collaboration: a shared version control system and project management through a ticket system with automated testing and deployment frameworks[7].

The rapid expansion of web applications has transformed the IT sector through three key technology advantages, which include independent platform capabilities and hardware delivery with a standardized digital data processing system through cloud-based solutions[8]. Big data analyses through these platforms join big data analytics to improve the user experience and decision making in the

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14959407

large scale information sharing. Technology stacks are the foundation of modern web applications and are based on the principle that replacing any one of the components is feasible and provides flexibility by replacing the individual components on a given application with new frameworks, APIs and database solutions to ensure application efficiency[9][10]. However, technological advancement and the growing need for response, scalable apps have made full-stack development more complex [11]. In response, Agile development approaches like Scrum, Kanban and Extreme Programming (XP) have come into practice. These approaches are flexible and foster development, adaptation, and cooperation, which help teams to quickly adjust to changing needs and technical breakthroughs. Agile strategies play a crucial role in managing the complexities of full-stack development, ensuring efficient software delivery while maintaining scalability and maintainability[12].

This paper explores full-stack development in the cloud-native era, focusing on technologies that enhance scalability, efficiency, and maintainability. It covers cloud platforms (AWS Amplify, Google Firebase), serverless computing (AWS Lambda, Google Cloud Functions), and containerization (Docker, Kubernetes) for seamless deployment. Key components like front-end (React, Angular), back-end (Node.js, Django), and cloud databases (MongoDB, AWS DynamoDB) are examined alongside DevOps practices, CI/CD pipelines, and IaC for automation. A challenge described in the study is security and performance bottlenecks and solutions of the form best practices such as microservices and API-driven development. These approaches make clouds more flexible, more scalable, and also cheaper so that you can have better, robust and scalable cloud native applications.

➢ *Organization of the Study*

This paper is structured to provide a comprehensive understanding of modern full-stack development in cloud-native environments. Section II offers an overview of software full-stack development. Section III explores the technologies used in full-stack development. Section IV delves into the fundamentals of cloud-native development. Section V presents a literature review, analyzing existing research and advancements in full-stack development. Section VI ends with several important takeaways and suggestions for further investigation.

II. SOFTWARE FULL-STACK DEVELOPMENT: AN OVERVIEW

The term "web development," which includes "full stack development," is used to describe the process of creating websites that may be hosted on either an intranet or the internet. It entails creating an app from the ground up, including the front end (often called a client-side) and the back end (sometimes called a server-side). The advent of cloud computing has brought about tremendous shifts in the Full Stack Development industry and its impact on software development[13][14]. Figure 1 illustrates the core components of full-stack development, each weighted equally at 20%, highlighting their collective importance. Scalability & Maintenance underscores the need for robust systems that can grow and adapt, while System Architecture emphasizes the foundational design crucial for a project's success. These skills are necessary to perform Frontend Development to create user interfaces and Backend Development or Support to handle server-side logic and data management. Finally, Team Leadership is necessary to coordinate work from the team members and provide the services and help one another. It indicates that this balance distribution is indicative of the fact that you have to be proficient in all of these areas to be regarded a successful full stack developer[15].

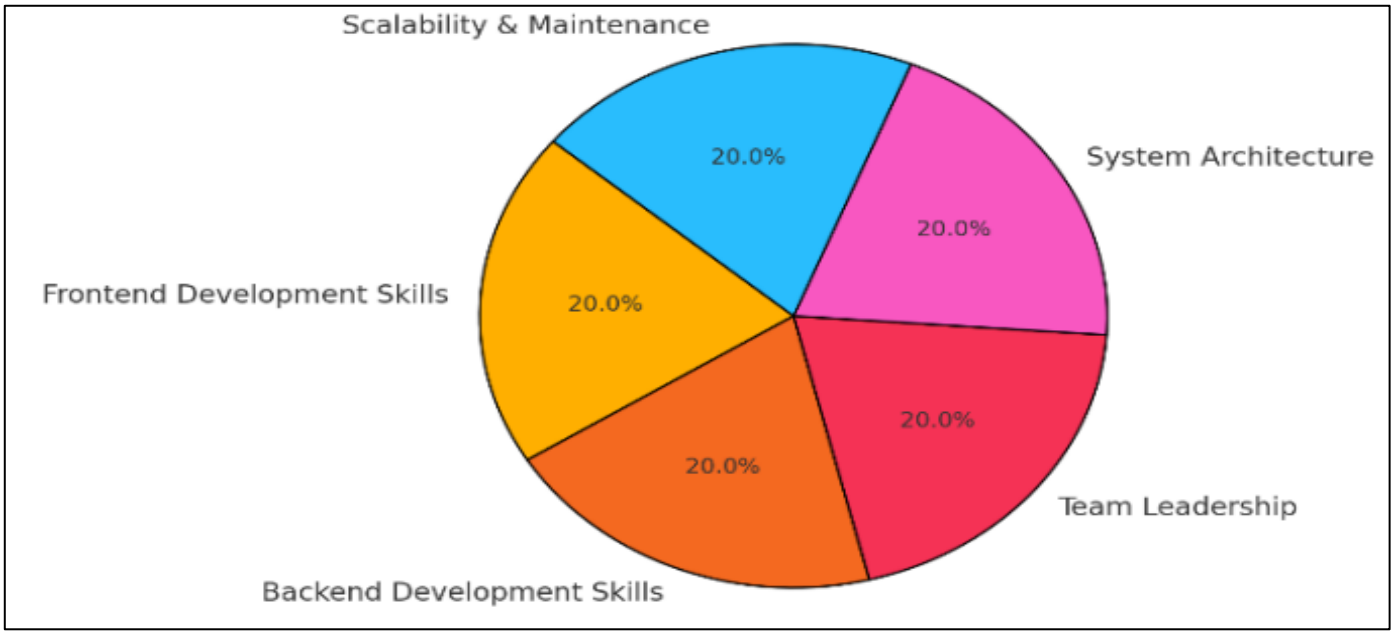

Fig 1: Key Components of Full Stack Development in Practice[15]

