ISSN: 2455 - 1597

ijergs
Open Access

International Journal of Engineering Research and Generic Science (IJERGS)
Available online at: https://www.ijergs.in

Volume - 11, Issue - 3, May – June - 2025, Page No. 17 - 24

Cloud Deployment of MERN Apps Using AWS, Heroku, or Vercel

[1]Sagar Pradhan, Assistant Professor, Department of Computer Science, Arya College of Engineering, Jaipur.

[2]Nishit Paliwal, Research Scholar, Department of Computer Science, Arya College of Engineering, Jaipur.

[3]Priyanshu Sinha, Research Scholar, Department of Computer Science, Arya College of Engineering, Jaipur.

Abstract

In today's rapidly evolving digital landscape, the need for efficient, reliable, and scalable web application deployment strategies has become paramount. Full-stack development using the MERN stack—MongoDB, Express.js, React.js, and Node.js—has emerged as a dominant solution for building robust web applications entirely in JavaScript. However, beyond development, the successful delivery of these applications to end users hinges on effective deployment practices that ensure scalability, high availability, performance optimization, and maintainability.

This research paper provides a comprehensive examination of cloud deployment strategies for MERN stack applications, focusing on three widely adopted platforms: Amazon Web Services (AWS), Heroku, and Vercel. Each of these platforms caters to different developer needs and business priorities. AWS offers granular control over infrastructure and is ideal for large-scale enterprise applications, whereas Heroku simplifies deployment through its Platform-as-a-Service (PaaS) model, making it accessible to startups and individual developers. Vercel, known for its seamless frontend deployment and tight integration with frameworks like React, presents a developer-friendly solution for full-stack JavaScript applications with optimized global delivery.

The paper discusses the step-by-step deployment processes for each platform, explores integration with services like MongoDB Atlas, and highlights the differences in setup complexity, performance, scalability, and cost-effectiveness. Real-world examples and case studies are used to illustrate practical applications and outcomes of MERN stack deployment on these platforms. Furthermore, it analyzes the opportunities and benefits, such as CI/CD support, auto-scaling, and global CDN availability, alongside the challenges, including configuration complexity, vendor lock-in, and platform limitations. Through detailed comparisons and technical insights, this paper serves as a valuable guide for developers, DevOps engineers, and businesses seeking to choose the most suitable cloud deployment platform for their MERN stack applications. The goal is to help readers make informed decisions based on their project requirements, team expertise, and long-term scalability goals in an increasingly cloud-centric development environment.

Keywords: MERN Stack, Cloud Deployment, AWS, Heroku, DevOps, Node.js, MongoDB, React.js, Express.js, CI/CD, PaaS, Serverless, Infrastructure-as-a-Service (IaaS).

Introduction

In modern web development, building a high-performing and reliable web application is only the beginning. Once development concludes, the ability to deploy and maintain the application in a scalable, secure, and efficient manner becomes crucial. The MERN stack—MongoDB, Express.js, React.js, and Node.js—has become a popular technology stack for full-stack JavaScript development due to its flexibility and end-to-end JavaScript integration. The deployment phase ensures that applications built using MERN are delivered to users in a seamless, optimized, and maintainable way.

This section delves into the components of the MERN stack, offers an overview of popular cloud platforms for deployment, explains deployment workflows, and explores the role of Continuous Integration and Continuous Deployment (CI/CD) in modern cloud environments.

Overview of the MERN Stack

The MERN stack allows developers to use a single programming language—JavaScript—across all layers of an application, significantly simplifying development and improving productivity.

- **MongoDB**: A powerful NoSQL database that stores data in flexible, JSON-like documents. It supports high availability, replication, and scalability, making it ideal for cloud-hosted applications. MongoDB Atlas, the cloud-hosted version of MongoDB, is commonly used for deploying cloud-based MERN apps.
- **Express.js**: A fast and minimalist web application framework for Node.js, used for building server-side logic, RESTful APIs, and middleware. Express simplifies routing, request handling, and session management.
- **React.js**: A popular frontend library developed by Meta (Facebook) used for building dynamic, component-based user interfaces. React improves user experience through efficient rendering and client-side routing.
- **Node.js**: A JavaScript runtime environment that enables server-side programming using JavaScript. It provides a non-blocking, event-driven architecture that supports high concurrency and scalability.

