



# AUTOMATED DEVOPS PIPELINE FOR DEPLOYING MERN STACK WEB APPLICATION IN KUBERNETES ON AWS

<sup>1</sup>Dr. Mahalakshmi Shetty, <sup>2</sup>Satya Tulasi Jalandhar C H, <sup>3</sup>Sahitya Nandaluru, <sup>4</sup>S Rekha,  
<sup>5</sup>Ranjithkumar T R

<sup>1</sup>Associate Professor, <sup>2</sup>Student, <sup>3</sup>Student, <sup>4</sup>Student, <sup>5</sup>Student  
Computer Science and Engineering,

R. L. Jalappa Institute of Technology, Doddaballapur, Bengaluru, India.

**Abstract:** This paper presents an Automated DevOps Pipeline that is specifically designed for MERN-Stack web applications, catering to the ever-changing needs of cloud computing. The pipeline focuses on AWS and incorporates Jenkins for Continuous Integration, ArgoCD for Continuous Deployment on Kubernetes, and Terraform for scalable infrastructure provisioning. The structured methodology includes source code change detection, CI with Jenkins, dynamic Kubernetes resource creation with Terraform, containerization, CD using ArgoCD, and final deployment on AWS. By utilizing Jenkins, ArgoCD, Terraform, Kubernetes, Docker, and AWS, the project ensures zero downtime, efficient resource utilization, and seamless CI/CD integration. The results showcase improved scalability and a comprehensive solution for contemporary challenges in application deployment. The project concludes with a successful implementation, emphasizing efficiency, scalability, and zero downtime in the deployment process. Future work includes exploring advanced Terraform modules, integrating monitoring tools, expanding pipeline support, and enhancing security measures. Positioned as a robust and forward-looking solution, this project demonstrates the adaptability and efficiency of DevOps methodologies in addressing modern challenges in cloud-based application deployment.

**Index Terms – DevOps, CICD, Jenkins Pipeline, MERN stack, AWS, Kubernetes, ArgoCD, Deployment**

## I. INTRODUCTION

This paper describes an Automated DevOps Pipeline created specifically for MERN-Stack web applications to meet the ever-changing needs of cloud computing. The pipeline is focused on AWS and includes Jenkins for Continuous Integration, ArgoCD for Continuous Deployment on Kubernetes, and Terraform for scalable infrastructure provisioning. The organized methodology consists of source code change detection, CI with Jenkins, dynamic Kubernetes resource creation with Terraform, containerization, CD with ArgoCD, and final AWS deployment. The project uses Jenkins, ArgoCD, Terraform, Kubernetes, Docker, and AWS to achieve minimal downtime, effective resource use, and seamless CI/CD integration. The results demonstrate enhanced scalability and a comprehensive answer to today's application deployment difficulties. The project ends with a successful implementation that emphasizes efficiency, scalability, and zero downtime in the deployment process.

Future development will involve investigating advanced Terraform modules, integrating monitoring tools, expanding pipeline support, and improving security mechanisms. This project, positioned as a strong and forward-thinking solution, exhibits DevOps approaches' adaptability and efficiency in solving modern issues in cloud-based application deployment.

## II. OBJECTIVES

DevOps strives to improve software development by bringing developers and operations teams together, eliminating barriers, and automating tasks. The purpose is to streamline software delivery and improve efficiency.

DevOps promotes shared accountability, continuous improvement, and automation to provide high-quality software, increase operational efficiency, and generate commercial value.

DevOps promotes experimentation, innovation, and continuous learning, enabling teams to rapidly adapt to changing customer needs and market conditions, resulting in faster and more effective value delivery. Future development will involve investigating advanced Terraform modules, integrating monitoring tools, expanding pipeline support, and improving security mechanisms. This

project, positioned as a strong and forward-thinking solution, exhibits DevOps approaches' adaptability and efficiency in solving modern issues in cloud-based application deployment.