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14959407

Development staff must design scalable methods to accommodate complex internal requirements after writing code. Leading development teams requires full technical proficiency as well as superior people skills in this environment. Full-stack developers manage developmental systems by directing teams on the creation of system architecture and agile processes as well as technological stack decisions when working in leadership roles. Their responsibility includes technical alignment with organizational targets and fostering teamwork together with active communication between frontend and backend teams. Modern development practices primarily depend on scalability together with flexibility capabilities. The expanding user base demands systems that full-stack developers must build for their applications to maintain optimal performance during changes in technology platforms.

A. Full-Stack Development Frameworks

A framework is a group of software elements that may be reused to speed up the process of creating new applications. It contains tools like debuggers, compilers, code libraries, and APIs. It is easier to conform to software security standards, save development time, and increase code quality when using frameworks. All full-stack developers use the following frameworks[16]. The list is only an overview of well-known frameworks and is not exhaustive.

- **Ruby on Rails (RoR):** Rails is a Ruby-based framework for building web applications; it's sometimes called RoR. Web development ideas such as DRY and CoC were popularized by it. As it simplifies development of both the front end and the back end, Rails qualifies as a full-stack framework. To make it more versatile, it comes with a plethora of jewels or libraries.
- **Django:** A sophisticated Python web framework, Django encourages rapid development and clean, uncomplicated design. Complying with the batteries-included principle, Django offers almost all the features developers may desire "out of the box." Integrating it with other Python libraries is simple since it is developed in Python. The majority of the setting is managed by Django, freeing developers to concentrate entirely on creating applications. Building dependable and scalable web apps is one of its many uses.
- **Spring Boot:** An addition to the Spring framework, Spring Boot aims to streamline the initial setup and development process. Spring Boot, which is written in Java, aims to reduce the amount of boilerplate code and setup that is typical of Java development in order to rapidly create production-ready applications. Building enterprise-level applications is a suitable match for it because of its great flexibility and compatibility with almost all application needs.
- **Laravel:** Laravel is a beautiful and well documented framework for PHP web applications. A number of features, including an ORM, routing, caching, and authentication, are available in Laravel, which is comparable to Ruby on Rails. It makes creating and maintaining web applications faster with its extensive library and built-in techniques.

B. Several Tools and Technologies Connect Full-Stack Development in Cloud Native:

- **Cloud Development Platforms:** These platforms offer a comprehensive environment for Full-Stack Development, including integrated development tools, continuous integration and deployment services, and access to cloud resources. Examples include Heroku, AWS Amplify, and Google Firebase.
- **Serverless Computing:** Full-Stack Developers can build and deploy applications without managing servers, thanks to serverless computing. AWS Lambda, Google Cloud Functions, and Azure Functions enable event-driven, auto-scaling code execution.
- **Containers:** Containers offer a way to package and deploy applications consistently across different environments. Tools like Docker and Kubernetes enable Full-Stack Developers to easily deploy and scale their applications on cloud infrastructure.
- **Cloud-based Databases:** Scalable cloud databases like AWS DynamoDB, Google Cloud Fire Store, and Microsoft Azure CosmosDB simplify Full-Stack Development by providing managed databases.
- **Cloud IDEs:** Full-Stack Devs can code anywhere with cloud IDEs like AWS Cloud9 and Repl, no local software required.

These tools and technologies make it easier for Full-Stack Developers to take advantage of the benefits of Cloud Computing, leading to increased efficiency and faster time-to-market for applications.

III. TECHNOLOGIES USED IN FULL STACK DEVELOPMENT

An assortment of front-end and back-end technologies are used by the best full-stack apps[17][18]. Figure 2 illustrates the breadth of technologies encompassed within full-stack development. It breaks down the field into five key areas: Front-end, Back-end, Database, DevOps, and Mobile App Development. Each area further branches into specific technologies and concepts [19]. Examples of front-end technologies include Angular and React, in addition to standard languages like JavaScript and CSS. Node.js and Python are examples of back-end languages, whereas Django and Express are examples of frameworks. Database options range from relational databases like MySQL to NoSQL databases like MongoDB. DevOps highlights automation tools like CI/CD and cloud platforms like AWS. Finally, Mobile App Development includes cross-platform frameworks like React Native and Ionic.

