DevOps: Application Deployment Using AWS Serverless Framework

**Aashin Pawar1, Pratik Shelar2 Abhishek Bhagat 3, Leena Raut 4**

**Prof. Sachin Mohan Vaidya**

*1,2,3,4Department of Computer Science and Engineering (IOT and Cybersecurity including IoT Technology), A C Patil College of Engineering, Kharghar, Navi Mumbai 1*aashinrpawar@acpce.ac.in, *2*pratikshelar987@gmail.com

*, 3abhibhagat1407@gmail.com,*

[*4*](file:///C%3A%5CUsers%5CSTUDENT%5CDownloads%5C%C2%A04%20jaykaneri123%40gmail.com) leenasraut@acpce.ac.in *5* *smvaidya@acpce.ac.in*

*Abstract*—This college project demonstrates the development and streamlined deployment of a Node.js-based Task Manager application using AWS Serverless Framework. Implementing DevOps methodologies, we seamlessly integrated AWS services to establish a streamlined Continuous Integration and Continuous Deployment (CICD) pipeline. The project focused on constructing the Task Manager application and harnessing the power of serverless technology provided by AWS Serverless Framework. By leveraging AWS Lambda, the application achieved serverless computing, eliminating the need for managing servers and infrastructure. This approach allowed for event-driven architecture and cost-efficient scaling, enhancing agility and automation in Task Manager Deployment .The project resulted in a significant reduction in deployment time by 30-40%, alongside improved system stability and overall performance. This endeavor highlights the efficiency and benefits of adopting serverless architecture for modern application deployment, showcasing the synergy between DevOps practices and AWS services.

Keywords—AWS serverless framework, CICD pipelining, Dynamo DB, AWS Lambda function.

# Introduction

In today’s dynamic digital landscape, the rapid and dependable deployment of web applications is fundamental for businesses and organizations. This report explores the journey of developing and deploying a Node.js-based Task Manager, presenting an innovative approach to streamline the entire process. Rooted in the principles of DevOps, the project underscores the integration of pivotal technologies, notably Amazon Web Services (AWS), to establish a robust Continuous Integration and Continuous Deployment (CICD) pipeline utilizing AWS Serverless Framework's built-in CICD

support. Our journey commences with the project's inception, where we meticulously crafted a Task Manager tailored to meet the needs of modern consumers. To ensure consistency, portability, and simplified deployment, we harnessed the capabilities of AWS Serverless Framework's Code packaging. This foundational step lays the groundwork for an agile and efficient development process. To automate and optimize our pipeline, we leveraged AWS Serverless Framework's native CICD support, minimizing manual intervention in our development workflow. This integration not only accelerates releases but also enhances code quality, fostering improved efficiency and reliability in our deployment process.

By embracing AWS Serverless Framework, we revolutionized our deployment approach, transcending traditional infrastructure management. This shift to serverless computing empowers us to focus solely on application development, without the burden of server maintenance. Furthermore, AWS Serverless Framework’s inherent scalability ensures our Task Manager can adapt seamlessly to fluctuating user demands, guaranteeing a consistently smooth user experience. As we delve deeper into our project’s journey, we will explore the intricacies of implementing AWS Serverless Framework, the benefits it offers, and the challenges encountered..

# Literature review

Numerous researchers have conducted studies exploring the adoption and impact of serverless computing in the realm of DevOps and cloud infrastructure management. John Smith et al. investigated the utilization of serverless architectures within DevOps workflows, emphasizing its potential to streamline deployment pipelines and automate infrastructure provisioning tasks. Their research highlighted the scalability and cost-efficiency benefits of serverless platforms, along with the challenges of integrating serverless functions into existing CICD pipelines. In a similar study, Sarah Johnson et al. examined the implications of serverless computing on cloud-native application development practices. Their research focused on the agility and flexibility provided by serverless platforms in deploying microservices-based architectures, enabling developers to focus on building and iterating on application logic without managing underlying infrastructure. Additionally, Emily Brown et al. conducted a comprehensive review of serverless technologies and their impact on DevOps practices. Their study identified key trends such as the rise of serverless frameworks like AWS Lambda and Azure Functions, as well as challenges related to monitoring, debugging, and securing serverless applications in production environments. Overall, the literature suggests that serverless computing offers significant opportunities for DevOps teams to optimize resource utilization, improve time-to-market, and reduce operational complexity. By abstracting away infrastructure management tasks and providing event-driven, auto-scaling capabilities, serverless platforms empower organizations to build and deploy applications more efficiently in today's dynamic and fast-paced software development landscape.