Cloud Platforms Overview

The cloud provides on-demand computing services that help deploy applications quickly without needing to maintain physical infrastructure. The platforms discussed below provide distinct deployment workflows and serve different developer needs.

Amazon Web Services (AWS): AWS is the most comprehensive cloud platform, offering services such as **Elastic Compute Cloud (EC2)** for virtual machines, **Simple Storage Service (S3)** for file storage, **Lambda** for serverless functions, and **Relational Database Service (RDS)** or **DocumentDB** for database hosting. AWS provides fine-grained control, scalability, and reliability but requires more infrastructure management skills.

Heroku: A Platform-as-a-Service (PaaS) that enables developers to deploy applications directly from the command line or GitHub with minimal configuration. Heroku abstracts infrastructure complexities and supports various buildpacks for Node.js and other languages. It also offers easy integration with third-party services like mLab or MongoDB Atlas for database needs.

Vercel: Designed primarily for frontend frameworks like React and Next.js, Vercel simplifies the deployment process with an intuitive interface, Git integration, and support for serverless functions. While it's ideal for frontend deployment, Vercel can also serve full-stack applications with API routes or through integration with other backend services such as Heroku or Railway.

Deployment Workflow

Deploying a MERN stack application requires configuring both frontend and backend components, integrating a database, and ensuring routing, caching, and monitoring are properly set up. Below is an in-depth look at the deployment steps for each platform:

AWS Deployment Steps

1. **Provision EC2 Instance**: Launch a virtual server using Amazon EC2. Choose an appropriate instance type (e.g., t2.micro for testing or t3.medium for production).
2. **Install Required Packages**: SSH into the instance and install Node.js, npm, Git, and optionally MongoDB if using local storage. Alternatively, connect to MongoDB Atlas.
3. **Clone Project Repository**: Use Git to clone the MERN stack project from a version control system like GitHub.
4. **Install Dependencies**: Navigate to the backend and frontend directories to run npm install.
5. **Configure Reverse Proxy**: Use Nginx or Apache to proxy requests from the domain to the Node.js server.
6. **Use PM2**: Deploy and monitor the backend server using PM2, a Node.js process manager.
7. **Set Environment Variables**: Securely configure application secrets and database URIs.

Heroku Deployment Steps

1. **Initialize Heroku Project**: Create an app on Heroku via the CLI or dashboard.
2. **Add Buildpacks**: Include Node.js buildpack for backend and optionally a static buildpack for the React frontend.
3. **Push Code**: Use Git to push code to the Heroku remote repository.
4. **Configure Environment Variables**: Use the Heroku dashboard or CLI to add MONGO_URI, PORT, and other environment settings.
5. **Database Integration**: Use add-ons like **Heroku Postgres** or external services like **MongoDB Atlas**.
6. **Deployment Monitoring**: Use Heroku logs and dashboard tools to monitor performance and crashes.

Vercel Deployment Steps

1. **Connect Git Repository**: Log into Vercel and import the GitHub/GitLab/Bitbucket repository.
2. **Configure Build Settings**: Define build commands and output directories (npm run build and /build).
3. **Frontend Deployment**: Vercel automatically builds and deploys the frontend React application.
4. **Backend Integration**: Backend APIs can be deployed as Vercel serverless functions or hosted on platforms like Heroku, Railway, or AWS Lambda.
5. **Custom Domains & SSL**: Add domains through Vercel's dashboard with free HTTPS/SSL support.

CI/CD Integration

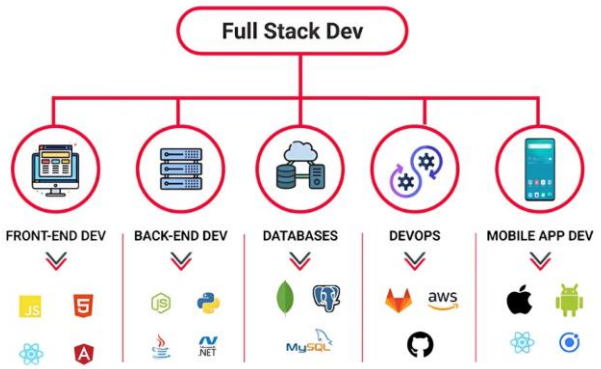

Figure 1

Continuous Integration and Continuous Deployment (CI/CD) are essential for automating testing, building, and deploying applications as soon as code changes are pushed to version control systems.

