Exploring the Integration of Blockchain and Distributed DevOps for Secure, Transparent, and Traceable Software Development

Junaid Nasir Qureshi  
University of Management and Technology

Muhammad Shoaiq Farooq  
University of Management and Technology

Usman Ali  
University of Management and Technology

Uzma farooq  
University of Management and Technology

Momina Shaheen (momina.shaheen@roehampton.ac.uk)  
University of Roehampton

Article

Keywords: DevOps, Blockchain, Smart Contracts, Distributed DevOps

Posting Date: July 27th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3187369/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License. Read Full License
Abstract

Distributed DevOps is a software development methodology that aims to integrate the work of development and operations teams without being bound by geographical constraints. This methodology excels in enhancing collaboration and the speed of software development. However, it does suffer from a lack of security, transparency, and traceability, which can result in project delays, a lack of trust between stakeholders, and even project failure. This paper addresses these issues of Distributed DevOps by implementing Blockchain technology. In this paper, we propose a framework that utilizes the benefits of blockchain technology to eliminate the shortcomings of Distributed DevOps. We present performance results that show the effectiveness of our framework in a real-world scenario, highlighting its ability to improve transparency, traceability, and the security of the DevOps pipeline. In conclusion, our research contributes to the growing literature on the intersection of blockchain and DevOps. It provides a practical framework for organizations looking to leverage blockchain technology to improve their development processes.

1. Introduction

Software development has been becoming more complex day by day, and to overcome the complexities and challenges of software development, one of the best approaches is using DevOps. DevOps is a relatively new technique used to give new software updates with maximum reliability and accuracy while doing continuous integration and continuous development (CI/CD) [1]. DevOps combines the Development Team and Operation Team’s people, processes, and technology to work in a collaborative environment for delivering software as a service [2]. The DevOps eliminate the gap between the Development Team and Operation Team [3], and because of that, the software update can be released successfully and rapidly. Figure 1 shows a visual representation of the DevOps workflow. On the other hand, Distributed DevOps is an approach to developing softwares that involve distributed teams working together to build, deploy, and maintain the software in a collaborative manner [4]. In traditional DevOps models, developers and operations teams work together to build and deploy software, but this collaboration often occurs within a single geographical location. In contrast, Distributed DevOps involves geographically dispersed teams that may not be part of the same organization. The potential advantage of distributed DevOps is that it allows organizations to tap into a wider pool of talent and expertise, as they are not limited to a single location. It can also enable organizations to be more agile and responsive to changing market conditions, as they can quickly bring new resources online as needed.

The adoption of DevOps practices in distributed teams can bring numerous benefits, but it also presents significant challenges around transparency, trust, security, and traceability. It is critical to establish trust and transparency, particularly when teams are working across different locations and time zones [6]. Security is also paramount to prevent malicious actors from disrupting or compromising the project [7]. Moreover, software traceability also holds significant importance as it allows the traceability of different artifacts, including requirements, design models, and code [8].
The development pipeline is susceptible to cyber threats and malicious attacks [9], which can result in data breaches, system downtime, and financial losses if there is no security in the system. Moreover, without appropriate traceability, it can be difficult to identify the root cause of errors or bugs, which can lead to delays in the development cycle and increased costs. The lack of transparency and trust can result in poor collaboration and communication between team members [10], leading to misunderstandings and misaligned goals. This can also result in delays, wasted effort, and decreased productivity.

Blockchain technology can significantly impact Distributed DevOps for several reasons: its decentralized nature, immutability, and distributed ledger. Blockchain can be defined as an interconnected sequence of blocks, where each block consists of multiple transactions, resulting in decentralized and tamper-proof data [11]. Since its emergence, blockchain technology has undergone significant advancements, enabling the construction of Smart Contracts that automatically store and execute code on the blockchain. These smart contracts are also excellent in streamlining digital interactions and transactions [12]. To overcome the challenges in software development with Distributed DevOps, an efficient blockchain-based framework is essential.