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14959407

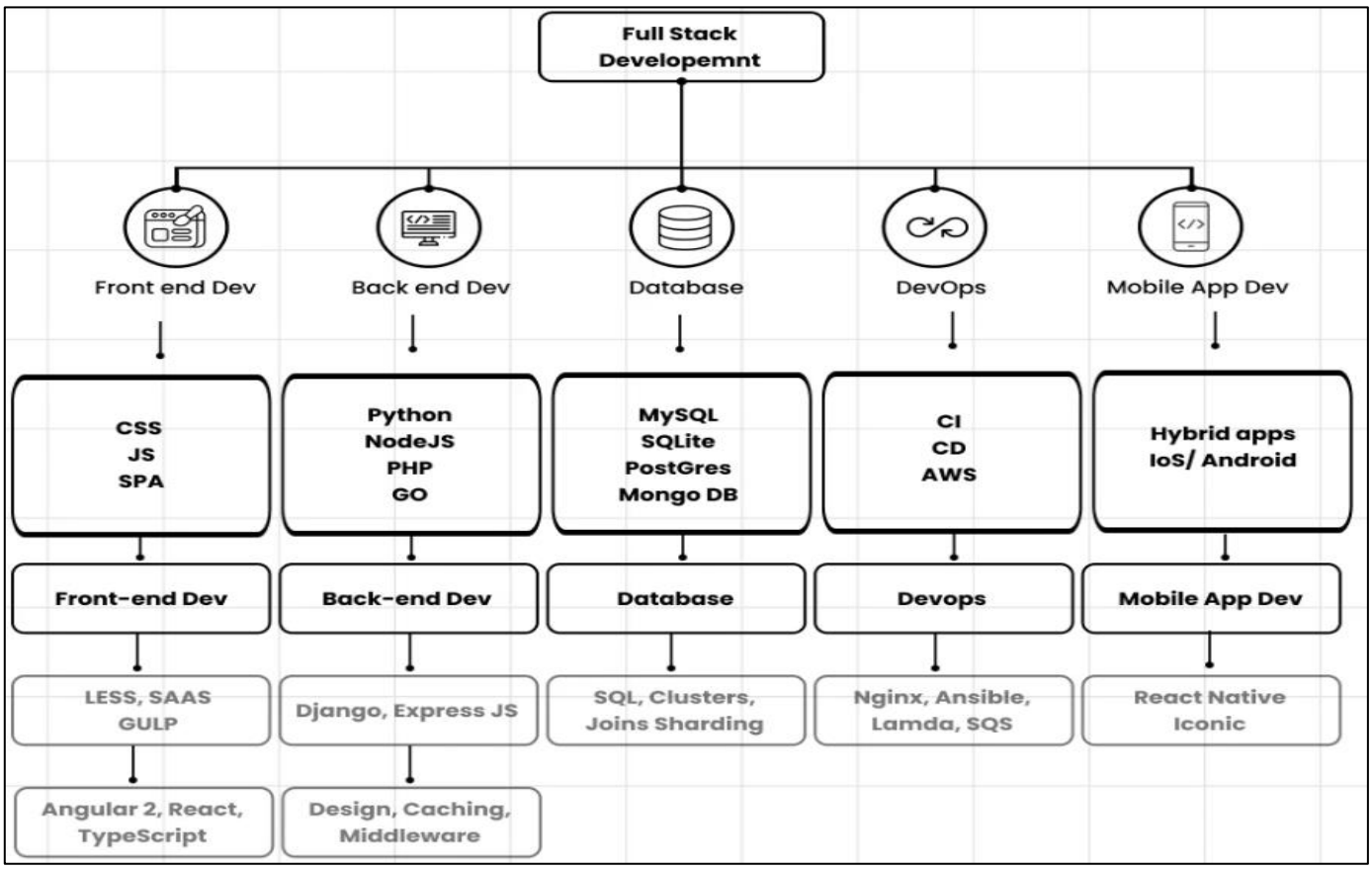

Fig 2: Technologies of Full Stack Development

The Figure effectively demonstrates the diverse skill set required for a full-stack developer, encompassing everything from user interface design to server-side logic and deployment.

A. Front-End

The portion that people view when i visit websites and online apps is known as the front end [20]. The term "front-end development" describes the steps used to bring a design to life on a website. Structure, data, design, content, and functionality are the many layers that make up a website's pages. A website's design, navigation menus, messages, images, videos, and more make up what is known as the front end, which is where visitors may see and interact with the GUI and command line. All the usual suspects like HTML, CSS, and JavaScript are included, along with popular front-end frameworks and libraries like AngularJS, React.js, jQuery, and SASS. In terms of front-end design, the two primary categories are:

- User Experience (UX)
- User Interface (UI)

Despite their seeming similarities, some things are different as we learn more about them. Visual components, animations, images, videos, and other items that seem beautiful on a website but are challenging to build are examples of strong UI but poor user experience (UX); a well-designed website should offer an intuitive experience that doesn't need the user to think too much.

- **HTML & CSS:** HTML (HyperText Markup Language) structures web pages using elements defined by tags, while CSS styles these elements by controlling colors, fonts, backgrounds, and spacing. Frameworks like Bootstrap enhance HTML and CSS management for efficient web design.
- **JavaScript:** A widely used programming language for both frontend and backend development, JavaScript makes web pages interactive. Frameworks like jQuery, React.js, Angular, and Vue simplify development, enabling dynamic and responsive web applications.

B. Back-End

The back end of a website is the unseen component that developers must take into account[21]. Developers working on the back end of a system do things like create libraries, write APIs, and communicate directly with the system's components. Node.js, JavaScript, Python, C++, Java, and PHP are among the many back-end languages and frameworks available. Numerous back-end frameworks are available, including Express, Django, Rails, Laravel, Spring, and many more. Few of the most well-known languages for back-end development and scripting include C#, Ruby, REST, GO, and others. There are two main categories for the back-end component.

- The server
- The application

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14959407

The server responds to every request made by the application's code. The server maintains code synchronization with applications on a regular basis due to the many links between application source code and server requests. Code blocks like async-wait, try-catch, and sync are excellent instances of this language pattern. In response to these requests, the server handles them either on the client side or on the server side, depending on the kind of callbacks included in the code pertaining to the request type. The requests-responses tier is an integral part of the server architecture that facilitates communication for effective synchronization and application smooth running.