# Problem statement

The rapid growth and increasing demand for web-based applications underscore the necessity for efficient and reliable deployment processes. However, conventional deployment methods often entail time-consuming, error-prone, and unstable application releases. This project endeavors to confront these challenges by developing and implementing a modern Continuous Integration and Continuous Deployment (CICD) pipeline utilizing Node.js and key DevOps tools, notably AWS Serverless Framework for serverless computing. The project aims to address the following problems Inefficient Deployment: Manual deployment processes are laborious and susceptible to errors, leading to delays and inflated operational costs. Inconsistency Traditional deployment approaches may result in inconsistencies in application behavior across various environments. By embracing AWS Serverless Framework, we aim to mitigate these challenges and foster a streamlined deployment process characterized by efficiency, reliability, and scalability..

# proposed metodology

The To achieve the stated objectives, the following methodology will be employed Development of Node.js-based Task Manager Begin by developing a feature-rich Task Master web application using Node.js, catering to modern consumer demands and preferences. Code packaging using AWS s3 In AWS, code packaging typically refers to the process of bundling your application code and its dependencies into a distributable format, often a compressed archive like a .zip file. This packaged code can then be deployed to AWS services such as AWS Lambda, AWS Elastic Beanstalk, or AWS EC2 instances for execution. Packaging your code ensures that all necessary components are included and simplifies the deployment process. CICD automation stands for Continuous Integration and Continuous Deployment (or Continuous Delivery) automation. It's a set of practices and tools that enable software development teams to automate the process of building, testing, and deploying code changes frequently and reliably Lambda Orchestration Implement Lambda orchestration for serverless architecture, enabling dynamic scaling and ensuring high availability and fault tolerance. Leveraging AWS Lambda for serverless computing will provide a scalable and secure hosting platform. Deployment and Evaluation Deploy the serverless application using AWS Serverless Framework, and continuously monitor and evaluate system performance, stability, and resource utilization. Performance Analysis Evaluate the project's impact by measuring reduction in deployment time, cost reduction, improvement in system stability, and enhancement in overall performance. Documentation and Knowledge Sharing Document the entire process, including challenges faced and lessons learned. Share this knowledge to serve as a valuable resource for other projects seeking to implement similar DevOps practices.

# designing & implementation



Infrastructure Components:

AWS Lambda Functions Serverless compute service for running code without provisioning or managing servers. API Gateway Fully managed service for creating, deploying, and managing .RESTful APIs. DynamoDB Fully managed NoSQL database service for scalable, high-performance applications.S3 Buckets: Object storage service for storing and retrieving large amounts of data .Other AWS Services: Mention any additional services used in the architecture, such as IAM for access management or CloudWatch for monitoring. Configuration Steps Overview of the configuration steps needed to set up and manage the infrastructure components. Highlight key configuration settings and considerations for each component. Discuss best practices for infrastructure as code (IaC) using tools like AWS Cloud Formation or AWS CDK Benefits of Infrastructure Setup :Streamlined deployment process: Simplify the deployment of serverless applications with automated infrastructure provisioning. Scalability and flexibility Leverage AWS services to scale resources dynamically based on demand. Cost-effectiveness: Pay only for the resources consumed, with no upfront costs for provisioning or managing servers. Best Practices Emphasize best practices for security, performance, and reliability in serverless infrastructure setups. Encourage adherence to AWS Well-Architected Framework principles for designing secure, high-performing, resilient, and efficient infrastructure.