In this paper, we have proposed a framework that combines DevOps with blockchain that can utilize the benefits of blockchain technology to create a more secure, transparent, traceable, trustable, and efficient development environment. This framework involves developing, integrating, and deploying applications in a distributed environment, leveraging blockchain's decentralized and tamper-proof properties to create a trustworthy and traceable, secure environment. The framework includes smart contracts, which can automate some tasks of Distributed DevOps, reducing the time and effort required to complete the software development cycle. It also includes consensus mechanisms, which enable multiple parties to agree on the state of the code and ensure that all parties have access to the same information. The framework includes robust security measures, such as cryptographic hashing and encryption, to protect the code and sensitive data from cyber threats and malicious attacks. Moreover, the blockchain's scalability issue is also solved by using InterPlanetary File System.

The novelty of our proposed framework is that it enhances the security, traceability, and transparency of software development using DevOps for distributed teams. We offer a strategy for implementing DevOps with Blockchain, reducing payment issues between clients and software development teams.

In the remaining paper, the sections are as follows: In Section 2, the related work has been presented, and in Section 3, we have elaborated Preliminaries used in the model. The proposed framework is shown in Section 4, and Section 5 contains the implementation and performance. Furthermore, in Section 6, the discussion is shown, and finally, Section 7 describes the conclusion and future work.

2. Related Work

There has been a significant amount of research done on the intersection of blockchain with DevOps and other software development models. Here are a few examples of research in this area:
Sandip Bankar and Deven Shah [13] also presented a paper that intersects blockchain with DevOps. This paper describes how blockchain technology can be used in DevOps for software development. The benefits of this include improved quality and performance, as well as increased security. This is achieved by storing all project artifacts in a decentralized and secure blockchain environment. Although, the proposed solution does not provide a solution when teams are working in a distributed environment.

Muhammad Azeem Akbar et. al. [14] have investigated the potential benefits of using blockchain technology in a DevOps paradigm. A framework is proposed which helps merge the characteristics of blockchain with the DevOps paradigm. This provides an effective means for the adoption of blockchain in DevOps while ensuring its advantages over other solutions are maximized. However, the proposed framework does not address any parts related to payment.

Faizan Tariq & Ricardo Colomo-Palacios [15] conducted an analysis of the usage and benefits of Blockchain Smart Contracts in Software Engineering. The study also highlighted several challenges that have yet to be addressed by current methodologies. The findings suggest that a greater practical application of this system has the potential to pave the way for future research opportunities. However, this study does not mention any about DevOps or Distributed DevOps.

Md Jobair Hossain Faruk, Santhiya Subramanian, Hossain Shahriar et. al. [16] examine the impact of existing software engineering processes. They consider the need to adopt new concepts and evolve current software engineering processes for blockchain systems. The study also looks at the role of software project management in the development of blockchain-oriented software. Nevertheless, this was a systematic study and does not present any kind of framework.

N Sudha and Raju Ramakrishna Gondkar [17] present a novel framework that integrates Blockchain technology into Agile software development processes. This paper explores the potential of Blockchain as a means to address these challenges by enabling robust tracking and traceability mechanisms within Agile software development. Thus, this research is only for agile software development and not DevOps.

These are just a few examples of the many research efforts focused on the blockchain with DevOps and other software development models. It is worth noting that while these studies have the potential to bring many benefits to DevOps and other software development processes, but they do not present any framework that is suitable for Distributed DevOps where teams are not in the same place or even time zone.

The novelty of our proposed framework is that it could make it simpler for distributed teams to communicate and share data in a secure, traceable, and immutable environment. We have presented a comprehensive strategy for implementing DevOps methods with respect to Blockchain to successfully execute projects while reducing conflicts. We have solved the issues of traceability, trust, and transparency of Distributed DevOps so that everyone can work in a trusted and more secure environment. DevOps phases; project initiation, continuous development, continuous integration & delivery, continuous deployment, and continuous monitoring are shown, and how they will work with the Blockchain is also...
described. Using cryptocurrencies and digital wallets, we have also resolved payment-related issues. In addition, InterPlanetary File System is used to address the scalability issue of the Blockchain.

3. Preliminaries

This section showcases the preliminaries for the proposed framework. The major components that will be used in this framework will be described in this section including, IPFS, Decentralized Applications, Blockchain, Smart Contracts, and Jenkins.

3.1 IPFS (InterPlanetary File System)

IPFS (InterPlanetary File System) enables decentralized storage and sharing of data in a content-addressable distributed file system [18]. It operates by storing clusters of hashed files in individual nodes of the system [19]. IPFS uses a peer-to-peer network to make it easy for people to share files without having to go through central authorities or servers. This makes it resistant to censorship and seizure.