- **Node.js:** To enable JavaScript code execution outside of the browser, Node.js was developed. It is an open-source, backend runtime that is based on Chrome's V8 engine. It supports both frontend and backend execution, making it popular for developers using JavaScript as a backend language.
- **Express.js:** A free, open-source Node.js framework used for building web applications and APIs. It simplifies backend development by handling requests and responses efficiently, making it one of the most widely used frameworks for JavaScript-based backend development.

C. Database

The database is an essential part of full-stack web development because it stores all the data in tables with rows (tuples) and columns (attributes). When the application needs to access this data, it sends it through a secure transmission channel, which allows it to handle large amounts of data dynamically and receive and send it all at once[22][23]. Databases include things like MY-SQL, MongoDB, CouchDB, MS-SQL, and so on. while developing robust apps that rely on solid backend infrastructure[24].

- **Relational Database:** A structured database based on the relational model, storing data in tables with rows and columns. Managed by RDBMS using SQL, it maintains predefined relationships between data. Common examples include Oracle, MySQL, and SQL Server[23].
- **Non-Relational Database:** Unlike relational databases, it stores data in flexible formats like JSON instead of tables. Optimized for specific requirements, it provides more scalability and is commonly used for unstructured or semi-structured data[25].

D. Version Control

An integral part of full stack web development, version control (or source control) allows developers to track and manage the evolution of their code. A version control system may be broadly classified into three categories:

- Local Version Control System
- Centralized Version Control System
- Distributed Version Control System

E. Deployment

The process of installing and configuring software on a server allows it to run applications. If everyone else is using our application online, then it must be great. Additionally, having an app deployed involves getting it operating on a particular device, whether it a production environment, a test server, or even simply the user's PC or mobile phone.

F. Web Stack

Web application stacks, or simply web stacks, are collections of software components designed to run websites and online applications. The term "stack" describes the arrangement of adjacent layers. "Stack" describes the way the parts of the system are assembled from different parts. A script interpreter, a database, an operating system, and a web server are the fundamental components of a web stack. Along with the right server hardware, this package guarantees that the information that enquiring clients—typically web browsers—need about comparable web projects is sent to them[14].

➢ *A Few Types of Web Stacks*

A LAMP stack is an example of a web stack; it is a collection of open-source tools for building online apps and web pages. The LAMP stack came first, and then a number of additional stacks. As a result of this technology's constant advancement and the creation of new software, other LAMP stack variants have emerged. Several well-known instances are as follows:

- WAMP (Windows as an OS)
- MAMP (Mac OS X as an OS)
- XAMPP (platform-agnostic FTP server; any OS, Perl, and PHP script interpreters)
- The web stack that is most often used is XAMPP. Linux formerly worked with MySQL databases but has since switched to MariaDB, which is incompatible with DVWA-based SQL Injection attacks.

➢ *Deployment*

The process of starting a program on a server is known as software deployment. It indicates that everyone else is using our program online. Application deployment is the process of getting an application up and running on a particular device, such as a test server, a production environment, or even simply the user's personal computer or smartphone.

IV. FUNDAMENTALS OF CLOUD-NATIVE DEVELOPMENT IN WEB APPLICATIONS

As software development, operation, and maintenance technologies have advanced in recent years, cloud-native architecture has gained appeal due to its special qualities, such as enabling developers to build adaptable, scalable, and modular applications[26][27]. 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

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14959407

like container orchestrators, microservices etc.[28]. 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 adapt automatically to changing workloads, ensuring optimal performance and cost efficiency. CI/CD pipelines, which promote a culture of cooperation and fast iteration, are also often used in development process[29]. The following programs were developed specifically for use in the cloud, as shown in Figure 3:

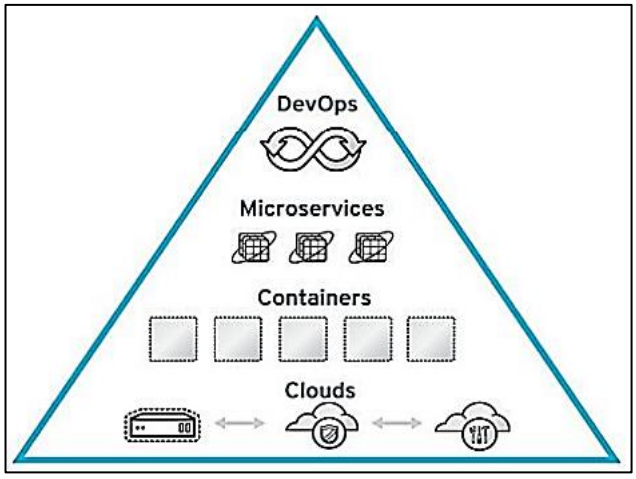

Fig 3: A Pyramid Of Modern Cloud-Native Applications[30].

The layers of cloud-native development. At the base is cloud infrastructure, enabling scalability and flexibility. Figure 3 mentions the containers for portability and resource efficiency, followed by microservices, which ensure modular, independent development. 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[31].
- **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[32].
- **Serverless Computing:** Serverless computing, also known as Function as a Service (FaaS), is a promising new approach to deploying applications in the cloud, spurred by the present trend in corporate application architecture towards microservices and containers. An analysis of cloud computing's serverless designs, namely Function as a Service (FaaS).
- **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.

A. 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 making use of capabilities offered by the application runtime[33].
- **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.
- **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.

B. Best Practices for Building Scalable Cloud-Native Web Applications

➢ *The Following are Best Practices for Building Scalable Cloud-Native Web Applications:*

- *Continuous Integration and Continuous Deployment (CI/CD)*

Automatic software integration is a crucial aspect of Continuous Integration (CI), involving frequent construction and testing of modified source code, often triggered by each commit to version control. CI ensures that software remains changeable and deployable, forming a foundation for Continuous Delivery (CD), which automates the release process up to the staging environment.