### III. EXISTING SYSTEM

The system presents a whole automated workflow, beginning with detecting changes in the Java-based web application source code, then establishing new resources in the Kubernetes cluster to host this new version, and lastly deploying the containerized application to AWS. The solution adheres to DevOps best practices and uses Jenkins for continuous integration. The uniqueness here is that we used Ansible for Continuous Deployment, which increased scalability and overall simplicity of use. Despite combining six different technologies and requiring relatively few processing resources, the solution provides zero downtime and demonstrates speed.

### IV. PROPOSED SYSTEM

To deploy a MERN Stack web application with multi-node support, several steps are involved to ensure efficiency and reliability. Firstly, incorporating Ansible modules facilitates advanced infrastructure configuration, streamlining the setup process and enabling smoother deployments. Additionally, integrating supplementary monitoring and logging tools enhances observability, providing valuable insights into the application's performance and aiding in troubleshooting. Expanding the pipeline to support infrastructure provisioning further optimizes the deployment process, allowing for seamless scaling as needed. Moreover, incorporating security measures and compliance checks within the pipeline ensures that the deployment adheres to best practices and regulatory requirements, bolstering the application's resilience against potential threats. Overall, these steps collectively contribute to a robust deployment process, facilitating the efficient delivery of a high-quality MERN Stack web application with multi-node support.