3.2 Blockchain and Smart Contracts

Blockchain is a decentralized ledger system that offers a safe, transparent, and permanent record of transactional data between two parties [20]. Through the use of cryptographic techniques, it produces a record of transactions that cannot be altered. This record takes the form of a chain of blocks, each of which references the block that came before it.

Smart Contracts are code snippets designed to execute versatile tasks [21]. They are kept on a blockchain. They make it possible to execute complicated transactions in a way that is tamper-proof, transparent, and efficient, which could potentially reduce the need for intermediaries.

3.3 Decentralized Applications

DApps are open-source, decentralized applications that can operate without human intervention [22]. These applications are made with smart contracts and have a front and backend that run on a decentralized peer-to-peer network.

3.4 Jenkins

Jenkins is a widely-used open-source automation tool for CI/CD in software development [23]. Its flexibility, extensibility, and support for pipeline-as-code make it a popular choice among development teams. Its web-based user interfaces, collection of plugins, and other various features make it easy to manage and monitor the build, test, and deployment process, which leads to more efficient and streamlined software development.

4. Proposed Framework
In this section, a proposed framework is presented that will make the CI/CD more traceable, secure, and trustworthy even when the teams are scattered in different locations. In this framework, the Blockchain, DApps, Jenkins, Smart Contracts, and IPFS have been used to make DevOps more secure, transparent, traceable, and immutable without disturbing the automation of DevOps. Figure 2. shows the flow chart of the proposed framework.

The main goal of this study is to bring data like source code, files, and artifacts to distributed ledger so that it can become accessible easily for every stakeholder while still being tamperproof and secure.

4.1 High-level Architecture

The high-level architecture for the proposed framework is shown in Fig. 3. It shows the overall working of the model that will use Blockchain, an InterPlanetary File System, and a Smart contract. In the architecture, the main focus is on data being decentralized without being in control of a central authority.

4.2 Layered Architecture

The framework adheres to the blockchain architectural style, and we have presented the 7 layered architecture in Fig. 4.

Presentation Layer

The proposed system's Presentation Layer includes a user-facing interphase and a decentralized application (DApps). Its primary goal is to connect clients and DevOps teams to the system.

Application Layer: The metadata relating to transaction records, payments, mockups, prototypes, etc., along with agreements made between DevOps teams and clients in the form of videos, text, and audio, are present within this layer. Digital currencies, including ETH, USDT, BUSD, or BTC, are also in this layer to facilitate transactions after the completion of one successful iteration. The primary function of this layer is to enable seamless communication among stakeholders while serving as an intermediary between the Presentation Layer and Business Logic Layer.

Business Logic Layer

This layer has all the smart contracts that govern the terms and conditions of interactions within the system. It serves as an active database for these contracts, enabling the acknowledgment, execution, and enforcement of communication rules. This layer plays a critical role in facilitating the functioning of the system by ensuring that all interactions are carried out in accordance with established rules and regulations.

Trust Layer: The Trust Layer of the layered architecture of distributed DevOps is responsible for managing the system's consensus algorithms, such as Proof of Stake or Proof-of-Work. The trust layer plays a critical role in ensuring the security and reliability of the system by implementing robust consensus algorithms and security protocols.
**Transaction Layer:** The transaction layer is responsible for facilitating the developers and customers by enabling them to trigger transactional smart contracts. This layer also oversees the processes of mining/staking and validating the blocks containing these transactions. The transaction layer plays a critical role in the functioning of the proposed system.

**Hardware/Infrastructure Layer:** This layer includes the peer-to-peer network that validates transactions and a distributed storage system that stores and retrieves files on the decentralized storage systems.

**Security Layer:** The Security Layer is responsible for security measures to protect the network from potential attacks. The security layer works parallel with the rest of the system, incorporating algorithms and security protocols to safeguard the blockchain network.

### 4.3 Detailed Architecture

This section describes every step of DevOps in detail with the implementation of Blockchain technology. The Detailed Architecture of the system is shown in Fig. 5.

**Project Initiation:** The manager or owner can create the project at the project's initiation. After creating a project, the manager/owner can add members to the project using data that include the Name, Username, Email, and Contact Number of developers. After that, all members will be given unique identity materials like a "Key." Members can log in to the project using unique keys to do their tasks. All unique keys, along with members’ data and basic project details going to upload on the blockchain using a smart contract.