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14959407

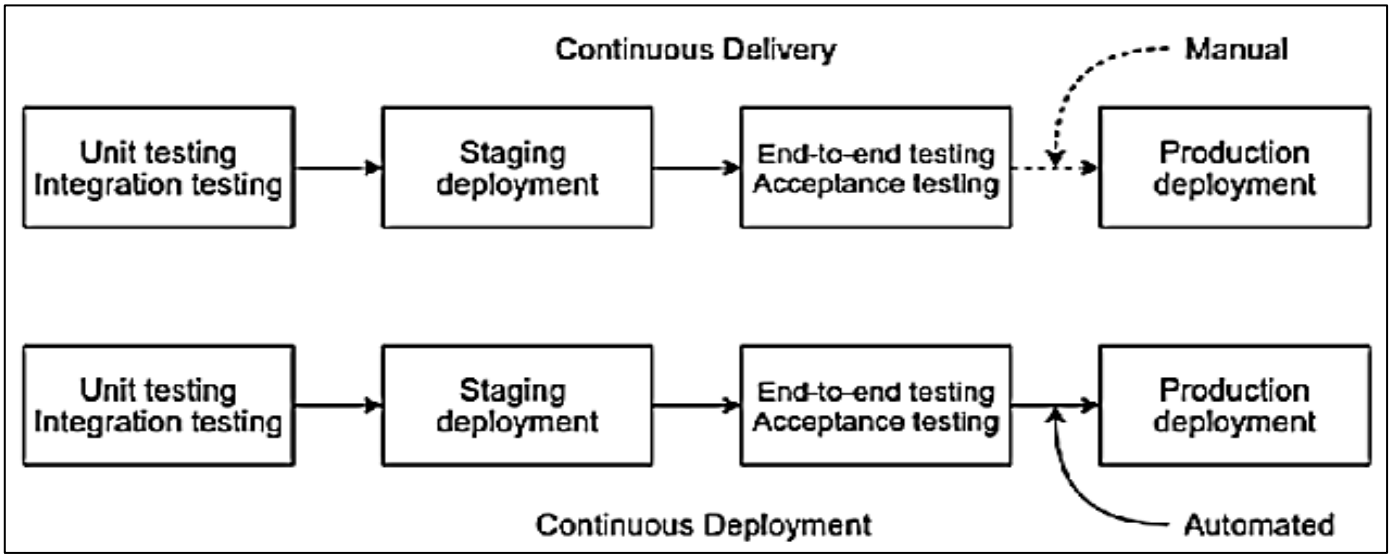

Fig 4: Relations of Continuous Integration, Delivery and Deployment

Continuous Deployment (CDp) takes this further by deploying software directly to production once it passes automated tests. While CI, CD, and CDp are closely linked, they serve distinct purposes, with CI and CD being prerequisites for CDp. Organizations must implement robust CI/CD pipelines to achieve seamless and reliable software deployment, as illustrated in Figure 4.

- *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[34].

- *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.

- *Orchestration*

There are security considerations with orchestration tools that are native to the cloud, such as Kubernetes. As part of this, we will secure the Kubernetes control plane, set up secure configuration procedures, and protect critical components like the data store.

- *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[35], As part of the DevSecOps approach, security monitoring and continuous security testing are essential components that should be seamlessly incorporated into the development process.

C. 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[36].
- **Kubernetes and Container Orchestration Frameworks:** Container orchestration frameworks like Kubernetes have become essential for deploying and managing cloud-native applications[37].
- **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 solute. For Docker containers, it benefits from being deeply ingrained in the Docker ecosystem and making use of its own API.

Table I presents a structured overview of modern full-stack development practices essential for building scalable and maintainable cloud-native applications. It categorizes key technologies, their applications, common challenges faced, and practical solutions to optimize performance, security, and efficiency. By leveraging best practices such as containerization, CI/CD automation, serverless computing, and database optimization, developers can enhance system reliability, scalability, and maintainability in cloud environments.

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14959407

Table 1: Full-Stack Development: Technologies, Applications, Challenges, and Solutions in Cloud native

Technology	Application	Challenges	Solutions
HTML & CSS	Frontend UI design, responsiveness	Inconsistent design, browser compatibility	Use frameworks like Bootstrap, Tailwind, CSS
JavaScript & Frameworks (React, Angular, Vue)	Interactive frontend, SPA development	Performance issues, complex state management	Implement Virtual DOM, use state management libraries (Redux, Vuex)
Node.js & Express.js	Backend API development, server-side logic	High request latency, memory leaks	Optimize API calls and implement caching mechanisms
Relational Databases (MySQL, PostgreSQL)	Structured data storage, transactional integrity	Scalability limitations, complex queries	Use indexing, database sharding
Non-Relational Databases (MongoDB, Firebase)	Unstructured data storage, flexible schema	Data consistency issues, lack of ACID compliance	Implement proper indexing, use hybrid database models
RESTful & GraphQL APIs	Communication between frontend and backend	Over-fetching/under-fetching data	Use GraphQL for precise data fetching
Cloud Computing (AWS, GCP, Azure)	Scalable infrastructure, serverless computing	High costs, vendor lock-in	Optimize resource allocation, use multi-cloud strategies
Docker & Kubernetes	Containerization, microservices deployment	Orchestration complexity, scaling issues	Use automated CI/CD pipeline monitoring tools like Prometheus
CI/CD (Jenkins, GitHub Actions, GitLab CI)	Automated testing, deployment pipelines	Configuration errors, slow build times	Implement pipeline caching, optimize test execution
Security Practices (JWT, OAuth, TLS/SSL)	Authentication, secure data transmission	Vulnerabilities, data breaches	Implement strong encryption, multi-factor authentication
Serverless Computing (AWS Lambda, Google Cloud Functions)	Event-driven architecture, cost efficiency	Cold start latency, limited execution time	Use warm-up techniques, optimize function execution
AI & ML Integration	Predictive analytics, chatbots, personalization	Data quality issues, high computation costs	Use cloud-based AI services, preprocess data effectively

V. LITERATURE REVIEW

This section presents a study on cloud-native applications, with a specific focus on full-stack development for building highly available scalable web applications. Table II summarizes the key studies reviewed in the survey.