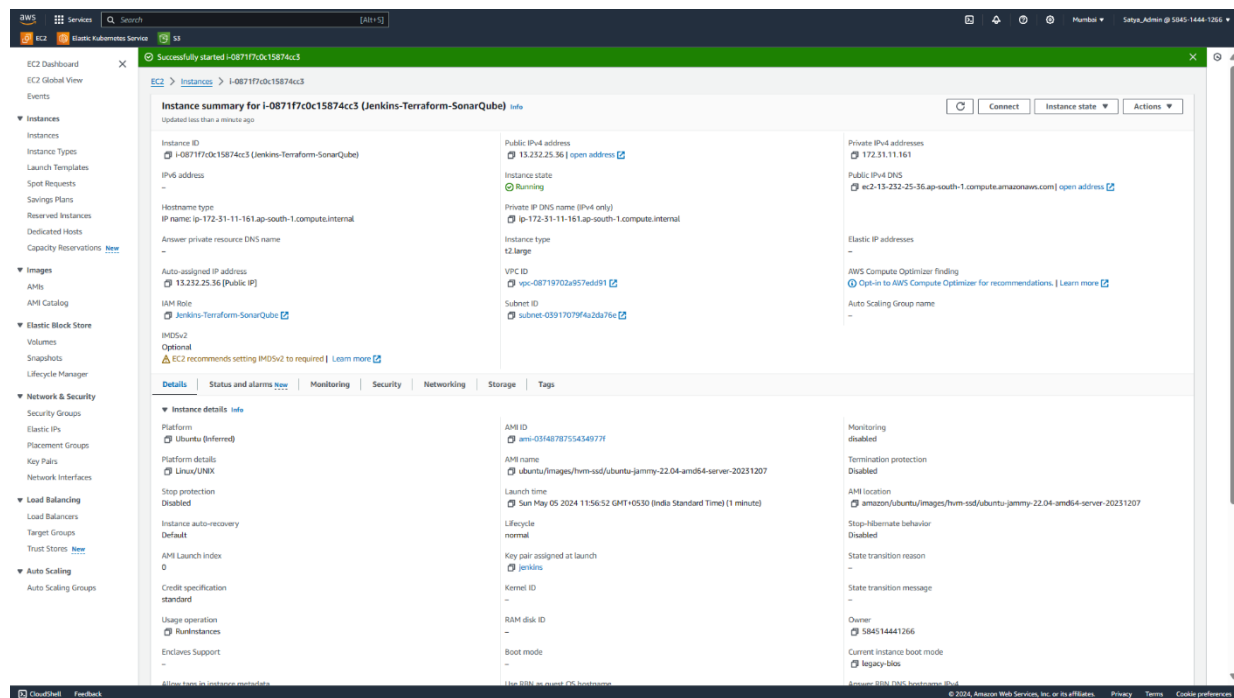
### V. LITERATURE SURVEY

- [1]. Pratik Thantharate "Intelligent Monitor: Empowering DevOps Environments with Advanced Monitoring and Observability", International Conference on Information Technology (ICIT) 979-8-3503-2006-0/23 ©2023 IEEE. Challenges of monitoring and observability in complex, distributed systems, such as data overload, alert fatigue, and visibility issues.
- [2]. Rizki Sadewa, Muharman Lubis, Hanif Fakhurroja "Rational of Study: Speed Up Production Development by DevOps Culture. It compares different deployment approaches, such as agile, traditional, and DevOps.
- [3]. Artur Cepuc, Robert Botez, Ovidiu Craciun, Iustin-Alexandru Ivanciu, Virgil Dobrota "Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services Using Jenkins, Ansible and Kubernetes", 2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet) | 978-0-7381-1265-7/20 ©2020 IEEE | DOI: 10.1109/ROEDUNET51892.2020.9324857.
- [4]. Mali Senapathi, Jim Buchan, Hady Osman "DevOps Capabilities, Practices, and Challenges: Insights from a Case Study", June 28, 2018, Christchurch, New Zealand © 2018 ACM. 978-1-4503-6403-4/18/06 <https://doi.org/10.1145/3210459.3210465>. The main challenges of DevOps were finding and training skilled staff and implementing appropriate technologies and tools.
- [5]. Fowler, M. (2006). Continuous Integration. [martinfowler.com](https://martinfowler.com). Retrieved from <https://martinfowler.com/articles/continuousIntegration.html>
- [6]. Humble, J., & Farley, D. (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional.



## VII. IMPLEMENTATION

## AWS CONFIGURATION AND RESOURCE ALLOCATION



Configuring AWS services according to project requirements involves a meticulous setup of various components to ensure a secure, scalable, and efficient infrastructure. This typically begins with the creation of a Virtual Private Cloud (VPC), which provides an isolated section of the AWS cloud where resources can be launched.

Within the VPC, subnets are established to segment the network and control traffic flow, with each subnet typically spanning a specific availability zone to enhance fault tolerance. Security groups are then implemented to control inbound and outbound traffic to AWS resources, acting as virtual firewalls that dictate which traffic is allowed or denied.

Additionally, Identity and Access Management (IAM) roles are configured to manage permissions and access to AWS resources securely. IAM roles define what actions users, services, and resources can perform within the AWS environment, helping enforce the principle of least privilege and bolstering security.

## KUBERNETES CLUSTER DEPLOYMENT AND CONFIGURATION

Deploying a Kubernetes cluster on AWS involves leveraging tools such as kops (Kubernetes Operations) or Amazon EKS (Elastic Kubernetes Service) to orchestrate the setup and management of the cluster infrastructure. With Kops, users can define the cluster's desired state using declarative configuration files, specifying parameters like instance types, networking options, and high availability settings. Alternatively, Amazon EKS provides a managed Kubernetes service, abstracting away the underlying complexities of cluster provisioning and management while offering features like automated version upgrades and integrated logging and monitoring.

```

[Pipeline] sh
+ terraform init

[0m[1mInitializing the backend...[0m

[0m[1mInitializing provider plugins...[0m
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v5.39.1

[0m[1m[32mTerraform has been successfully initialized![0m[32m[0m
[0m[32m
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.[0m
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (terraform validate)
[Pipeline] sh
+ terraform validate
[32m[1mSuccess![0m The configuration is valid.
[0m
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage

```

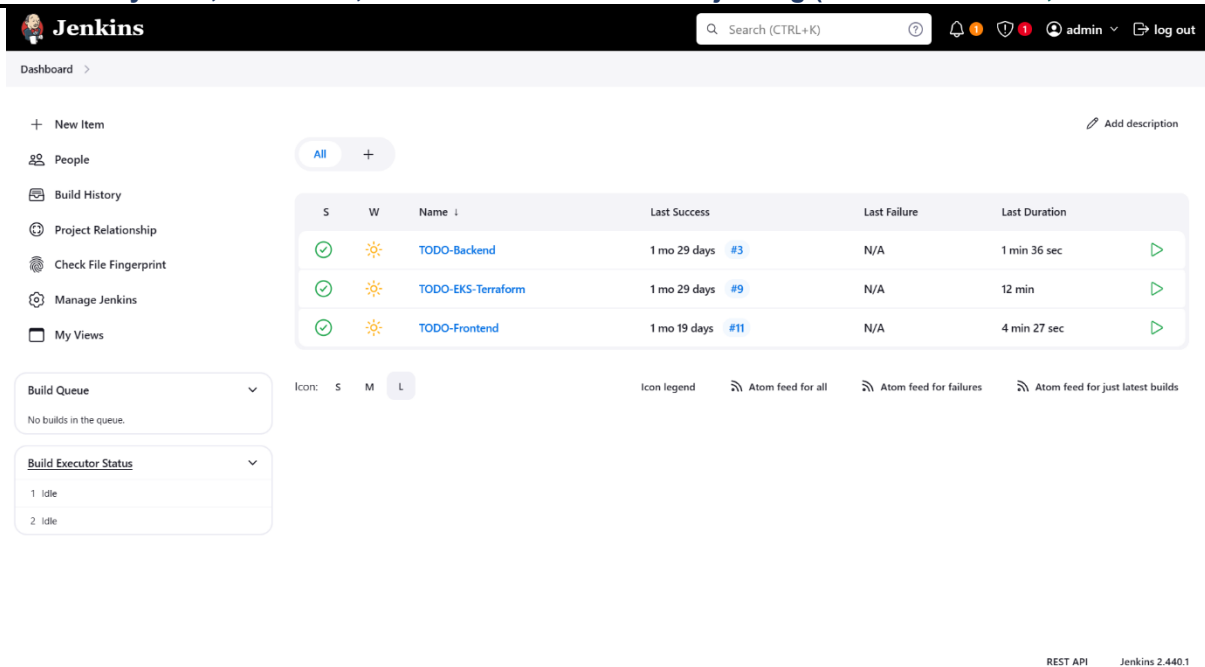
Configuring the Kubernetes cluster for high availability and scalability entails designing a resilient architecture that spans multiple availability zones, ensuring fault tolerance and minimizing downtime. This often involves deploying worker nodes across different AWS regions and distributing workload replicas across these nodes for redundancy. Additionally, implementing features like auto-scaling enables the cluster to dynamically adjust its capacity based on resource demands, ensuring optimal performance and cost efficiency.

By deploying a Kubernetes cluster on AWS and configuring it for high availability and scalability, organizations can harness the power of container orchestration to streamline application deployment, enhance reliability, and scale their infrastructure to meet evolving business needs.

## JENKINS CONFIGURATION FOR BUILD AND DEPLOYMENT PIPELINES

Jenkins is a Continuous Integration and Continuous Delivery Tool used to automate the Build, Test, and Deployment process. According to the below figure it consists of three jobs as follows:

1. TODO-Backend: This Job performs the build and uploads the Backend docker image to the docker hub container registry.
2. TODO-EKS-Terraform: This Job performs the setup of the infrastructure of the AWS Cluster to deploy the TODO application.
3. TODO-Frontend: This Job performs the build and uploads the Frontend docker images to the docker hub container registry.