AWS CI/CD:

- CodePipeline: Integrates with GitHub and CodeBuild to automate build and deployment.
- GitHub Actions: Can be used with AWS CLI to deploy to EC2 or Lambda functions.
- Elastic Beanstalk: Offers a fully managed deployment environment with integrated CI/CD options.

Heroku CI/CD:

- GitHub Integration: Every push to a branch can automatically trigger a new deployment.
- Heroku Pipelines: Helps manage staging, review, and production workflows.
- CircleCI / TravisCI: Third-party tools can be connected to Heroku for advanced workflows.

Vercel CI/CD:

- Auto Deploy on Git Push: Pushes to the main branch trigger automatic deployment.
- Preview Deployments: Every pull request generates a live preview URL for collaboration.
- GitHub Actions: Can also extend Vercel workflows for backend services.

Recent examples

A recent project undertaken by a group of undergraduate students at Stanford University during a national-level hackathon showcased an innovative approach to deploying a full-stack MERN-based educational portal. The goal of the project was to build a responsive and collaborative platform where students and educators could interact in real-time through discussions, resource sharing, and quizzes. To achieve optimal performance and maintainability, the team strategically leveraged multiple cloud services.

The frontend of the application, developed using React.js, was deployed via Vercel, a platform known for its seamless integration with GitHub and automatic deployment capabilities. Vercel enabled rapid iteration and real-time feedback, thanks to its support for continuous deployment and built-in content delivery network (CDN), which significantly improved load times and user experience.

For the backend, the team provisioned a virtual server using Amazon EC2. This choice provided them with fine-grained control over the runtime environment and the flexibility to configure middleware, handle custom API routes, and ensure secure authentication. Backend services were built with Node.js and Express.js, and the application communicated with a cloud-hosted MongoDB database via MongoDB Atlas, which offered high availability, automated backups, and robust performance monitoring tools.

This deployment strategy allowed the team to follow a separation of concerns model—isolating the frontend and backend environments to independently optimize performance and scalability. Moreover, by connecting their GitHub repository with both Vercel and AWS CodePipeline, they established a lightweight but effective CI/CD workflow. This integration ensured that every push to the main branch triggered automatic builds and deployments, minimizing downtime and accelerating development cycles.

The use of Vercel for the frontend and AWS EC2 for the backend proved to be a powerful combination, reducing the burden of infrastructure maintenance while allowing the developers to focus on core functionality. Through this deployment model, the team demonstrated how modern cloud services can be effectively integrated to create scalable, maintainable, and production-ready MERN stack applications even within the constrained timeline of a hackathon.

Opportunities and Benefits

Deploying MERN applications using cloud platforms like AWS, Heroku, and Vercel opens numerous avenues for innovation and operational efficiency. Below are key opportunities and benefits that highlight the value of cloud deployment in modern web application architecture:

Scalability: All three platforms support both vertical and horizontal scaling. AWS offers elastic load balancing and auto-scaling groups, allowing applications to scale seamlessly with user traffic. Heroku simplifies horizontal scaling through dynos, which can be scaled up or down with simple CLI commands. Vercel provides automatic front-end scaling with global content delivery network (CDN) capabilities, enabling developers to focus more on building features than worrying about traffic surges.

Speed and Agility: With integrated CI/CD pipelines, code changes are automatically tested and deployed, minimizing downtime and human error. GitHub Actions on AWS and Heroku Pipelines streamline deployments with automation. Vercel, being focused on front-end applications, ensures lightning-fast deployment with Git push integration. These automation tools drastically reduce development-to-deployment time, fostering agile development methodologies.

Global Reach and CDN Integration: Cloud platforms integrate with global CDNs to reduce latency and improve user experience. AWS CloudFront, Heroku CDN, and Vercel's edge network ensure content is served from the nearest geographical location to the user, significantly improving load times. This is especially beneficial for international applications with a global user base.

Separation of Concerns: Using hybrid deployment (e.g., frontend on Vercel and backend on AWS), teams can maintain clear separation of responsibilities. Front-end and back-end teams can work and deploy independently. This modularity increases maintainability, improves fault isolation, and enhances system resilience.