Iqbal (2024) explores the many aspects of full-stack web development in great detail. The field of full-stack web development is quickly becoming an important subset of computer science and engineering, and it is already making a big impact on how the IT industry will evolve in the future. Web apps and websites rely heavily on full-stack developers, who oversee both the front-and back-end development processes[13].

Perveen and Edward (2024) explore the core principles and practices associated with cloud-native architectures, including microservices, containerization, and orchestration, and examine their role in enhancing application scalability, resilience, and efficiency. It delves into the benefits of adopting cloud-native approaches, such as improved resource utilization, faster time-to-market, and increased flexibility. The study also addresses the challenges organizations face when transitioning to cloud-native architectures, such as managing complex dependencies, ensuring security, and optimizing performance[38].

Kamau et al. (2024) examine advances in full-stack development frameworks, emphasizing their security and compliance models to address modern-day challenges such as data breaches, unauthorized access, and regulatory non-compliance. The study explores prominent full-stack frameworks, including MEAN (MongoDB, Express.js, Angular, Node.js), MERN (MongoDB, Express.js, React, Node.js), and Django, highlighting their inherent security features. These include robust authentication mechanisms, encryption protocols, and defense against common threats such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF)[39].

Olasehinde (2024) explores the synergies between RDMA, Java, and AngularJS in the development of scalable web applications. We discuss how RDMA enhances the communication layer in full-stack systems, reducing latency and improving overall application performance, especially in distributed environments. By examining the role of Java and AngularJS in the server and client sides of full-stack applications, we highlight how these technologies, when integrated with RDMA, can deliver seamless, scalable web experiences[40].

Dhanveer Prakash and Sharma (2023) DevOps helps to upscale the speed, durability, and efficiency of software development by emphasizing collaboration, automation, and continuous improvement in the cloud. With the ability to scale and resource versatility, the cloud is a great platform

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14959407

for applying DevOps approaches. The CI/CD pipelines, infrastructure as code (IaC), containerization, orchestration, and automated testing play a major role in integrating DevOps in a cloud environment. In the end, the author has provided an overview of how DevOps and cloud integration may increase innovation, and efficiency to build a culture of cooperation[41].

Bharadwaj and Premananda (2022) present a thorough analysis of the cloud-native approach, contrasts it with the conventional method of application design, development, and deployment, and emphasize the need to make the transition. Containers, continuous delivery, devops, and microservices make up the four main tenets of cloud native architecture. A cloud native application's development stack and the many considerations that should go into its architecture are also covered extensively. The article delves into tools like microservices, API-based architecture, and the 12-factor application, all of which are crucial for cloud native apps. Applications built specifically for the cloud may help alleviate some of the additional difficulties associated with cloud computing[1].

Table II highlights the key research contributions in full-stack development and cloud-native computing, highlighting their focus areas, findings, challenges, and contributions.

Table 2: Summary of Related Work on Full Stack Development in Cloud Native Applications

Author(s) & Year	Focus Area	Key Technologies /Concepts	Benefits	Challenges	Future Recommendation
Iqbal (2024)[13]	Full Stack Web Development	Front-end and Back-end Development	Comprehensive website and application management	Keeping up with emerging technologies, managing complexity	Explore integration of AI-driven tools in development
Jelani, Perveen, & Edward (2024)[38]	Cloud-Native Architectures	Microservices, Containerization, Orchestration	Improved scalability, resilience, flexibility, faster time-to-market	Managing complex dependencies, ensuring security, optimizing performance	Develop automated security solutions for cloud-native environments
Kamau et al. (2024)[39]	Full-Stack Development Frameworks and Security	MEAN, MERN, Django; Authentication, Encryption, XSS, CSRF defense	Enhanced security, compliance, and protection against data breaches	Addressing modern security challenges, regulatory compliance	Investigate AI-based threat detection methods
Olasehinde (2024)[40]	Scalable Web Applications using RDMA	RDMA, Java, AngularJS	Reduced latency improved performance in distributed environments	Integration complexities	Enhance RDMA integration with emerging web frameworks
Dhanveer Prakash and Sharma (2023) [41]	DevOps integration in cloud computing	CI/CD pipelines, Infrastructure as Code (IaC), Containerization, Orchestration, Automated Testing	Increased speed, durability, and efficiency of software development; enhanced collaboration and automation	Implementation complexity, security concerns, and managing dependencies	Strengthening DevOps and cloud integration to foster innovation and cooperation
Bharadwaj and Premananda (2022)[1]	Cloud-native architecture	Microservices, DevOps, Continuous Delivery, Containers, 12-factor application, API-based design	Scalability, agility, and flexibility in application development; efficient resource utilization	Transition challenges from traditional to cloud-native approaches; managing microservices complexity	Enhancing cloud-native solutions to address evolving cloud computing challenges