The owner or manager will write all basic requirements and terms and conditions at the project's initiation. Tables 1 and 2 shows the JSON file for the agreement between the client and the organization.

**Table 1. File for Customer Agreement**

```json
Customer's Agreement
{
  "Total budget": "$1000"
  "Project deadline": "30/10/2023"
  "Each Iteration Duration": "2 Weeks"
  "Requirements": "All project requirements should be completed within budget and time"
}
```

**Table 2. File for Organization's Agreement**
All stakeholders need to accept the terms and conditions set by the other side to reach a consensus. After that, the data can be stored on the blockchain using a smart contract after the consensus is met between the manager/owner, developers, and clients. The complete work of the Project Initiation Phase is shown in Fig. 6.

**Continuous Development Phase**: In the context of DevOps, the continuous development phase refers to the process of continuously planning and coding.

**Planning**

In this phase, the developer, managers, and all other stakeholders directly or indirectly involved in software development make a plan on how to complete the project successfully. Because the teams are located in different locations, the planning can happen via online video calling platforms like Zoom or Google Meetings. All the details of the planning phase, like video conference recordings and notes, can be stored in the IPFS because of Blockchain’s scalability issue [24]. Only the returned hash file is stored on the Blockchain to make it tamperproof and more secure, so no one can change it, as shown in Fig. 7. This data is visible to all the stakeholders. In this way, anyone who can access details can see them but not change them.

In traditional planning, there is no involvement of a secured and tamper-proof system, which makes the planning phase vulnerable to documents being deleted, manipulated, or tampered with, resulting in confusion and clashes between stakeholders. However, with Blockchain in place, the data can be securely stored.

**Coding**

Git has been one of the most popular tools for distributed revision control systems [25] and is widely used in DevOps. However, the InterPlanetary File System (IPFS) can be used as a decentralized version control system in software development [26]. IPFS provides a content-addressed version control system for managing files. With IPFS’s distributed storage, version control, and content addressing features, a decentralized and resilient file management system can be made that operates similarly to Git. However,
it's worth noting that while IPFS provides some of the foundational components for a Git-like system, additional tooling and interfaces may be required to fully replicate the entire range of Git's features and workflows.

**Continuous Integration & Delivery Phase:** Continuous Integration (CI) and Continuous Delivery (CD) are the main components of a DevOps workflow, enabling teams to build, test, and deliver software quickly and reliably. In this blockchain-based framework, CI/CD can be further enhanced by leveraging the immutability and transparency of the blockchain.

To implement CI/CD, our framework is using an open-source tool, Jenkins. Jenkins is one of the most popular automation tools for developers [27]. It is used to automate parts of the software development process, such as building, testing, and releasing code changes.

In the proposed framework, whenever developers update the repository, the code goes through the Jenkins server, which performs a series of automated operations, including code review, unit testing, integration testing, and building an executable package (e.g., WAR or JAR). Once the package is built, it will upload to a decentralized storage system IPFS. The IPFS hash of the package, along with the results of the code review, unit tests, and integration tests, should store on the blockchain. This provides a transparent and immutable record of the file changes and the testing performed. If there is a code error during code review, unit test, or integration testing, the system generates an alert that notifies all developers to remove the error in the code and update the repo as soon as possible.

To enable continuous delivery, smart contracts are used. A smart contract will be placed that automatically deploys the package to the server or cloud provider when certain conditions are met (e.g., passing all tests). The smart contract can also track the deployment process, ensuring that the package is delivered correctly and providing an audit trail of the delivery.

The information on the executable file that should be stored on the blockchain is shown in Table 3.

<table>
<thead>
<tr>
<th>Info</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>The time when files upload</td>
</tr>
<tr>
<td>Date</td>
<td>Date of uploading of file</td>
</tr>
<tr>
<td>Name</td>
<td>Name of file</td>
</tr>
<tr>
<td>Version</td>
<td>Version of file</td>
</tr>
</tbody>
</table>

After the file is uploaded on Blockchain, all developers get notified about the file. The complete working of CI/CD is shown in Fig. 8.
Continuous Deployment: Continuous Deployment ensures the automatic release of code changes to the production environment [28].