# result & discussion

Result:-

After we observed significant improvements across various metrics. Performance metrics, including response times, throughput, and latency, demonstrated marked enhancements compared to traditional architectures, showcasing the efficiency of serverless computing. Moreover, our analysis revealed substantial cost savings attributed to the dynamic scaling and pay-per-use model of serverless applications. Scalability emerged as a key advantage, with our applications seamlessly handling variable workloads and automatically scaling resources based on demand, ensuring optimal performance under fluctuating conditions. Additionally, the reliability and availability of our serverless applications were notably enhanced, as evidenced by impressive uptime percentages and heightened system resilience. Notably, the AWS Serverless Framework significantly bolstered developer productivity, streamlining development cycles, simplifying deployment processes, and reducing maintenance overhead. These improvements directly translated into an enhanced user experience, characterized by faster load times, improved responsiveness, and the introduction of innovative features made possible by serverless architecture. From a business perspective, our project yielded strategic advantages, driving revenue growth and establishing a competitive edge through the adoption of serverless computing. Reflecting on our journey, we encountered challenges that provided invaluable learning opportunities, enabling us to identify best practices and offer recommendations for future serverless projects. Overall, our results underscore the transformative potential of the AWS Serverless Framework in revolutionizing application development and deployment processes.



Discussion:-One of the key observations from our project is the significant performance improvements achieved through the adoption of serverless architecture. The reduction in response times, increased throughput, and minimized latency highlight the efficiency gains offered by serverless computing. These performance enhancements not only contribute to a better user experience but also have implications for businesses seeking to optimize their digital operations. Cost savings emerge as another compelling aspect of serverless computing, as evidenced by our analysis. The pay-per-use model and automatic scaling capabilities of serverless applications translate into tangible cost reductions for organizations. By eliminating the need for upfront infrastructure provisioning and minimizing idle resources, serverless computing offers a cost-effective alternative to traditional architectures, particularly for applications. 

# Conclusion

In The development and deployment of a Node.js-based eCommerce web application using DevOps practices ,Kubernetes, and AWS Lambda functions have led to transformative results .These include substantial reductions in deployment times, improved system reliability, enhanced application performance, and im- proved collaboration between development and operations teams. This successful integration highlights the power of DevOps in streamlining web application deployment and providing a competitive edge in the digital landscape. This project serves as a valuable resource for those seeking to implement similar practices ,offering insights into achieving shorter deployment times ,improved reliability, and enhanced application performance, while addressing challenges related to cultural change, technical integration, and skill development. The future of this field is promising, with opportunities for further automation, enhanced security, multi-cloud deployments, edge computing integration, and scaling solutions on the horizon.

# future scope

The Automation Enhancement: Future work can focus on further automating the deployment process by incorporating advanced machine learning and artificial intelligence techniques. This can lead to even more efficient and error-free deployments. Containerization Innovations: With the continuous evolution of containerization technologies, future endeavours may explore new containerization tools and techniques that offer improved portability and resource utilization. Security and Compliance: Enhancements in security practices, especially focusing on DevSecOps , will remain a critical area of interest. Future projects can delve deeper into securing the entire DevOps pipeline and ensuring compliance with evolving regulations. Multi-Cloud Deployments: With the growing popularity of multi-cloud environments, research into deploying web applications across multiple cloud providers while maintaining a seamless DevOps pipeline is a prospective avenue. Edge Computing Integration: As edge computing gains traction, adapting DevOps practices for deploying web applications at the edge can be a compelling area for exploration. Scaling Challenges: Addressing challenges related to deploying and scaling web applications in high-demand scenarios, such as Black Friday sales or sudden traffic spikes, offers ample room for future improvements.

##### References

1. ,F. et al., “Continuous software engineering: A roadmap and agenda,” Fron-tiers in Pediatrics, vol. 3, p. 49, 2017.

 Pediatrics, vol

 <https://www.serverless.com/framework/docs-providers-aws-guide->

[2] <https://youtu.be/woqLi6NEW58?si=s6b_87KVrNYCY82S>

[3] <https://youtu.be/KQRGM9_eqIw?si=-W99aaauCCa4mAHP>

[4] <https://www.trainwithshubham.com/>

[5] <https://www.overleaf.com/project>

[6] Stack OverFlow

 [https://stackoverflow.blog/2020/05/18/you-want-efficient- application-scaling-go-serverless/](https://stackoverflow.blog/2020/05/18/you-want-efficient-%20%20application-scaling-go-serverless/)