VI. CONCLUSION AND FUTURE WORK

Modern full-stack development in cloud-native environments empowers developers to build scalable, efficient, and maintainable applications by leveraging advanced tools and methodologies. Cloud platforms, serverless computing, containerization, and microservices architectures provide flexibility, automation, and cost-effective solutions for application deployment and management. DevOps practices, CI/CD pipelines, and

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14959407

Infrastructure as Code (IaC) further streamline development, ensuring continuous integration, delivery, and monitoring. Despite challenges such as security risks, resource optimization, and complexity in managing distributed systems, best practices like container orchestration, database optimization, and proactive security measures enable robust and resilient applications. By adopting these modern full-stack development strategies, organizations can enhance performance, scalability, and maintainability, ensuring seamless user experiences in dynamic cloud-native ecosystems. Future research can focus on optimizing microservices orchestration, enhancing serverless security, integrating AI-driven DevOps (AIOps), and exploring blockchain for secure applications. Advancements in edge computing and sustainable cloud practices will further improve scalability, efficiency, and resilience in cloud-native development.

REFERENCES

[1]. D. Bharadwaj and B. S. Premananda, "Transition of Cloud Computing from Traditional Applications to the Cloud Native Approach," in *2022 IEEE North Karnataka Subsection Flagship International Conference, NKCon 2022*, 2022. doi: 10.1109/NKCon56289.2022.10126871.

[2]. Q. Zeng, M. Kavousi, Y. Luo, L. Jin, and Y. Chen, "Full-stack vulnerability analysis of the cloud-native platform," *Comput. Secur.*, 2023, doi: 10.1016/j.cose.2023.103173.

[3]. N. P. Hirenkumar Mistry Kumar Shukla, "Securing the Cloud: Strategies and Innovations in Network Security for Modern Computing Environments," *Int. Res. J. Eng. Technol.*, vol. 11, no. 04, p. 11, 2024.

[4]. T. Mattila, "Building a Complete Full-Stack Software Development," Turku University, 2018.

[5]. S. Arora and S. R. Thota, "Automated Data Quality Assessment And Enhancement For Saas Based Data Applications," *J. Emerg. Technol. Innov. Res.*, vol. 11, pp. i207–i218, 2024, doi: 10.6084/m9.jetir.JETIR2406822.

[6]. V. S. Thokala, "Improving Data Security and Privacy in Web Applications : A Study of Serverless Architecture," *Int. Res. J.*, vol. 11, no. 12, pp. 74–82, 2024.

[7]. T. K. K. and S. Rongala, "Implementing AI-Driven Secure Cloud Data Pipelines in Azure with Databricks," *Nanotechnol. Perceptions*, vol. 20, no. 15, pp. 3063–3075, 2024, doi: https://doi.org/10.62441/nano-ntp.vi.4439.

[8]. E. Nikulchev, D. Ilin, and A. Gusev, "Technology Stack Selection Model for Software Design of Digital Platforms," *Mathematics*, vol. 9, no. 4, 2021, doi: 10.3390/math9040308.

[9]. A. Ramírez, J. R. Romero, and S. Ventura, "Interactive Multi-Objective Evolutionary Optimization of Software Architectures," *Inf. Sci. (Ny).*, vol. 463–464, pp. 92–109, Oct. 2018, doi: 10.1016/j.ins.2018.06.034.

[10]. Y. Yang, B. Yang, S. Wang, T. Jin, and S. Li, "An Enhanced Multi-Objective Grey Wolf Optimizer for Service Composition in Cloud Manufacturing," *Appl. Soft Comput.*, vol. 87, Feb. 2020, doi: 10.1016/j.asoc.2019.106003.

[11]. H. Cherukuri, R. Gupta, S. Shukla, A. Rajan, and S. Aravind, "The Impact of Agile Development Strategies on Team Productivity in Full Stack Development Projects," *Int. J. Intell. Syst. Appl. Eng.*, pp. 175–184, 2024.

[12]. A. Goyal, "Optimising Cloud-Based CI/CD Pipelines: Techniques for Rapid Software Deployment," *Tech. Int. J. Eng. Res.*, vol. 11, no. 11, pp. 896–904, 2024.

[13]. K. Iqbal, "Full Stack Web Development: Vision, Challenges and Future Scope," *Int. J. Sci. Res. Eng. Manag.*, vol. 08, no. 04, pp. 1–5, 2024, doi: 10.55041/ijsrem30338.

[14]. Akshat Dalmia and Abhishek Raj Chowdary, "The New Era of Full Stack Development," *Int. J. Eng. Res.*, 2020, doi: 10.17577/ijertv9is040016.

[15]. T. M. C. Venkata Ashok Kumar Boyina, "Fullstack Development in Practice Leading Teams and Architecting Scalable Solutions," *Int. Res. J. Mod. Eng. Technol. Sci.*, vol. 06, no. 12, pp. 4697–4703, Jan. 2024, doi: 10.56726/IRJMETS65797.

[16]. Vasudhar Sai Thokala, "Enhancing Test-Driven Development (TDD) and BDD Methodologies in Full-Stack Web Applications," *Int. J. Sci. Res. Arch.*, vol. 10, no. 1, pp. 1119–1129, Oct. 2023, doi: 10.30574/ijsra.2023.10.1.0815.

[17]. V. P, "Full Stack Development-A New Horizon in Technologies," *Int. Res. J. Mod. Eng. Technol. Sci.*, vol. 05, no. 06, pp. 2370–2372, 2023, doi: 10.56726/IRJMETS42018.

[18]. Y. Baiskar, "MERN: A Full-Stack Development," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 1, pp. 1029–1035, 2022.