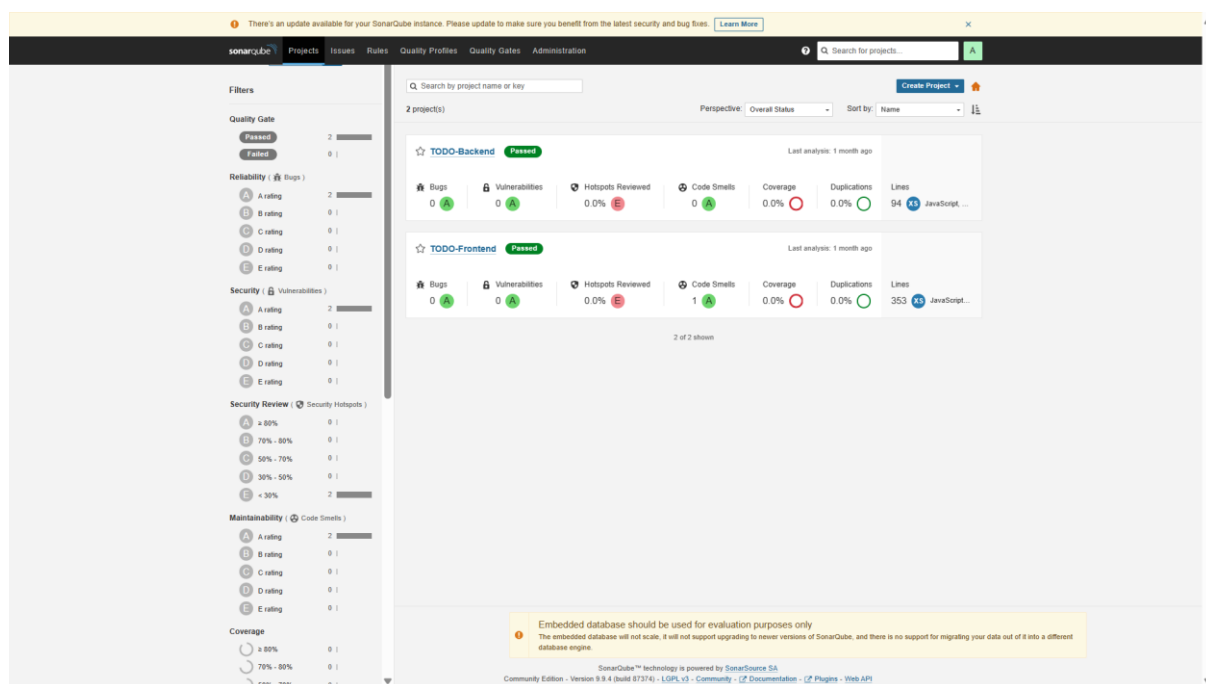


The screenshot shows the Jenkins Dashboard. At the top, there's a search bar and user information (admin, log out). The left sidebar contains navigation links: New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, and My Views. The main area displays a table of builds with columns: S, W, Name, Last Success, Last Failure, and Last Duration. The table lists three builds: TODO-Backend, TODO-EKS-Terraform, and TODO-Frontend. Below the table, there are sections for Build Queue (No builds in the queue) and Build Executor Status (1 idle, 2 idle). At the bottom right, it says REST API and Jenkins 2.440.1.

| S | W | Name               | Last Success     | Last Failure | Last Duration |
|---|---|--------------------|------------------|--------------|---------------|
| ✓ | ☀ | TODO-Backend       | 1 mo 29 days #3  | N/A          | 1 min 36 sec  |
| ✓ | ☀ | TODO-EKS-Terraform | 1 mo 29 days #9  | N/A          | 12 min        |
| ✓ | ☀ | TODO-Frontend      | 1 mo 19 days #11 | N/A          | 4 min 27 sec  |

The dashboard also shows that there are no builds in the queue and that there are two idle build executors.

## SONARQUBE INTEGRATION FOR CODE ANALYSIS



The screenshot shows the SonarQube dashboard. At the top, there's a notification about an update. The left sidebar contains filters for Quality Gate, Reliability, Security, Security Review, Maintainability, and Coverage. The main area displays a list of projects: TODO-Backend and TODO-Frontend. Each project has a 'Passed' status and a 'Last analysis' timestamp. Below the project list, there's a table with columns: Bugs, Vulnerabilities, Hotspots Reviewed, Code Smells, Coverage, Duplications, and Lines. The table shows data for both projects. At the bottom, there's a warning about the embedded database and a footer with SonarQube version information.