Smart Contracts play a major role in implementing continuous deployment in our proposed framework. Before deploying the main file, the code has to go through a smart contract specifically created to check quality standards, security requirements, and compliance with regulatory requirements. The smart contract should be already integrated into the CI/CD pipeline to enforce the deployment criteria specified in the contract. This can be done by creating a custom script or plugin that interacts with the contract or by using an existing blockchain integration tool. Once the main file is deployed, the owners, managers, clients, and developers are notified about this change. The status of this implemented functionality changes to “Deployed”.

Continuous monitoring: In this phase, all the metrics related to the performance and availability of the software and infrastructure components are recorded on the blockchain with the help of smart contracts. By recording these metrics on the blockchain, they become immutable and secure.

When a problem or alert is detected in the system, it will be recorded on the blockchain with the help of a Smart Contract. In parallel, all the developers who are part of the DevOps team will be notified about it immediately. This notification ensures that the problem can be stored and then resolved as soon as possible, minimizing the impact on end-users.

To monitor the process, any preferred software can be used, such as Nagios, Splunk, Prometheus, etc. However, the software must be compatible with the blockchain framework being used in the distributed DevOps environment. This can be done by creating a custom script or plugin. This ensures that the monitoring data can be recorded on the blockchain and accessed by all the relevant stakeholders securely and transparently.

Payments Phase: The payment phase automatically gets enabled when one iteration gets completed by the organization or after two weeks. The payment can be set to be paid after 2 weeks or after one successful deployment of functionality, depending on the decision made at the time of project initiation.

To complete the payment process, every stakeholder in the project must have a Digital Wallet. The payment can be made with digital currencies like ETH (Ethereum) or BTC (Bitcoin). The managers, developers, or testers can change this digital currency to their local currency, like PKR, USD, or EUR, using cryptocurrency exchanges, as shown in Fig. 9.

The client has to pay the payment; otherwise, as a result, the work cannot proceed further. The client must pay the team within the time frame settled while initiating the project. The payment requirements are written in a smart contract, so they cannot be changed or altered. The JSON object for payment, customer’s penalty (in case of no payment), and developer’s penalty (in case of iteration delay) are shown in Tables 4, 5, and 6, respectively.
Table 4
File for Payment

<table>
<thead>
<tr>
<th>Payment</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
</tr>
<tr>
<td>“Project Budget”: “$1000”</td>
</tr>
<tr>
<td>“Payment After 1 Successful Iteration”: “$100”</td>
</tr>
<tr>
<td>OR</td>
</tr>
<tr>
<td>“Payment After 2 Weeks”: “$100”</td>
</tr>
<tr>
<td>“Payment Due”: “Yes”</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

Table 5
File For Customer Penalty (In Case Of No Payment)

<table>
<thead>
<tr>
<th>Customer’s Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
</tr>
<tr>
<td>“Overall Project Progress”: “50%”</td>
</tr>
<tr>
<td>“Payment Due”: “Yes”</td>
</tr>
<tr>
<td>“Payment Status”: “Pending”</td>
</tr>
<tr>
<td>“In Case of Non-Payment”: “Stop Project’s Functions”</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

Table 6
File For Developer’s Penalty (In Case of Iteration Delay)

<table>
<thead>
<tr>
<th>Developer’s Penalty</th>
</tr>
</thead>
<tbody>
<tr>
<td>{</td>
</tr>
<tr>
<td>“Overall Project Progress”: “50%”</td>
</tr>
<tr>
<td>“Iteration Due Date”: “25/05/2023”</td>
</tr>
<tr>
<td>“Iteration Status”: “Not Complete”</td>
</tr>
<tr>
<td>“Penalty”: “$5 Deduction”</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>
5. Implementation and Performance

In this section, we have tested the efficiency of the framework to demonstrate its effectiveness in a real-world scenario.

5.1 Performance Assessment

To test the efficiency and performance of our proposed model, we utilized specific tools to implement and test a blockchain network. We employed Spyder IDE Version 5.4.1 for implementing the blockchain in Python, and Postman Version 10.14.2 was utilized to test the network's performance. Additionally, we have used the Python library Matplotlib for plotting graphs in order to do graphical representations of our results.