[19]. D. D. Rao, D. Dhabliya, A. Dhore, M. Sharma, S. S. Mahat, and A. S. Shah, "Content Delivery Models for Distributed and Cooperative Media Algorithms in Mobile Networks," in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, IEEE, Jun. 2024, pp. 1–6. doi: 10.1109/ICCCNT61001.2024.10724905.

[20]. N. K. Bharali, "Full Stack Web Development Of Redux-Based Web Applications with Dynamic Microservices (Case Study - Idea Repository)," *Int. Res. J. Mod. Eng. Technol. Sci.*, vol. 05, no. 01, Jan. 2023, doi: 10.56726/IRJMETS33219.

[21]. M. Khorasani, M. Abdou, and J. H. Fernández, *Web Application Development with Streamlit: Develop and Deploy Secure and Scalable Web Applications to the Cloud Using a Pure Python Framework*. 2022. doi: 10.1007/978-1-4842-8111-6.

[22]. A. G. Milavkumar Shah, "Distributed Query Optimization forPetabyte-Scale Databases," *Int. J. Recent Innov. Trends Comput. Commun.*, vol. 10, no. 10, 2022.

ISSN No:-2456-2165 https://doi.org/10.5281/zenodo.14959407

[23]. S. S. S. Neeli, "A Comparative Analysis of SQL and NoSQL Database Management within Cloud Architectures for Mission-Critical Business Systems," *ESP Int. J. Adv. Comput. Technol.*, vol. 2, no. 4, pp. 140–149, 2024.

[24]. G. A. Nys and R. Billen, "From consistency to flexibility: A simplified database schema for the management of CityJSON 3D city models," *Trans. GIS*, 2021, doi: 10.1111/tgis.12807.

[25]. B. Boddu, "Importance Of Nosql Databases: Business Strategies With Administration Tactics," *Int. J. Core Eng. Manag.*, vol. 7, no. 2, 2022.

[26]. S. Murri, S. Chinta, S. Jain, and T. Adimulam, "Advancing Cloud Data Architectures: A Deep Dive into Scalability, Security, and Intelligent Data Management for Next-Generation Applications," *Well Test. J.*, vol. 33, no. 2, pp. 619–644, 2024, [Online]. Available: https://welltestingjournal.com/index.php/WT/article/view/128

[27]. V. Ugwueze, "Cloud Native Application Development: Best Practices and Challenges," *Int. J. Res. Publ. Rev.*, vol. 5, pp. 2399–2412, 2024, doi: 10.55248/gengpi.5.1224.3533.

[28]. S. S. S. Neeli, "Leveraging Docker and Kubernetes for Enhanced Database Management," *J. Artif. Intell. Mach. Learn. Data Sci.*, vol. 1, no. 1, p. 5, 2022.

[29]. Godavari Modalavalasa, "The Role of DevOps in Streamlining Software Delivery: Key Practices for Seamless CI/CD," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 1, no. 12, pp. 258–267, Jan. 2021, doi: 10.48175/IJARSCT-8978C.

[30]. S. Chippagiri and P. Ravula, "Cloud-Native Development: Review of Best Practices and Frameworks for Scalable and Resilient Web Applications," vol. 8, pp. 13–21, 2021.

[31]. 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.

[32]. 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.

[33]. I. Jana and A. Oprea, "AppMine: Behavioral analytics for web application vulnerability detection," in *Proceedings of the ACM Conference on Computer and Communications Security*, 2019. doi: 10.1145/3338466.3358923.

[34]. O. C. Oyeniran, O. T. Modupe, A. A. Otitoola, O. O. Abiona, A. O. Adewusi, and O. J. Oladapo, "A Comprehensive Review of Leveraging Cloud-Native Technologies for Scalability and Resilience in Software Development," *Int. J. Sci. Res. Arch.*, vol. 11, no. 2, pp. 330–337, Mar. 2024, doi: 10.30574/ijsra.2024.11.2.0432.

[35]. S. S. S. Neeli, "Optimizing Database Management with DevOps: Strategies and Real-World Examples," *J. Adv. Dev. Res.*, vol. 11, no. 1, p. 8, 2020.

[36]. A. Hakli, D. Taibi, and K. Systa, "Towards Cloud Native Continuous Delivery: An Industrial Experience Report," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, IEEE, Dec. 2018, pp. 314–320. doi: 10.1109/UCC-Companion.2018.00074.

[37]. B. Boddu, "Unleashing the Power of Docker and Kubernetes for Databases," *North Am. J. Eng. Res.*, vol. 3, no. 3, p. 5, 2022.

[38]. U. Jelani, K. Perveen, and E. Edward, "Cloud-Native Architectures: Building and Managing Applications at Scale," *Int. J. Mach. Learn. Res. Cybersecurity Artifcial Intell.*, vol. 15, no. 1, 2024.

[39]. E. Kamau, A. Collins, G. Babatunde, and A. Alabi, "Advances in Full-Stack Development Frameworks: A Comprehensive Review of Security and Compliance Models," *Int. J. Multidiscip. Res. Growth Eval.*, vol. 5, pp. 1172–1185, 2024, doi: 10.54660/.IJMRGE.2024.5.1.1172-1185.

[40]. T. Olasehinde, "Full-Stack Web Development Trends: Leveraging RDMA, Java, and AngularJS for Scalable Applications," 2024.

[41]. M. P. Dhanveer Prakash and N. Sharma, "The Convergence of DevOps and Cloud Computing: A Redefining Software Development," in *2023 Seventh International Conference on Image Information Processing (ICIIP)*, 2023, pp. 800–805. doi: 10.1109/ICIIP61524.2023.10537710.