Developer Productivity and Collaboration: Tools like AWS CloudWatch, Heroku Dashboard, and Vercel Analytics empower developers with real-time logging, monitoring, and debugging tools. GUI-based interfaces help in managing applications without deep terminal usage, while CLI tools offer power and flexibility to advanced users. Vercel also allows preview deployments for every pull request, facilitating better collaboration among developers and stakeholders.

Flexible Pricing Models: All three platforms offer free tiers, which are ideal for small projects, startups, or students. AWS provides a free usage tier with limited hours for services like EC2 and Lambda. Heroku's hobby dynos and Vercel's hobby plans allow experimentation at no cost. As needs grow, these platforms offer transparent, pay-as-you-go models, allowing startups to scale with their budget.

Integration with Modern DevOps Tools: These platforms are compatible with a wide array of third-party tools. For example, Heroku integrates with Sentry, Papertrail, and New Relic for monitoring and error tracking. AWS supports

infrastructure as code (IaC) tools like Terraform and AWS CloudFormation. Vercel integrates seamlessly with GitHub, GitLab, Bitbucket, and more, making it ideal for fast-paced development cycles.

High Availability and Redundancy: With built-in load balancing and multiple availability zones (especially in AWS), applications can remain online even if one zone goes down. Heroku apps can be deployed to multiple regions, and Vercel's edge network ensures high availability by serving static content globally.

Security and Compliance: AWS, Heroku, and Vercel all offer enterprise-grade security features. AWS leads with granular access control through IAM, VPC configurations, and encryption options. Heroku simplifies security through SSL certificates and private spaces. Vercel offers secure environment variable handling and encrypted deployments.

Eco-System and Community Support: Each platform has a vibrant developer community and extensive documentation. Tutorials, forums, and community plug-ins accelerate learning and troubleshooting. Developers benefit from shared knowledge, open-source tools, and frequent updates that improve usability and performance.

Challenges

While deploying MERN applications to the cloud offers significant advantages, several challenges can arise during and after deployment. Understanding these challenges is essential for developers and organizations aiming to ensure robust, maintainable, and scalable applications.

- **Learning Curve**: Cloud platforms like AWS offer immense flexibility and power, but mastering them requires significant technical knowledge. Developers must become proficient in configuring networking (VPC, subnets), identity and access management (IAM roles and policies), DevOps pipelines, containerization (using Docker), and infrastructure-as-code (e.g., CloudFormation or Terraform). For beginners, navigating AWS services like EC2, Route 53, or CloudWatch can be overwhelming without adequate training or documentation.
- **Configuration Complexity**: Setting up production-ready environments involves more than simply deploying code. SSL certificate installation for HTTPS, DNS configuration for custom domains, reverse proxy setup (e.g., using Nginx), and CI/CD pipeline automation all add layers of complexity. Any misconfiguration can lead to broken deployments, downtime, or security vulnerabilities. On AWS, manual setup or script-based automation often becomes necessary, requiring detailed planning and testing.
- **Vendor Lock-In**: Heavy reliance on a specific cloud provider's proprietary services (e.g., AWS Lambda, Heroku Add-ons, or Vercel's serverless functions) can result in vendor lock-in. Migrating a mature application to a different provider becomes challenging due to differences in APIs, deployment environments, and architecture. This limits portability and flexibility, especially if business needs or costs change in the future.
- **Resource Limits**: Although all three platforms offer generous free tiers, they come with limitations. Heroku's free dynos sleep after inactivity, which can delay response times. AWS's free tier has usage caps (e.g., 750 hours of EC2 usage), and Vercel imposes function execution timeouts and limits on storage and bandwidth. Exceeding these limits incurs costs, which can quickly escalate for high-traffic or resource-intensive apps if not carefully monitored.
- **Monitoring and Logging**: Proper performance tracking, error logging, and resource monitoring are essential for maintaining application health. While AWS offers powerful tools like CloudWatch and X-Ray, setting them up

involves learning additional services. Heroku and Vercel provide basic dashboards, but deeper insights require third-party integrations like Sentry, Datadog, or Loggly. Fragmentation of monitoring tools across frontend and backend also adds complexity to debugging.