5.2 Performance Evaluation

In this section, we present the performance results of our model in a real-world scenario. The Postman tool is responsible for HTTP requests, specifically 'GET' and 'POST', to interact with APIs. Table 7 shows each function's name that is employed to evaluate the framework along with its purpose.

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetChain</td>
<td>To demonstrate the entire Chain.</td>
</tr>
<tr>
<td>ConnectNode</td>
<td>To establish connections between nodes.</td>
</tr>
<tr>
<td>MineBlock</td>
<td>To successfully mine a block within a blockchain network.</td>
</tr>
<tr>
<td>AddTransaction</td>
<td>To include a single transaction within a block.</td>
</tr>
</tbody>
</table>

A total of 500 blocks have been mined by sending HTTP requests. As depicted in Fig. 10, the chain size consistently increases with each block mined. On average, the single block size is approximately 465B. Throughout the process, starting from the 1st block and continuing to the 500th block, the chain size has increased from 290B to 232 KB.

It is essential to highlight that mining each block's latency (measured in milliseconds) appears to be random. There is no visible pattern or significant correlation observed in the latency values. As shown in Fig. 11, the latency fluctuates from 12ms to 1359ms as 500 blocks are mined in the blockchain.

This variability in latency suggests that the time taken to mine a block can differ significantly for each instance. Factors such as network congestion, computational resources, and the complexity of the block
being mined can contribute to these fluctuations. It is crucial to consider and analyze the latency distribution to gain insights into the overall performance and responsiveness of the blockchain network.

Conclusively, these evaluation results have provided valuable insights into the behavior of our proposed model in a real-world scenario. The evaluation of our model using specific tools and metrics has allowed us better to understand the efficiency and performance of the blockchain network. Further analysis and optimization can be pursued to enhance the overall reliability and responsiveness of the blockchain system.

5.3 Comparison of Presented Framework with Related Work

In this subsection, we have presented a comprehensive comparison between our proposed framework for distributed DevOps and the related work that has been conducted in this domain. We have carefully examined the existing literature and research efforts in the field of blockchain-based software engineering, taking into account the various aspect. By comparing our proposed framework with the related work, we aim to highlight the unique contributions and advantages offered by our approach. Table 8 shows the comparison of our work with related work.

*Blockchain-Based Study:* The presented framework is based on blockchain, enabling secure, immutable, transparent, and traceable software development.

*Framework Presented*

We have introduced a comprehensive framework that addresses the shortcomings of related work. The framework demonstrates the successful execution of each step in DevOps.

*Distributed DevOps*

This study focuses specifically on distributed DevOps, a relatively new software development technique that involves teams located in different locations. We have successfully addressed the challenges of trust and transparency that are often encountered in distributed settings.

*Payment Solution:* This framework also includes a payment solution, ensuring seamless and uninterrupted payment transactions after each process or iteration.
Table 8
Comparison of Proposed Framework with Related Work

<table>
<thead>
<tr>
<th>Reference</th>
<th>Paper</th>
<th>Blockchain-Based Study</th>
<th>Framework Presented</th>
<th>Distributed DevOps</th>
<th>Payment Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>[13]</td>
<td>Blockchain based framework for Software Development using DevOps</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[14]</td>
<td>Toward Effective and Efficient DevOps Using Blockchain</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[15]</td>
<td>Use of Blockchain Smart Contracts in Software Engineering: A Systematic Mapping</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[16]</td>
<td>Software Engineering Process and Methodology in Blockchain-Oriented Software Development: A Systematic Study</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>[17]</td>
<td>Blockchain in Agile Software Development</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>This Paper</td>
<td>Blockchain-Based Framework for Distributed DevOps</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

6. Discussion

In this section, the performance of our proposed framework is formally discussed. This framework is designed to facilitate the successful development of software projects with Distributed DevOps while maintaining decentralization, security, traceability, transparency, and coordination. The proposed framework addresses potential challenges and problems associated with Distributed DevOps, focusing on security, transparency, and traceability while enhancing coordination and collaboration.

Notably, our framework's performance results demonstrate that combining Blockchain technology with Distributed DevOps can resolve many problems that may lead to project failures, delays, and financial losses. The integration of Blockchain with Distributed DevOps offers several benefits, including:

- Increased project decentralization.
- Enhanced collaboration between development teams and operations teams in a secure and transparent environment.
- Elimination of payment-related issues when teams are distributed across different locations.
- Improved client satisfaction through increased transparency provided by Blockchain.
- Enhanced security for all stakeholders, enabling them to work in a secure environment.
Utilization of Blockchain technology as a useful tool for tracking work progress and maintaining records.