| Project       | Status | Last analysis | Bugs | Vulnerabilities | Hotspots Reviewed | Code Smells | Coverage | Duplications | Lines |
|---------------|--------|---------------|------|-----------------|-------------------|-------------|----------|--------------|-------|
| TODO-Backend  | Passed | 1 month ago   | 0    | 0               | 0.0%              | 0           | 0.0%     | 0.0%         | 94    |
| TODO-Frontend | Passed | 1 month ago   | 0    | 0               | 0.0%              | 1           | 0.0%     | 0.0%         | 353   |

Integrating SonarQube into the CI/CD pipeline is a pivotal step toward enhancing code quality and ensuring the reliability of software deployments. SonarQube serves as a robust static code analysis tool, offering comprehensive insights into codebase health and identifying potential vulnerabilities, bugs, and code smells.

## Todo List

Connected to backend

### Todo List

| Task | Status        | Deadline              | Actions                           |
|------|---------------|-----------------------|-----------------------------------|
| hi   | Pending/To-Do | 4/22/2024, 6:49:00 AM | <div>Edit</div> <div>Delete</div> |

### Add Task

Task

Status

Deadline

📅

Add Task

Our project offers a cutting-edge Todo application aimed at transforming task and deadline management within our team. This comprehensive application provides users with a smooth platform to input task names, specify their status as pending or completed, and allocate deadlines to each task.

## CONCLUSION

Developing and using an automated DevOps pipeline for deploying MERN stack web applications on AWS with Kubernetes offers many benefits. By combining containerization, orchestration, and cloud infrastructure, we've created a streamlined and automated deployment process. Continuous integration and deployment practices have sped up our development cycles and reduced time-to-market, while infrastructure as code principles ensure consistency and reproducibility.

Monitoring and logging capabilities provide real-time insights into application performance, enabling proactive troubleshooting. Automated testing maintains software quality throughout development. Overall, this project shows the power of modern DevOps practices and cloud-native technologies for building and managing applications. Such automated pipelines not only make deployment faster but also enhance reliability and scalability. As organizations embrace cloud and containerization, these pipelines will drive innovation and agility in software development and deployment.

## REFERENCES

- [1]. Pratik Thantharate "IntelligentMonitor: Empowering DevOps Environments With Advanced Monitoring and Observability" , International Conference on Information Technology (ICIT) 979-8-3503-2006-0/23 ©2023 IEEE.
- [2]. Rizki Sadewa, Muharman Lubis, Hanif Fakhurroja "Rational of Study: Speed Up Production Development By DevOps Culture.
- [3]. Artur Cepuc, Robert Botez, Ovidiu Craciun, Iustin-Alexandru Ivanciu, Virgil Dobrota "Implementation of a Continuous Integration and Deployment Pipeline for Containerized Applications in Amazon Web Services Using Jenkins, Ansible and Kubernetes", 2020 19th RoEduNet Conference: Networking in Education and Research (RoEduNet) | 978-0-7381-1265-7/20 ©2020 IEEE | DOI: 10.1109/ROEDUNET51892.2020.9324857.
- [4]. Mali Senapathi, Jim Buchan, Hady Osman "DevOps Capabilities, Practices, and Challenges: Insights from a Case Study", June 28 2018, Christchurch, New Zealand © 2018 ACM. 978-1-4503-6403-4/18/06 <https://doi.org/10.1145/3210459.3210465>
- [5]. Fowler, M. (2006). Continuous Integration. [martinfowler.com](https://martinfowler.com). Retrieved from <https://martinfowler.com/articles/continuousIntegration.html>
- [6]. Humble, J., & Farley, D. (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley Professional.
- [7]. Newman, S. (2019). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.
- [8]. Burns, B., & Vapnitchnaia, N. (2021). Kubernetes Best Practices: Blueprints for Building Successful Applications on Kubernetes. O'Reilly Media.
- [9]. Levere, J., & Squillante, P. (2020). Terraform: Up & Running: Writing Infrastructure as Code. O'Reilly Media.