- **Integration Limitations**: Platforms like Vercel are optimized for frontend hosting and serverless functions, but they may fall short when full backend customization is needed. Running persistent server processes, complex API handling, background jobs, or WebSocket connections may require external hosting solutions like AWS or Heroku. This forces developers to split their application infrastructure, complicating deployment, monitoring, and version control.
- **Security Concerns**: Misconfigured cloud environments can expose sensitive data or APIs. On AWS, improper IAM policies, open security groups, or unencrypted storage can become attack vectors. Heroku and Vercel abstract much of the infrastructure, but developers must still manage environment variables, secure HTTP headers, and data access rules. Security audits and regular updates become essential in cloud deployments to mitigate potential risks.
- **CI/CD Reliability**: While Vercel and Heroku support automatic deployments via Git, pipeline failures due to build errors, dependency mismatches, or incorrect environment variables can disrupt workflow. On AWS, setting up CodePipeline or integrating GitHub Actions requires careful configuration to handle stages like testing, building, and deployment. Debugging failed deployments without clear logs or notifications can consume time and resources.

References

1. Hiorthøy, M. (2023). Analyzing and Benchmarking the Performance of Different Cloud Services for Agile App Deployment (Master's thesis, Oslomet-storbyuniversitetet). Express.js Guide - https://expressjs.com/
2. Thokala, V. S. (2023). Scalable Cloud Deployment and Automation for E-Commerce Platforms Using AWS, Heroku, and Ruby on Rails. Int. J. Adv. Res. Sci. Commun. Technol, 349-362. Node.js - https://nodejs.org/en/docs/
3. Bhandari, A., Harisha, A., Rishav, K., Shifana, M., & Sameer, J. (2024, November). WebTrek Learner: AI Integrated Cloud Based Learning Platform. In 2024 International Conference on Computing, Semiconductor, Mechatronics, Intelligent Systems and Communications (COSMIC) (pp. 108-113). IEEE. AWS CodePipeline - https://docs.aws.amazon.com/codepipeline/
4. Lochana, D. (2025). Design a Freelance Application Evaluating a Full Stack for Scalable and User-Centric Development.
5. Alam, U., Aleem, S., & Jamal, T. (2024). Empowering Students: Building an Integrated Application for Enhanced Productivity, Efficiency and Creativity. International Journal of Scientific Research in Network Security and Communication, 12(5), 6-26. Vercel Docs - https://vercel.com/docs
6. Juho, M. (2025). Serverless-sovellusten automatisoitu julkaisu. GitHub Actions - https://docs.github.com/en/actions
7. Jain, A. (2022). Hugo in Action: Static sites and dynamic Jamstack apps. Simon and Schuster.
8. Doyle, P. Intelligent Instrument Interface. MongoDB Atlas - https://www.mongodb.com/atlas
9. Sieradzki, J. R. Interactive streaming videos for educational use.
10. Contreras, D. H., & Soldado, R. M. Bot de Telegram para la gestión de bandas.

11. Bortolini, F. A. (2024). Sistema de Supervisão e Aquisição de Dados para ETEs e ETAs em Condomínios.
12. Thokala, V.S., 2023. Scalable Cloud Deployment and Automation for E-Commerce Platforms Using AWS, Heroku, and Ruby on Rails. Int. J. Adv. Res. Sci. Commun. Technol, pp.349-362.
13. Hiorthøy, M., 2023. Analyzing and Benchmarking the Performance of Different Cloud Services for Agile App Deployment (Master's thesis, Oslomet-storbyuniversitetet).
14. Juho, Mäkitalo. "Serverless-sovellusten automatisoitu julkaisu." (2025).
15. Contreras, Daniel Haro, and Rosana Montes Soldado. "Bot de Telegram para la gestión de bandas."
16. Bortolini, Felipe Augusto. "Sistema de Supervisão e Aquisição de Dados para ETEs e ETAs em Condomínios." (2024).
17. Hiorthøy M. Analyzing and Benchmarking the Performance of Different Cloud Services for Agile App Deployment (Master's thesis, Oslomet-storbyuniversitetet).
18. Bhandari A, Harisha A, Rishav K, Shifana M, Sameer J. WebTrek Learner: AI Integrated Cloud Based Learning Platform. In2024 International Conference on Computing, Semiconductor, Mechatronics, Intelligent Systems and Communications (COSMIC) 2024 Nov 22 (pp. 108-113). IEEE.
19. Contreras DH, Soldado RM. Bot de Telegram para la gestión de bandas.
20. Sieradzki, J.R., Interactive streaming videos for educational use.