Decreased risk of conflicts and errors by implementing Blockchain with distributed DevOps, thereby facilitating smoother collaboration on complex projects.

Overall, our research work proves that implementing Blockchain technology with Distributed DevOps can increase security, transparency, and traceability in software development. Although previous studies have been conducted in this area, none have specifically addressed Distributed DevOps. Our study fills this gap by presenting an efficient approach to software development with CI/CD when teams are located worldwide.

However, there are certain limitations associated with this model. While the DevOps processes can be streamlined, the performance and speed of data uploading or loading may be slower due to the relatively slower nature of Blockchain compared to traditional databases. Another drawback is the implementation cost of this framework. Organizations operating on limited budgets may face constraints in implementing this framework due to cost and the complexity of the model.

In conclusion, the findings of this research highlight the significant potential of Blockchain, a major technological advancement, to revolutionize software development.

7. Conclusion and Future Work

The current distributed DevOps for software development needs more security, data protection, privacy, transparency, and trackability. These shortcomings are why the operations and development teams need help working together properly in a distributed environment. That is why this proposed model will allow the collaboration of development and operation teams more smoothly. The framework utilizes smart contracts and a decentralized architecture to enable secure and efficient collaboration among distributed teams in the development and operations of software systems. We discussed the benefits of using blockchain technology in DevOps, such as increased transparency, enhanced traceability, improved collaboration, and increased efficiency. We also presented the performance results, which demonstrated the effectiveness of the proposed framework in a real-world scenario.

Several directions for future work can be pursued based on the proposed framework. One potential area of research is to investigate this framework with other emerging technologies, such as Artificial Intelligence and Machine Learning, to enhance its capabilities and automation. Additionally, it will be interesting to study the real-world adoption and usage of the framework by different organizations and industries. Furthermore, there is also a scope to explore the regulatory compliance aspect of the framework and its adoption in different geographical regions. Inclusively, the proposed framework provides a promising solution for addressing the challenges of Distributed DevOps, and there is much potential for further research and development in this area.
Declarations

Funding
No Funding Available for this research.

Conflicts of Interest
The authors declare no competing interests.

Ethical Statement
Hereby, I Muhammad Shoaib Farooq consciously assure that for the manuscript “Exploring the Integration of Blockchain and Distributed DevOps for Secure, Transparent, and Traceable Software Development“ the following is fulfilled:

1. This material is the authors' own original work, which has not been previously published elsewhere.
2. The paper is not currently being considered for publication elsewhere.
3. The paper reflects the authors' own research and analysis in a truthful and complete manner.
4. The paper properly credits the meaningful contributions of co-authors.
5. The results are appropriately placed in the context of prior and existing research.
6. All sources used are properly disclosed (correct citation). Literally copying of text must be indicated as such by using quotation marks and giving proper reference.
7. All authors have been personally and actively involved in substantial work leading to the paper, and will take public responsibility for its content.
8. The violation of the Ethical Statement rules may result in severe consequences.

I agree with the above statements and declare that this submission follows the policies as outlined in the Guide for Authors and in the Ethical Statement.

Data Availability
The dataset available publicly on GitHub.

https://github.com/usmanali9525/Blockchain-Distributed-DevOps

Authors Contribution STATEMENT
Muhamad Shoaib Farooq and Usman Ali performed the measurements and analysis of the article. Junaid Nasir Qureshi and Muhammad Shoaib Farooq were involved in planning and supervised the research work. Usman Ali, Junaid Nasir Qureshi, and Muhammad Shoaib Farooq processed the experimental data, performed the analysis, drafted the manuscript and designed some of the figures. Some figures were drafted by Muhammad Shahraiz Durrani. Muhammad Shoaib Farooq and Usman Ali obtained the dataset and characterized it. Usman Ali and Muhammad Shoaib Farooq performed the experimental work and worked on different analysis tools and article repositories. Muhammad Shahraiz Durrani, Muhammad Shoaib Farooq and Junaid Nasir Qureshi aided in interpreting the results and worked on drafting the manuscript. All authors discussed the results and commented on the whole manuscript.

References


